The Use of Meta-level Information in Learning Situation-Specific Coordination*

M V Nagendra Prasad and Victor R Lesser
Department of Computer Science
University of Massachusetts, Amherst, MA 01003.
{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 algorithm that endows agents with the capability to choose the appropriate coordination algorithm from a set of available coordination algorithms based on meta-level information about their problem solving situations. We present empirical results that strongly indicate the effectiveness of the learning algorithm.

1 Introduction

Coordination is the act of managing interdependencies in a multi-agent system[Decker & Lesser, 1995]. 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 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[Durfee & Lesser, 1988; Decker & Lesser, 1995; Nagendra Prasad *et al.*, 1996]. For example, when the agents are under severe time pressure and the load of the activities at the agents is high, it is often difficult for them to rearrange their local activities so that they can exploit the results generated by other agents or to generate results that the other agents can exploit. In these situations, sophisticated agent coordination strategies may not pay-off. 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 an 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 decides the types of information to be exchanged and the kind of reasoning about local and non-local activities the agents perform to achieve coherent activity. 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 rest of the paper is organized as follows. After placing our work in context, we briefly review the TÆMS task structure representation for coordination problems. We then describe our learning algorithm that learns to choose among three coordination modes of different levels of sophistication. We then present some of our experimental results and conclude.

2 Related Work

Much of the literature in multi-agent learning relies on reinforcement learning and classifier systems as learning algorithms. In Sen, Sekaran and Hale[Sen, Sekaran, & Hale, 1994] and Crites and Barto[Crites & Barto, 1996], the agents

*This material is based upon work supported by the National Science Foundation under Grant Nos. IRI-9523419. The content of this paper does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

do not communicate with one another and an agent treats the other agents as a part of the environment. Weiss[Weiss, 1994] uses classifier systems for learning appropriate multi-agent hierarchical organization structuring relationships. In Tan[Tan, 1993], the agents share perception information to overcome perceptual limitations or communicate policy functions. In Sandholm and Crites[Sandholm & Crites, 1995] the agents are self-interested and an agent is not free to ask for any kind of information from the other agents. The MAS that this paper deals with contain 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, an agent tries to resolve the uncertainty about other agents' activities by communicating meta-level information to resolve the uncertainty to the extent possible (there is still the environmental uncertainty that the agents cannot do much about). In cooperative systems, an agent can "ask" other agents for any information that it deems relevant to appropriately situate its learned local coordination knowledge. Agents sharing perceptual information as in Tan[Tan, 1993] or bidding information as in Weiss[Weiss, 1994] do not make explicit the notion of situating the local control knowledge in a more global abstract situation. The information shared is weak and they are studied in domains such as predator-prey[Tan, 1993] or blocks world [Weiss, 1994] where the need for sharing meta-level information and situating learning in it is not apparent.

Sugawara and Lesser[Sugawara & Lesser, 1993] 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 relies on deep domain knowledge and agent homogeneity assumptions to learn to progressively refine situations based on failure-driven explanation and comparative analysis of problem solving traces. They test the theory on 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.

3 Task Analysis, Environment Modeling, and Simulation

3.1 TAEMS

The TÆMS framework (Task Analysis, Environment Modeling, and Simulation) [Decker & Lesser, 1993] represents coordination problems in a formal, domain-independent way. A TÆMS model of a task environment specifies what actions are available to agents and how those actions relate to one another and to the performance of the system as a whole. A coordination problem instance is represented as a *task group* $\mathcal{T}$. Each task group has an arrival time $\mathrm{Ar}(\mathcal{T})$, and a deadline $\mathrm{D}(\mathcal{T})$. A task group, consisting of a set of computationally related actions, is represented by a directed acyclic graph. The quality of a task T at a particular time t ($Q(T,t)$) is a function of the quality of its subtasks (in this paper this function is either minimum(AND-like) or maximum(OR-like)). *Quality* is used as a catch-all term representing acceptability characteristics like certainty, precision completeness, other than temporal characteristics. Leaf tasks are called *methods* M and they represent domain actions, such as executing a blackboard knowledge source, running an instantiated plan, or executing a piece of code with its data. Executable methods have base level quality and duration. 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.

Besides tasks/subtask relationships, there can be other interrelationships between tasks in a task group[Decker & Lesser, 1993]. In this paper, we will be dealing with two such interrelationships:

- `facilitates` relationship or soft interrelationship: If the results of execution of Task A are available for Task B before it starts executing, then Task B will have increased quality and/or decreased duration.
- `enables` relationships or hard interrelationship: Task A must be executed before Task B can be executed.

A design-to-time scheduling[Garvey & Lesser, 1993] algorithm heuristically enumerates a promising subset of quality and time trade-offs to produce schedules that maximize quality given the deadlines. In a cooperative multi-agent system, the goal of the agents is to work together to produce the highest possible quality for as many task groups as possible.

3.2 Environment-Specific Coordination Mechanisms

In order to bring to bear different collections of coordination mechanisms for different multi-agent problem-solving environments, we use the Generalized Partial Global Planning (GPGP) approach[Decker & Lesser, 1995]. GPGP consists of several coordination mechanisms, each of which notices certain features in its local partial view of the task structures of other agents and their relationship with its own task structure and responds by taking certain communication or information gathering actions, or by proposing new commitments to other agents. In GPGP, a coordination strategy can be derived by activating a subset of the coordination mechanisms. Specifically we will be investigating the effect of three coordination strategies:

Balanced (or dynamic-scheduling): 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. "Rough commitments" are a form of tacit social contract between agents about the completion times of their tasks.

The latter two coordination strategies are the alternatives normally used in the distributed data processing domain[Nagendra Prasad *et al.*, 1996]. [Decker & Lesser, 1995] proposed `balanced` as a sophisticated strategy that exploits a number of mechanisms to achieve coordination.

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[Aha, Kibler, & Albert, 1991] originally proposed for supervised classification learning. We, however, use the IBL-paradigm for unsupervised learning of decision-theoretic choice.

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 how to activate 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 quasi-online algorithm. In the initial stages it just explores and in the later stages it just exploits the learned information. A more typical online learning algorithm interleaves exploration and exploitation. Studying COLLAGE in this kind of a setup is high on our agenda of future work.

At end of each run of the coordination episode with a selected coordination strategy, the performance of the system is registered. This is represented as a vector of four performance measures: total quality, number of methods executed, number of communications, and termination time. 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 $\{situation, coordination\_strategy, performance\}$ 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.

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:
(a) 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 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.
(b) 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.

(c) The third and fourth components represent an approximation of the effect of detecting hard coordination interrelationships 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`.
(e) 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.
(f) The sixth component represents the load. It is obtained as the ratio of execution time under actual deadline and the execution time under no time pressure. 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.

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. However, more sophisticated models would need to take into consideration these factors too. We view this a one of our future directions of research.

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. We can have a number of composition functions but 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 all the corresponding local situation vector components.

For example, one 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`[Decker & Lesser, 1995] 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`[Decker & Lesser, 1995] do better in this scenario.

Note, however, that in most situation vectors, these tradeoffs 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. Hence, hand-coding the strategies by a designer is not a practical alternative.

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[Gilboa & Schmeidler, 1995]. 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 function 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 (in our experiments, the neighborhood radius was heuristically set to 0.05). We observed that such an averaging process in a neighborhood around the present situation vector was more robust than taking the nearest neighbor. 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 experiments on learning coordination were conducted in the domain of distributed data processing[Nagendra Prasad *et al.*, 1996]. This 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 the interrelationships between the tasks at these centers. [Nagendra Prasad *et al.*, 1996] developed a graph-grammar-based stochastic task structure description language and generation tool for modeling task structures arising in a domain such as this. They present the results of empirical explorations of the effects of varying deadlines and crisis task group arrival probability. Based on the experiments, they 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).

Experiments

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. Figure 1 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 the 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 versus, pairwise tests at each communication cost). This test revealed significant differences between each of the learning algorithms (both Mode 1 and Mode II) and each of 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[1]. 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. Figure 2 gives an example of situation-specific choice of coordination strategies for Mode 1 learner in 100 test runs when the crisis task group probability was 1.0. The Z-axis shows the number of times a particular coordination strategy was chosen in the 100 runs at a particular communication cost. X-axis shows the communication cost and the Y-axis shows the coordination strategy.

When not to learn!

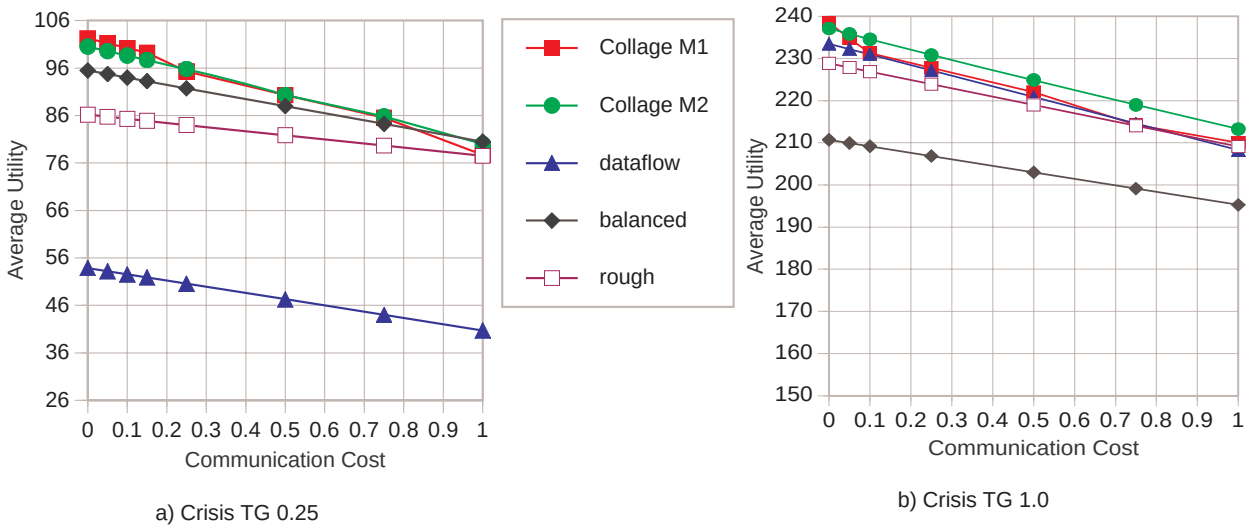
In order to test COLLAGE on interesting scenarios with a range of characteristics, we created a number of "synthetic domain theories" using graph grammar formalisms[Nagendra Prasad *et al.*, 1996]. Space limitations do not permit us to discuss all the results but we would like to briefly talk about a very interesting result seen in the synthetic grammar "G3". 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 a coordination strategy called `tough`[2] coordination performs slightly better than COLLAGE. 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 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.

5.2 Discussion

COLLAGE chooses an appropriate coordination strategy by projecting decisions from past similar experience into the newly perceived situation. COLLAGE agents performed better than using any single coordination strategy across all the

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

[2] Definition of the `tough` coordination strategy is not important here.



a) Crisis TG 0.25

b) Crisis TG 1.0

Figure 1: Average Quality versus Communication Cost

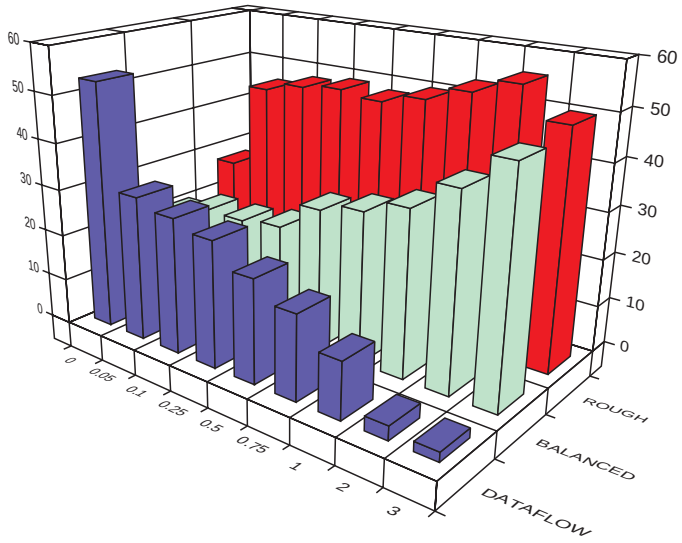


Figure 2: Strategies chosen by COLLAGE M1 for Crisis TG Probability 1.0 at various communication costs

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 best coordination strategy and the best coordination strategy 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 situation. In this paper, we presented a learning system, called COLLAGE, 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. Our experiments provide strong empirical evidence of the benefits of learning situation-specific coordination.

However, one important limitation of 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 [Sugawara & Lesser, 1993], 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 algorithm, they can choose pairwise or group-wise. 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.

Acknowledgments

We would like to thank Keith Decker and Alan Garvey for their help and input during the course of this work. Thanks also go to Daniel Neiman, Mike Chia and the anonymous referees for their comments and feedback on draft versions of this paper.

References

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

[Crites & Barto, 1996] Crites, R. H., and Barto, A. G. 1996. Improving elevator performance using reinforcement learning. In *Advances in Neural Information Processing Systems 8*.

[Decker & Lesser, 1993] Decker, K. S., and Lesser, V. R. 1993. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 217–224.

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

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

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

[Gilboa & Schmeidler, 1995] Gilboa, I., and Schmeidler, D. 1995. Case-based Decision Theory. *The Quaterly Journal of Economics* 605–639.

[Nagendra Prasad *et al.*, 1996] Nagendra Prasad, M. V.; Decker, K. S.; Garvey, A.; and Lesser, V. R. 1996. 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: AAAI Press.

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

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

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

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

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