Designing a Family of Coordination Algorithms [1]

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

UMass Computer Science Technical Report 94–14
August 9, 1995

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). The set of coordination mechanisms described here assists in scheduling activities for teams of cooperative computational agents. The GPGP approach has several unique features. First, it is not tied to a single domain. Each mechanism is defined as a response to certain features in the current task environment. We show that different combinations of mechanisms are appropriate for different task environments. Secondly, the approach works in conjunction with an agent's existing local planner/scheduler. Finally, the initial set of five mechanisms presented here generalizes and extends the Partial Global Planning (PGP) algorithm. In comparison to PGP, GPGP schedules tasks with deadlines, it allows agent heterogeneity, it exchanges less global information, and it communicates at multiple levels of abstraction. We analyze the performance of several GPGP algorithm family members and one centralized upper bound reference algorithm, using data from simulations of multiple agent teams working in abstract task environments. We show how to decide if adding a new mechanism is useful, and suggest a way to prune the search for an appropriate combination of mechanisms in an environment.

[1] A shorter version of this paper appeared in the *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, June 1995. This work was supported by DARPA contract N00014-92-J-1698, Office of Naval Research contract N00014-92-J-1450, and NSF contract IRI-9321324. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

1 Introduction

This paper presents a formal description of the implementation of a domain independent scheduling coordination approach which we call Generalized Partial Global Planning (GPGP). The GPGP approach consists of an extendable set of modular coordination mechanisms, any subset or all of which can be used in response to a particular task environment. Each mechanism is defined using our formal framework for expressing coordination problems (TÆMS [8]). GPGP both generalizes and extends the Partial Global Planning (PGP) algorithm [10].

Our approach has several unique features:

- Each mechanism is defined as a response to certain features in the current subjective task environment. Each mechanism can be removed entirely, or can be parameterized so that it is only active for some portion of an episode. New mechanisms can be defined; an initial set of five mechanisms is examined that together approximate the original PGP behavior. Eventually we intend to develop a library of reusable coordination mechanisms. The individual coordination mechanisms rest on a shared substrate that arbitrates between the mechanisms and the agent's local scheduler in a decision-theoretic manner.

- GPGP works in conjunction with an existing agent architecture and local scheduler. The experimental results reported here were achieved using a 'design-to-time' real-time local scheduler developed by Garvey [13].

- GPGP, unlike PGP, is not tied to a single domain. GPGP allows more agent heterogeneity than PGP with respect to agent capabilities. GPGP mechanisms in general exchange less information than the PGP algorithm, and the information that GPGP mechanisms exchange can be at different levels of abstraction. PGP agents communicated complete schedules at a single, fixed level of abstraction. GPGP mechanisms communicate scheduling commitments to particular tasks, at any convenient level of abstraction.

The GPGP approach views coordination as *modulating* local control, not replacing it. This process occurs via a set of domain-independent coordination mechanisms that post constraints to the local scheduler about the importance of certain tasks and appropriate times for their initiation and completion. An example of a GPGP coordination mechanism is the one that handles simple method redundancy. If more than one agent has an otherwise equivalent method for accomplishing a task, then an agent that schedules such a method will commit to executing it, and will notify the other agents of its commitment. If more than one agent should happen to commit to a redundant method, the mechanism takes care of retracting all but one of the redundant commitments.

By concentrating on the creation of local scheduling constraints, we avoid the sequentiality of scheduling in the original PGP algorithm 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 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). Nothing in TÆMS the underlying

task structure representation, requires agents to be cooperative, antagonistic, or simply self-motivated.

Besides the obvious connections to the earlier PGP work, GPGP builds on work by von Martial [20] in detecting and reacting to relationships (such as von Martial's "favor" relationship). GPGP also uses a notion of social commitments similar to those discussed by [2, 19, 1, 15]. Durfee's newer work [9] is based on a hierarchical behavior space representation that like GPGP allows agents to communicate at multiple levels of detail. The mechanisms presented in this paper deal with coordination while agents are scheduling (locating in time) their activities rather than while they are planning to meet goals. This allows them to be used in distributed scheduling systems, agenda-based systems (like blackboard systems), or systems where agents instantiate previous plans (like case-based planning systems). The focus on mechanisms for coordinating schedules is thus slightly different from work that focuses on multi-agent planning [14, 11]. Shoham and Tennenholtz's 'social laws' approach [18] can be viewed as one which tries to change the (perceived) structure of the tasks by, for example, restricting the agents' possible activities. Intelligent agents might use all of these approaches at one time or another.

The next section will briefly re-introduce our framework for representing coordination problems, and summarize the assumptions we make about an agent's internal architecture. We then describe the GPGP substrate and five coordination mechanisms.[1] Previous work has shown how the GPGP approach can duplicate and extend the behaviors of the PGP algorithm [5]; Section 4 summarizes several new results that are reported in [4] concerning this approach's performance, adaptability, and extendibility. We conclude with a look at our future directions.

1.1 Representing The Task Environment

Coordination is the process of managing interdependencies between activities [17]. If we view an agent as an entity that has some beliefs about the world and can perform actions, then the coordination problem arises when any or all of the following situations occur: the agent has a *choice* of actions it can take, and that choice affects the agent's performance; the *order* in which actions are carried out affects performance; the *time* at which actions are carried out affects performance. The coordination problem of choosing and temporally ordering actions is made more complex because the agent may only have an incomplete view of the entire task structure of which its actions are a part, the task structure may be changing dynamically, and the agent may be uncertain about the outcomes of its actions. If there are multiple agents in an environment, then when the potential actions of one agent are related to those of another agent, we call the relationship a *coordination relationship*. Each GPGP coordination mechanism is a response to some coordination relationship.

The TÆMS framework (Task Analysis, Environment Modeling, and Simulation) [8] represents coordination problems in a formal, domain-independent way. We have used it to represent coordination problems in distributed sensor networks, hospital patient scheduling, airport resource management, distributed information retrieval, pilot's associate, local area network diagnosis, etc. [4]. In this paper we will describe an agent's current subjective beliefs

[1] These five mechanisms are oriented towards producing PGP-like 'cooperative team' behavior. Mechanisms for self-interested agents are also possible.

about the structure of the problem it is trying to solve by using the TÆMS framework [8, 4]. For this purpose, there are two unique features of TÆMS. The first is the explicit, quantitative representation of task interrelationships as functions that describe the effect of activity choices and temporal orderings on performance. The second is the representation of task structures at multiple levels of abstraction. The highest level of abstraction is called a *task group*, and contains all tasks that have explicit computational interrelationships. A *task* is simply a set of lower-level subtasks and/or executable methods. The components of a task have an explicitly defined effect on the quality of the encompassing task. The lowest level of abstraction is called an executable *method*. An executable *method* represents a schedulable entity, such as a blackboard knowledge source instance, a chunk of code and its input data, or a totally-ordered plan that has been recalled and instantiated for a task. A method could also be an instance of a human activity at some useful level of detail, for example, "take an X-ray of patient 1's left foot".

A coordination 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 $\mathbf{E} = \langle \mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n \rangle$. Figure 1 shows an objective[2] task group and agent A's subjective view of that same task group. A common performance goal of the agent or agents is to maximize the sum of the quality achieved for each task group before its deadline. 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). Tasks at the leaves of the tree represent executable *methods*, which are the actual instantiated computations or actions the agent will execute that produce some amount of quality (in the figure, these are shown as boxes). The circles higher up in the tree represent various subtasks involved in the task group, and indicate precisely how quality will accrue depending on what methods are executed and when. 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.

2 Summary of the GPGP algorithm family approach

This section will provide a quick overview of the GPGP approach. Figure 2 shows a simple two-agent example that we will use. Each agent has as part of its architecture a belief database, local scheduler, and coordination module. The local scheduler uses the information in the belief database to schedule method execution actions for the agent in an attempt to maximize its performance. We add to this a coordination module that is in charge of communication actions, information gathering actions, and in making and breaking *commitments* to complete tasks in the task structure. The coordination module consists of several coordination mechanisms, each of which notices certain features in the task structures in the belief database, and responds by taking certain communication or information gathering actions, or by proposing new commitments. The coordination mechanisms rest in a shared coordination module substrate that keeps track of local commitments and commitments received from other agents, and that chooses from among multiple schedules if the local scheduler returns multiple schedules.

[2] The word 'objective' refers to the fact that this is the true, real structure.

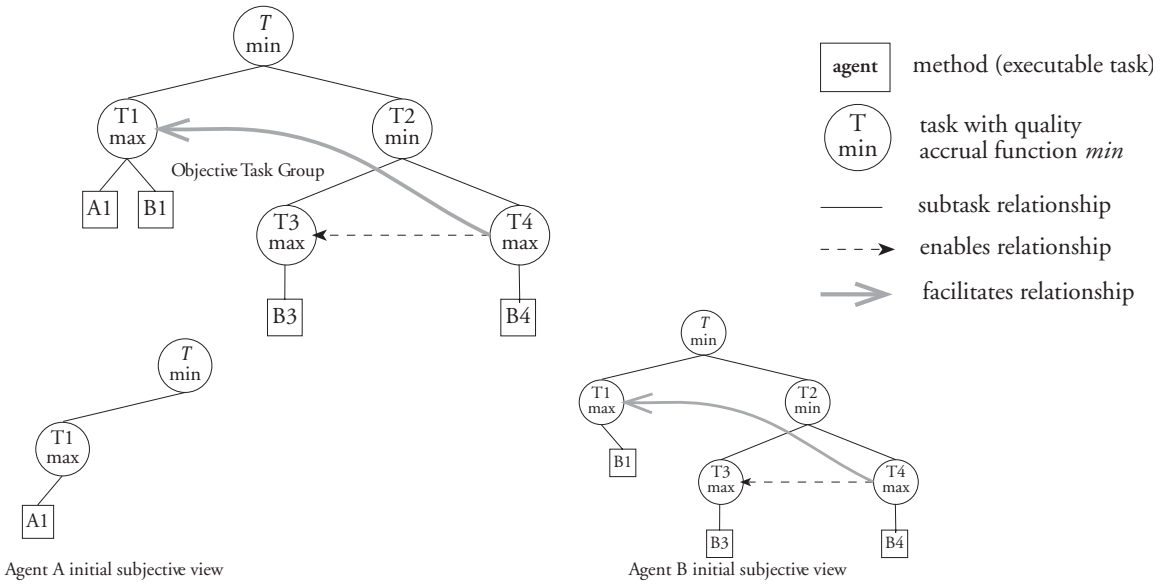

Figure 1: Agent A and B's subjective views (bottom) of a typical objective task group (top)

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. We will show examples of the effect of the environment on the performance of a GPGP family member, and show an environment where family member A is better than B, and a different environment where B is better than A. We will return to the demonstration of meta-level information being more useful when there is a large amount of variance between episodes in an environment.

Here is a short example intended only to give the reader a feel for the overall approach. In Figure 2, both agents have executed an initial information gathering action, and have their initial views of the task structure (everything in the agents' belief database *except* for the shaded tasks (Tasks 2, 5, D and E), and the relationships touching the shaded tasks). One of the coordination mechanisms (Mech. 1, update non-local views) performs an information gathering action to determine which tasks may be related to tasks at other agents ("detect coordination relationships"). These tasks are then exchanged between the agents, resulting in the belief databases shown in the figure (including the shaded tasks). Other mechanisms react to the task structure. One mechanism (Mech. 5, handle soft predecessors) notices that Task 2 at Agent Y facilitates Task 5 at Agent X. In order that Agent X might schedule to take advantage of this, Agent Y's mechanism makes a local intermediate deadline commitment to complete its Task 2 by time 7 with minimum quality 45 (you and I may infer that Y intends to execute Method B, but that local information is not a part of the commitment). A commitment is made in two stages: first it is made locally to see if it is possible as far as the agent's local scheduler is concerned, and then it is made non-locally and communicated to the other agents that are involved. Note that the deadline on the non-local version of this commitment is later (time 8) to take into account the communication delay (here, 1 time unit). Similarly, Agent

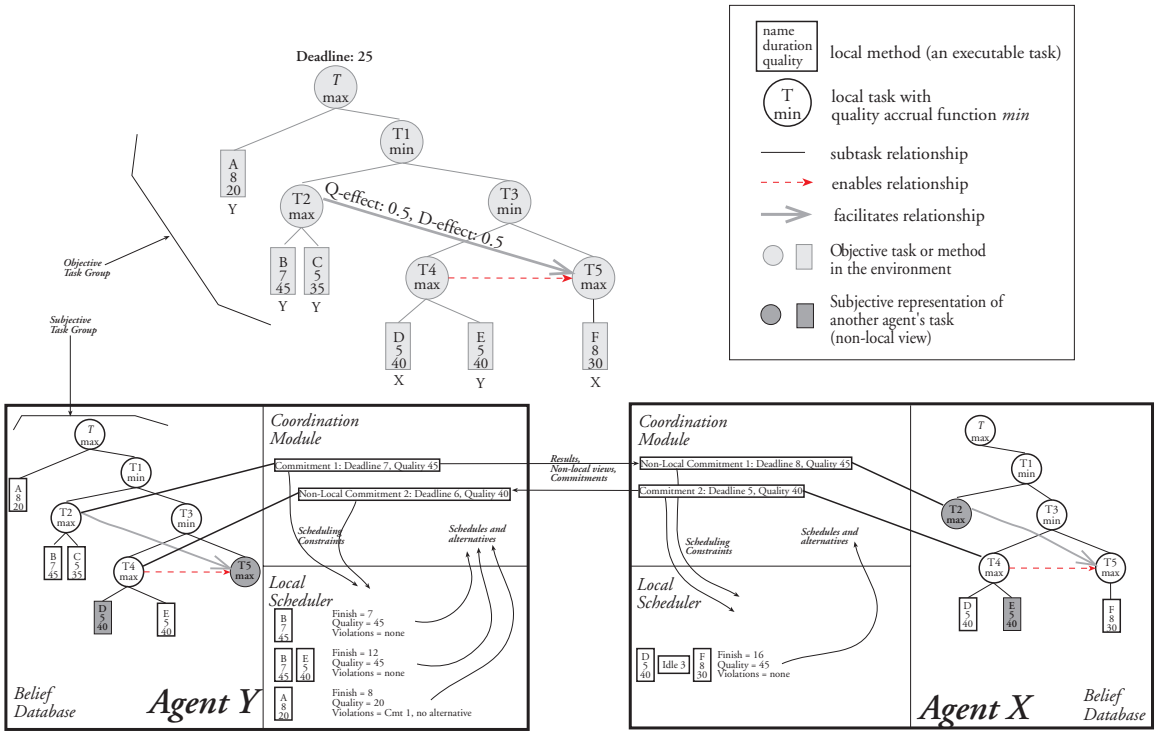

Figure 2: An Overview of Generalized Partial Global Planning

X has a mechanism (Mech. 3, handle simple redundancy) that notices that either agents X or Y could do Task 4. Agent X does eventually commit to this task (the process is a bit more complicated as will be explained later) and communicates this commitment to Agent Y.

In both cases the agents' local schedulers use the information about the task structure they have in their belief database, and the local and non-local commitments, to construct schedules. The local scheduler may return multiple schedules for several reasons we explain later. Each schedule is evaluated along the dimensions of the performance criteria (such as total final quality and termination time) and for what (if any) local commitments are violated. If a commitment is violated, the local scheduler may suggest an alternative (for instance, relaxing a quality or intermediate deadline constraint). The coordination module chooses a schedule from this set, and handles the retraction of any violated commitments.

2.1 The Agent Architecture

We make few assumptions about the architecture of the agents. The agents have a database that holds their current beliefs about the structure of the tasks in the current episode; we represent this information using TÆMS. The agents can do three types of actions: they can execute methods from the task structure, send direct messages to one another, and do "information gathering". Information gathering actions model how new task structures or communications get into the agent's belief database. This could be a combination of external actions (checking the agent's incoming message box) and internal planning. 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.

Formally, we write $B_A^t(x)$ to mean agent A subjectively believes x at time t (from Shoham[19]). We will shorten this to $B(x)$ when the particular agent or time is not important. An agent's subjective beliefs about the current episode include the agent's beliefs about task groups, subtasks, executable methods, and interrelationships (e.g., $B(\mathcal{T}_i \in \mathbf{E}), B(T_a, M_b \in \mathcal{T}_i), B(\mathsf{enables}(T_a, M_b))$).

The GPGP family of coordination mechanisms also makes a stronger assumption about the agent architecture. 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. The local scheduler attempts to maximize a (possibly changing) utility function. The current set of GPGP coordination mechanisms are for cooperative teams of agents—they assume that agents do not intentionally lie and that agents believe what they are told. 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. In this paper we will refer to that domain-dependent code as the information gathering action called *detect-coordination-relationships*; we will describe this action more in Section 3.2.

2.2 The Local Scheduler

Each GPGP agent contains a local scheduler that takes three types of input information and produces a set of schedules and alternatives. The first input is the current, subjectively believed task structure. Using information about the potential duration, potential quality, and interrelationships, the local scheduler chooses and orders executable methods in an attempt to maximize a pre-defined utility function. In this paper the utility function is the sum of the task group qualities $\sum_{\mathcal{T} \in \mathbf{E}} Q(\mathcal{T}, \mathrm{D}(\mathcal{T}))$, where $Q(T,t)$ denotes the quality of T at time t as defined in [8]. Quality does not accrue after a task group's deadline.

The second input is a set of *commitments* $\mathbf{C}$. These commitments are produced by the GPGP coordination mechanisms, and act as 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 one of agent A's coordination mechanisms may *commit* agent A to do method 1. Commitments are *social*—directed to particular agents in the sense of the work of Shoham and Castelfranchi [1, 19]). A local commitment C by agent A becomes a non-local commitment when received by another agent B. 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 $[Q(T,t) \geq q] \wedge [t \leq t_{dl}]$. When a commitment is sent to another agent, it also implies that the task result will be communicated to the other agent (by the deadline, if it is a deadline commitment).

The third input to the local scheduler is the set of non-local commitments $\mathbf{NLC}$ made by other agents. 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 $\mathsf{enables}$ method M_2 at agent B (and

agent B knows this), 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, $\langle M_2, t\rangle \in S \;\Rightarrow\; t \geq t_1$ (in other words, agent B only schedules the enabled method after the deadline that agent A has committed to.

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. 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 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 function $U_{\mathrm{est}}(\mathbf{E}, S, \mathbf{NLC})$ returns the estimated utility at the end of the episode if the agent follows schedule S and all non-local commitments in $\mathbf{NLC}$ 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 [12]. 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 $\hat{S}_{\bar{V}}$. 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: $\mathrm{U}(\mathbf{E}, S_U, \mathbf{NLC}) \leq \mathrm{U}(\mathbf{E}, \hat{S}_U, \mathbf{NLC})$.

3 Five GPGP Coordination Mechanisms

The role of the coordination mechanisms is to provide information to the local scheduler that allows the local scheduler to construct better schedules. This information can be in the form of modifications to portions of the subjective task structure of the episode or in the form of local and non-local commitments to tasks in the task structure. 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 shown in [5]. 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 will return at least one violated constraint in all its schedules. Since the local scheduler typically satisfices instead of optimizes, it may do this even if constraints are not inconsistent (i.e. it does not search exhaustively). The next section describes how a schedule is chosen by the coordination module substrate.

3.1 The GPGP Coordination Module Substrate

All the specific coordination mechanisms rest on a common substrate that handles information gathering actions, invoking the local scheduler, choosing a schedule to execute (including dealing with violated or inconsistent commitments), and deciding when to terminate processing on a task group. Information gathering actions include noticing new task group arrivals and receiving communications from other agents. Information gathering is done at the start of problem solving, when communications are expected from other agents, and when the agent is otherwise idle. 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 $\mathbf{NLC}$. This is the minimal general information gathering policy. Termination of processing on a task group occurs for an agent when the agent is idle, has no expected communications, and no outstanding commitments for the task group.

Choosing a schedule is more complicated. The agent's local scheduler may return multiple schedules because it cannot find a single schedule that both maximizes utility and meets all commitments. From the set of schedules $\mathbf{S}$ 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. The coordination substrate may receive 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.

If both schedules have the same utility, 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 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 [4] and plan to analyze it in future work.

3.2 Mechanism 1: Updating Non-Local Viewpoints

Remember that each agent has only a partial, subjective view of the current episode. 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). The process of detecting coordination relationships between private and shared parts of a task structure is in general very domain specific, so 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 scheduled whenever a new task group arrives.

The set $\mathbf{P}$ of privately believed tasks or methods at an agent A (tasks believed at arrival time by A only) is then $\{x \mid \mathit{task}(x) \wedge \forall a \in \mathbf{A} \setminus A,\ \neg B_A(B_a^{\mathbf{Ar}(x)}(x))\}$, where $\mathbf{A}$ is the set of all agents and $\mathbf{Ar}(x)$ 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. $B_A(B_B(T_2))$) [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 $\{a \mid B_A(B_a(T_2))\}$.

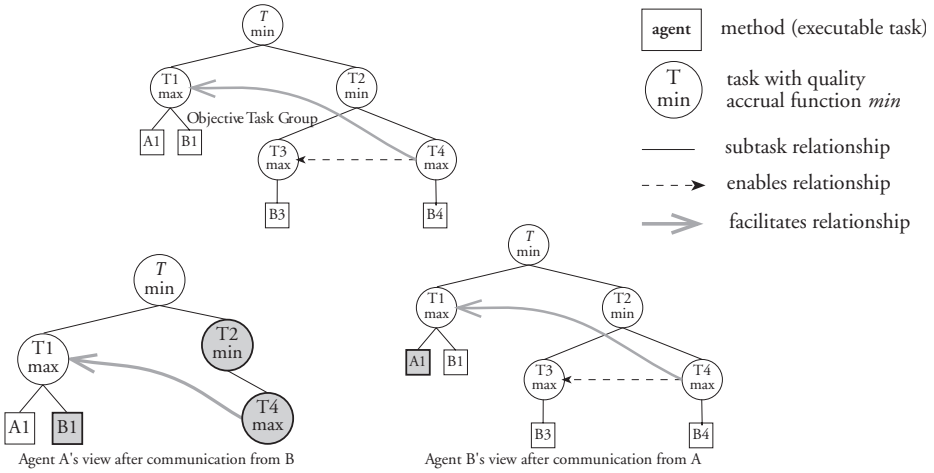

Figure 3: Agents A and B's local views after receiving non-local viewpoint communications via mechanism 1 (shaded objects). Figure 1 shows the agents' initial states.

For example, Figure 3 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 $\mathbf{P}$, $\mathsf{facilitates}(T_4, T_1, \phi_d, \phi_q) \in \mathbf{PCR}$ (the facilitation relates a private task to a mutually believed task), and $\mathsf{enables}(T_4, T_3)$ 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 concern to the local scheduler. The purpose of this coordination mechanism is the exchange of information that expands the set of coordination relationships $\mathbf{CR}$. Without this mechanism in place, $\mathbf{CR}$ will consist of only non-private relationships, and none that are in $\mathbf{PCR}$. Since the primary focus of the coordination mechanisms is the creation of social commitments in response to coordination relationships (elements of $\mathbf{CR}$), 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 sucessfully omitted.

3.3 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[3]). 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 $\{A \in \mathbf{A} \mid B(B_A(C(T))\}$.

3.4 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 [16]). 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 [6]—it does not try to load balance, and uses one communication action (because in the general case the agents do not know beforehand,

[3] Such a policy is all that is needed in many simple environments.

without communication, that certain methods are redundant[4]). 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 $\mathrm{Others}(\mathbf{M})$ (non-local agents who also have a method in $\mathbf{M}$):

$$[\langle M, t_M \rangle \in S] \wedge [\mathsf{subtask}(T, \mathbf{M}, \min) \in \mathbf{RCR}] \wedge [M \in \mathbf{M}] \Rightarrow$$
$$[C(\mathsf{Do}(M, \mathrm{Q}_{\mathrm{est}}(M, \mathrm{D}(M), S))) \in \mathbf{C}] \wedge [\mathsf{comm}(M, \mathrm{Others}(\mathbf{M}), t) \in \mathcal{I}]$$

See for example the top of figure 4—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 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 4 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.

3.5 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 $\mathbf{CR}$. The hard coordination 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 $\mathbf{HPCR}$, and commits to its execution by a certain deadline both locally and socially:

$$[\mathrm{Q}_{\mathrm{est}}(T, \mathrm{D}(T), S) > 0] \wedge [\mathsf{enables}(T, \mathbf{M}) \in \mathbf{HPCR}] \Rightarrow$$
$$[C(\mathsf{DL}(T, \mathrm{Q}_{\mathrm{est}}(T, \mathrm{D}(T), S), t_{\mathrm{early}})) \in \mathbf{C}] \wedge [\mathsf{comm}(C, \mathrm{Others}(\mathbf{M}), t) \in \mathcal{I}]$$

The next question is, by what time (t_{early} above) do we commit to providing the answer? One solution, usable with any local scheduler that fits our general description in Section 2.2,

[4]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.

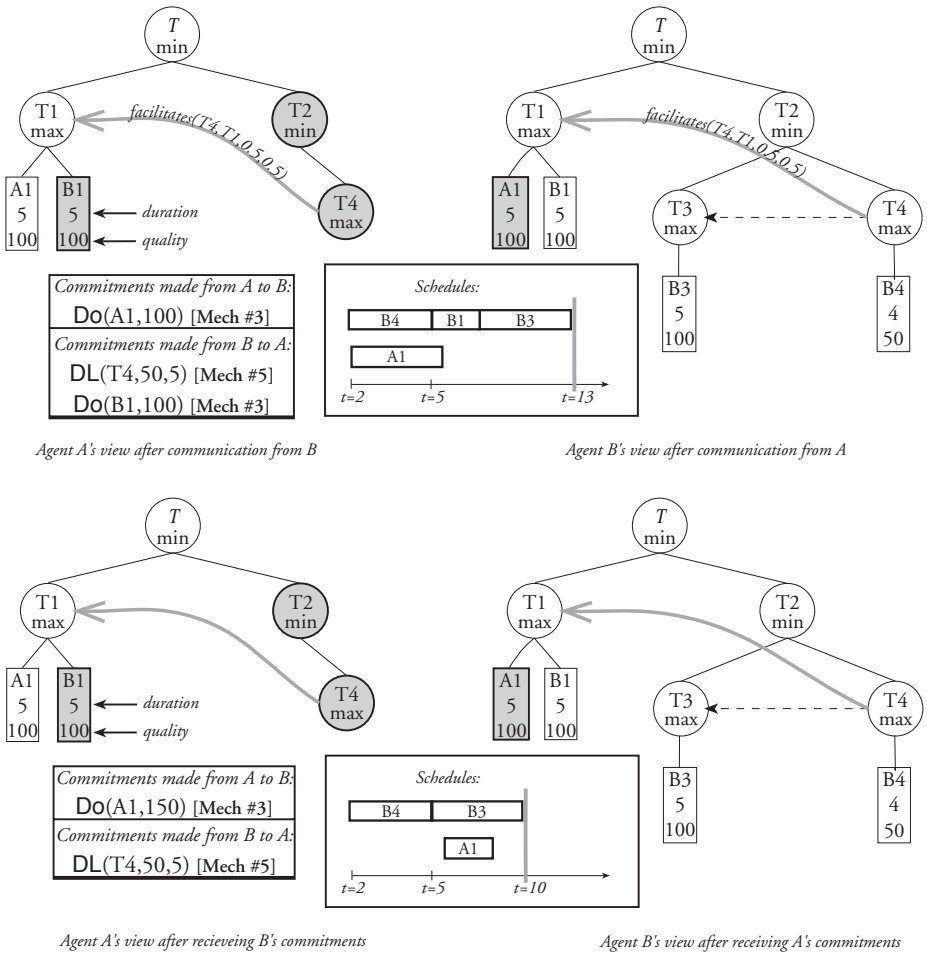

Figure 4: A continuation of Figures 1 and 2. At top: agents A and B propose certain commitments to one another via mechanisms 3 and 5. At bottom: after receiving the initial commitments, mechanism 3 removes agent B's redundant commitment.

is to use the $\min t$ such that $\mathrm{Q}_{\mathbf{est}}(T, \mathrm{D}(T), S) > 0$. 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 $C(\mathsf{DL}(T, \mathrm{Q}_{\mathbf{est}}(T, \mathrm{D}(T), S), \text{"early"}))$ to the local commitment set $\mathbf{C}$, 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 pro-active mechanism, providing infor-

mation 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.

3.6 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$(M_1, M_2, \phi_d, \phi_q)$ 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 [8] for the details). A more situation-specific version of this coordination mechanism might ignore relationships with very low 'power'. The relationship hinders$(M_1, M_2, \phi_d, \phi_q)$ 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 pro-active 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 4 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.

4 Experimental Results

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; see Section 5). We will also discuss statistical techniques (e.g. paired-response simulations) to deal with the large between-episode variances that occur when using randomly-generated problems.

4.1 GPGP Simulation: Issues

Our model of an abstract task environment, used in these experiments, has ten parameters; Table 1 lists them and the values used in the experiments described in the next two sections.[5]

[5] Our earlier work focussed on the analysis of distributed sensor network task environments [6, 7].

Figure 2 shows a small example task group.

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 $\mathbf{RCR}$ (redundant coordination relationships), $\mathbf{HPCR}$ (hard predecessor coordination relationships), $\mathbf{SPCR}$ (soft predecessor coordination relationships), 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.

Mechanism *setting*	Communications	Information Gathering	Scheduler	*Other Overhead*
substrate	0	$\mathbf{E}$+*idle*	L	$O(L\mathbf{C})$
nlv *none*	0	0	0	0
some	$O(d\mathbf{P})$	$\mathbf{E}$*detect-CRs*	0	$O(T \in \mathbf{E})$
all	$O(\mathbf{P})$	$\mathbf{E}$*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

4.2 General Performance Issues

We examined the general performance of the most complex (all mechanisms in place) and least complex (all mechanisms off) members of the GPGP family in comparison to each other, and in comparison to a centralized scheduler reference implementation (as an upper bound). We looked at performance measures such as the total final quality achieved by the system, the amount of work done, the number of deadlines missed, and the termination time. The centralized schedule reference system is not an appropriate solution to the general coordination problem, even for cooperative groups of agents, for several reasons:

- The centralized scheduling agent becomes a possible single point of failure that can cause the entire system to fail (unlike the decentralized GPGP system).
- The centralized scheduling agent requires a complete, global view of the episode—a view that we mentioned earlier is not always easy to achieve. We do not account for any costs in building such a global view in the reference implementation (viewing it as an upper bound on performance). We do not allow dynamic changes in the episodic task structure (which might require rescheduling).
- The centralized reference scheduler uses an *optimal* single-agent schedule as a starting point. The problem of scheduling actions in even fairly simple task structures is in NP, and the optimal scheduler's performance grows exponentially worse with the number of methods to be scheduled. Since the centralized reference scheduler has a global view and schedules all actions at all agents, the size of the centralized problem always grows faster than the size of the scheduling problems at GPGP agents with only partial views and heuristic schedulers.

We conducted 300 paired response experiments, using the three algorithms. "Balanced" refers to all mechanisms being on, with partial non-local views and communication of committed results and completed task groups. "Simple" refers to all mechanisms being off, with no non-local view and broadcast communication of all results. "Parallel" refers to the centralized reference scheduler that uses a heuristic parallelization of an optimal single agent schedule using a complete global view. The experiments were based on the same environmental parameters as the facilitation experiments (Table 1). There are several important things to note about this class of environments:

- The size of the episodes was kept artificially small so that the centralized reference scheduler could find an optimal schedule in a reasonable amount of run time.
- The experiments had very low (10%)numbers of enables relationships and a low (20%) number of MIN quality accrual functions because they penalize the simple algorithm—we demonstrate this in Section 4.4.
- Deadline pressure was also kept low (it also makes the simple algorithm perform badly).

In our experiments, 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 than the distributed scheduler and cannot be scaled up to larger numbers of methods or task groups. The centralized scheduler also starts with a global view of the entire episode. Table 3 shows the results for all four measured criteria by summarizing within-block (paired-response) comparisons. For total final quality and number of deadlines missed, "better" simply refers to an episode where the algorithm in question had a greater total final quality or missed fewer deadlines, respectively. With respect to method execution time (a measure of system load) and termination time, "better" refers to the fact that one algorithm produced both a higher quality and missed fewer deadlines than the other algorithm, or if the two algorithms were the same, then the better algorithm had a lower total method execution time (lower load) or terminated sooner.[6]

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. Coordinated agents ("balanced") execute far fewer methods because of their ability to avoid redundancy. The redundant execution of methods proves a much more hindering element to the uncoordinated agents when acting under severe time pressure [4]. Table 4 summarizes the results.

	Parallel better	**Balanced Better**	**Same**	*Significant?*
Total Final Quality	57%	7%	36%	yes
Method Execution Time	80%	7%	13%	yes
Deadlines Missed	1%	1%	98%	no
Termination Time	67%	15%	18%	yes

Table 3: Performance comparison: Centralized Parallel Scheduler vs. Balanced GPGP Coordination and Decentralized DTT Scheduler

	Simple better	**Balanced Better**	**Same**	*Significant?*
Total Final Quality	8%	21%	71%	yes
Method Execution Time	12%	72%	16%	yes
Deadlines Missed	0%	4%	96%	yes
Termination Time	9%	58%	33%	yes

Table 4: Performance comparison: Simple GPGP Coordination vs. Balanced GPGP Coordination

[6] Termination within two time units was considered "the same" because the "balanced" algorithm has a fixed 2-unit startup cost. The average task duration is 10 time units.

4.3 Taking Advantage of a Coordination Relationship: When to Add a New 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 (Mechanism 5), 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.

4.4 Different Family Members for Different Environments

In this section we show a particular example of how different family members do better and worse in different environments. We will concentrate on two distinct family members—the 'modular agent' archetype (all CR modules on, non-local views, communicate commitments and completed task groups), and the 'simple agent' (no CR modules on, no non-local views, broadcast all completed methods). The environmental parameter we will vary (derived from the screening data collected in Section 5) is *QAF-min*, the percentage of tasks that have *min* as their quality accumulation function ('AND' semantics). Our hypothesis was that the modular agents would do better than the simple agents as QAF-min increased (as more tasks needed to be done). We ran 250 paired-response experiments at 5 levels of QAF-min (0, 0.25, 0.5, 0.75, 1.0) with enables-probability varying also at the same 5 levels, no time pressure, overlaps of 0.5, 5 task groups, and 4 agents per run. The performance (in terms of total final quality) of the two coordination styles was significantly different by the Wilcoxon matched-pairs signed-ranks test (199 different pairs, $Z = -3.27$, $p \leq 0.0005$). More interestingly, we can see the difference in performance widening with the value of QAF-min. Figure 5 shows the probability of one coordination style or the other doing better (calculated simply from the frequencies) plotted verses the value of QAF-min. This allows you to see graphically the difference in the styles as QAF-min changes.

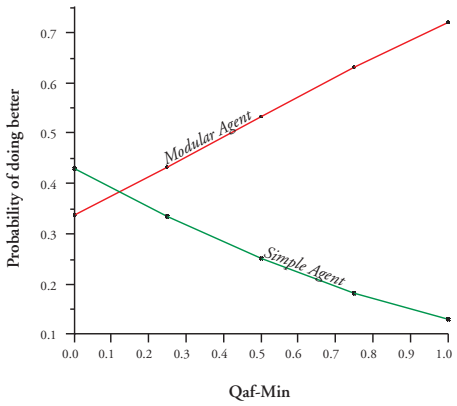

Figure 5: Plot of the probability of the modular or simple coordination styles doing better than the other (total final quality) verses the probability of task quality accumulation being MIN (AND-semantics)

4.5 Meta-level Communication: Return to Load Balancing through Dynamic Reorganization

Another question we have examined is the effect of task structure variance on the performance of load balancing algorithms. This work is a logical follow-on to the analysis of static, dynamic, and negotiated reorganization detailed in [6]. A *static* organization divides the load up *a priori*—in the case below, by randomly assigning redundant tasks to agents. A *one-shot dynamic* reorganization, like that analyzed in [7], assigns redundant tasks on the basis of the *expected* load on other agents. A *meta-level communication* (MLC) reorganization assigns redundant tasks on the basis of actual information about the particular problem-solving episode at hand. Because it requires extra communication, the MLC reorganization is more expensive, but the extra information pays off as the *variance* in static agent loads grows.

A MLC coordination mechanism (mechanism 6) can be implemented in GPGP. Many such implementations are possible; the one that we chose works by altering the way redundant commitments are handled. When a commitment is sent to another agent, it is modified to include the current *load* of the agent making the commitment (to be precise, the amount of work for the agent in the current schedule). Whenever a decision about redundant commitments need to be made at another agent (in mechanisms 3, 4, and 5—simple redundancy, hard, and soft successor relationship handling) the load of the agents with the redundant commitments are taken into account at the point where ties would have been broken randomly. The agent with the lowest load keeps the commitment instead. If the loads are equal, the tie is broken randomly as before.

The effect of this mechanism on the general GPGP environments when agents use the default Design-To-Time scheduler is minimal. The heuristics used by the DTT scheduler are focused at providing the highest possible total final quality for the agent without violating deadlines—this is not the same as terminating quickly, and the scheduler has no heuristics to prefer earlier termination times (nor, frankly, should it have them). In a randomly-generated task environment, where the methods are assigned to agents randomly (and therefore, somewhat evenly) there is rarely any significant change in termination time.

However, if you recall one of our results from [6, 7], you will remember that MLC coordination is most useful in environments with high *variance* in the task structures presented to agents. We can look at our experiments in this light, by calculating an endogenous input variable for each run that represents the amount of variance in redundant tasks (the ones that would *potentially* be eligible for a load-balancing mechanism decision). Figure 6 shows how the probability of terminating more quickly with the MLC load balancing algorithm grows as the standard deviation in the total durations of redundant tasks at each agent grows.

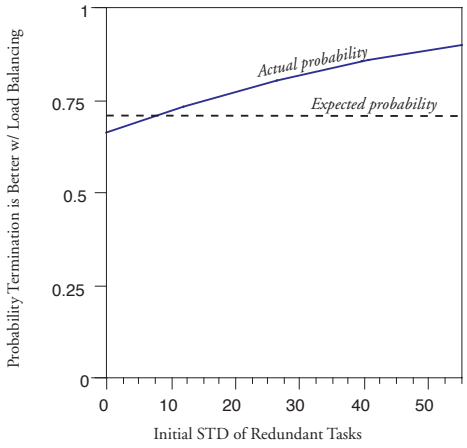

Figure 6: Probability that MLC load balancing will terminate more quickly than static load balancing, fitted using a loglinear model from actual TÆMS simulation data.

5 Exploring the Family Performance Space

Finally, we looked at the multidimensional performance space for the family of coordination algorithms over four different performance measures. At the most abstract level, each of the five mechanisms are parameterized independently (the first two have three possible settings and the last three can be 'in' or 'out') for a total of 72 possible coordination algorithms. 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 when searching for an appropriate algorithm family member in a new environment.

The analysis proceeded as follows: we generated one random episode in each of 63 randomly chosen environments, 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

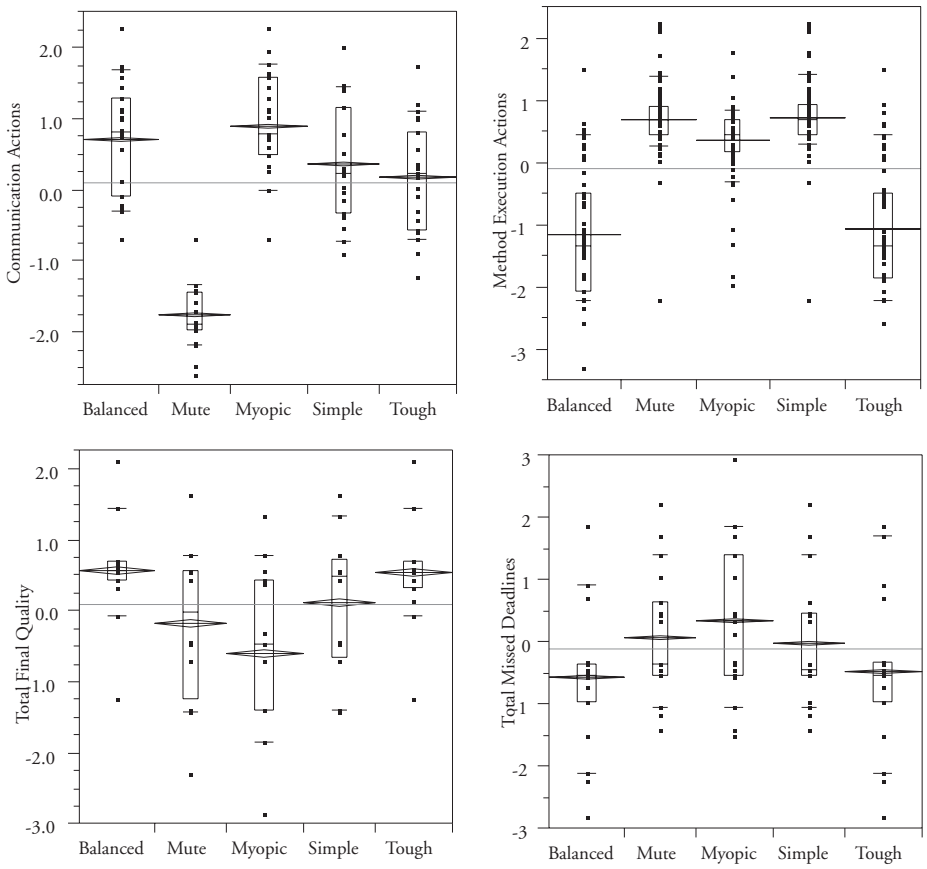

Figure 7: Standardized Performance by the 5 named coordination styles.

(SYSTAT JOIN with 'complete linkage'[7]) 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: Agent that makes no soft commitments.

Mute: No communication whatsoever[8]

Figure 7 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 algorithms' performances are significantly different by Tukey Kramer HSD except for: Method Execution

[7]Distances are calculated between the farthest points in each cluster. Other distance measures (Euclidean, centroid, or Pearson correlation) gave similar results.

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

(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 8 shows an example of the effect of the amount of overlap (method redundancy) 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) 1) the algorithms only differ in the way that they handle facilitation, and 2) only half the experiments had any facilitation, and when it was present was only at 50% power.

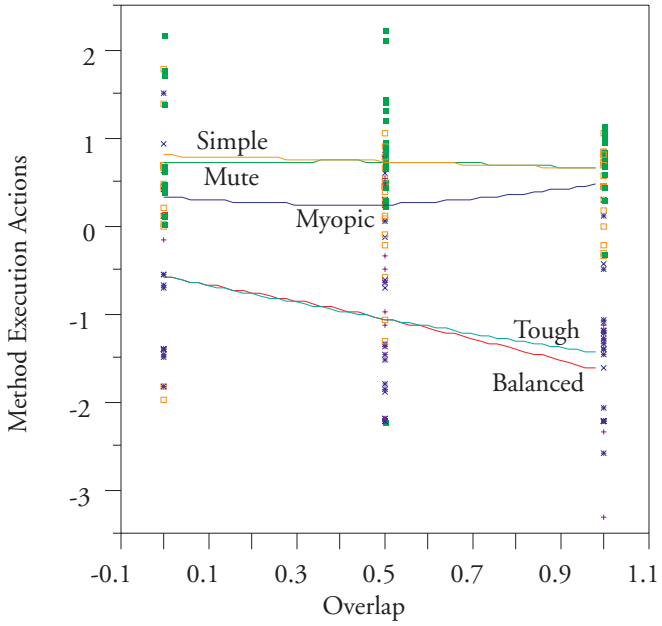

Figure 8: The effect of overlaps in the task environment on the standardized method execution performance by the 5 named coordination styles (smoothed splines fit to the means).

A linear clustering algorithm, SYSTAT KMEANS, produces a similar result as hierarchical clustering, and also produces the mean value of each performance measure for each group. For example, 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. The "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 original PGP algorithm replicated by this set of mechanisms.

6 Conclusions and Future Work

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

mechanisms 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. We have used TÆMS to model a simple distributed sensor network problem, the original DVMT domain, and a hospital scheduling environment. 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. We believe this separation is not only useful for applying our coordination mechanisms to 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).

The GPGP coordination approach as described in this paper has been fully implemented in the TÆMS simulation testbed. Significant experimental validation of the GPGP approach is documented in [4]. This paper showed how to decide if the addition of a new GPGP mechanism was useful. It showed the general performance of two GPGP family algorithms compared to a centralized parallel reference algorithm; GPGP with all mechanisms on produces 85% of the quality of the centralized reference scheduler in a random environment. Such performance is reasonable and we feel could be made even better by developing better local scheduling algorithms and new coordination mechanisms.

We also demonstrated how a feature of the task environment (the probability of task quality accumulation being MAX) can cause different GPGP family members to be preferred. We also discussed a sixth mechanism, a load balancing mechanism that communicates meta-level information, and showed that it was somewhat more useful when the variance in duration of the agents' overlapping tasks was high. This section thus ties-in back to the discussion in [6, 7] on the usefulness of meta-level communication (in this case, the transmission of local load information) when the inter-episode variance (in this case, in the initial agent loads) is high.

Finally, we gave a sense of the performance space of the five broadly-parameterized mechanisms using a clustering technique. Clustering can be a useful method for dealing with large algorithm spaces to prune search for an appropriate combination of mechanisms. Such methods may also lead to ways to learn situation-specific knowledge about the application of certain mechanisms in certain situations (perhaps using case-based reasoning techniques).

We believe that GPGP can become a reusable, domain-independent basis for multi-agent coordination when used in conjunction with a library of coordination mechanisms and a learning mechanism. We intend to develop such a library of reusable coordination mechanisms. For example, mechanisms that work from the successors of hard and soft relationships instead of the predecessors, negotiation mechanisms, mechanisms for behavior such as contracting, or mechanisms that can be used by self-motivated agents in non-cooperative environments. Many of these mechanisms can be built on the existing work of other DAI researchers. 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 mechanism set for an environment). Finally, we are also beginning work on using the GPGP approach in applications ranging from providing human coordination assistance to distributed information gathering.

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):213–261, 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, 1995.

[5] 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.

[6] Keith S. Decker and Victor R. Lesser. An approach to analyzing the need for meta-level communication. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 360–366, Chambéry, France, August 1993.

[7] Keith S. Decker and Victor R. Lesser. A one-shot dynamic coordination algorithm for distributed sensor networks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 210–216, Washington, July 1993.

[8] 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.

[9] 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.

[10] 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.

[11] E. Ephrati and J.S. Rosenschein. Divide and conquer in multi-agent planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 375–380, Seattle, 1994. AAAI Press/MIT Press.

[12] 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.

[13] Alan Garvey and Victor Lesser. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1491–1502, 1993.

[14] B. Grosz and S. Kraus. Collaborative plans for group activities. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Chambéry, France, August 1993.

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

[16] 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.

[17] 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.

[18] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies (preliminary report). In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 276–281, San Jose, July 1992.

[19] 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.

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