Learning Procedural Knowledge to Better Coordinate*

Andrew Garland and **Richard Alterman**
Volen Center for Complex Systems
Brandeis University
Waltham, MA 02254
{aeg,alterman}@cs.brandeis.edu

Abstract

A fundamental difficulty faced by groups of agents that work together is how to efficiently coordinate their efforts. This paper presents techniques that allow heterogeneous agents to more efficiently solve coordination problems by acquiring procedural knowledge. In particular, each agent autonomously learns *coordinated procedures* that reflect her contributions towards successful past joint behavior. Empirical results validate the significant benefits of coordinated procedures.

1 Introduction

Research on groups of agents that work together is a large and growing field that covers topics such as autonomous robots, software agents, and smart objects. A fundamental difficulty faced by such agents is how to coordinate their efforts when they have overlapping objectives. This coordination problem is both ubiquitous and challenging, especially in environments where agents have limited knowledge about, and control over, other agents and the world around them.

This work is part of a line of research interested in groups of agents that interleave planning and execution [desJardins *et al.*, 1999; Grosz and Sidner, 1990; Levesque *et al.*, 1990; Durfee and Lesser, 1991; Decker and Lesser, 1992; Grosz and Kraus, 1996; Tambe, 1997]. In the present work, individual agents are motivated by personal objectives and do not reason about group-wide objectives or attempt to establish or maintain group-wide mental attitudes. The extent to which their activity can be successful depends on the degree to which the individuals' objectives converge.

One approach to the coordination problem is to design agents to have common built-in knowledge about the group, such as knowledge of the planning or execution abilities of all agents. This common knowledge makes agents more likely to be able to efficiently solve coordination problems that occur at runtime. Unfortunately, for many domains there will continue to be coordination problems that lie outside the initial design because of the difficulty of foreseeing all possible interactions in complex, dynamic environments.

*This work was supported in part by the ONR (grants N00014-96-1-0440 and N00014-97-1-0604).

In this paper, an agent acquires knowledge about the environment and other agents from experience, supplementing any *a priori* common knowledge she might have. Thus, individual agents learn to better coordinate their actions so that the agents' future behavior more accurately reflects what works in practice.

The learning techniques are memory-based and a novel contribution is a technique to learn *coordinated procedures* based on past, possibly unplanned, successful joint behavior. These procedures are extracts of execution traces, which are the result of multiple planning sessions occurring at various times during the activity, and are composed around, and organized by, *coordination points* [Alterman and Garland, 2001]. Unexpected requests and responses allow an agent to acquire coordination knowledge about other agents, and constitute the building blocks of learned coordinated procedures.

This paper begins by outlining the framework within which the coordination problem is studied. The next section presents the key technical details that allow agents to learn coordinated procedures. Empirical results then demonstrate that coordinated procedures provide a statistically significant improvement in run-time performance and are used efficiently when planning. The paper ends with a discussion of related work.

2 Coordinating independent agents

This section describes the aspects of the system framework that are relevant to studying the coordination problem. The most noteworthy features are the autonomy of the agents, the distribution of both problem-solving knowledge and execution ability, and the role of communication as a coordination mechanism. Given these attributes, even seemingly simple problems create imposing hurdles to efficient coordination.

2.1 An example of a coordination problem

In the test-bed domain, called MOVERS-WORLD, the task is to move boxes from a house onto a truck or *vice versa*. MOVERS-WORLD has multiple agents of different types: some are "lifters" and some are "hand-truck operators". The agents do not know their type or even have an internal representation of the concept of type. Most actions are type-specific, but all agents are able to move and communicate. The duration of conversations between two agents varies

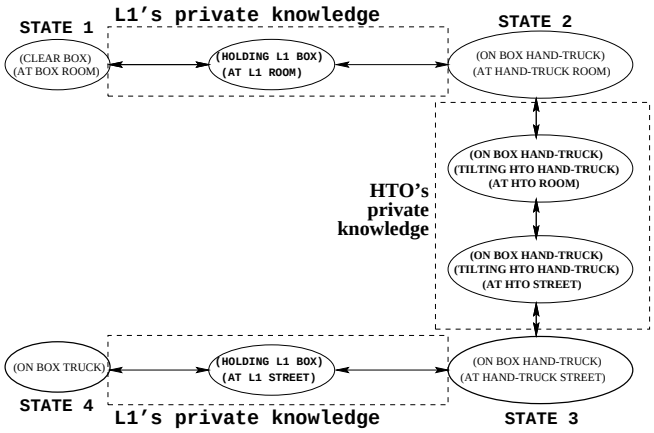

Figure 1: The distribution of problem-solving knowledge.

based on the content of the dialog. The agents have no built-in planning knowledge about the execution abilities of agents of other types. Each agent has the autonomy to decide which goal(s) to work on at any time.

The rest of this subsection discusses how coordination problems arises in MOVERS-WORLD. The plans a lifter can generate and the plans a hand-truck operator can generate are contrasted in the context of moving a single box onto a truck. By expanding the problem to include more boxes, the difference in planning ability makes achieving optimal coordination impossible and achieving even near-optimal behavior unlikely.

A solution path for a simple situation where a lifter (L1) and a hand-truck operator (HTO) could work together to get a box onto the truck is shown in Figure 1. Over the course of the solution, the box "moves" through eight different states (represented by ovals listing the salient predicates). Only L1 knows how to transform the box from state 1 to state 2. Only HTO knows how to get from state 2 to state 3. And only L1 knows how to get the box from state 3 to state 4.

L1 cannot generate a plan to match this solution path on her own since L1 has no planning knowledge of the hand-truck or of HTO's capabilities. Backward chaining can identify state 3 as a precursor to state 4 and forward-chaining can identify state 2 as a successor to state 1. However, L1 cannot distinguish the pair of states (state 2, state 3) that is relevant to this solution path from the many other pairs of states that are not relevant to any solution path.

Coordination becomes more of an issue when the agents want to move two boxes onto the truck in as little time as possible. Optimal performance involves no communication whatsoever and would take 244 ticks of a simulated clock. However, without communication, L1 would never construct a plan to load BOX1 onto the hand-truck! L1's expectation at the outset would be to move both boxes to the truck by carrying them; HTO's expectation is that both boxes would be moved via the hand-truck. Furthermore, neither knows that the other is working on the same two goals.

```
Ticks   1 to  15: HTO and L1 converse
  "L1, would you help me achieve (ON BOX1 HANDTR)?"
  "HTO, I'll help, but you'll have to wait a bit."
Ticks  16 to  45: <LIFT L1 BOX1>
Ticks  46 to  80: <LOAD L1 BOX1 HANDTR>
Ticks  81 to 100: <TILT-HANDTR HTO HANDTR>
Ticks  81 to 112: <LIFT L1 BOX2>
Ticks 101 to 125: <PUSH-HANDTR HTO HANDTR STREET>
Ticks 113 to 152: <CARRY L1 BOX2 STREET>
Ticks 126 to 145: <STAND-HANDTR HTO HANDTR>
Tick         146: HTO trying to contact L1 ...
Ticks 153 to 167: HTO and L1 converse
  "L1, would you help me achieve (ON BOX1 TRUCK)?"
  "HTO, I'll help, but you'll have to wait a bit."
Ticks 168 to 202: <LOAD L1 BOX2 TRUCK>
Ticks 203 to 237: <UNLOAD L1 BOX1 HANDTR>
Ticks 238 to 272: <LOAD L1 BOX1 TRUCK>
```

Figure 2: A near-optimal solution to a coordination problem.

The closest to optimal that is possible given baseline coordination abilities is 272 ticks as shown in Figure 2. That solution requires that L1 agrees to help HTO on both occasions and that L1 correctly adapts her plan. While none of these is unlikely independently, neither agent possesses enough knowledge to reliably act in this way. As the complexity of the problems increases by including other agents and increasing the number of boxes, there is an exponential increase in the number of decisions that all must be made "correctly" for the community to perform even this close to optimal.

Coordination problems in this domain are not solely a consequence of the distribution of planning knowledge. Even if both agents had common goals and planning knowledge, there is ambiguity about the order on which to work on the goals. Also, when there is uncertainty about the outcome of actions, they may have different preferences among alternative solutions.[1] Finally, agents may have different beliefs about the current state of the world, leading to different beliefs about the best course of action.

2.2 Communication as a coordination mechanism

In terms of the coordination problems faced by the agents, a central feature of this system is that communication, cooperation, and coordination are shaped by the autonomy of the agents. Agents do not communicate at planning time; they plan independently, act independently, and only communicate when necessary to establish cooperation or to maintain coordination. Each problem includes some goal(s) that can only be solved by agents that work together, so communication is an essential part of the community activity.

Communication happens at *coordination points*, which are defined as points in the activity where an actor cannot progress without the assistance of someone else. If all actions are reversible and the goals of the agents do not conflict, conversing at individual coordination points at the time when they arise is sufficient to ensure that the activity will be completed.

[1] The fact that actions are not deterministic will not be addressed in detail in this paper.

Agents' decision-making strategies are based upon personal, rather than group-wide, objectives. If an agent is willing to cooperate, she may be unable to construct a plan to do so; an agent who is unwilling or unable to assist can propose an alternative that the original requester may now contemplate adopting. After agreeing to cooperate, an agent can "opt out" at any time, without obligation to notify other agents.

When cooperation is first established during communication, the agents must determine how they will coordinate. Sometimes, nothing needs to be done to begin coordinating; more often, though, the requester will idle for several time steps — for example, if a lifter is not currently ready to lift the box. An agent will stop waiting if another agent initiates communication, either to establish a new agreement or to indicate progress on a current agreement (e.g., the other lifter now indicates she is ready to act). The agent will also stop waiting upon observing the completion of the request (e.g., the box appears on the hand-truck). Finally, if an agent is idle "too" long (i.e., longer than a pre-set threshold), the agent inquires about the status of her request (possibly discovering that the other agent has opted out).

Communication is the only mechanism whereby agents can check if they are working on the same goals since there are no global structures, such as blackboards, available. While observation is sufficient to engineer the exit from coordination problems, the agents are not assumed to possess the common goals and knowledge of each other required in order to solve coordination problems without any communication [cf. Genesereth *et al.*, 1986; Huber and Durfee, 1995].

3 Leveraging past experience

This section will describe the case-based reasoning [Kolodner, 1993] techniques that individual agents use to acquire and use coordinated procedures in order to better coordinate. A guiding principle of these techniques is that memory should be organized around coordination points. There is no communication between the agents during the learning process; the memories are created and maintained by each agent independently.

3.1 Learning coordinated procedures

Coordination points influence three facets of the learning process. Most importantly, coordination devices that represent past successful joint achievement of coordination points form the skeleton of future plans. Next, memories are primarily indexed based on expected future requests in order to make the memory more likely to be recalled during communication. Third, coordination points influence the determination of state-based secondary indices.

Figure 3 contains pseudo-code for LEARNCOORDINATEDPROCEDURE, which is the method by which agents transform experience into memories. Coordinated procedures are derived from execution traces, which are quite noisy because actions and requests can fail or be ineffective for other reasons. There are many details involved in the process of transforming such noisy data into something suitable for learning that have not been addressed in prior work on procedural learning. Space prevents covering many of them here; the interested reader is directed elsewhere [Garland, 2000] for more about cleaning, segmenting, and removing inefficiencies from execution traces. There are two other non-trivial procedures, KEEP and ADDCASEBASEENTRY, whose pseudo-code is not given but whose important aspects will be described below.

```
LEARNCOORDINATEDPROCEDURES (executionTrace) ≡
  cleanTrace ← CLEAN(executionTrace)
  segmentGoalsPairs ← SEGMENT(cleanTrace)
  forall ⟨segment, goals⟩ in segmentGoalsPairs
    coreSegment ← REMOVEINEFFICIENCIES(segment, goals)
    STORECOORDINATEDPROCEDURE(coreSegment, goals)

STORECOORDINATEDPROCEDURE (coreSegment, goals) ≡
  procedure ← LIST()
  times ← LIST()
  requests ← LIST()
  forall step in REVERSE(coreSegment)
    if KEEP(step, procedure)
      procedure ← PUSH(step, procedure)
      times ← LIST()
      requests ← LIST()
    times ← PUSH(STARTTIME(step), times)
    if step is an agreement to achieve a coordination point
      requests ← PUSH(step, requests)
  if HEAD(procedure) is not a member of requests
    ADDCASEBASEENTRY(procedure, goals, null, times)
  forall r in requests
    rTimes ← REMOVELATERTHAN(times, STARTTIME(r))
    ADDCASEBASEENTRY(procedure, goals, r, rTimes)
```

Figure 3: Algorithms to learn coordinated procedures.

STORECOORDINATEDPROCEDURE converts an execution trace segment into a coupling of a coordinated procedure and its indexing information. The primary tasks of STORECOORDINATEDPROCEDURE are:

1. Construct a coordinated procedure by summarizing and optimizing the execution trace segment.
2. Determine the set of expected requests that the agent could use as retrieval cues in the future.
3. For each entry to be added to the case base, determine the set of states in which the agent should consider retrieving the entry.

The motives for summarizing an execution trace segment are: the particular time at which subordinate goals were achieved at runtime may be misleading; the stored plan will be more easily adapted in the future (at the cost of regenerating the original action if it is needed again); and the state-based indices for case-base entries are generalized each time the same procedure is stored under different indices. Optimizations allow agents to improve upon, rather than just repeat, the way in which some coordination points are jointly achieved.

Storing procedures under several indices is a good heuristic when agents do not share indexing information and there is uncertainty about the setting at the outset of cooperation. In addition, the state of the world at the start of the first step of

the coordinated procedure is not the only reasonable benchmark for determining future settings in which the procedure will be effective. Alternative indexing states are provided by the (removed) steps in the execution trace summary that precede the first kept step.

The pseudo-code in Figure 3 sets forth how STORECOORDINATEDPROCEDURE records indexing information while adding actions to the coordinated procedure. A call to the KEEP function determines which actions from the execution trace segment are added to the procedure. For each step, whether it is kept or not, the time at which it was started is added to a list of indexing times and agreements are added to a list of expected requests. When a step is added to the procedure, these lists are reset.

After the coordinated procedure has been determined, it is stored into memory without any reference to an expected request (unless the first step in the plan is an expected request). Next, the procedure is stored again for each expected request so that the request is part of the primary storage index. Indexing in this way facilitates retrieving coordinated procedures during conversations. The means by which coordination points influence the state-based secondary indices is less direct, and is determined by ADDCASEBASEENTRY.

ADDCASEBASEENTRY makes representational changes to the procedure and maintains the structure of the underlying case base. In terms of the organization of memory, a notable division is that STORECOORDINATEDPROCEDURE computes the relevant times to check the state of the world; ADDCASEBASEENTRY converts those states into concrete case-base indices. For this work, the secondary indexing scheme is influenced by the top-level goals and the expected requests of the entry. The literals from these are culled and then all relevant predicates in the state relating to the literals are identified. Relevancy is determined from the surface features of the environment, rather than from analysis of the stored plan [cf. Hammond, 1990; Veloso and Carbonell, 1993].

The most interesting aspect of KEEP is the treatment of coordination points. The principle summarization criteria is to remove actions that are planner-reconstructible — this produces a skeleton of coordination mechanisms that is fleshed out by the essential individual actions needed to support the achievement of coordination. An important characteristic of summarization is that it never removes: (1) coordination mechanisms for agreements, (2) coordination points that correspond to unexpected requests, or (3) actions that end a shift between goals. Requests for joint action require special handling to ensure that the initiator only keeps the joint action and the assistant only remembers to expect a request.

The heuristic optimizations currently implemented are intended to eliminate some conversations altogether. For example, based on past experience, a lifter learns to load the hand-truck without being explicitly told to do so. Likewise, the hand-truck operator learns to expect the lifter to load the hand-truck without the hand-truck operator's guidance. The risk of the heuristics is that if agents do not recall compatible memories, time and effort may be wasted.

One optimization rule (implemented in KEEP) is that an agent who agreed to a request removes the corresponding coordination mechanism from the coordinated procedure. This guideline reflects an optimistic belief that the agent knows the right time to accomplish the request without being specifically asked. A second guideline, dual to the first, is part of the representational changes performed by ADDCASEBASEENTRY. For an initiator of a request, the procedure passed into ADDCASEBASEENTRY contains a coordination mechanism that will prompt the agent to establish the same request again in the future. This mechanism is heuristically converted into an (optimistic) expectation that the request for service will be satisfied without a direct request. These two optimizations currently only relate to requests for service and not to requests for joint action, which require more precise timing.

```
Time         Action                          Outcome

  1  agreed with L2 to ...                  Summarized
 21  <LIFT-TOGETHER XLBOX1>                 Summarized
 91  HTO asked to (ON XLBOX1 HANDTR3)       Optimized
105  L2 agreed to ...                       Summarized
126  <LOAD-TOGETHER XLBOX1 HANDTR3>         Kept
161  <LIFT MBOX0>                           Summarized
191  <CARRY MBOX0 STREET1>                  Summarized
231  <LOAD MBOX0 TRUCK3>                    Kept
266  HTO asked to (ON XLBOX1 TRUCK3)        Optimized
279  L2 agreed to ...                       Summarized
300  <UNLOAD-TOGETHER XLBOX1 HANDTR3>       Summarized
335  L2 agreed to ...                       Summarized
353  <LOAD-TOGETHER XLBOX1 TRUCK3>          Kept

L1 adding case-base entry MEM75
Procedure: (<LOAD-TOGETHER ?L1-177 ?L1-174>
            <LOAD ?L1-175 ?L1-176>
            <LOAD-TOGETHER ?L1-177 ?L1-176>)
Top-level goals:  ((ON ?L1-177 ?L1-176)
                   (ON ?L1-175 ?L1-176))
Request: NIL
State indices based on ticks 1, 21, 91, 105, 126

L1 adding case-base entry MEM76 derived from MEM75
Request: (LIFT-TOGETHER ?L1-177) by ?L1-180
State indices based on tick 1

L1 adding case-base entry MEM77 derived from MEM75
Request: (ON ?L1-177 ?L1-174) by ?L1-173
State indices based on ticks 1, 21, 91
```

Figure 4: Three entries for a single coordinated procedure.

Figure 4 shows some (lightly edited) output when agent L1 is adding multiple case-base entries for the same coordinated procedure. In this example, the agent is creating one top-level entry and two request entries. The performance that led to learning this procedure was exceptional (33% fewer ticks than average) and was chosen for illustrative purposes; execution traces are normally much more chaotic. The only failure, which is not shown in the figure, is an attempt at time 71 by the two lifters to jointly carry the box to the street.

3.2 Acting from shared past experience

Acting from shared past experience can lead agents to coordinate more efficiently. For example, when an agent receives a familiar request, she can retrieve a plan which anticipates future requests rather than merely creating a plan to satisfy that single request. When different agents anticipate the same

points of coordination, they can coordinate more effectively for three reasons:

1. They will not waste time discussing alternatives that will prove to be unproductive.
2. They will not waste time negotiating over two viable alternatives.
3. In some cases, they can eliminate communication entirely.

Unfortunately, acting from past experience does not guarantee that agents will coordinate more efficiently. First of all, there may not be regularity in the types of coordination problems faced by the community. Second, agents will sometimes assess the same situation in disparate manners. This reflects both differences in experience between agents and the open-endedness of interpretation in general. Thus, it is important to store procedures so that different agents are likely to retrieve compatible memories in many situations. In this paper, compatible viewpoints on which past activities to recall develop when storage is guided by coordination points and surface features of the environment.

When planning, agents prefer coordinated procedures from similar past activities to plans constructed by the baseline planner. However, an agent does not search her case base when a planning session immediately follows the failure of a retrieved plan. For the experiments in the next section, coordinated procedures are recalled during planning slightly more than 20% of the time during the last problem-solving episode.

An agent measures the similarity of a case-base entry to the current setting by determining what percentage of the stored predicates can be made true given the possible mappings of literals in the current state to required role-fillers for the entry. For those entries that meet or exceed the current highest similarity (at least above a pre-set threshold of 0.50), the coordinated procedure stored in the entry is checked to see if it can be adapted to the current state of the world. If so, the current highest similarity is updated. Finally, the best of the best is selected from the entries with maximal similarity. There are various ways to determine the best of the best such as randomly or by the number of storages for the entry. The results in the next section rank the plans using the same heuristic as the baseline planner (most time-efficient) and break any remaining ties randomly.

In experiments conducted so far, the number of entries in each agent's case base has been less than 50, so neither storage nor retrieval was a bottleneck. In general, a more refined secondary indexing scheme, such as indexing by differences [Kolodner, 1983a,b] might be needed. Further refinements such as distinguishing between short and long term memory or selectively "forgetting" past cases [Smyth and Keane, 1995] may also be fruitful.

4 Empirical results and analysis

This section supports the claim that learning coordinated procedures leads agents to better coordinate with empirical studies from an implemented testbed. The experimental methodology takes into account the possible influence of sampling bias and ordering effects; each data point reported is the average of 600 trials.

In all of the experiments in this section, agents acquire coordination knowledge independently of learning coordinated procedures — agents learn more accurate probability estimates of the likelihood of success of possible actions. Agents use probabilities when planning from scratch, when deciding who to ask for help, when deciding whether to cooperate, and when adapting coordinated procedures to match the current problem setting. Having more accurate probability estimates, therefore, can cause agents to behave more efficiently. A learning structure, based on COBWEB [Fisher, 1987] trees, enables agents to increase the accuracy of probability estimates by generalizing past experiences interacting with the domain and other agents. For more details, see Garland [2000].

There are many ways to measure the performance of the agent community, such as the number of primitive actions attempted and the number conversations that occur. The best overall measure of community effort, however, is the number of ticks of a simulated clock that transpire during the course of the community solving the problem. The number of ticks measures both action and communication effort, in addition to time when the agents are idle for one reason or another (see Figure 2).

In order to measure the advantages of learning coordinated procedures under various conditions, the type of coordinated procedures learned and the initial ability of agents to solve coordination problems were varied. The possible types of coordinated procedures are:

Basic A basic coordinated procedure is not optimized and only achieves a single goal.

Improved An improved coordinated procedure is optimized and may achieve multiple goals (as in Figure 4).

A third possibility is to learn better probability estimates only and not learn any coordinated procedures.

The three initial levels of ability to solve coordination problems are:

Minimal The minimal amount of coordination knowledge is a predicate-based communication language based on goals, coordination points, and the ability to make unambiguous external references.

Basic Basic agents have additional planning knowledge of other agents and the ability to make goal-selection choices based on past experience. The coordination information built into basic agents was determined by analyzing the coordination knowledge implicitly acquired by basic coordinated procedures.

Expert Expert agents have additional hand-crafted goal-selection and coordination strategies, including an extension of the heuristic optimizations.

Note that basic and expert agents have abilities that exceed those assumed in Section 2.

Table 1 presents a tabular summary of the results of running the system for each of the combinations of initial ability and type of coordinated procedures learned. For each of these nine runs, the chart shows the number of ticks required to solve the tenth problem-solving episode faced by the community. The data in the first column indicate that improving

Built-in Coordination Ability	Type of Coordinated Procedures Learned		
	None	Basic	Improved
Minimal	846.3	747.2	623.5
Basic	789.2	743.5	640.6
Expert	687.4	641.5	594.1

Table 1: Comparison of the amount of community effort required for the tenth problem-solving episode.

the initial coordination ability of non-learning agents substantially reduces the number of ticks. Scanning across each row of Table 1 shows that, regardless of the initial coordination ability of the agents, learning basic coordinated procedures led to significant reductions in the number of ticks (with 99% confidence). Improved coordinated procedures are always significantly more effective than basic ones.

The second row of Table 1 warrants special attention. By design, agents with basic initial coordination abilities have access to the same goal-selection and planning knowledge as agents can learn over time through basic coordinated procedures. The fact that agents with basic abilities nonetheless benefit from learning such procedures shows that these pieces of knowledge are more useful as a unit, when retrieved from the case base, than when accessed separately.

The benefits of learning coordinated procedures are immediate. This is evinced by the learning curves, plotted with their 99% confidence intervals, shown in Figure 5. A comparison of curves MN and EN, two runs when agents are not learning coordinated procedures, points out that the impact of augmenting the initial ability to solve coordination problems does not diminish over time. A comparison of curves MI and MN, two runs when agents have minimal initial abilities to solve coordination problems, shows that the importance of learning coordinated procedures grows over time. Most importantly, comparing curves MI and EN reveals that learning coordinated procedures is more effective than initial expertise by the second problem-solving episode.

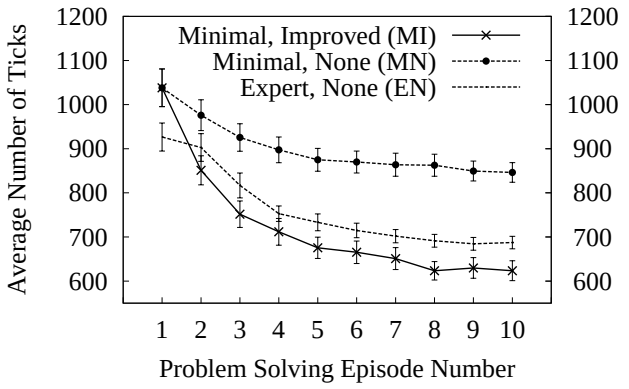

Figure 5: Comparing runtime effort.

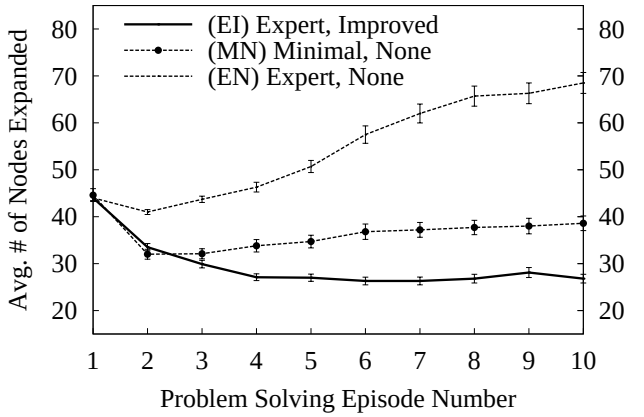

Figure 6: Comparing planning effort.

Another significant advantage of learning coordinated procedures is in controlling the amount of planner search. Figure 6 depicts planning effort, as measured by the average number of planning nodes expanded per call to the planner. The disparity between curves MN and EN provides evidence that initial expertise comes with a price — the increased amount of planning information can lead to a large increase in planner search. However, learning and acting from past experience controls planner search so effectively that experts who learn coordinated procedures (curve EI) expanded fewer planning nodes than agents with minimal initial knowledge who do not learn them (curve MN).[2]

Learning coordinated procedures leads to similar statistical improvement in the number of conversations, the number of ticks spent conversing, the number of attempted actions, and the number of successful actions. Furthermore, these results hold across a wide spectrum of possible goal-selection strategies, cooperation strategies, and communication costs [Garland, 2000].

Overall, the results clearly demonstrate that coordinated procedures are an effective resource for agents to learn to better coordinate their run-time activities. Learning coordinated procedures benefits agents regardless of the initial ability to solve coordination problems. There is leverage in storing and retrieving plans based on the surface features of the environment, rather than having access to similar information that is accessed at separate times. Finally, the techniques are very effective in preventing increased planner search.

5 Related Research

Learning coordination knowledge in multi-agent systems has been studied in frameworks with different assumptions than those made in this paper, such as in homogeneous systems [Sugawara and Lesser, 1998] and communication-free domains [Haynes and Sen, 1998; Ho and Kamel, 1998]. In a system with similar underpinnings, Prasad and Lesser [1999] implement a learning system that extends the generalized partial global planning [Decker and Lesser, 1992] architecture

[2]CPU time, another common measure of planning effort, is also reduced significantly by learning coordinated procedures.

by allowing the agents to choose a coordination mechanism (from a commonly known set of choices) based upon the results of training runs. All agents make the same choice because agents communicate their local viewpoints to all other agents to form a consistent global viewpoint and each agents records the same training data. By contrast, in this paper, each agent learns independently on the basis of their own experiences. Also, the techniques are applicable in the absence of built-in common knowledge and agents acquire procedures in addition to compatible viewpoints about how to coordinate activity.

In this work, procedural learning is based on run-time behavior, which differs from learning techniques based upon the output of planning sessions [Carbonell, 1983; Veloso and Carbonell, 1993; Laird *et al.*, 1986; Kambhampati and Hendler, 1992; Sugawara, 1995]. Reusing a plan derivation will not produce a sequence of actions to better solve a similar problem in the future if the derivation was deficient (due to the agent's incomplete knowledge). On the other hand, execution traces encapsulate the history of both planned and unplanned agent interactions with the domain. Consequently, procedures are learned that were not developed in a single (or multiple) planning histories. Thus, some coordinated procedures stored in memory can represent unplanned successes. Remembering examples of past successes differs from previous approaches to changing run-time behavior that have emphasized learning from failures [Hammond, 1990; Haynes and Sen, 1998].

References

Richard Alterman and Andrew Garland. Convention in joint activity. *Cognitive Science*, 25(4), 2001.

Jaime Carbonell. Derivational analogy and its role in problem solving. In *Proc. Third National Conference on Artificial Intelligence*, pages 64–69, 1983.

Keith S. Decker and Victor R. Lesser. Generalized partial global planning. *Intl. Journal of Intelligent and Cooperative Information Systems*, 1(2):319–346, 1992.

Marie E. desJardins, Edmund H. Durfee, Charles L. Ortiz, and Michael J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 20(4):13–22, 1999.

Edmund H. Durfee and Victor R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183, 1991.

Douglas H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

Andrew Garland. *Learning to Better Coordinate in Joint Activities*. PhD thesis, Brandeis University, 2000.

Michael Genesereth, Matt Ginsberg, and Jeffrey Rosenschien. Cooperation without communication. In *Proc. Fifth National Conference on Artificial Intelligence*, pages 51–57, 1986.

Barbara Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86:269–357, 1996.

Barbara Grosz and Candace Sidner. Plans for discourse. In Philip R. Cohen, Jerry Morgan, and Martha E. Pollack, editors, *Intentions in Communication*, pages 417–444. Bradford Books, 1990.

Kristian J. Hammond. Case-based planning: A framework for planning from experience. *Cognitive Science*, 14:385–443, 1990.

Thomas Haynes and Sandip Sen. Learning cases to resolve conflicts and improve group behavior. *Intl. Journal of Human-Computer Studies*, 48:31–49, 1998.

Fenton Ho and Mohamed Kamel. Learning coordination strategies for cooperative multiagent systems. *Machine Learning*, 33(2-3):155–177, 1998.

Marcus J. Huber and Edmund H. Durfee. Deciding when to commit to action during observation-based coordination. In *Proc. First Intl. Conference on Multiagent Systems*, pages 163–170, 1995.

Subbarao Kambhampati and James A. Hendler. Control of refitting during plan reuse. *Artificial Intelligence*, 55:193–258, 1992.

Janet L. Kolodner. Maintaining organization in a dynamic long-term memory. *Cognitive Science*, 7:243–280, 1983.

Janet L. Kolodner. Reconstructive memory: A computer model. *Cognitive Science*, 7:281–328, 1983.

Janet L. Kolodner. *Case-Based Reasoning*. Morgan Kaufmann Publishers, San Mateo, CA, 1993.

John E. Laird, Paul S. Rosenbloom, and Alan Newell. Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.

Hector J. Levesque, Philip R. Cohen, and José H. T. Nunes. On acting together. In *Proc. Eighth National Conference on Artificial Intelligence*, pages 94–99, July 1990.

M. V. Nagendra Prasad and Victor R. Lesser. Learning situation-specific coordination in cooperative multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 2:173–207, 1999.

Barry Smyth and Mark T. Keane. Remembering to forget. In *Proc. Fourteenth Intl. Joint Conference on Artificial Intelligence*, pages 377–382, 1995.

Toshiharu Sugawara and Victor Lesser. Learning to improve coordinated actions in cooperative distributed problem-solving environments. *Machine Learning*, 33(2-3):129–153, 1998.

Toshiharu Sugawara. Reusing past plans in distributed planning. In *Proc. First Intl. Conference on Multiagent Systems*, pages 360–367, 1995.

Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

Manuela Veloso and Jaime Carbonell. Derivational analogy in PRODIGY: Automating case acquisition, storage, and utilization. *Machine Learning*, 10:249–278, 1993.