Learning Situation-Specific Coordination in Cooperative Multi-agent Systems

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

Abstract

Achieving effective cooperation in a multi-agent system is a difficult problem for a number of reasons such as limited and possibly out-dated views of activities of other agents and uncertainty about the outcomes of interacting non-local tasks. In this paper, we present a learning system called COLLAGE, that endows the agents with the capability to learn how to choose the most appropriate coordination strategy from a set of available coordination strategies. COLLAGE relies on meta-level information about agents' problem solving situations to guide them towards a suitable choice for a coordination strategy. We present empirical results that strongly indicate the effectiveness of the learning algorithm.

Keywords: *Multi-agent Systems, Coordination, Learning*

1 Introduction

Coordination is the process of effectively managing interdependencies between activities distributed across agents so as to derive maximum benefit from them[21, 6]. Based on structure and uncertainty in their environment, agents have to choose and temporally order their activities to mitigate the effects of harmful interdependencies and exploit the beneficial interdependencies among them. Researchers in both human and computational organizational theories have pointed out that there is no single organization or coordination protocol that is the best for all environments. In human organizations, environmental factors such as dynamism and task uncertainty have a strong effect on what coordinated actions are and how organizationally acceptable outcomes arise[18, 9, 33]. These effects have been observed in purely computational organizations as well[8, 7, 6, 23, 26].

Achieving effective coordination in a multi-agent system (MAS) is a difficult problem for a number of reasons. An agent's local control decisions about what activity to do next or what information to communicate and to whom or what information to ask others may be inappropriate or suboptimal due to its limited view of the interactions between its own activities and those of the other agents. In order to make more informed control decisions, the agents have to acquire a view of the task structures of other agents. To the extent that this resolves agents' uncertainty about the non-local problem solving activities, they can act coherently. However, an agent has to expend computational resources in acquiring and exploiting such non-local views of other agents' activities. This involves communication delays and the computational cost of providing this information and assimilating the information from other agents. Given the inherent uncertainty in agents' activities and the cost of meta-level processing, relying on sophisticated coordination strategies to acquire non-local views of task structures may not be worthwhile for all problem-solving situations. In certain situations, coordination protocols that permit some level of non-coherent activity and avoid the additional overhead for coordination may lead to better performance[7, 6, 23, 34]. For example, when the agents are under severe time pressure and the load of the activities at the agents is high, sophisticated agent coordination strategies do not generally payoff. Agents may not have the time to benefit from the increased awareness they derive through coordination in such situations. In this paper, we will be dealing with how agents can learn to dynamically choose the appropriate coordination strategy in different coordination problem instances. We empirically demonstrate that even for a narrow class of agent activities, learning to choose the appropriate coordination strategy based on meta-level characterization of the global problem solving state outperforms using any single coordination strategy across all problem instances.

In order to accomplish learning, we break the coordination problem into two phases. In the first phase, the agents exchange meta-level information not directly used for coordination. This information is used by the agents to derive a prediction of the effectiveness of various coordination mechanisms in the present problem solving episode. These mechanisms differ in the amount of non-local information they acquire and use, and in the complexity of analysis of interactions between activities at the agents. Agents choose an appropriate subset of the coordination mechanisms (or a coordination strategy) based on the meta-level information and enter Phase II. In this phase, the coordination strategy selected in Phase I decides the types of information to be exchanged and the kind of reasoning about local and non-local activities the agents need to perform to achieve coherent behavior. We call the meta-level information a *situation* and the two phase process *situation-specific coordination*. Learning situation-specific coordination involves associating appropriate views of the global situation with the knowledge learned about the effectiveness of the coordination mechanisms.

The agents in a cooperative multi-agent system can learn to coordinate by proactively resolving some of their uncertainty about the global problem solving state and situating their learning in more global views.

The rest of the paper is organized as follows. We first place our work in context and discuss its relationship to the existing work on learning in multi-agent systems. We then briefly review the TÆMS task structure representation for coordination problems and the GPGP model for designing coordination strategies. We subsequently describe our learning algorithm that learns how to choose among coordination strategies of different levels of sophistication. We then present some of our experimental results and conclude.

2 Related Work

Previous work related to learning in multi-agent systems is limited and much of this work relies on techniques derived from reinforcement learning[1, 35], genetic algorithms[16] and classifier systems[17]. In Sen, Sekaran and Hale[29] the agents use reinforcement learning to evolve complimentary policies in a box pushing task, and in Crites and Barto[3] a team of reinforcement learning agents optimize elevator dispatching performance. In both these works, the agents do not communicate with one another and any agent treats the other agents as a part of the environment. Weiss[37] uses a variant of Holland's[17] bucket brigade algorithm for learning hierarchical organization structuring relationships in a block world domain via strengthening promising chains of actions through bid-reward cycles. Tan[36] deals with a predator-prey domain in a grid world. The agents share perception information to overcome perceptual limitations or communicate policy functions learned through reinforcement learning. Grefenstette[14] uses genetic algorithms to learn reactive decision rules for agents in a predator-prey domain similar to that in [36]. Sandholm and Crites[28] study the emergence of cooperation in the Iterated Prisoner's Dilemma problem, where the agents are self-interested, and an agent is not free to ask for any kind of information from the other agents. Haynes and Sen[15] present studies in learning cases to resolve conflicts among agents in a predator-prey domain similar to that used by Tan[36]. There is no communication between the agents, and the cases result from the agent's perception of the problem solving state.

The multi-agent systems that this paper deals with consist of complex cooperative agents (each agent is a sophisticated problem solver), and an agent's local problem solving control interacts with that of the other agents' in intricate ways. In our work, rather than treating other agents as a part of the environment and learning in the presence of increased uncertainty, agents communicate meta-level information to resolve the uncertainty to the extent possible (there is still the environmental uncertainty that the agents cannot do much about). Agents sharing perceptual information as in Tan[36] and Greffenstette[14] or bidding information as in Weiss[37] do not make explicit the notion of situating the local control knowledge in a more global, abstract situation. The information shared is weak, and the studies were conducted in domains such as predator-prey[36, 14] or blocks world[37] where the need for sharing meta-level information and situating learning in this information is not apparent.

Our previous work explored learning situation-specific organizational roles in a heterogeneous multi-agent system[26]. An organizational role represents a set of tasks an agent can perform on a composite solution. Agent use abstractions of problem solving state to learn to play the roles they are best suited for in a multi-agent search process for cooperatively constructing an overall solution. The system was tested in a parametric design domain and the learning agents produced designs that, on an average, were better than those produced by a system with agents playing roles hand-coded by a human expert.

Sugawara and Lesser[34] also recognize the need for situation specificity in learning coordination, though they do have the notion of two-phase coordination. They are concerned with learning to make the situations more discriminating to avoid using an inappropriate coordination strategy in the domain of distributed network diagnosis. Their learning mechanisms rely on deep domain knowledge and agent homogeneity assumptions in order to progressively refine situations based on failure-driven explanation and comparative analysis of problem solving traces. They test the theory on a very limited number of coordination situations, and the evidence was anecdotal. It is not clear how such knowledge-intensive learning can be generalized to other instances without significant knowledge engineering and the development of more sophisticated explanation-based learning techniques. Despite these limitations, combining their work on learning situation representations with the learning presented here on situation-based choice of coordination could have interesting implications for situation-specific learning.

Garland and Alterman[10] discuss issues in knowledge reuse in multi-agent systems. Agents working in a simulated MOVERS-WORLD domain have to collaborate to move objects too heavy for a single agent to move. Agents use their past experience to anticipate coordination rather than dynamically communicate to establish it. Motivation for this work is different from ours. In their work, agents exploit their past experience to reduce the control effort needed to achieve coordination in future problem instances. Similar use of past experience has been studied in Nagendra Prasad, Lander and Lesser[25].

Certain multi-agent learning systems in the literature deal with a different task from that presented in this paper. Systems like ILS[32] and MALE[31] use multi-agent techniques to build hybrid learners from multiple learning agents. On the other hand, we deal with learning problem-solving control for multi-agent systems.

3 Coordination in Multi-agent Systems

In this paper, we show the effectiveness of learning situation-specific coordination by extending a domain-independent coordination framework called Generalized Partial Global Planning (GPGP)[6, 4]. GPGP is a flexible and modular method that recognizes the 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. Each mechanism can be invoked in response to the detection of certain features of the environment and interdependencies between problem solving activities of agents. The coordination mechanisms can be configured (or parameterized) in different ways to derive different coordination strategies. In GPGP (without the learning extensions), once a set of mechanisms are configured (by a human expert) to derive a coordination strategy, that strategy is used across all problem instances in the environment. Experimental results have already verified that for some environments a subset of the mechanisms is more effective than using the entire set of mechanisms [6, 23]. In this work, we present a learning extension, called COLLAGE, that endows the agents with the capability to choose a suitable subset of the coordination mechanisms based on the present problem solving situation, instead of having a fixed subset across all problem instances in an environment.

GPGP coordination mechanisms rely on the coordination problem instance being represented in a framework called TÆMS[4, 5]. The TÆMS framework (Task Analysis, Environment Modeling, and Simulation) [4, 5] represents coordination problems in a formal, domain-independent way. It captures the essence of the coordination problem instances, independent of the details of the domain, in terms of trade-offs between multiple ways of accomplishing a goal, progress towards the achievement of goals, various interactions between the activities in the problem instance and the effect of these activities on the performance of the system as a whole. TÆMS can model aspects of coordination in complex *worth-oriented* domains[27] where states have functions that rate their acceptability (not necessarily a binary rating). There are deadlines associated with the tasks and some of the subtasks may be interdependent, that is, cannot be solved independently in isolation. There may be multiple ways to accomplish a task and these tradeoff the quality of a result produced for the time to produce that result. *Quality* is used as a catch-all term representing acceptability characteristics like certainty, precision, and completeness, other than temporal characteristics. A set of related tasks is called a task group or a task structure. A task structure is the elucidation of the structure of the problem an agent is trying to solve. The quality of a task is a function of the quality of its subtasks. The leaf tasks are denoted as methods with base level quality and duration. Methods represent domain actions, like executing a blackboard knowledge source, running an instantiated plan, or executing a piece of code with its data. A task may have multiple ways to accomplish it, represented by multiple methods, that trade off the time to produce a result for the quality of the result. Figure 1 is an example of a simple tasks structure. Besides task/subtask relationships, there can be other interdependencies, called coordination interrelationships, between tasks in a task group[5]. In this paper, we will be dealing with two such interrelationships:

- facilitates relationship or soft interrelationship: Task A facilitates Task B if the the execution Task A produces a result that affects an optional parameter of Task B. Task B could have been executed without the result from Task A, but the availability of the result provides constraints on the execution of Task B, leading to improved quality and reduced time requirements.

- enables relationships or hard interrelationship: If Task A enables Task B then the execution Task A produces a result that is a required input parameter for Task B. The results of execution of Task A must be communicated to Task B before it can be executed.

In GPGP, 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 comprising the agent's knowledge about the task structures in the environment. The local scheduler of an agent uses the information in the belief database to build schedules of method execution actions in order to maximize utility. In this work we use a design-to-time scheduler[12] that can take into account, the constraints provided by the coordination module and provide schedules that maximize the global utility measure. The 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 mechanisms are parameterized independently so that each combination of the parameter settings leads to a coordination

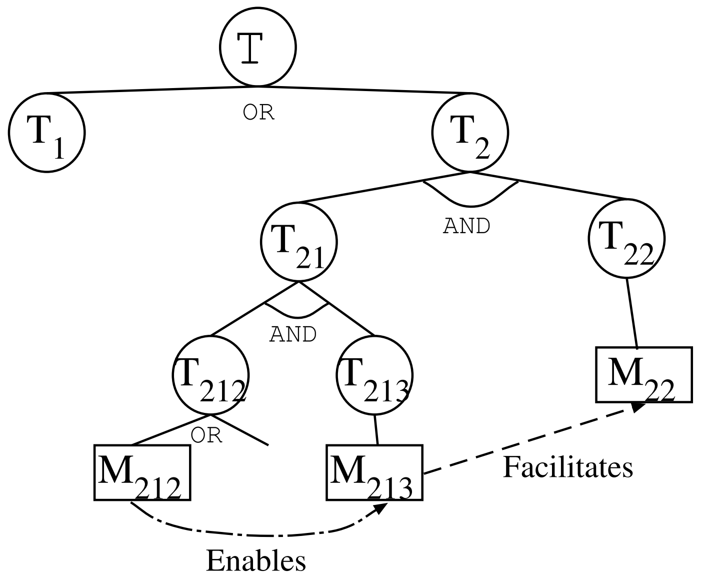


Figure 1: Example of a Task Structure

strategy. Different mechanisms require different types of information to operate. Thus, by choosing different sets of coordination mechanisms, the amount of communication and other overheads associated with coordination can be varied. The interface between the coordination mechanisms and the local scheduler is bidirectional and negotiation-based, permitting the coordination module to ask “what-if” questions[11]. This is a crucial ability whose utility becomes obvious later, when we describe our learning algorithm in more detail.

In GPGP, a coordination strategy can be derived by activating a subset of the coordination mechanisms. Given that these mechanisms can be parameterized independently, there are large number of possible coordination strategies. Learning to choose the best strategy from amongst a large set can be quite cumbersome. We suggest two ways to prune this search space, leading to a more manageable set of distinct coordination strategies:

- Use domain knowledge to prune down the set of possible coordination strategies to small number of interesting distinct strategies. In the experiments to be described later, we will be dealing with the distributed data processing domain. In this domain, we investigate three strategies:
  - `balanced` (or dynamic-scheduling): It is the same as previously described. Agents coordinate their actions by dynamically forming commitments. Relevant results are generated by specific times and communicated to the agents to whom corresponding commitments are made. Agents schedule their local tasks trying to maximize the accrual of quality based on the commitments made to it by the other agents, while ensuring that commitments to other agents are satisfied. The agents have the relevant non-local view of the coordination problem, detect coordination relationships, form commitments and communicate the committed results.
  - `data-flow` strategy: An agent communicates the result of performing a task to all the agents and the other agents can exploit these results if they still can. This represents

the other extreme where there are no commitments from any agent to any other agent.

  - `rough` coordination: This is similar to `balanced` but commitments do not arise out of communication between agents but are known *a priori*. Each agent has an approximate idea of when the other agents complete their tasks and communicate results based on its past experience. The agents have the relevant non-local view of the coordination problem, detect coordination relationships, but use rough commitments and communicate the committed results. "Rough commitments" are a form of tacit social contract between agents about the completion times of their tasks. It might be possible to view rough commitments as precompiled social laws [30][1].

  The latter two coordination strategies are the alternatives normally used in the distributed data processing domain[23]. Decker and Lesser[6] proposed `balanced` as a sophisticated strategy that exploits a number of mechanisms to achieve coordination. In our experiments, agents learn to choose among these three coordination strategies in the domain of distributed data processing. In this domain, three types of task groups arise: routine task groups, crisis task groups, and low priority task groups (see Section 5 for details). The latter two task groups arise only occasionally (i.e. with a probability) and hence it is unrealistic to expect commitments on crisis tasks and low priority tasks to follow such tacit *a priori* rough commitments. So this coordination type uses rough commitments for routine tasks but behaves just like data flow for the non-routine crisis and low priority tasks.

- Cluster the entire set of possible strategies on the performance space and choose a prototypical instance from each cluster. The agent performance can be characterized along a number of dimensions like total quality, number of methods executed, number of communications, and termination time. Decker and Lesser[6] explore the performance space of the coordination strategies based on these four performance measures and identify five prototypical coordination strategies:

  - `balanced` with all mechanisms for detecting soft and hard coordination interrelationships and forming commitments on them turned on. Only relevant partial views are exchanged between agents for detecting these interrelationships and results that satisfy commitments are exchanged.
  - `simple` with no hard or soft coordination mechanisms or exchange of relevant partial non-local views. All results obtained due to task executions are broadcast to all agents.
  - `myopic` with all commitment mechanisms on but no non-local view.
  - `tough` with no soft coordination but otherwise same as `balanced`
  - `mute` with no hard or soft coordination mechanisms and no non-local-views and no communication whatsoever.

  These five coordinations strategies will be used in our experiments later on some synthetic domains.

[1]Rough commitments could also be learned by the agents.

4 COLLAGE: Learning Coordination

4.1 Learning Coordination

Our learning algorithm, called COLLAGE, uses abstract meta-level information about coordination problem instances to learn to choose, for the given problem instance, the appropriate coordination strategy from the three strategies described previously. Learning in COLLAGE (**CO**ordination **L**earner for mu**L**tiple **AGE**nt systems) falls into the category of Instance-Based Learning algorithms[2] originally proposed for supervised classification learning. We, however, use the IBL-paradigm for unsupervised learning of decision-theoretic choice. We would like to note that we could as well have used some other method like neural nets or decision trees. Our interest in this work is not in studying the comparative performances of various algorithms.

Learning involves running the multi-agent system on a large number of training coordination problem instances and observing the performance of different coordination strategies on these instances. When a new task group arises in the environment, each of the agents has its own partial local view of the task group. Based on its local view, each agent forms a *local situation* vector. A local situation represents an agent's assessment of the utility of reacting to various characteristics of the environment. Such an assessment can potentially indicate the need for activating the various GPGP mechanisms and consequently has a direct bearing on the type of coordination strategy that is best for the given coordination episode. The agents then exchange their local situation vectors and each of the agents composes all the local situation vectors into a global situation vector. All agents agree on a choice of the coordination strategy and the choice depends on the kind of learning mode of the agents:

Mode 1: In this mode, the agents run all the available coordination strategies and note their relative performances for the each of the coordination problem instances. Thus, for example, agents run each of `data-flow`, `rough`, and `balanced` for a coordination episode and store their performances for each strategy.

Mode 2: In this mode, the agents choose one of the coordination strategies for a given coordination episode and observe and store the performance only for that coordination strategy. They choose the coordination that is represented the least number of times in the neighborhood of a small radius around the present global situation. This is done to obtain a balanced representation for all the coordination strategies across the space of possible global situations.

Mode 2 is a quasi-online algorithm. In the initial stages it just explores and in the later stages it just exploits the learned information. Studying COLLAGE in a setup that is more typical of online learning algorithms, where exploration and exploitation are interleaved, is high on our agenda of future work.

At end of each run of the coordination episode with a particular coordination strategy, the performance of the system is registered. This is represented as a vector of performance measures such as total quality, number of communications, and termination time[2]. Learning involves simply adding the new instance formed by the performance of the coordination strategy along with the associated problem solving situation to the "instance-base". Thus, the training phase builds a set of

[2]In the work presented here, the cost of scheduling is ignored and the time for scheduling is considered negligible compared to the execution time of the methods. Our work could easily be generalized to take into account these other forms of coordination overhead like the number of calls to the scheduler and total number of methods executed. We view this as one of our future directions of research.

$\{situation, coordination\_strategy, performance\}$ triplets for each of the agents. Here the global situation vector is the abstraction of the global problem solving state associated with the choice of a coordination-strategy. 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. All agents form identical instance-bases because they build the same global situation vectors through communication.

4.1.1 Forming a Local Situation Vector

The situation vector is an abstraction of the coordination problem and the effects of the coordination mechanisms in GPGP. It is composed of six components:

- The first component represents an approximation of the effect of detecting soft coordination relationships on the quality component of the overall performance. An agent creates virtual task structures from the locally available task structures by letting each of the facilitates coordination relationships potentially affecting a local task to actually take effect with maximum strength and calls the scheduler on these task structures. In order to achieve this, the agent detects all the facilitates interrelationships that affect its tasks. An agent can be expected to know the interrelationships affecting its tasks though it may not know the exact tasks in other agents that affect it without communicating with them. The agent then produces another set of virtual task structures, but this time with the assumption that the facilitates relationships are not detected and hence the tasks that can potentially be affected by them are not affected in these task structures. The scheduler is again called with this task structure. The first component, representing the effect of detecting facilitates is obtained as the ratio of the quality produced by the schedule without facilitates relationships and the quality produced by the schedule with facilitates relationships. Note that that this ratio is only an approximation of the actual effects of the facilitates interrelationships. In actual problem solving process, it is highly unlikely that all facilitates interrelationships with be exploited with their full strength.

- The second component represents an approximation of the effect of detecting soft coordination relationships on the duration component of the overall performance. It is formed using the same techniques discussed above for quality but using the duration of the schedules formed with the virtual task structures.

- The third component represents an approximation of the effect of detecting hard coordination interrelationships on the quality the local task structures at an agent. They are obtained in a manner similar to that described for facilitates inter-relationship. The third component is obtained as the ratio of the quality produced by the schedule without enables relationships and the quality produced by the schedule with enables relationships.

- The fourth component represents an approximation of the effect of detecting hard coordination relationships on the duration component of the overall performance. It is formed using methods similar to that discussed above for quality but using the duration of the schedules formed with the virtual task structures.

- The fifth component represents the time pressure on the agent. In a design-to-time scheduler, increased time pressure on an agent will lead to schedules that will still adhere to the deadline requirements as far as possible but with a sacrifice in quality. Under time pressure, lower quality, lower duration methods are preferred over higher quality, higher duration methods for achieving a particular task. In order to get an estimate of the time pressure, an agent generates virtual task structures from its local task structures by setting the deadlines of the task groups, tasks and methods to ∞ (a large number) and scheduling these virtual task structures. The agents schedule again with local task structures set to the actual deadline. Time pressure is obtained as the ratio of the schedule quality with the actual deadlines and the schedule quality with large deadlines.

- The sixth component represents the load. It is formed using methods similar to that discussed above for time pressure but using the duration of the schedules formed with the virtual task structures. The time taken by the schedule when it has no time pressure (deadline is ∞) represents the amount of work the agent would have done under ideal conditions. However, time pressure makes the agent work almost right up to the deadline. It may not work all the way to the deadline as it may not find a method that can be fitted into the last available chunk of time. Thus, load is obtained as the ratio of execution time under actual deadline and the execution time under no time pressure.

4.1.2 Forming a Global Situation Vector

Each agent communicates its local situation vector to all other agents. An agent composes all the local situation vectors: its own and those it received from others to form a global situation vector. A number of composition functions are possible. The one we used in the experiments reported here is simple: component-wise average of the local situation vectors. Thus the global situation vector has six components where each component is the average of the corresponding components of the local situation vectors.
An example global situation vector looks as follows: `(0.82 0.77 0.66 0.89 1.0 0.87)`. Here the low value of the third component represents large quality gains by detecting and coordinating on hard interrelationships. Thus two of the more sophisticated coordination strategies called `balanced` and `tough`[6] are found to be better performers in this situation. On the other hand, in a global situation vector such as `(0.80 0.90 0.88 0.80 0.61 0.69)` the low values of fifth and sixth components indicate high time pressure and load in the present problem solving episode. Even if the agents use sophisticated strategies to coordinate, they may not have the time to benefit from it. Hence, relatively simple coordination strategies like `simple` or `mute`[6] do better in this scenario.

Note, however, that in most situation vectors, these trade-offs are subtle and not as obvious as the above examples. It is difficult for a human to look at the situations and easily predict which strategy is the best performer. The trade-offs may be very specific to the kinds of task structures that occur in the domain. Hence, hand-coding the strategies by a designer is not a practical alternative.

Once the entire instance-base is formed, each of the features for each instance is normalized by the range of maximum and minimum for that feature within the entire instance-base. This is done in order to avoid biasing the similarity metric in favor of any particular feature.

4.2 Choosing a Coordination Strategy

COLLAGE chooses a coordination strategy based on how the set of available strategies performed in similar past cases. We adopt the notation from Gilboa and Schmeidler[13][3]. Each case c is triplet

$$\begin{aligned} \langle p, a, r \rangle &\in C_i \\ C_i &\subseteq P \times A \times R \end{aligned}$$

where $p \in P$ and P is the set of situations representing abstract characterization of coordination problems, $a \in A$ and A is the set of coordination choices available, $r \in R$ and R is the set of results from running the coordination strategies.

Decisions about coordination strategy choice are made based on similar past cases. Outcomes decide the desirability of the strategies. We define a similarity function and a utility functions as follows:

$$\begin{aligned} s : P^2 &\rightarrow [0, 1] \\ u : R &\rightarrow \Re \end{aligned}$$

In the experiments presented later, we use the Euclidean metric for similarity.

The desirability of a coordination strategy is determined by a similarity-weighted sum of the utility it yielded in the similar past cases in a small neighborhood around the present situation vector. We observed that such an averaging process in a neighborhood around the present situation vector was more robust than taking the nearest neighbor, possibly because the averaging process was less sensitive to noise. Let M be the set of past similar cases to problem $p_{new} \in P$ (greater than a threshold similarity).

$$m \in M \quad \Leftrightarrow \quad s(p_{new}, m) \geq s_{threshold}$$

For $a \in A$, let $M_a \equiv \{m = \langle p, \alpha, r \rangle \in M | \alpha = a\}$. The utility of a is defined as

$$U_{(}p_{new}, a) = \frac{1}{|M_a|} \sum_{\langle q, a, r \rangle \in M_a} s(p_{new}, q) u(r)$$

5 Experiments

5.1 Experiments in the DDP domain

Our earlier work on learning coordination[24] relied on a weak task environment generator where the task structures were generated randomly. The experimenter was limited to setting certain numerical parameters like mean of the task structure depth or mean and variance of the number

[3]Gilboa and Schmeidler[13] describe case-based decision theory as a normative theory of human behavior during decision making. Even though, we adopt their notation, there are crucial differences in the motivations and structure of the two works. Gilboa and Schmeidler are primary concerned with a descriptive theory of human decision making while our work, developed independently, is concerned primarily with building computational systems based on case-based decision theory.

of interrelationships in task structures. This often gives rise to a wide range of task structures and a huge variance in the types of capabilities needed by the system to effectively handle them. Accordingly, the learning algorithms showed at best modest gains in performance[24]. More importantly, it is unlikely that most real applications involve an infinite variety of task structures. The domain semantics dictate and limit morphology of the task structures. While there is bound to be some randomness in these structures, it is highly unlikely that the only regularity that can be modeled in the task structure representations of a coordination problem instance are a few parameters like its mean depth or branching factor. Nagendra Prasad et al.[23] developed a graph-grammar-based stochastic task structure description language and generation tool for modeling task structures arising in a domain. Such a graph-grammar-based task structure specification language is powerful enough to model the topological relationships occurring in task structures representing many real life applications. For the experiments below, we use this tool to model a distributed data processing domain as in [23]. We now briefly introduce this domain and refer the interested reader to [23] for further details.

The distributed data processing domain consists of a number of geographically dispersed data processing centers (agents). Each center is responsible for conducting certain types of analysis tasks on streams of satellite data arriving at its site: "routine analysis" that needs to be performed on data coming in at regular intervals during the day, "crisis analysis" that needs to be performed on the incoming data but with a certain probability and "low priority analysis", the need for which arises at the beginning of the day with a certain probability. Low priority analysis involves performing specialized analysis on specific archival data. Different types of analysis tasks have different priorities. A center should first attend to the "crisis analysis tasks" and then perform "routine tasks" on the data. Time permitting, it can handle the low-priority tasks. The processing centers have limited resources to conduct their analysis on the incoming data and they have to do this within certain deadlines. Results of processing data at a center may need to be communicated to other centers due to the interrelationships between the tasks at these centers. In [23], we give details of how a stochastic graph-grammar can model task structures arising in a domain such as this. In the same work, we also present the results of empirical explorations of the effects of varying deadlines and crisis task group arrival probability. Based on the experiments, we noted the need for different coordination strategies in different situations to achieve good performance. In this section, we intend to demonstrate the power of COLLAGE in choosing the most appropriate coordination strategy in a given situation.

We performed two sets of experiments varying the probability of the centers seeing crisis tasks. In the first set of experiments, the crisis task group arrival probability was 0.25 and in the second set it was 1.0. For both sets of experiments, low priority tasks arrived with a probability of 0.5 and the routine tasks were always seen at the time of new arrivals. A day consisted of a time slice of 140 time units and hence the deadline for the task structures was fixed at 140 time units. In the experiments described here, utility is the primary performance measure. Each message an agent communicates to another agent penalizes the overall utility by a factor called $comm\_cost$. However, achieving a better non-local view can potentially lead to higher quality that adds to the system-wide utility. Thus, $utility = quality - total\_communication \times comm\_cost$. The system consisted of three agents (or data processing centers) and they had to learn to choose the best coordination strategy from among `balanced`, `rough`, and `data-flow`.

5.1.1 Results

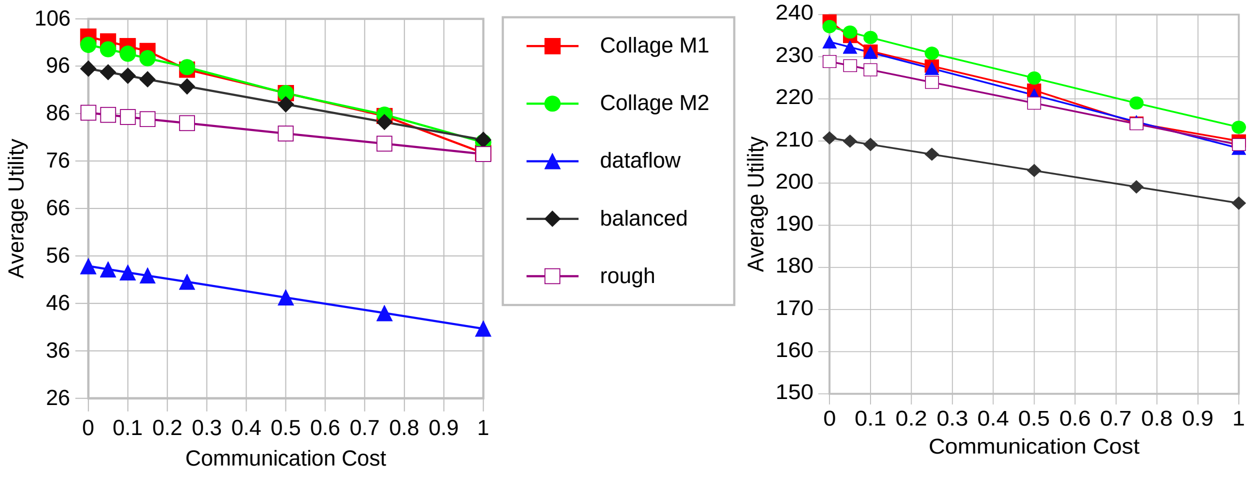
For the experiments where crisis task group arrival probability was 0.25, COLLAGE was trained on 4500 instances in Mode 1 and on 10000 instances in Mode 2. For the case where crisis task group arrival probability was 1.0, it was trained on 2500 instances in Mode 1 and on 12500 instances in Mode 2. In our experiments, the size of the case-base for Mode 1 learning was determined by plotting the performance of the learner, averaged over 500 instances randomly generated by the grammar, for various sizes of the instance-base, and at communication cost of 0. The learning was stopped when the performance improvement tapered off. Mode 2, was trained till it achieved the performance of Mode 1 learner, at communication cost of 0. Figure 2 shows the average quality over 100 runs for different coordination strategies at various communication costs. The curves for both Mode 1 and Mode 2 learning algorithms lie above those for all the other coordination strategies for the most part in both experiments. We performed a Wilcoxon matched-pair signed ranks analysis to test for significant differences (at significance level 0.05) between average performances of the strategies across communications costs upto 1.0 (as opposed to pairwise tests at each communication cost). This test revealed significant differences between the learning algorithms (both Mode 1 and Mode II) and the other three coordination strategies indicating that we can assert with a high degree of confidence that the performance of the learning algorithms across various communication costs is better than statically using any one of the family of coordination strategies[4]. As the communication costs go up, the mean performance of the coordination strategies go down. For crisis task group arrival probability of 0.25, the `balanced` coordination strategy performs better than the learning algorithms at very high communication costs because, learning algorithms use additional units of communication to form the global situation vectors. At very high communication costs, even the three additional meta-level messages for local situation communication (one for each agent) led to large penalties on utility of system. At communication cost of 1.0, Mode 1 learner and Mode 2 learner average at 77.72 and 79.98 respectively, whereas, choosing `balanced` always produces an average performance of 80.48. Similar behavior was exhibited at very high communication costs when the crisis task group arrival probability was 1.0. Figures 3 gives the number of coordination strategies of each type chosen in the 100 test runs for Mode 1 learning when crisis task group probability was 1.0[5].

5.2 Experiments with Synthetic domains

In order to test COLLAGE on interesting scenarios with a range of characteristics, we created a number of "synthetic domain theories" using graph grammar formalisms[23]. A synthetic grammar represents a domain theory that is not grounded in any real life application. We tested COLLAGE on three different synthetic environments generated by their corresponding grammars. In these experiments, there were four agents in the system and each agent has only a partial view of a new task group in the environment. Upon seeing a task group, a COLLAGE agent forms a local situation vector and communicates it to other agents to enable them to get a more global view of the problem solving process. Each agent forms a global situation vector (the same for all agents) and indexes into its instance base of past experience to choose a good coordination strategy for the

[4]Testing across communication costs is justified because in reality, the cost may vary during the course of the day.

[5]This figure shows distributions at communication costs 1.0, 2.0, and 3.0 that are not shown in Figure 2. We avoided this in Figure 2 because, including it would congest the display at communication costs between 0 - 1.0.



a) Crisis TG 0.25

b) Crisis TG 1.0

Figure 2: Average Quality versus Communication Cost

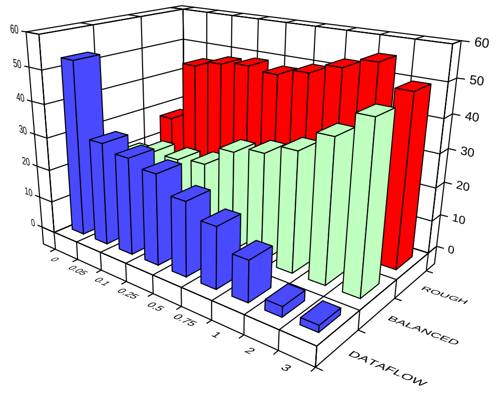


Figure 3: Strategies chosen by COLLAGE M1 for Crisis TG Probability 1.0 at various communication costs

present situation from among `balanced`, `mute`, `myopic`, `simple`, and `tough` coordination strategies.

5.2.1 Grammar 1 & Grammar 2

For the domain represented by G1, COLLAGE was trained on 2500 instances in Mode 1 and 18000 instances in Mode 2. Figure 4a shows the average quality over the same 100 runs for different coordination strategies at various communication costs. The curves for both Mode 1 and Mode 2 learning algorithms lie above those for all the other coordination strategies. We performed a Wilcoxon matched-pair signed ranks analysis to test for significant differences (at significance level 0.05) between average performances of the strategies across communications costs. This test revealed significant differences between the learning algorithms (both Mode 1 and Mode 2) and the other five coordination strategies, indicating that we can assert with a high degree of confidence that the performance of the learning algorithms across various communication costs is better than statically using any one of the family of coordination strategies. As with the DDP domain, when the communication costs go up, the mean performance of the learning algorithms goes down. At some high value of communication cost, the performance of the learning algorithm falls below that of `mute` because learning agents use four units of communication to publish their local situation vectors. The cost of communicating these local situation vectors can overwhelm the benefits derived from a better choice of the coordination strategy. When the communication cost was as high as 0.25, the performances of Mode 1 and Mode 2 learners were 12.495 and 12.88 respectively. On the other hand `mute` coordination gave a mean performance of 13.55.

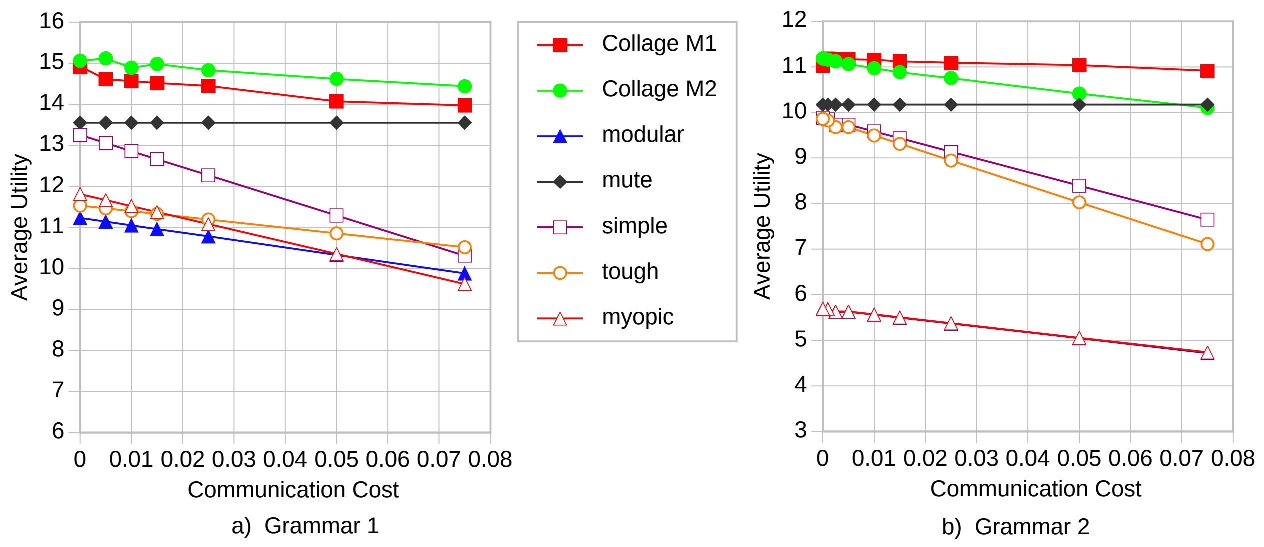


Figure 4: Average Quality versus Communication Cost

Grammar 2 is similar to Grammar 1 in structure but the distribution of quality and duration of the leaf methods is different from those in Grammar 1. Figure 4b shows the average quality for different coordination strategies at various communication costs. The performance behavior of the learning algorithms is similar as in the case of Grammar 1. For COLLAGE Mode 2, the average performance in fact falls below that of `mute` at communication cost 0.075. When the communication cost was as high[6] as 0.25, the performance of COLLAGE Mode 1 was at 10.762 which is still higher than `mute` at 10.17. However, COLLAGE Mode 2 fell to 7.735. A closer examination of Figure 5 and Figure 6, that give the number of coordination strategies of each type chosen in the 100 test runs by COLLAGE in Mode 1 and Mode 2, revealed the reasons for this. As communication costs increase, `mute` becomes the overwhelming favorite but even at as high communication costs as 0.08, there are still problems where `mute` is not the best choice. COLLAGE Mode 1 chose `mute` most of the instances (74%) except a few where `mute` is not a good coordination strategy. This led to a better performance that compensated for the increased communication cost. On the other hand, COLLAGE Mode 2 chose more of the other communication intensive coordination strategies (39 %) leading to poorer performances. We believe that more training in the Mode 2 may be able to rectify this situation. Note that we stopped the training of Mode 2, when its performance was around that of Mode 1 learner at communication cost 0. The divergence between their performances occurred at non-zero communication costs, and hence was not detected during learning.

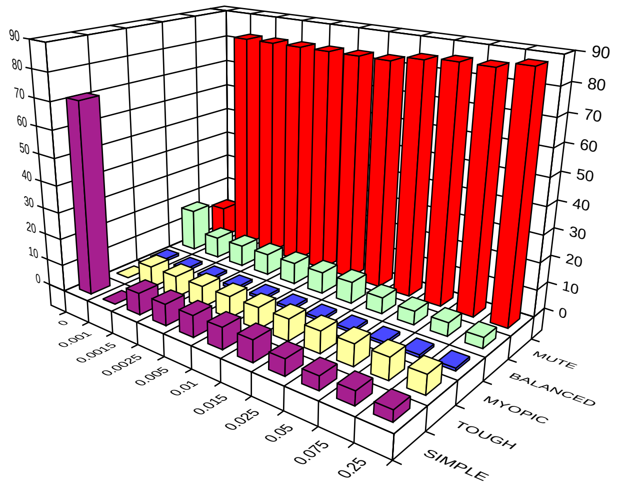


Figure 5: Coordination Strategies chosen by COLLAGE M1 for G2

5.2.2 When not to Learn

Synthetic grammar "G3" is an interesting case study. We trained COLLAGE on the G3 domain in both Mode 1 and Mode 2 and tested them on 100 runs for different coordination strategies at various communication costs. We found that `tough` coordination strategy performs slightly better than COLLAGE (see Figure 7). Upon closer examination of the problem instances, it was noted that `tough` was the best performer in 81% of the instances and other coordination strategies did better in the rest of the 19%. COLLAGE learns to choose the right coordination strategy in all the 100 instances. However, the agents require additional units of communication of meta-level information to form the global situation vector and decide that `tough` is the strategy of choice (in most cases). The lesson we learn from this grammar is that, if there is an overwhelming favorite for best performance in the family of strategies, then it may not pay to use COLLAGE to determine

[6]These environments are entirely different from those for the previous experiments. What is a "high" communication cost obviously depends on the environments.

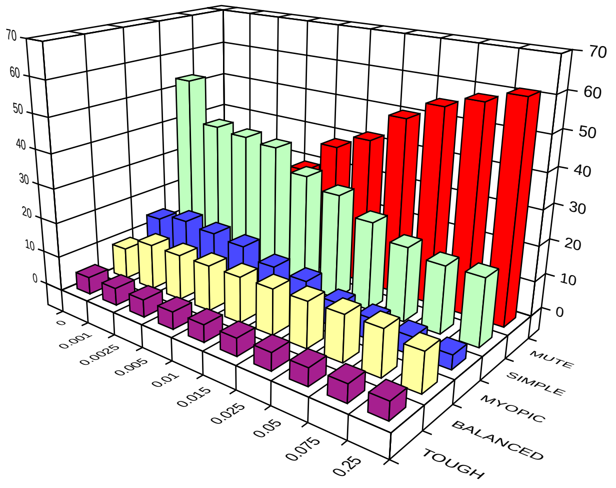


Figure 6: Coordination Strategies chosen by COLLAGE M2 for G2

the best performer through additional situation communication. Sticking to the favorite without awareness of the nonlocal situation may yield as good a performance. However, if the few cases that warrant the choice of another strategy give far superior performance, then the gains from choosing a strategy can more than compensate for the additional communication. This, however, was not the case in environments produced by grammar G3.

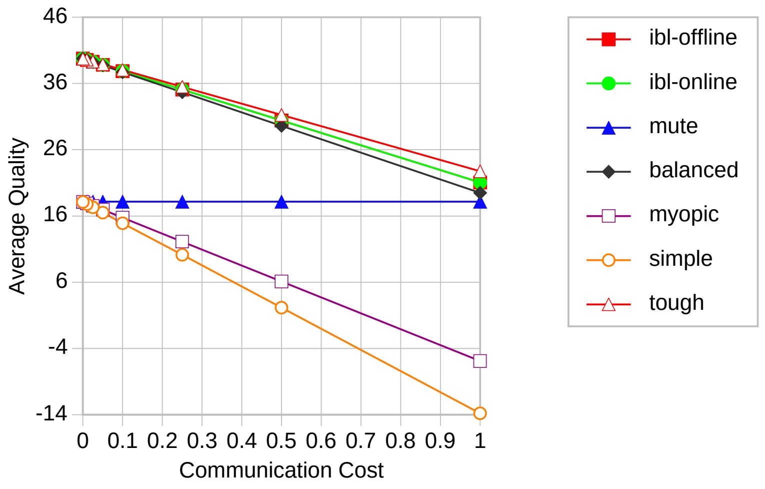


Figure 7: Average Quality versus Communication Cost for Grammar G3

5.3 Discussion

COLLAGE chooses an appropriate coordination strategy by projecting decisions from past similar experience into the newly perceived situation. COLLAGE agents performed better than those using a single coordination strategy across all the 100 instances in all domains we experimented with, except G3. In these domains, the cost incurred by additional communication for detecting global situation is offset by the benefits of choosing a coordination strategy based on globally grounded learned knowledge. Domain G3, however, is distinguished by the fact that there is little variance in the choice of the best coordination strategy and it was almost always `tough`. This highlights the fact that learning is especially beneficial in more dynamic environments.

6 Conclusion

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 dynamically choose a coordination strategy based on the problem solving situation. In this paper, we presented the COLLAGE system, that uses meta-level information in the form of abstract characterization of the coordination problem instance to learn to choose the appropriate coordination strategy from among a class of strategies. In the first phase of coordination, agents exchange local situations that are abstractions of the their views of the problem instances and form a global situation. They use this global situation to guide their learning and the subsequent exploitation of the learned coordination knowledge. Our experiments provide strong empirical evidence of the benefits of learning situation-specific coordination.

However, the scope situation-specific learning goes beyond learning coordination. The ideas presented here are relevant to the more general problem of learning *cooperative control* in a multi-agent system. Effective cooperation in a multi-agent system requires that the global problem-solving state influence the local control decisions made by an agent. We call such an influence cooperative control[20]. Coordination strategies are a form of cooperative control. An agent with a purely local view of the problem solving state cannot learn to make effective problem solving control decisions about coordination because these decision may have global implications. We thus advocate the need for learning *globally-situated* local cooperative control knowledge. An agent associates appropriate aspects of the global problem solving state with the cooperative control knowledge that it learns. In cooperative systems, an agent can "ask" other agents for any information that it deems relevant to appropriately situate its local control knowledge. However, as mentioned previously, an agent cannot utilize the entire global problem solving state even if other agents are willing to communicate all the necessary information because of the limitations on its computational resources, representational heterogeneity[7] and phenomenon like distraction[19][8]. This leads us to the importance of communicating only relevant information at suitable abstractions to other agents. We call such meta-level information a *situation*. An agent thus needs to associate appropriate views of the global situation with the knowledge learned about effective control decisions. We call this form of knowledge *situation-specific control*. We have looked at the relevance of the theory of situation-specific learning in the context of learning other forms of problem solving control in cooperative multi-agent systems[25, 26].

Moreover, COLLAGE demonstrates the utility and viability of learning coordination in complex, realistic multi-agent systems. The domains in this work are realistic and coordination strate-

[7]Representational heterogeneity implies that the agents may be using incompatible representations of their domain and control knowledge, eliminating the possibility of one agent knowing about the problem solving state of another agent at very deep level of detail. For example, a rule-based agent cannot understand the workings of a case-based agent even if that agent provided all the details about its internal state (similarity metric used, most similar case retrieved and adapted etc.).

[8]The phenomenon of distraction arises when incorrect or irrelevant information provided by another agent with weak constraint knowledge could lead the receiving agent to explore along unproductive directions[19].

gies are complex. Some of the work in multi-agent robotic systems[22] also deals with realistic and complex systems but that work is primarily concerned with homogeneous agents. The work in this paper deals with heterogeneous agents. They need to speak a common language to the extent of being able to understand TÆMS representations. They can organize their local computations and scheduling in any manner they desire.

The work here is exciting to us as much for the kinds of questions and challenges it uncovers as it is for the results obtained so far. Some of the important questions that it raises are “What is a suitable level of abstraction for a situation and how can one determine it?”, “How specific is the instance base to a particular domain and how much of it can be transfered across domains?”.

One important concern about COLLAGE is its scalability. As the number of coordination alternatives become large in number, the learning phase could become computationally very intensive and the instance-base size could increase enormously with respect to Mode 2. We are looking at how to integrate methods for progressively refining situation vectors such as those in [34], ways to organize the instance-base to access and detect regions where there is insufficient learning and also ways to do more directed experimentation during learning rather than randomly sampling the problem space.

In COLLAGE, all the agents form identical instance-bases. We could as well have done with one designated agent forming the instance-base and choosing the coordination strategy. However, our configuration was set up with a more general scheme in mind. Instead of all agents choosing the same coordination strategy, they can choose in a pairwise or a group-wise manner so that a subset of the agents coordinate to choose the same strategy. This will lead to different case-bases at different agents and an agent may have more than one case-base if it is a part of more than one group. This leads us to another scalability issue: the number of agents. If there are a large number of agents, then common situation vectors may lose “too many” details about the situations. Pairwise or group-wise coordination may be a better option. However, we have to deal with issues such as inconsistent and conflicting knowledge among the case-bases, formation of appropriate groups, and different amounts of learning for different groups.

References

[1] R. Sutton A. Barto and C. Watkins. Learning and sequential decision making. In M. Gariel and J. W. Moore, editors, *Learning and Computational Neuroscience*, Cambridge, MA, 1990. MIT Press.

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

[3] Robert H. Crites and Andrew G. Barto. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*, MIT Press, Cambridge, MA, 1996.

[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. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 217–224, Washington, July 1993.

[6] 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.

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

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

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

[10] A. Garland and R. Alterman. Multi-agent learning through collective memory. In *Proceedings of the 1996 AAAI Spring Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems*, Stanford, CA, 1996.

[11] 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.

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

[13] Itzhak Gilboa and David Schmeidler. Case-based Decision Theory. *The Quaterly Journal of Economics*, pages 605–639, August 1995.

[14] John J. Grefenstette. The Evolution of Strategies for multi-agent environments. *Adaptive Behavior*, 1(1):65–89, 1992.

[15] Thomas Haynes and Sandip Sen. Learning cases to resolve conflicts and improve group behavior. Submitted, 1996.

[16] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, 1975.

[17] J. H. Holland. Properties of bucket brigade algorithm. In *First International Conference on Genetic Algorithms and their Applications*, pages 1–7, Pittsburgh, PA, 1985.

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

[19] V. R. Lesser and L. D. Erman. Distributed interpretation: A model and an experiment. *IEEE Transactions on Computers*, C-29(12):1144–1163, December 1980.

[20] Victor R. Lesser. A retrospective view of FA/C distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1347–1362, 1991.

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

[22] M. J. Mataric. Learning to behave socially. In *Proceedings of the Third International Conference on Simulation of Adaptive Behavior (SAB-94)*, 1994.

[23] M. V. Nagendra Prasad, Keith S. Decker, Alan Garvey, and Victor R. Lesser. Exploring Organizational Designs with TAEMS: A Case Study of Distributed Data Processing. In *Proceedings of the Second International Conference on Multi-Agent Systems*, Kyoto, Japan, December 1996. AAAI Press.

[24] M. V. Nagendra Prasad and V. R. Lesser. Learning Situation-Specific Coordination in Generalized Partial Global Planning. In *1996 AAAI Spring Symposium on Adaptation, Co-evolution and Learning in Multi-agent Systems,* Stanford, CA, 1996. AAAI Press.

[25] M. V. Nagendra Prasad, V. R. Lesser, and S. E. Lander. Cooperative learning over composite search spaces: Experiences with a multi-agent design system. In *Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, Portland, Oregon, August 1996. AAAI Press.

[26] M. V. Nagendra Prasad, V. R. Lesser, and S. E. Lander. Learning organizational roles in a heterogeneous multi-agent system. In *Proceedings of the Second International Conference on Multi-Agent Systems*, Kyoto, Japan, December 1996. AAAI Press.

[27] J. S. Rosenschein and G. Zlotkin. Designing conventions for automated negotition. *AI Magazine,* pages 29–46, Fall 1994.

[28] T. Sandholm and R. Crites. Multi-agent reinforcement learning in the repeated prisoner's dilemma. to appear in Biosystems, 1995.

[29] Sandip Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 426–431, Seattle, WA, July 1994. AAAI.

[30] 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.

[31] S. S. Sian. Extending learning to multiple agents: issues and a model for multi-agent machine learning. In *Proceedings of Machine Learning - EWSL 91*, pages 440–456, Springer-Verlag, 1991.

[32] B. Silver, W. Frawely, G. Iba, J. Vittal, and K. Bradford. A framework for multi-paradigmatic learning. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 348–358, 1990.

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

[34] T. Sugawara and V. R. Lesser. On-line learning of coordination plans. In *Proceedings of the Twelfth International Workshop on Distributed AI*, Hidden Valley, Pa, May 1993.

[35] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[36] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, 1993.

[37] G. Weiss. Some studies in distributed machine learning and organizational design. Technical Report FKI-189-94, Institut für Informatik, TU München, 1994.