Designing a Family of Coordination Algorithms *

Keith S. Decker and Victor R. Lesser
Department of Computer Science
University of Massachusetts, Amherst, MA 01003
Email: DECKER@CS.UMASS.EDU

May 27, 1994

Abstract

Many researchers have shown that there is no single best organization or coordination mechanism for all environments. This paper discusses the design and implementation of an extendable family of coordination mechanisms, called Generalized Partial Global Planning (GPGP), that form a basic set of coordination mechanisms for teams of cooperative computational agents. The important features of this approach include a set of modular coordination mechanisms (any subset or all of which can be used in response to a particular task environment); a general specification of these mechanisms involving the detection and response to certain abstract *coordination relationships* in the incoming task structure that are not tied to a particular domain; and a separation of the coordination mechanisms from an agent's local scheduler that allows each to better do the job for which it was designed. We will also discuss the interactions between these mechanisms and how to decide when each mechanism should be used, drawing data from simulation experiments of multiple agent teams working in abstract task environments.

1 Introduction

This paper presents a formal description of the Generalized Partial Global Planning (GPGP) coordination approach [6] and extends it to create a family of coordination algorithms that can be adapted to different environments. The GPGP family of algorithms is based on recognizing and reacting to the characteristics of certain coordination relationships, an approach shared with Von Martial's work on the *favor* relationship [23]. The approach described here is based on modular components (called 'mechanisms') that work in conjunction with, but do not replace, a fully functional agent with a local scheduler. Each component or mechanism can be added as required in reaction to the environment in which the agents find themselves a part.

*This work was supported by DARPA contract N00014-92-J-1698, Office of Naval Research contract N00014-92-J-1450, and NSF contract CDA 8922572. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

An individual algorithm in the family is defined by a particular set of active mechanisms and their associated parameters.

This paper will also summarize experimental results that involve a complete implementation of GPGP and a separately developed real-time local scheduler [16, 15]. We will show how to decide when a particular mechanism is useful, how some family members perform relative to a centralized algorithm, and what the space of possible coordination algorithms looks like for the five mechanisms currently defined. We analyze the performance of this family of algorithms through simulation in conjunction with the heuristic real-time local scheduler and randomly generated abstract task environments. In these environments, the agents attempt to maximize the system-wide total utility (a quantity called 'quality', described later) by executing sequences of interrelated 'methods'. The agents do not initially have a complete view of the problem solving situation, and the execution of a method at one agent can either positively or negatively affect the execution of other methods at other agents.

The GPGP approach specifies three basic areas of the agent's coordination behavior: how and when to communicate and construct non-local views of the current problem solving situation; how and when to exchange the partial results of problem solving; how and when to make and break *commitments* to other agents about what results will be available and when. The use of commitments in the GPGP family of algorithms is based on the ideas of many other researchers [2, 21, 1, 18]. Each agent also has a heuristic local scheduler that decides what actions the agent should take and when, based on its current view of the problem solving situation (including the commitments it has made), and a utility function. The coordination mechanisms supply non-local views of problem solving to the local scheduler, including *what* non-local results will be available locally, and *when* they will be available. The local scheduler creates (and monitors the execution of) schedules that attempt to maximize group quality through both local action and the use of non-local actions (committed to by other agents).

One way to think about this work is that the GPGP approach views coordination as *modulating* local control, not supplanting it—a two level process that makes a clear distinction between coordination behavior and local scheduling. This process occurs via a set of coordination mechanisms that post constraints to the local scheduler about the importance of certain tasks and appropriate times for their initiation and completion. By concentrating on the creation of local scheduling constraints, we avoid the sequentiality of scheduling in the original PGP algorithm [12] that occurs when there are multiple plans. By having separate modules for coordination and local scheduling, we can also take advantage of advances in real-time scheduling algorithms to produce cooperative distributed problem solving systems that respond to real-time deadlines. We can also take advantage of local schedulers that have a great deal of domain scheduling knowledge already encoded within them. Finally, our approach allows consideration of termination issues that were glossed over in the PGP work (where termination was handled by an external oracle). This paper will also discuss how to decide when each mechanism should be used and how the mechanisms interact, drawing on simple models of generic task structures and data from simulation experiments where multiple agent teams work in abstract task environments.

The way that we specify the family of algorithms in a general, domain-independent way (as responses to certain environmental situations and interactions with a local scheduler) is very important. It leads to the eventual construction of libraries of reusable coordination components that can be chosen (customized) with respect to certain attributes of the target

application.

1.1 The Task Environment

The observation that no single organization or coordination algorithm is 'the best' across environments, problem-solving instances, or even particular situations is a common one in the study of both human organizational theory (especially contingency theory) [19, 14, 22] and cooperative distributed problem solving [13, 11, 10, 8]. Key features of task environments demonstrated in both these threads of work that lead to different coordination mechanisms include those related to the structure of the environment (the particular kinds and patterns of **interrelationships** or dependencies that occur between tasks) and environmental **uncertainty** (both in the *a priori* structure of any problem-solving episode and in the outcome's of an agent's actions; for example, the presence of both uncertainty and concomitant high variance in a task structure). This makes it important for a general approach to coordination (i.e., one that will be used in many domains) to be appropriately parameterized so that the overhead activities associated with the algorithm, in terms of both communication and computation, can be varied depending upon the characteristics of the application environment.

So far we have made two important statements: that there is ample evidence that no single coordination algorithm is 'the best' across different environments, and that we are going to describe a *family* of coordination algorithms. That we need a family of algorithms follows from the first statement; we will now briefly discuss how that family is organized. At the most abstract level, each of the five mechanisms we are about to describe are parameterized independently (the first two have three possible settings and the last three can be in or out) for a total of 72 combinations. Many of these combinations do not show significantly different performance in randomly generated episodes, as will be discussed in the experiments (Section 3), although they may allow for fine-tuning in specific applications. More mechanisms can (and have) been added to expand the family, but the family can also be enlarged by making each mechanism more situation-specific. For example, mechanisms can have their parameters set by a mapping from dynamic meta-level measurements such as an agent's load or the amount of real-time pressure. Mechanisms can be 'in' or 'out' for individual *classes* of task groups, or tasks, or even specific coordination relationships, that re-occur in particular environments. The cross product of these dynamic environmental cues provides a large but easily enumerated space of potential coordination responses that are amenable to the adaptation of the coordination mechanisms over time by standard machine learning techniques. In the experimental section of this paper we will only consider the coarsest parameterization of the mechanisms. The reader should keep in mind when reading about the various coordination techniques that we have explicitly stated that they will not be useful in every situation. Instead, our purpose is to carefully describe the mechanisms and their interactions so that a decision about their inclusion in an application can be made analytically or experimentally (we will present an example in Section 3).

1.1.1 Uncertainty

Less uncertainty in the environment means less uncertainty in the existence and extent of the task interdependencies, and less uncertainty in local scheduling—therefore the agents need less complex coordination behaviors (communication, negotiation, partial plans, etc) [20]. For

example, one can have cooperation without communication [17] if certain facts are known about the task structure by all agents. In this paper agents will not have *a priori* information about the structure of an episode, but they will be able to get information about a subset of the structure after the start of problem solving—no single agent working alone will be able to construct a view of the entire problem (task structure) facing the group. This lack of information (another form of environmental uncertainty) can cause the local scheduler to make sub-optimal decisions. Some of this uncertainty will arise from disparities in the objective (real) task structure and the subjective (agent-believed) task structure. For example, an agent may not know the true duration of a method and if the execution variance is high may not even have a good estimate. In this paper we will be looking at environments that do not have this characteristic (of large variance between objective characteristics and subjective estimates of those characteristics). Instead we will focus on a second source of structural uncertainty—each agent has only a partial subjective view of the current episode.

1.1.2 Task interrelationships

Task interrelationships include the relationships of tasks to the performance criteria by which we will evaluate a system, to the control decision structures of the agents which make up a system, and to the performance of other tasks. We will represent, analyze, and simulate the effects of task interrelationships using the TÆMS (Task Analysis, Environment Modeling, and Simulation) framework [9]. The important features of TÆMS include its layered description of environments (*objective* reality, *subjective* mapping to agent beliefs, *generative* description of the other levels across single problem-solving instances); its acceptance of any performance criteria (based on temporal location and *quality* of task executions); and its non-agent-centered point of view that can be used by researchers working in either formal systems of mental-state-induced behavior or experimental methodologies. The TÆMS objective **subtask** relationship indicates the relationship of tasks to system performance criteria; the subjective mapping indicates the initial beliefs available to an agent's control decision structures; various non-local effects such as **enables** and **facilitates** indicate how the execution of one task affects the duration or quality of another task. When a relationship extends between parts of a task structure that are subjectively believed by different agents, we call it a *coordination relationship.* The basic idea behind Generalized Partial Global Planning (GPGP) is to detect and respond appropriately to these coordination relationships.

1.1.3 Representing task environments

A problem instance (called an *episode* $\mathbf{E}$) is defined as a set of task groups, each with a deadline $\mathrm{D}(\mathcal{T})$, such as $\mathbf{E} = \langle \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n \rangle$. The task groups (or subtasks within them) may arrive at different times ($\mathrm{Ar}(x)$ is the arrival time of task x), but in this paper they will all arrive at the start of problem solving (like the classic DVMT). While task groups are independent of one another computationally[1], the tasks within a single task group are in general not independent. Figure 1 shows an objective task group and agent A's subjective view of that task group. A task group consists of a set of tasks related to one another by a **subtask** relationship that forms an acyclic graph (here, a tree). The circles higher up in the tree represent various subtasks involved

[1] Except for the use of the processor(s) or other physical resources.

in the task group, and indicate precisely how quality will accrue depending on what leaf tasks are executed and when. Tasks at the leaves of the tree (without subtasks) represent *methods*, which are the actual computations or actions the agent will execute (in the figure, these are shown as boxes). The arrows between tasks and/or methods indicate other task interrelationships where the execution of some method will have a positive or negative effect on the quality or duration of another method. The presence of these interrelationships make this an NP-hard scheduling problem; further complicating factors for the local scheduler include the fact that multiple agents are executing related methods, that some methods are redundant (executable at more than one agent), and that the subjective task structure may differ from the real objective structure. This notation and associated semantics are formally defined in [9].

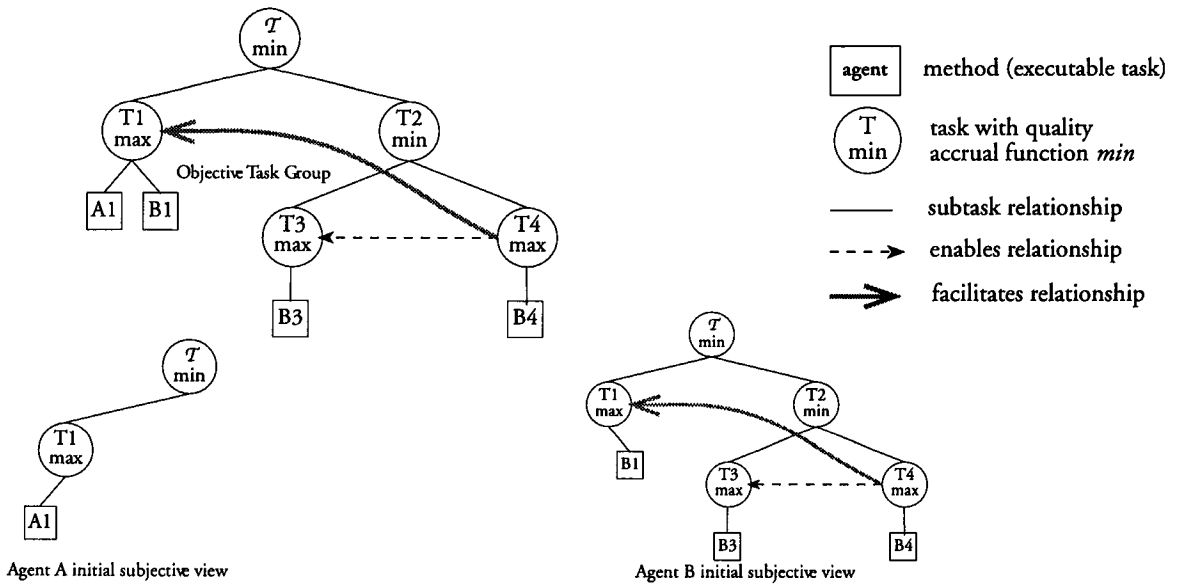

Figure 1: Agent A and B's subjective views (bottom) of a typical objective task group (top)

1.2 The Agent Architecture

The TÆMS framework makes very few assumptions about the architecture of agents—agents are loci of belief and action. Agents have some control mechanism that decides on actions given the agent's current beliefs. There are three classes of actions: method execution, communication, and information gathering. Method execution actions cause quality to accrue in a task group (as indicated by the task structure). Communication actions are used to send the results of method executions (which in turn may trigger the effects of various task interrelationships) or meta-level information. Information gathering actions add newly arriving task structures, or new communications, to an agent's set of beliefs.

Formally, we write $B_A^t(x)$ to mean agent A subjectively believes x at time t (from Shoham[21]). We will shorten this to $B(x)$ when we don't have a particular agent or time in mind. An agent's subjective beliefs about the current episode $B(\mathbf{E})$ includes the agent's beliefs about various task groups (e.g., $B(\mathcal{T}_i \in \mathbf{E})$), and an agent's beliefs about each task group includes beliefs about the tasks in that task group (e.g., $B(T_a, M_b \in \mathcal{T}_i)$) and the relationships between these tasks (e.g., $B(\mathbf{enables}(T_a, M_b))$).

The GPGP family of coordination algorithms makes stronger assumptions about the agent architecture. Most importantly, it assumes the presence of a local scheduling mechanism (to be described in the next section) that can decide what method execution actions should take place and when. It assumes that agents do not intentionally lie and that they believe what they are told (i.e. if agent $A1$ tells agent $A2$ at time t_1 with communication delay δ that $B_{A1}(\mathsf{enables}(T_a, M_b))$, then $B^{t_2}_{A2}(\mathsf{enables}(T_a, M_b))$ where $t_2 \geq t_1 + \delta$ is the earliest time after the communication arrives that agent $A2$ performs a new communication information gathering action to read the message from $A1$). However, because agents can believe and communicate only subjective information, they may unwittingly transmit information that is inconsistent with an objective view (this can cause, among other things, the phenomena of *distraction*). Finally, the GPGP family approach requires domain-dependent code to detect or predict the presence of coordination relationships in the local task structure [5]. In this paper we will refer to that domain-dependent code as the information gathering action *detect-coordination-relationships*; we will describe this action more in Section 2.2.

1.3 The Local Scheduler

Each GPGP agent contains a local scheduler that takes as input the current, subjectively believed task structure and produces a schedule of what methods to execute and when. Using the information in the subjective structure about the potential duration, potential quality, and relationships of the methods, it chooses and orders executable methods in an attempt to maximize a pre-defined utility measure for each task group $\mathcal{T}$. In this paper the utility function is the sum of the task group qualities. The local scheduler attempts to maximize this utility function $\mathrm{U}(\mathbf{E}) = \sum_{\mathcal{T} \in \mathbf{E}} Q(\mathcal{T}, \mathrm{D}(\mathcal{T}))$, where $Q(T, t)$ denotes the quality of T at time t as defined in [9].[2]

Beside the subjective task structure, the local scheduler should accept a set of *commitments* $\mathbf{C}$ from the coordination component. These commitments are extra constraints on the schedules that are produced by the local scheduler. For example, if method 1 is executable by agent A and method 2 is executable by agent B, and the methods are redundant, then agent A's coordination mechanism may *commit* agent A to do method 1 both locally and socially (commitments are directed to particular agents in the sense of the work of Shoham and Castelfranchi [1, 21]) by communicating this commitment to B (so that agent B's coordination mechanism records agent A's commitment, see the description of non-local commitments $\mathbf{NLC}$ below). This paper will use two types of commitments: $C(\mathsf{Do}(T, q))$ is a commitment to 'do' (achieve quality for) T and is satisfied at the time t when $Q(T, t) \geq q$; the second type $C(\mathsf{DL}(T, q, t_{dl}))$ is a 'deadline' commitment to do T by time t_{dl} and is satisfied at the time t when $[Q(T, t) \geq q] \wedge [t \leq t_{dl}]$.[3]

A schedule S produced by a local scheduler will consist of a set of methods and start times: $S = \{\langle M_1, t_1\rangle, \langle M_2, t_2\rangle, \ldots, \langle M_n, t_n\rangle\}$. The schedule may include idle time, and the local scheduler may produce more than one schedule upon each invocation in the situation where not all commitments can be met. The different schedules represent different ways of partially satisfying the set of commitments (see [15] and the next section). The function $\mathrm{Violated}(S)$ returns the set of commitments that are believed to be violated by the schedule. For violated deadline commitments $C(\mathsf{DL}(T, q, t_{dl})) \in \mathrm{Violated}(S)$ the function $\mathrm{Alt}(C, S)$ returns an

[2] Note that quality does not accrue after a task group's deadline.

[3] Other commitments, such as to the earliest start time of a task, may also prove useful.

alternative commitment $C(\mathsf{DL}(T, q, t^*_{dl}))$ where $t^*_{dl} = \min t$ such that $Q(T, t) \geq q$ if such a t exists, or NIL otherwise. For a violated Do commitment an alternative may contain a lower minimum quality, or no alternative may be possible.

The final piece of information that is used by the local scheduler is the set of non-local commitments made by other agents **NLC**. This information can be used by the local scheduler to coordinate actions between agents. For example the local scheduler could have the property that, if method M_1 is executable by agent A and is the only method that **enables** method M_2 at agent B (and agent B knows this $B_B(\mathsf{enables}(M_1, M_2))$), and $B_A(C(\mathsf{DL}(M_1, q, t_1))) \in B_B(\mathbf{NLC})$, then for every schedule S produced by agent B, $\langle M_2, t\rangle \in S \Rightarrow t \geq t_1$. The function $U_{\mathsf{est}}(\mathbf{E}, S, \mathbf{NLC})$ returns the estimated utility at the end of the episode if the agent follows schedule S and all non-local commitments in **NLC** are kept. Thus we may define the local scheduler as a function $\mathrm{LS}(\mathbf{E}, \mathbf{C}, \mathbf{NLC})$ returning a set of schedules $\mathbf{S} = \{S_1, S_2, \ldots, S_m\}$. More detailed information about this kind of interface between the local scheduler and the coordination component may be found in [15].

This is an extremely general definition of the local scheduler, and is the minimal one necessary for the GPGP coordination module. Stronger definitions than this will be needed for more predictable performance, as we will discuss later. Ideally, the optimal local scheduler would find both the schedule with maximum utility $\hat{S}_U$ and the schedule with maximum utility that violates no commitments $\hat{S}_V$. In practice, however, a heuristic local scheduler will produce a set of schedules where the schedule of highest utility S_U is not necessarily optimal: $\mathrm{U}(\mathbf{E}, S_U, \mathbf{NLC}) \leq \mathrm{U}(\mathbf{E}, \hat{S}_U, \mathbf{NLC})$.

2 The Coordination Mechanisms

The role of the coordination mechanisms is to provide constraints to the local scheduler (by modifying portions of the subjective task structure of the episode **E** or by making commitments in **C**) that allow the local scheduler to construct objectively better schedules. The modules fulfill this role by (variously) communicating private portions of its task structures, communicating results to fulfill non-local commitments, and making commitments to respond to *coordination relationships* **between** portions of the task structure controllable by *different* agents or **within** portions controllable by *multiple* agents.[4]

The five mechanisms we will describe in this paper form a basic set that provides similar functionality to the original partial global planning algorithm as explained in [6]. Mechanism 1 exchanges useful private views of task structures; Mechanism 2 communicates results; Mechanism 3 handles redundant methods; Mechanisms 4 and 5 handle hard and soft coordination relationships. More mechanisms can be added, such as one to update utilities across agents as discussed in the next section, or to balance the load better between agents. The mechanisms are independent in the sense that they can be used in any combination. If inconsistent constraints are introduced, the local scheduler would return at least one violated constraint in all its schedules, which would be dealt with as discussed in the next section. Since the local scheduler is boundedly rational and satisfices instead of optimizing, it may do this even if constraints are not inconsistent (i.e. it does not search exhaustively).

[4] We say a subtree of a task structure is *controllable* by an agent if that agent has at least one executable method in that subtree.

2.1 The Substrate Mechanisms

All the specific coordination mechanisms rest on a common substrate that handles information gathering actions (new task group arrivals and receiving communications), invoking the local scheduler and choosing a schedule to execute (including dealing with violated commitments), and deciding when to terminate. Information gathering is done at the start of problem solving, whenever the agent is otherwise idle (but not ready to terminate), and when communications are expected from other agents. Communications are expected in response to certain events (such as after the arrival of a new task group) or as indicated in the set of non-local commitments **NLC**. This is the minimal general information gathering policy.[5] Termination occurs for an agent when the agent is idle, has no expected communications, and no outstanding commitments.

Choosing a schedule is more complicated. From the set of schedules **S** returned by the local scheduler, two particular schedules are identified: the schedule with the highest utility S_U and the best committed schedule S_C. If they are the same, then that schedule is chosen. Otherwise, we examine the sum of the changes in utility for each commitment. Each commitment, when created, is assigned the estimated utility U_{est} for the task group of which it is a part. This utility may be updated over time (when other agents depend on the commitment, for example). We then choose the schedule with the largest positive change in utility. This allows us to abandon commitments if doing so will result in higher overall utility. The coordination substrate does not use the local scheduler's utility estimate U_{est} directly on the entire schedule because it is based only on a local view, and the coordination mechanism may have received non-local information that places a higher utility on a commitment than it has locally.

For example, at time t agent A may make a commitment C_1 on task $T \in \mathcal{T}_1 \in \mathbf{E}$ that results in a schedule S_1. C_1 initially acquires the estimated utility of the task group of which it is a part, $U(C_1) \leftarrow U_{\text{est}}(\{\mathcal{T}_1\}, S_1, B_A(\mathbf{NLC}))$. Let $U(C_1) = 50$. After communicating this commitment to agent B (making it part of $B_B(\mathbf{NLC})$, agent B uses the commitment to improve $U_{\text{est}}(\{\mathcal{T}_1\}, S_2, B_B(\mathbf{NLC}))$ to 100. A coordination mechanism can detect this discrepancy and communicate the utility increase back to agent A, so that when agent A considers discarding the commitment, the coordination substrate recognizes the non-local utility of the commitment is greater than the local utility.[6]

If both schedules have the same impact, the one that is more negotiable is chosen. Every commitment has a negotiability index (high, medium, or low) that indicates (heuristically) the difficulty in rescheduling if the commitment is broken. This index is set by the individual coordination mechanisms. For example, hard coordination relationships like **enables** that cannot be ignored will trigger commitments with low negotiability.

If the schedules are still equivalent, the shorter one is chosen, and if they are the same length, one is chosen at random. After a schedule S is chosen, if $\text{Violated}(S)$ is not empty, then each commitment $C \in \text{Violated}(S)$ is replaced with its alternative $\mathbf{C} \leftarrow \mathbf{C} \setminus C \cup \text{Alt}(C, S)$. If the commitment was made to other agents, the other agents are also informed of the change in the

[5] The minimal policy would examine each element of **NLC** at the appointed time and if the local schedule had changed so that the reception of the information would no longer have any effect, the associated information gathering action could be skipped.

[6] While it is clear that without this policy the system of agents will perform non-optimally, it is not clear how often the situation occurs or what the performance hit is. Future work will have to examine the costs and benefits of this policy; for this reason we do not include this mechanism among the five examined in this paper.

commitment. While this could potentially cause cascading changes in the schedules of multiple agents, it generally does not for three reasons: first, as we mentioned in the previous paragraph less important commitments are broken first; secondly, the resiliancy of the local schedulers to solve problems in multiple ways tends to damp out these fluctuations; and third, agents are time cognizant resource-bounded reasoners that interleave execution and scheduling (i.e., the agents cannot spend all day arguing over scheduling details and still meet their deadlines). We have observed this useful phenomenon before [7] and plan to analyze it in future work.

2.2 Mechanism 1: Updating Non-Local Viewpoints

Remember that each agent has only a partial, subjective view of the current episode. Agents can, therefore, communicate the private portions of their subjective task structures to develop better, non-local, views of the current episode. They could even communicate all of their private structural information, in an attempt to develop a global subjective view . The GPGP mechanism described here can communicate no private information ('none' policy, no non-local view), or all of it ('all' policy, global view), or take an intermediate approach ('some' policy, partial view): an agent communicates to other agents only the private structures that are related by some coordination relationship to a structure known by the other agents. The process of detecting coordination relationships between private and shared parts of a task structure is in general very domain specific, so in the experiments presented later in the paper we model this process by a new information gathering action, *detect-coordination-relationships,* that takes some fixed amount of the agent's time. This action is chosen when a new task group arrives (which adds new information to the agent's private task structures).

The set $\mathbf{P}$ of privately believed tasks or methods at an agent A (tasks believed at arrival time by A only) is then $\{x \mid \mathbf{Task}(x) \wedge \forall a \in \mathbf{A} \setminus A,\ \neg B_A(B_a^{\mathbf{Ar}(x)}(x))\}$, where $\mathbf{A}$ is the set of all agents and $\mathrm{Ar}(x)$ is the arrival time of x. Given this definition, the action *detect-coordination-relationships* returns the set of private coordination relationships $\mathbf{PCR} = \{r \mid T_1 \in \mathbf{P} \wedge T_2 \notin \mathbf{P} \wedge [r(T_1, T_2) \vee r(T_2, T_1)]\}$ between private and mutually believed tasks. The action does not return what the task T_2 is, just that a relationship exists between T_1 and some otherwise unknown task T_2. For example, in the DVMT, we have used the physical organization of agents to detect that Agent A's task T_1 in an overlapping sensor area is in fact related to some unknown task T_2 at agent B (i.e. $B_A(B_B(T_2))$) [6, 5]. The non-local view coordination mechanism then communicates these coordination relationships, the private tasks, and their context: if $r(T_1, T_2) \in \mathbf{PCR}$ and $T_1 \in \mathbf{P}$ then r and T_1 will be communicated by agent A to the set of agents $\{a \mid B_A(B_a(T_2))\}$.

For example, Figure 2 shows the local subjective beliefs of agents A and B after the communication from one another due to this mechanism. The agents' initial local view was shown previously in Figure 1. In this example, T_3 and T_4 are two elements in Agent B's private set of tasks $\mathbf{P}$, $\mathbf{facilitates}(T_4, T_1, \phi_d, \phi_q) \in \mathbf{PCR}$ (the facilitation relates a private task to a mutually believed task), and $\mathbf{enables}(T_4, T_3)$ is completely local to Agent B (it relates two private tasks). At the start of this section we mentioned that coordination relationships exist **between** portions of the task structure controllable by different agents (i.e., in $\mathbf{PCR}$) and **within** portions controllable by multiple agents. We'll denote the complete set of coordination relationships as $\mathbf{CR}$; this includes all the elements of $\mathbf{PCR}$ and all the relationships between non-private tasks. Some relationships are entirely local—between private tasks—and are only of

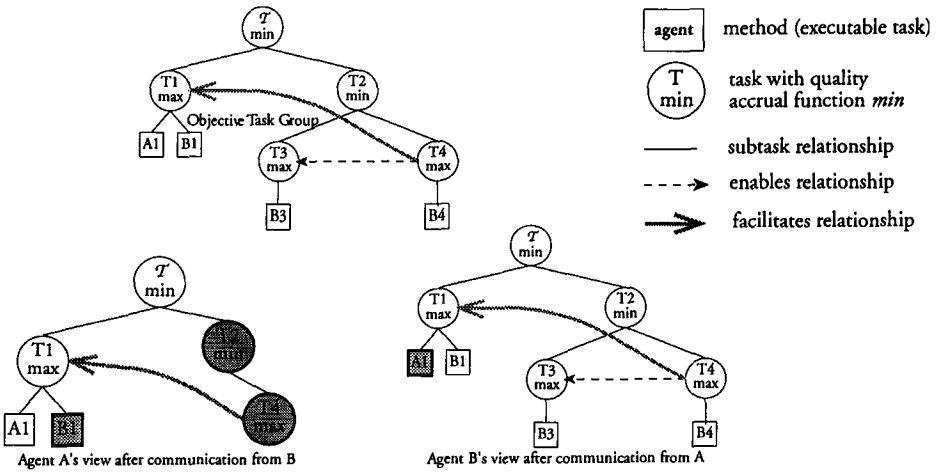

Figure 2: Agents A and B's local views after receiving non-local viewpoint communications via mechanism 1. Figure 1 shows the agents' initial states.

concern to the local scheduler. The purpose of this coordination mechanism is the exchange of information that expands the set of coordination relationships **CR**. Without this mechanism in place, **CR** will consist of only non-private relationships, and none that are in **PCR**. Since the primary focus of the coordination mechanisms is the creation of social commitments in response to coordination relationships (elements of **CR**), this mechanism can have significant indirect benefits. In environments where $|\mathbf{PCR}|$ tends to be small, very expensive to compute, or not useful for making commitments (see the later sections), this mechanism can be omitted.

2.3 Mechanism 2: Communicating Results

The result communication coordination mechanism has three possible policies: communicate only the results necessary to satisfy commitments to other agents (the minimal policy); communicate this information plus the final results associated with a task group ('TG' policy), and communicate all results ('all' policy). Extra result communications are broadcast to all agents, the minimal commitment-satisfying communications are sent only to those agents to whom the commitment was made (i.e., communicate the result of T to the set of agents $\{A \in \mathbf{A} \mid B(B_A(C(T))\}$.

2.4 Mechanism 3: Handling Simple Redundancy

Potential redundancy in the efforts of multiple agents can occur in several places in a task structure. Any task that uses a 'max' quality accumulation function (one possible semantics for an 'OR' node) indicates that, in the absence of other relationships, only one subtask needs to be done. When such subtasks are complex and involve many agents, the coordination of these agents to avoid redundant processing can also be complex; we will not address the general redundancy avoidance problem in this paper (see instead [20]). In the original PGP algorithm and domain (distributed sensor interpretation), the primary form of potential redundancy was simple method redundancy—the same result could be derived from the data from any of a number of sensors. The coordination mechanism described here is meant to address this

simpler form of potential redundancy.

The idea behind the simple redundancy coordination mechanism is that when more than one agent wants to execute a redundant method, one agent is randomly chosen to execute it and send the results to the other interested agents. This is a generalization of the 'static' organization algorithm discussed by Decker and Lesser [8]—it does not try to load balance, and uses one communication action (because in the general case the agents do not know beforehand, without communication, that certain methods are redundant[7]). The mechanism considers the set of potential redundancies $\mathbf{RCR} = \{r \in \mathbf{CR} \mid [r = \mathsf{subtask}(T, \mathbf{M}, \min)] \wedge [\forall M \in \mathbf{M}, \mathit{method}(M)]\}$. Then for all methods in the current schedule S at time t, if the method is potentially redundant then commit to it and send the commitment to Others(M) (non-local agents who also have a method in M):

$$\begin{aligned} & [\langle M, t_M \rangle \in S] \wedge \\ [\mathsf{subtask}(T, \mathbf{M}, \min) \in \mathbf{RCR}] & \wedge \\ [M \in \mathbf{M}] \quad \Rightarrow \quad & [C(\mathsf{Do}(M, Q_{\mathsf{est}}(M, \mathrm{D}(M), S))) \in C] \wedge \\ & [\mathsf{Comm}(M, \mathrm{Others}(\mathbf{M}), t) \in \mathcal{I}] \end{aligned}$$

See for example the top of figure 3—both agents commit to Do their methods for T_1.

After the commitment is made, the agent must refrain from executing the method in question if possible until any non-local commitments that were made simultaneously can arrive (the communication delay time δ). This mechanism then watches for multiple commitments in the redundant set $(\mathsf{subtask}(T, \mathbf{M}, \min) \in \mathbf{RCR}$, $M_1 \in \mathbf{M}$, $M_2 \in \mathbf{M}$, $C(\mathsf{Do}(M_1, q)) \in C$, and $B_B(C(\mathsf{Do}(M_2, q))) \in \mathbf{NLC})$[8] and if they appear, a unique agent is chosen randomly (but identically by all agents) from those with the best commitments to keep its commitment. All the other agents can retract their commitments. For example the bottom of figure 3 shows the situation after Agent B has retracted its commitment to Do B_1. If all agents follow the same algorithm, and communication channels are assumed to be reliable, then no second message (retraction) actually needs to be sent (because they all choose the same agent to do the redundant method). In the implementation described later, identical random choices are made by giving each method a unique random identifier, and then all agents choose the method with the 'smallest' identifier for execution.

Initially, all Do commitments initiated by the redundant coordination mechanism are marked highly negotiable. When a redundant commitment is discovered, the negotiability of the remaining commitment is lowered to medium to indicate the commitment is somewhat more important.

2.5 Mechanism 4: Handling Hard Coordination Relationships

Hard coordination relationships include relationships like $\mathsf{enables}(M_1, M_2)$ that indicate that M_1 must be executed before M_2 in order to obtain quality for M_2. Like redundant methods, hard coordination relationships can be culled from the set **CR**. The hard coordination

[7]The detection of redundant methods is domain-dependent, as discussed earlier. Since we are talking here about simple, direct redundancy (i.e. doing the exact same method at more than one agent) this detection is very straight-forward.

[8]Read "M_1 and M_2 are redundant, and I am doing M_1 and I know that B has committed to me to do M_2".

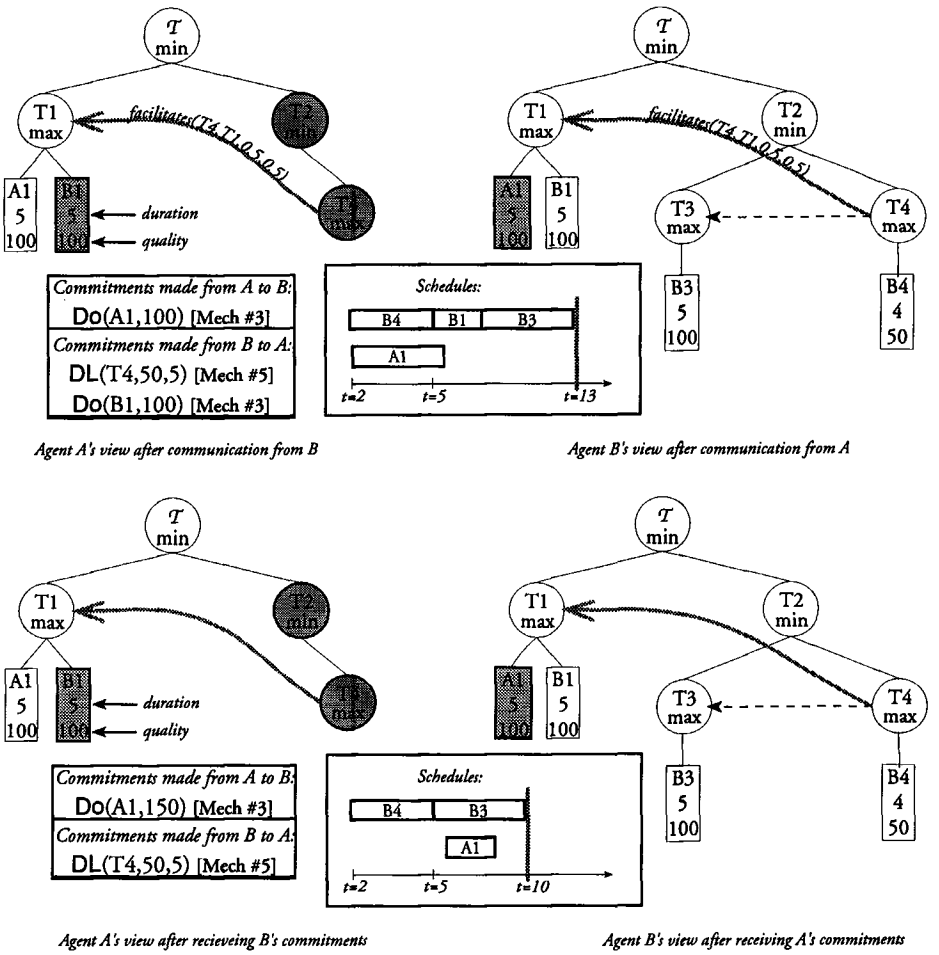

Figure 3: A continuation of Figures 1 and 2. At top: agents A and B propose certain commitments to one another via mechanisms 3 and 5. At bottom: after receiving the initial commitments, mechanism 3 removes agent B's redundant commitment.

mechanism further distinguishes the direction of the relationship—the current implementation only creates commitments on the predecessors of the **enables** relationship. We'll let $\mathbf{HPCR} \subset \mathbf{CR}$ indicate the set of potential hard predecessor coordination relationships. The hard coordination mechanism then looks for situations where the current schedule S at time t will produce quality for a predecessor in **HPCR**, and commits to its execution by a certain deadline both locally and socially:

$$\begin{aligned} &[Q_{\mathbf{est}}(T, \mathrm{D}(T), S) > 0] \ \wedge \\ &[\mathsf{enables}(T, \mathbf{M}) \in \mathbf{HPCR}] \ \Rightarrow \ [C(\mathsf{DL}(T, Q_{\mathbf{est}}(T, \mathrm{D}(T), S), t_{\mathbf{early}})) \in C] \ \wedge \\ &\qquad\qquad [\mathsf{Comm}(C, \mathsf{Others}(\mathbf{M}), t) \in \mathcal{I}] \end{aligned}$$

The next question is, by what time ($t_{\mathbf{early}}$ above) do we commit to providing the answer? One solution, usable with any local scheduler that fits our general description in Section 1.3, is to use the $\min t$ such that $Q_{\mathbf{est}}(T, \mathrm{D}(T), S) > 0$. In our implementation, the local scheduler provides a query facility that allows us to propose a commitment to satisfy as 'early' as possible (thus allowing the agent on the other end of the relationship more slack). We take

advantage of this ability in the hard coordination mechanism by adding the new commitment $C(\mathrm{DL}(T, Q_{\mathrm{est}}(T, \mathrm{D}(T), S), \text{"early"}))$ to the local commitments C, and invoking the local scheduler $LS(\mathbf{E}, \mathbf{C}, \mathbf{NLC})$ to produce a new set of schedules $\mathbf{S}$. If the preferred, highest utility schedule $S_U \in \mathbf{S}$ has no violations (highly likely since the local scheduler can simply return the same schedule if no better one can be found), we replace the current schedule with it and use the new schedule, with a potentially earlier finish time for T, to provide a value for $t_{\mathbf{early}}$. The new completed commitment is entered locally (with low negotiability) and sent to the subset of interested other agents.

If redundant commitments are made to the same task, the earliest commitment made by any agent is kept, then the agent committing to the highest quality, and any remaining ties are broken by the same method as before.

Currently, the hard coordination mechanism is a proactive mechanism, providing information that might be used by other agents to them, while not putting the individual agent to any extra effort. Other future coordination mechanisms might be added to the family that are reactive and request from other agents that certain tasks be done by certain times; this is quite different behavior that would need to be analyzed separately.

2.6 Mechanism 5: Handling Soft Coordination Relationships

Soft coordination relationships are handled analogously to hard coordination relationships except that they start out with high negotiability. In the current implementation the predecessor of a **facilitates** relationship is the only one that triggers commitments across agents, although **hinders** relationships are present. The positive relationship **facilitates**$(M_1, M_2, \phi_{\mathrm{d}}, \phi_{\mathrm{q}})$ indicates that executing M_1 before M_2 decreases the duration of M_2 by a 'power' factor related to ϕ_{d} and increases the maximum quality possible by a 'power' factor related to ϕ_{q} (see [9] for the details). A more situation-specific version of this coordination mechanism might ignore relationships with very low 'power'. The relationship **hinders**$(M_1, M_2, \phi_{\mathrm{d}}, \phi_{\mathrm{q}})$ is negative and indicates an *increase* in the duration of M_2 and a *decrease* in maximum possible quality. A coordination mechanism could be designed for **hinders** (and similar negative relationships) and added to the family. To be proactive like the existing mechanisms, a **hinders** mechanism would work from the *successors* of the relationship, try to schedule them late, and commit to an earliest start time on the successor. Figure 3 shows Agent B making a D commitment to do method B_4, which in turn allows Agent A to take advantage of the **facilitates**$(T_4, T1, 0.5, 0.5)$ relationship, causing method A_1 to take only half the time and produce 1.5 times the quality.

3 Experiments in Generalized Partial Global Planning

In this final section we will discuss experiments we have conducted with our implementation of these ideas:

- Methodologically, how should we decide when the addition of a particular mechanism is warranted?
- What is the performance of a system using all the mechanisms compared to a system that only broadcasts results? Compared to a system with a centralized scheduler?

- What is the performance space of the GPGP family, as delineated by the five existing mechanisms?

As we have stated several times in this paper, we do not believe that any of the mechanisms that collectively form the GPGP family of coordination algorithms are indispensable. What we can do is evaluate the mechanisms on the terms of their costs and benefits to cooperative problem solving both analytically and experimentally. This analysis and experimentation takes place with respect to a very general task environment that does not correspond to a particular domain. Doing this produces general results, but weaker than would be possible to derive in a single fixed domain because the performance variance between problem episodes will be far greater than the performance variance of the different algorithms within a single episode. Still, this allows us to determine broad characteristics of the algorithm family that can be used to reduce the search for a particular set of mechanism parameters for a particular domain (with or without machine learning techniques). We will also discuss statistical techniques to deal with the large between-episode variances that occur when using randomly-generated problems.

Our model of an abstract task environment has ten parameters; Table 1 lists them and the values used in the experiments described in the next two sections.

Parameter	Values (facilitation exps.)	Values (clustering exps.)
Mean Branching factor (Poisson)	1	1
Mean Depth (Poisson)	3	3
Mean Duration (exponential)	10	(1 10 100)
Redundant Method QAF	Max	Max
Number of task groups	2	(1 5 10)
Task QAF distribution	(20%/80% min/max)	(50%/50% min/max) (100%/0% min/max)
Hard CR distribution	(10%/90% enables/none)	(0%/100% enables/none) (50%/50% enables/none)
Soft CR distribution	(80%/10%/10% facilitates/hinders/none)	(0%/10%/90% facilitates/hinders/none) (50%/10%/40% facilitates/hinders/none)
Chance of overlaps (binomial)	10%	(0% 50% 100%)
Facilitation Strength	.1 .5 .9	.5

Table 1: Environmental Parameters used to generate the random episodes

The primary sources of overhead associated with the coordination mechanisms include action executions (communication and information gathering), calls to the local scheduler, and any algorithmic overhead associated with the mechanism itself. Table 2 summarizes the total amount of overhead from each source for each coordination mechanism setting and the coordination substrate. L represents the length of processing (time before termination), and d is a general density measure of coordination relationships. We believe that all of these amounts can be derived from the environmental parameters in Table 1, they can also be measured experimentally. Interactions between the presence of coordination mechanisms and these quantities include: the number of methods or tasks in $\mathbf{E}$, which depends on the non-local view mechanism; the number of coordination relationships $|\mathbf{CR}|$ or the subsets $\mathbf{RCR}, \mathbf{HPCR}, \mathbf{SPCR}$, which depends on the number of tasks and methods as well; and the number of commitments $|\mathbf{C}|$, which depends on each of the three mechanisms that makes commitments. The relationships between these quantities can be modeled and verified using methods similar to those in [8].

Mechanism *setting*	Communications	Information Gathering	Scheduler	*Other Overhead*
substrate	0	**E**+*idle*	L	$O(LC)$
nlv *none*	0	0	0	0
some	$O(d\mathbf{P})$	**E***detect-CRs*	0	$O(T \in \mathbf{E})$
all	$O(\mathbf{P})$	**E***detect-CRs*	0	$O(T \in \mathbf{E})$
comm *min*	$O(\mathbf{C})$	0	0	$O(\mathbf{C})$
TG	$O(\mathbf{C}+\mathbf{E})$	0	0	$O(\mathbf{C}+\mathbf{E})$
all	$O(M \in \mathbf{E})$	0	0	$O(M \in \mathbf{E})$
redundant *on*	$O(\mathbf{RCR})$	0	0	$O(\mathbf{RCR} * S + \mathbf{CR})$
hard *on*	$O(\mathbf{HPCR})$	0	$O(\mathbf{HPCR})$	$O(\mathbf{HPCR} * S + \mathbf{CR})$
soft *on*	$O(\mathbf{SPCR})$	0	$O(\mathbf{SPCR})$	$O(\mathbf{SPCR} * S + \mathbf{CR})$

Table 2: Overhead associated with individual mechanisms at each parameter setting

Deciding to add a mechanism. A practical question to ask is simply whether the addition of a particular mechanism will benefit performance for the system of agents. Here we give an example with respect to the soft coordination mechanism, which will make commitments to facilitation relationships. We ran 234 randomly generated episodes (generated with the environmental parameters shown in Table 1) with four agents both with and without the soft coordination mechanism. Because the variance between these randomly generated episodes is so great, we took advantage of the paired response nature of the data to run a non-parametric Wilcoxon matched-pairs signed-ranks test [3]. This test is easy to compute and makes very few assumptions—primarily that the variables are interval-valued and comparable within each block of paired responses. For each of the 234 blocks we calculated the difference in the total final quality achieved by each group of agents and excluded the blocks where there was no difference, leaving 102 blocks. We then replace the differences with the ranks of their absolute values, and then replace the signs on the ranks. Finally we sum the positive and negative ranks separately. A standardized Z score is then calculated. A small value of Z means that there was not much consistent variation, while a large value is unlikely to occur unless one treatment consistently outperformed the other. In our experiment, the null hypothesis is that the system with the soft coordination mechanism did the same as the one without it, and our alternative is that the system with the soft coordination mechanism did better (in terms of total final quality). The result here was $Z = -6.9$, which is highly significant, and allows us to reject the null hypothesis that the mechanism did not have an effect.

Performance Issues. Another question is how the performance of a fully configured system compares with optimal performance. While we have no optimal parallel scheduler with which to compare ourselves, we do have a single agent optimal scheduler and a centralized, heuristic parallel scheduler that takes the single-agent optimal schedule as its starting point. In 300 paired response experiments based on the same environmental parameters as the facilitation experiments (Table 1), the centralized parallel scheduler outperformed our distributed, GPGP agents 57% of the time (36% no difference, 7% distributed was better) using the total final quality as the only criterion. The GPGP agents produced 85% of the quality that the centralized parallel scheduler did, on average. These results need to be understood in the proper context—the centralized scheduler takes much more processing time that the distributed scheduler and can not be scaled up to larger numbers of methods or task groups. The centralized scheduler also starts with a global view of the entire episode. We also looked at performance without

any of the mechanisms; on the same 300 episodes the GPGP agents produced on average 1.14 times the final quality of the uncoordinated agents. These experiments had very low numbers of **enables** relationships, which tend to hurt the uncoordinated agents, and also remember that we are only looking at one performance measure—total final quality. Coordinated agents execute far fewer methods because of their ability to avoid redundancy, but this does not show up in final quality because there is no real-time pressure in these 300 episodes (we kept the time pressure constantly low). The redundant execution of methods proves a much more hindering element to the uncoordinated agents when acting under severe time pressure [4].

Examining the family's performance space. Next we looked at the multidimensional performance space for the family of coordination algorithms over four different performance measures. Earlier we mentioned that although our simple initial parameterization results in 72 possible coordination algorithms, many of these are not statistically different from one another in randomly generated environments. We applied two standard statistical clustering techniques to develop a much smaller set of significantly different algorithms. The resulting five 'prototypical' combined behaviors are a useful starting point in evaluating the family in a new environment.

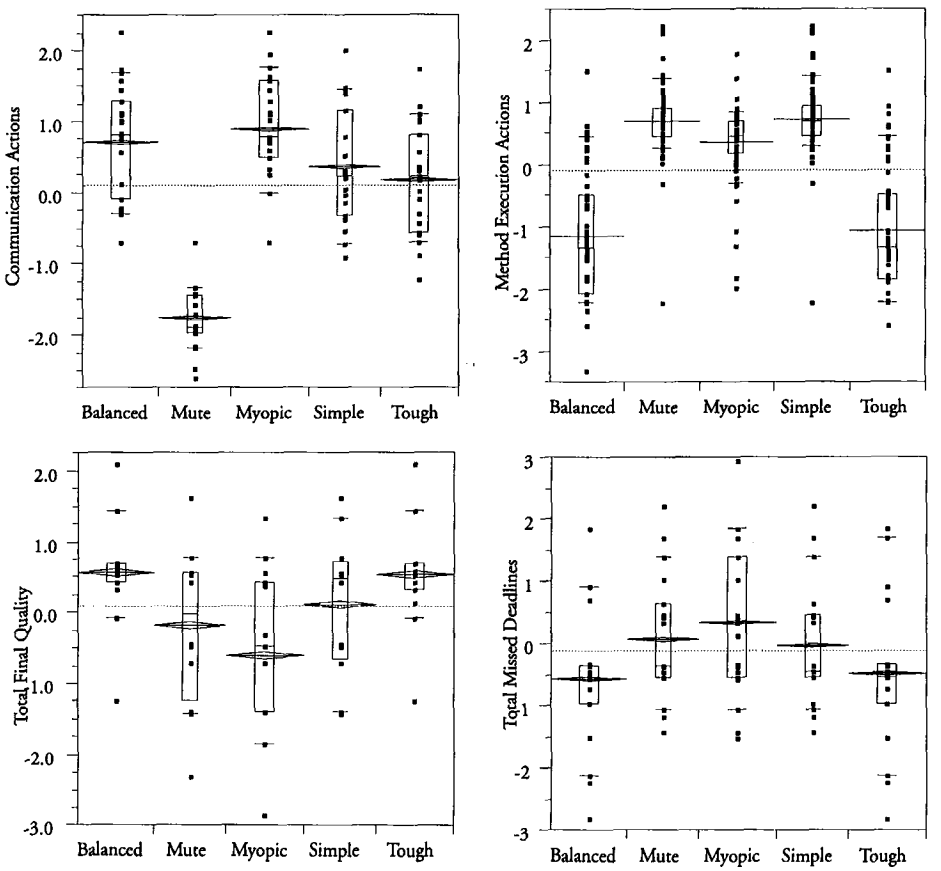

Figure 4: Standardized Performance by the 5 named coordination styles.

The analysis proceeded as follows: for 63 randomly chosen environments, we generated one random episode and ran each of the 72 "agent types" on the episode (4536 cases). We

collected four performance measures: total quality, number of methods executed, number of communication actions, and termination time. We then took this data and standardized each performance measure within an environment. So now each measure is represented as the number of standard deviations from the mean value in that environment. We then took summary statistics for each measure grouped by agent types—this boils the 4536 cases (standardized within each environment) into 72 summary cases (summarized across environments). Each of the 72 summaries correspond to the average standardized performance of one agent-type for the four performance measures. We then used both a hierarchical clustering algorithm (SYSTAT JOIN with complete linkage) and a linear clustering algorithm (SYSTAT KMEANS) to produce the following general prototypical agent classes (we chose one representative algorithm in each class):

Simple: No commitments or non-local view, just broadcasts results.
Myopic: All commitment mechanisms on, but no non-local view.
Balanced: All mechanisms on.
Tough-guy: Balanced agent that makes no soft commitments.
Mute: No communication whatsoever[9]

KMEANS also produces the mean value of each performance measure for each group, so you can in fact see that the non-communicating agents have a high negative mean "number-of-communications" (-1.16–remember these were averaged from standardized scores) but execute more methods on average and produce less final quality. They also terminate slightly quicker than average. Our "Balanced" group, in comparison, communicates a little more than average, executes *many* fewer methods (-1.29—way out on the edge of this statistic), returns better-than-average quality and about average termination time. This is reasonable, as 'avoiding redundant work' and other work-reducing ideas are a key feature of the GPGP algorithm.

Figure 4 shows the values of several typical performance measures for only the five named types. Performance measures were standardized *within each episode*, (i.e. across all 72 types). Shown for each are the means and 10, 25, 50, 75, and 90 percent quantiles. All agents' performances are significantly different by Tukey Kramer HSD except for: Method Execution (Simple vs. Mute), Total final quality (Balanced vs. Tough), Deadlines missed (simple vs. mute) and (balanced vs. tough).

We are also analyzing the effect of environmental characteristics on agent performance. Figure 5 shows an example of the effect of the *a priori* amount of overlap on the number of method execution actions for the five named agent types. Note again that the balanced and tough agents do significantly less work when there is a lot of overlap (as would be expected). The performance of the tough and balanced agents is similar because (from Table 1) only half the experiments had any facilitation, and when it was present was only at 50

4 Conclusions and Future Work

This paper discusses the design of an extendable family of coordination mechanisms, called Generalized Partial Global Planning (GPGP), that form a basic set of coordination mechanisms

[9]This algorithm makes no commitments (mechanisms 3, 4, and 5 off) and communicates (mechanism 2) only 'satisfied commitments'—therefore it sends no communications ever!.

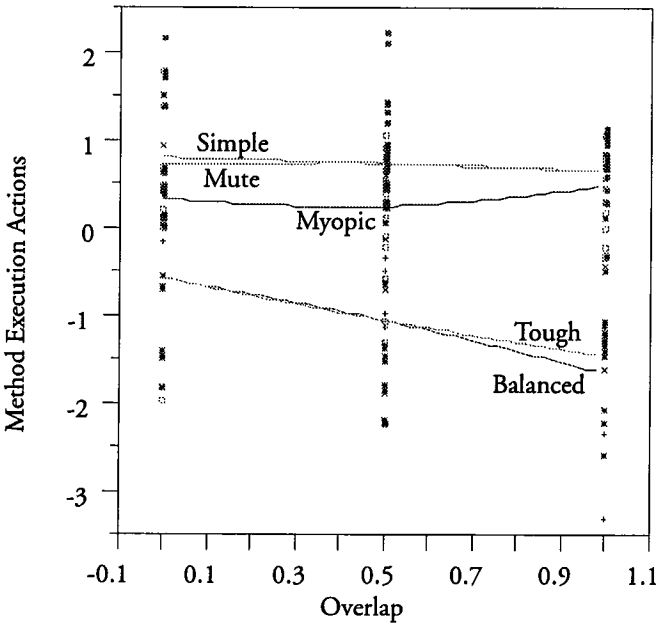

Figure 5: The effect of overlaps in the task environment on the standardized method execution performance by the 5 named coordination styles (smoothed splines fit to the means).

for teams of cooperative computational agents. An important feature of this approach includes an extendable set of modular coordination mechanisms, any subset or all of which can be used in response to a particular task environment. This subset may be parameterized, and the parameterization does not have to be chosen statically, but can instead be chosen on a task-group-by-task-group basis or even in response to a particular problem-solving situation. For example, Mechanism 5 (Handle Soft Predecessor CRs) might be "on" for certain classes of tasks and 'off' for other classes (that usually have few or very weak soft CRs). The general specification of the GPGP mechanisms involves the detection and response to certain abstract *coordination relationships* in the incoming task structure that were not tied to a particular domain. A careful separation of the coordination mechanisms from an agent's local scheduler allows each to better do the job for which it was designed. A complete discussion of the scheduler/decision-maker interface can be found in our companion paper [15]. We believe this separation is not only useful for applying our coordination mechanisms to existing problems with existing, customized local schedulers, but also to problems involving humans (where the coordination mechanism can act as an interface to the person, suggesting possible commitments for the person's consideration and reporting non-local commitments made by others).

Application designers will find this general specification of partial global planning coordination behaviors easier to adapt to their domains than the original distributed vehicle monitoring testbed implementation. What is required to use GPGP or similarly designed algorithms is a local scheduler that can handle local and non-local commitments and can communicate via task structures (the interface is fully described in [15]). GPGP also requires domain-dependent code to detect potential coordination relationships to other agents and verify their presence after non-local information has been received. This paper has outlined the overhead involved with each mechanism, and discussed experimental techniques for demonstrating improved performance in an application. Future work will involve elaborating and validating models

of these coordination mechanisms that link environmental parameters to both overhead and performance. We also discussed how to limit the initial search for a suitable application-specific coordination algorithm by using prototypical algorithm family members.

Work not reported here includes the development of a load-balancing coordination mechanism that works by providing better information to the agents with which to resolve redundant commitments. More mechanisms, and more complex mechanisms are possible; for example, mechanisms that work from the successors of hard and soft relationships instead of the predecessors, or negotiation mechanisms. Mechanisms for behavior such as contracting are also possible. As always, the important question will be "in what environments will the extra overhead of these mechanisms be worthwhile." Future work will also examine expanding the parameterization of the mechanisms and using machine learning techniques to choose the appropriate parameter values (i.e., learning the best family member for an environment).

References

[1] C. Castelfranchi. Commitments: from individual intentions to groups and organizations. In Michael Prietula, editor, *AI and theories of groups & organizations: Conceptual and Empirical Research.* AAAI Workshop, 1993. Working Notes.

[2] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence,* 42(3), 1990.

[3] W. W. Daniel. *Applied Nonparametric Statistics.* Houghton-Mifflin, Boston, 1978.

[4] Keith S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms.* PhD thesis, University of Massachusetts, 1994.

[5] 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 IJCAI,* pages 15–21, Sydney, Australia, August 1991. Extended version in International Journal of Pattern Recognition and Artificial Intelligence 7(2) 1993.

[6] Keith S. Decker and Victor R. Lesser. Generalizing the partial global planning algorithm. *International Journal of Intelligent and Cooperative Information Systems,* 1(2):319–346, June 1992.

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

[8] Keith S. Decker and Victor R. Lesser. An approach to analyzing the need for meta-level communication. In *Proceedings of the Thirteenth IJCAI,* Chambéry, August 1993.

[9] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence,* pages 217–224, Washington, July 1993.

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

[11] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(11):1275–1291, November 1987.

[12] 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.

[13] Mark S. Fox. An organizational view of distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(1):70–80, January 1981.

[14] J. Galbraith. *Organizational Design.* Addison-Wesley, Reading, MA, 1977.

[15] Alan Garvey, Keith Decker, and Victor Lesser. A negotiation-based interface between a real-time scheduler and a decision-maker. CS Technical Report 94–08, University of Massachusetts, 1994.

[16] 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.

[17] 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.

[18] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(3):223–250, 1993.

[19] Paul Lawrence and Jay Lorsch. *Organization and Environment.* Harvard University Press, Cambridge, MA, 1967.

[20] 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.

[21] Yoav Shoham. AGENT0: A simple agent language and its interpreter. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 704–709, Anaheim, July 1991.

[22] Arthur L. Stinchcombe. *Information and Organizations.* University of California Press, Berkeley, CA, 1990.

[23] Frank v. Martial. *Coordinating Plans of Autonomous Agents.* Springer-Verlag, Berlin, 1992. Lecture Notes in Artificial Intelligence no. 610.