1996

# Learning Situation-specific Coordination in Generalized Partial Global Planning

M. V. Nagendra Prasad
*University of Massachusetts - Amherst*

Victor R. Lesser
*University of Massachusetts - Amherst*

# Learning Situation-specific Coordination in Generalized Partial Global Planning

M V Nagendra Prasad, and Victor R Lesser
Department of Computer Science
University of Massachusetts, Amherst, MA 01002.
nagendra@cs.umass.edu

## 1 Introduction

Achieving effective cooperation in a multi-agent system is a difficult problem for a number of reasons. The first is that an agent's control decisions, based only on its local view of problem-solving task structures, may lead to inappropriate decisions about which activity it should do next, what results it should transmit to other agents and what results it should ask other agents to produce[5, 2]. If an agent has a view of the task structures of other agents it can make more informed choices. However, even with the availability of this type of meta-level information, there is still residual uncertainty about outcomes of tasks and what future tasks will be coming into the system, which may result in agents still exhibiting non-coherent behavior. These difficulties with achieving effective coordination are further exacerbated by the fact that an agent, in acquiring and exploiting a non-local view of other agents' activities, may expend significant computational resources. This expense is in terms of communication delays waiting for this information to arrive, as well as the computational cost of both providing this information in a suitable form to other agents and processing this information to make local control decisions. Thus, for specific problem-solving situations, due to the inherent uncertainty in agents' activities and the cost of meta-level processing, it may not be worthwhile to acquire a complete view of other agents' activities. Sophisticated coordination strategies are not cost-effective in all problem-solving situations, and permitting some level of non-coherent activity to occur may be the optimal coordination strategy[4]. In this case, local problem solving is done more efficiently where there is no additional overhead for coordination.

Generalized Partial Global Planning (GPGP) has recognized this need for creating tailored coordination strategies in response to the characteristics of a particular task environment. It is structured as an extensible set of modular coordination mechanisms so that any subset of mechanisms can be used. Experimental results have already verified that for some environments a subset of the mechanisms is more effective than using the entire set of mechanisms [3]. In this paper, we present a learning algorithm that endows agents with the capability to choose a suitable subset of the coordination mechanisms based on the present problem solving situation.

The rest of the paper is organized as follows. Section 2 discusses the coordination mechanisms in GPGP. Section 3 introduces our algorithm for situation-specific coordination. We then present a few early experiments in Section 4 and conclude.

## 2 Coordination in GPGP

Coordination is the process of managing interdependencies between activities[7]. Agents have to choose and temporally order their activities to mitigate the effects of harmful inter-relationships and exploit the beneficial inter-relationships between them. In the GPGP approach to coordination, a set of domain independent coordination mechanisms post constraints to modulate a local scheduler. These constraints deal with the importance of certain tasks due to their beneficial inter-relationships

with certain other non-local tasks, and appropriate times for the initiation and completion of such tasks. We will look at these in more detail below.

Each agent can be viewed as having three components: a local scheduler, a coordination module and a belief database. The belief database of an agent represents its subjective view of the present coordination problem instance and its comprises the agent's knowledge about the task structures in the environment, their tasks with their inter-relationships. A task structure is the elucidation of the structure of the problem an agent is trying to solve[2]. The local scheduler of an agent uses the information in the belief database to build schedules of method execution actions in order to maximize the agent's performance. A coordination module modulates the local scheduler by noticing the task structures in the belief database and doing a number of activities like gathering information about new task structures in the environment, communicating information about local beliefs to other agents or receiving information from them, and making or retracting commitments. A commitment represents a contract to achieve a particular quality by a specified deadline. The coordination module has a number of mechanisms which notice certain features of the task structures in the belief database and react in certain ways. We will briefly review five mechanisms that will be used for the experiments in this paper.

- **Mechanism 1: Updating Non-local Viewpoints**
  Each agent's subjective view of the current episode is only partial. This mechanism enables agents to communicate with each other to update their partial views. It has three options: 'none' policy i.e. no non-local view, or 'all of it' i.e. global view for each agent or 'some' policy i.e. partial view. The latter policy needs some explanation. The agents need not know all the information about another agent to perform effective coordination. The 'some' policy enables an agent to communicate coordination relationships, private tasks and their context only to the extent that the other agents can detect non-local inter-relationships with this agent.
- **Mechanism 2: Communicating Results**
  The result communication coordination mechanism has three options: 'minimum' policy that communicates only the results necessary to satisfy non-local commitments, 'TG' policy that communicates the information in minimum policy and also the final results of a task group, and 'all' policy that communicates all the results.
- **Mechanism 3: Handling Simple Redundancy**
  When more than one agent decides to execute a redundant method, one agent is randomly chosen to execute it and send the results to other interested agents. This mechanism can be 'active' or 'inactive'.
- **Mechanism 4: Handling Hard Coordination Relationships**
  Hard coordination involves the *enables*($M_1$,$M_2$) relationship that implies that $M_1$ must be executed before $M_2$ to obtain quality for $M_2$. The hard coordination mechanism examines the current schedule for those methods that are predecessors in a hard coordination relationship and commits, both locally and socially, to their execution by a certain deadline. The agent with the method at the successor end of such a hard coordination relationship can post this commitment as a constraint on the earliest starting time of this method. This mechanism can be 'active' or 'inactive'.
- **Mechanism 5: Handling Soft Coordination Relationships**
  Soft coordination involves the *facilitates*($M_1$,$M_2$) relationship that implies that executing $M_1$ before $M_2$ decreases the duration of $M_2$ by a certain 'power' factor $\phi_d$ and increases the quality by a certain 'power' factor $\phi_q$. This mechanism functions in a manner similar to the previous one. It can be 'active' or 'inactive'.

Each of the five mechanisms can be parameterized independently to obtain a total of 72 possible coordination algorithms. Decker and Lesser[3] explored the multidimensional performance space for these coordination algorithms and found that they could be clustered into a much smaller set of significantly different algorithms: **Simple** with no hard or soft coordination mechanisms or non-local views and broadcast all results, **Myopic** with all commitment mechanisms on but no non-local view, **Balanced** with all mechanisms on and only partial views updated, **Tough** with no soft coordination but otherwise same as **Balanced** and **Mute** with no hard or soft coordination mechanisms and no non-local-views and no communication whatsoever.

Our learning mechanism starts with these five coordination algorithms as the possible choices from which the agents chooses an algorithm, based on the abstraction of the present coordination problem situation.

## 3 Learning Coordination

Many researchers have shown that no single coordination mechanism is good for all situations. However, there is little in the literature that deals with how to choose a coordination strategy based on the situation. In this paper, we present a learning algorithm that uses an abstract characterization of the coordination problem instance to choose a coordination algorithm from among the five classes of algorithms discussed in the previous section.

Our learning algorithm falls into the category of Instance-Based Learning algorithms[1]. It involves subjecting the multi-agent system to a series of runs to obtain a set of situation vectors and the corresponding system performances (normalized based on estimates of the best possible system performances) for each of the five coordination algorithms discussed previously. During these training runs each of the five coordination algorithms are run against a problem instance to facilitate exploring the relative performance of the algorithms for various system configurations. Each agent builds and stores these situation vectors by observing the local situation and communicating with other agents about their situations to form its approximation of the global situation. At the end of the problem solving, each agent observes or communicates with other agents to obtain the system performance measures and registers them against the situation vector. Thus, the training phase builds a set of {situation-vector,coordination-algorithm,performance}-triplets for each of the agents. Note that at the beginning of a problem solving episode, all agents communicate their local problem solving situations to other agents. Thus, each agent aggregates the local problem solving situations to form a common global situation. Our learning is setup to be off-line, but extending an IBL method to be online is relatively straight forward.

After the learning is terminated, agents seeing a new coordination episode can communicate with each other to form a global situation vector and use it index into the database of previously seen situation instances and retrieve those that are in the neighborhood of the present situation. These "neighborhood instances" are used to determine the best coordination algorithm for the present situation. We are trying various methods for this choice process but the one that gave us the most promising results is based on choosing the coordination algorithm with maximum weighted total credibility score. Credibility score for a coordination algorithm in the present situation is obtained by incrementing, for each problem solving situation in the neighborhood, by similarity * performance (like total quality) in that past situation.

The situation vector is an abstraction of the coordination problem and the effects of the five coordination mechanisms discussed previously. In the present system, it has 8 components. The first two components represent an approximation of the effect of detecting soft coordination inter-

relationships on the quality and duration of the local task structures at an agent. We create a virtual task structure from the locally available task structure of an agent by letting each of the facilitates inter-relationships potentially effecting a local task to actually take effect[1]. We use a design-to-time real-time scheduler[6] to schedule this virtual task structure to see the effect of coordinating on facilitates inter-relationships for a particular agent. The next two components represent an approximation of the effect of detecting hard coordination inter-relationships on the quality and duration of the local task structures at an agent. They are obtained in a manner similar to that described for facilitates inter-relationship. The fifth component represents the time pressure on the agent and the sixth component represents the load. These are obtained as ratios of agent quality and time performances when the deadline is infinity versus when the deadline is the actual deadline on the task structures in the environment. The final two components capture certain organizational parameters that represent the strength of the soft coordination inter-relationships and the probability of redundancy in the task structures. Each agent is assumed to have the setting for these parameters in the form of some system-wide knowledge about the agent organization.

## 4 Experiments

We use the TAEMS framework[2] (Task Analysis, Environment Modeling, and Simulation) as a domain-independent representation for coordination problems. A random generator based on a generation template is used to create TAEMS task structure episodes. Each episode is seen at time 0 and the task structures are static (i.e. they do not change during execution). Each episode has two task structures and there are four agents in the environment. Each agent has only a partial view of the task structures and an agent's partial view may differ from another agent's partial view.

We trained the system with 1500 problem instances. The probability of facilitates was varied from 0 to 1.0 in steps of 0.25 and 300 instances were generated at each setting of facilitates probability. We then tested the system with 100 instances again varying the facilitates probability in steps of 0.25 from 0 to 1.0. The performance of the system is given by[2]:

$$Performance = total\_quality - communication\_cost * totalcommunication$$

Table 1 shows the average performance of the best of the five algorithms versus the learning algorithm at various communication costs. When the communication cost was 0.0, the situation-specific GPGP system learned to choose **balanced** in most of the situations. When the communication cost was set high to 1.5, then the learning system learned to choose mute most of the times. These experiments show that the learning system can take into consideration varying communication costs leading to the choice of different coordination algorithms under different circumstances. **Mute** outperforms **balanced** when communication cost is high, where as **balanced** outperforms **mute** when either communication cost is 0 or if it is not a factor in the final performance. The learning system chooses one of the five coordination algorithms based on the coordination problem instance. While it may not have always performed better than the best of the five mechanisms at a particular setting of communication cost, its performance is better than any one of the coordination algorithms across the range of communication costs.

[1] An agent can be expected to know the inter-relationships effecting its tasks though it may not know the exact tasks in other agents which effect it without communicating with them.

[2] Performance is considered better if the number representing performance is higher.

| *Communication cost* | *Coordination Algorithm* | | |
|---|---|---|---|
| | Learn | Balanced | Mute |
| 0.0 | 162.79 | 161.88 | 151.6 |
| 0.5 | 176.50 | 176.38 | 166.99 |
| 1.5 | 135.61 | 118.96 | 140.25 |
| Overall Average | 158.3 | 152.41 | 152.95 |

Table 1: Average performance at various communication costs

## 5 Conclusion

Early results from our experiments have revealed to us, the benefits of learning situation-specific coordination. However, we need more extensive studies dealing with various aspects of the problem of learning coordination. In addition, we need to investigate the implications of learning being distributed, as is the case in our system (each agent tries to learn about more global aspects of problem solving control by communicating with other agents)? We suspect that a number of issues that have cropped up in multi-agent system studies may also need to be tackled here, albeit in the context of learning.

## References


[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance, and Management*, 2(4):215–234, December 1993. Special issue on "Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior".

[3] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 73–80, San Francisco, CA, June 1995. AAAI Press.

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

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

[6] Alan Garvey, Marty Humphrey, and Victor Lesser. Task interdependencies in design-to-time real-time scheduling. pages 580–585, Washington, D.C., July 1993.

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