Machine Learning 33, 129–153 (1998)

Learning to Improve Coordinated Actions in Cooperative Distributed Problem-Solving Environments

TOSHIHARU SUGAWARA sugawara@ntt-20.ecl.net
NTT Laboratories, 3-9-11 Modori-cho, Musashino, Tokyo 185-8585, Japan

VICTOR LESSER lesser@cs.umass.edu
Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA

Editors: Michael Huhns and Gerhard Weiss

Abstract. Coordination is an essential technique in cooperative, distributed multiagent systems. However, sophisticated coordination strategies are not always cost-effective in all problem-solving situations. This paper presents a learning method to identify what information will improve coordination in specific problem-solving situations. Learning is accomplished by recording and analyzing traces of inferences after problem solving. The analysis identifies situations where inappropriate coordination strategies caused redundant activities, or the lack of timely execution of important activities, thus degrading system performance. To remedy this problem, situation-specific control rules are created which acquire additional nonlocal information about activities in the agent networks and then select another plan or another scheduling strategy. Examples from a real distributed problem-solving application involving diagnosis of a local area network are described.

Keywords: learning, coordination, multiagent systems

1. Introduction

Achieving globally coherent[1] activity in a cooperative, distributed multiagent system is a difficult problem for a number of reasons. One difficulty is that an agent's control decisions, based only on its local view of problem-solving task structures, may lead to inappropriate decisions about which activity it should do next, what results it should transmit to other agents and what results it should ask other agents to produce. If an agent has a view of the activities (task structures) of other agents, it can make more informed choices (Decker & Lesser, 1995; Durfee, Lesser, & Corkill, 1987; Durfee & Lesser, 1991; Lesser, 1991). Another difficulty is that even with this type of metalevel information, there is still residual uncertainty about the outcomes of tasks and which future tasks will be coming into the system that may result in agents still exhibiting incoherent behavior. The difficulties with achieving effective coordination are further exacerbated by the fact that an agent, in acquiring and exploiting a nonlocal view of other agents' activities, may expend significant computational resources. This expense is in terms of communication delays, as well as the computational cost of both providing this information in a suitable form to other agents and processing this information to make local decisions. Thus, for specific problem-solving situations, due to the inherent uncertainty in agents' activities and the cost of metalevel processing, it may not be worthwhile to acquire a complete view of other agents' activities, and thus a coordination strategy that does not eliminate all incoherent activity may be optimal (Lesser, 1991).

For example, coordination to avoid redundant activities may be unnecessary if processing resources are not overloaded and if communication channels are neither expensive nor overloaded. In this case, local problem solving is done more efficiently where there is no additional overhead for coordination. If a coordination strategy can be developed whose costs can be varied depending upon the amount and type of nonlocal information used to make coordination decisions, then it seems that only a selected, possibly situation-specific, view of other agents' activities is necessary (Lesser, 1998). The obvious next question is how to determine what the appropriate situation-specific view is and what type of coordination rules should be used in the situation. It is our hypothesis that for many multiagent applications, especially those operating in complex, open and possibly evolving environments, it is very difficult or impossible for the designer of a system to a priori anticipate all the problem-solving contexts and exactly what information and coordination strategy will be most cost-effective for each context. Thus, in this paper, we propose integrating into each agent a distributed learning component that agents can use to determine, through experience, what information is necessary for effective coordination in a specific situation and how to exploit this information locally to select and schedule its activities to achieve the desired coordination.

Another way of understanding our approach to learning coordination rules is to relate it to the GPGP/TÆMS coordination model (Decker & Lesser, 1993, 1995; Lesser et al., 1998). From the perspective of this model, each of the agents makes scheduling decisions based on a subjective view of its own and other agents' activities and its view of available resources. This subjective view is specified by relationships among these activities such as *enable*-, *facilitate*-, *overlap*- and *support*-relations[2] and resource usage patterns such as the *use*-relation. These relationships describe how an activity affects the duration and importance rating of other activities, as well as the quality of the outcomes. All coordination activities are based on the existence and quantitative characteristics of these relationships. In certain situations this subjective view will lead to an agent taking ineffective or inappropriate actions because the relationships among certain nonlocal activities or the nonlocal resources have not been specified as part of the subjective view, and thus have not been appropriately considered in making coordination decisions. This lack of effective coordination can also occur because the subjective view was based on default assumptions or out-of-date information. In our case, we start out with agents who, prior to learning, have subjective views used for coordination that are based solely on their local activities. The use of this totally local subjective view implicitly makes the assumption that there are sufficient computational and other resources so that the details of the activities and the states of other agents are not necessary for effective operation. Thus, the goal of the learning system can be thought of as adding nonlocal control information (and associated control rules) to minimally augment the subjective view of an agent so that it has sufficient information about other agents' activities for it to make a more effective control decision in a specific situation.

In the remainder of the paper, we develop this distributed learning component in detail, based on explanation based learning (EBL) techniques (Dejong, 1981; Mitchell, Keller, & Kedar-Cabelli, 1986) using a domain model and inductive techniques such as comparative analysis (Hudlická & Lesser, 1987). We discuss the implementation of these ideas in a real distributed problem-solving system, LODES (Sugawara, 1990; Sugawara & Murakami, 1992), which performs diagnosis of a computer communications network.

2. An example problem

The LODES network diagnosis system observes message traffic on the network in order to detect and analyze situations that indicate a hardware or software problem in the network. LODES is a multiagent system in which there is a diagnosis agent on each network segment. Each agent is responsible for monitoring traffic on its segment and diagnosing any problems that are recognized. An agent acquires, as a result of its normal monitoring functions, the status of the resources on its network segment. However, it does not normally have knowledge of nonlocal resources and only acquires this knowledge from other agents when it explicitly needs this information as part of its diagnostic inferencing process. Agents start out with the assumption that there is no need to explicitly coordinate with other agents on different segments of the network. Interaction among agents involves an occasional request for the execution of a diagnostic task[3] on an agent's local network segment, and the sharing of information about the network configuration and the final results of diagnosis. The basic assumption behind this lack of coordination is that there are sufficient communication and computational resources available on the network to sustain a certain level of noncoherent behavior, e.g., two agents diagnosing the same problem. This assumption is appropriate for diagnostic activities in most network environments and was the basis for how the LODES system was originally implemented. However, it is not always a valid assumption. For example, in network environments where diagnostic activities generate substantial message traffic, where there are multiple agents performing redundant diagnosis, and where the cost of communication is significant this assumption does not hold.

Consider the network environment shown in figure 1. L1 to L7 are LODES agents, Net1 to Net7 are network segments, and Net5 and Net6 are connected with a narrow-bandwidth line. Suppose a host, H_A on Net1, sends a broadcast to all hosts on Net7, but most of the hosts cannot understand that protocol. Upon receiving the broadcast, the hosts on Net7 that

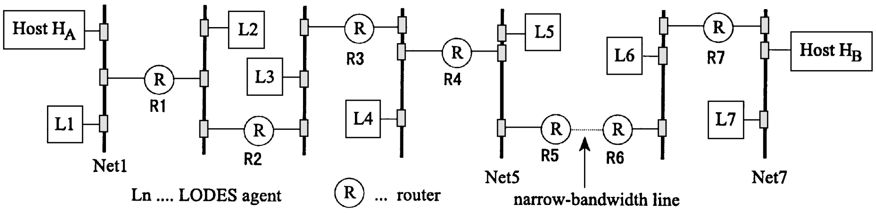


Figure 1. Network environment for the example problem.

cannot understand the protocol simultaneously send back error packets in order to inform H_A that they have discarded its broadcast packet. This use of an inappropriate broadcast protocol by H_A is called the primary problem. In this situation, agents L1, . . . , L7 will concurrently diagnose this problem since the return route of these error packets will traverse network segments monitored by each of these agents. In diagnosing the cause of these error packets, each agent may independently send diagnostic test packets through the network to assess which one of a variety of causes is actually responsible for the error messages. This redundant diagnostic message traffic may in turn lead to another problem if the network environment has network segments which are implemented as narrow-bandwidth lines, or if a tariff is associated with each message (e.g., in this case, the narrow band link between Net5 and Net6 gets congested). This secondary problem involving the overloading of an expensive or scarce resource, which is directly caused by LODES agents' activities, can be attributed to the lack of effective coordination among agents.

One of the interesting aspects of this example problem is that a LODES agent will detect the secondary problem as a normal part of its monitoring of the network. In this case, agents L5 and L6 will decide that the secondary problem is not tolerable because they know the existence of the narrow-bandwidth line. The existence of this problem and the fact that it was caused by the agents themselves will be the trigger for the learning component to be invoked. Other mechanisms for invoking the learning component are discussed in (Sugawara & Lesser, 1993).

3. The learning framework

3.1. Behaviors of planner and scheduler

Before presenting our learning framework, it is necessary to describe the control framework in LODES diagnostic agents, especially the behaviors of its planner and scheduler components. During problem solving, the planner of an agent selects a (diagnostic) goal and a task is generated to achieve this goal. A task consists of a partially ordered set of subtasks. A subtask may further be divided into smaller subtasks. A task that cannot be divided into subtasks is called an *operation* and is directly executable. The execution of a task means that all the subtasks are executed in a manner consistent with the specified partial order. There are usually a number of different high-level tasks that can be used to achieve a goal. The planner generates a set of alternative tasks that can achieve a specific diagnostic goal and then selects the most appropriate task among them. This selected task will be referred to as a plan. The scheduler selects an operation that should be executed next according to the partial order associated with this plan. An operation involves the execution of a subroutine and sometimes will also result in a message being sent to another diagnostic agent. When a message arrives at an agent, the executions of its scheduled operations are interleaved so as to allow the incoming message to be analyzed to understand its impact on the current diagnostic goals or to generate new diagnostic goals. Depending on the result of this analysis, the execution of the current plan is resumed or another more important plan is selected and its execution is initiated.

3.2. What should be learned for coordination actions

Our approach to learning situation-specific coordination strategies is to modify and extend the control rules used by the planning components of a diagnostic agent to select and prioritize its diagnostic activities. The learned control rules cause the agent to acquire the necessary and sufficient nonlocal information to make effective local control decisions so that its activities will be appropriately coordinated with those of other agents. To achieve coordinated actions, agents potentially need to know detailed information about the current and planned activities of other agents, and information these agents have acquired about the characteristics of the network and the partial results of their diagnoses. It is, however, costly to receive and analyze all this nonlocal data; more importantly, usually only a small subset of information is necessary to coordinate actions in a specific situation. Thus, on one hand, a lack of nonlocal data may cause an agent to execute redundant or unnecessary actions. While on the other hand, the transmission of unnecessary information can: contribute to overloading communication channels; add significant computational activities to agents in order to package the information for transmission and assimilate the information into their local databases; and distract an agent, causing it to delay the execution of important actions. Therefore, the process of identifying the appropriate information must be linked closely to the characteristics of the coordination situation and what information is really essential to know in order to avoid the specific incoherence in that situation which is problematic. Of course, performing redundant and unnecessary actions may cause no serious problems. In this case, no rules need to be learned. The result of the learning process is the generation of new information gathering rules that cause the planner to preferentially generate the operation sequence for acquiring needed information before it makes a decision on how to best satisfy a specific diagnostic goal. An alternative approach to situation-specific coordination that is motivated by the same concerns of this work is being pursued by Nagendra Prasad et al. (Nagendra Prasad, Lesser, & Lander, 1998; Nagendra Prasad & Lesser, 1996). The focus of their work is to develop a metalevel characterization of the agent activities and then statistically learn, based on this characterization, what the best agent coordination strategy is for a given metalevel state.

The learned rules are stored as knowledge for planning and coordination (cf., figure 2) and affect the planning activities in the situation-specific way as follows. First, when a plan is built and selected for the given goal, the planner, according to the learned rules, collects additional information that was not included before learning; this additional nonlocal information and associated control logic can lead the planner to no longer select a particular task to achieve a given goal if a specific nonlocal situation exists and instead choose another task. Second, the planner, as a result of new control rules, may alter the ratings of the generated tasks for a given goal so that in the current situation the appropriate plan will now be selected in a timely manner. Third, the planner may also introduce into the chosen plan (again on a situation-specific basis) coordination activities (such as sending messages and modifying the ratings of messages). Prior to learning, these messages were sent with the inappropriate ratings or were just not sent. Thus, unnecessary tasks were selected or needed tasks were not selected by the receiving agents.

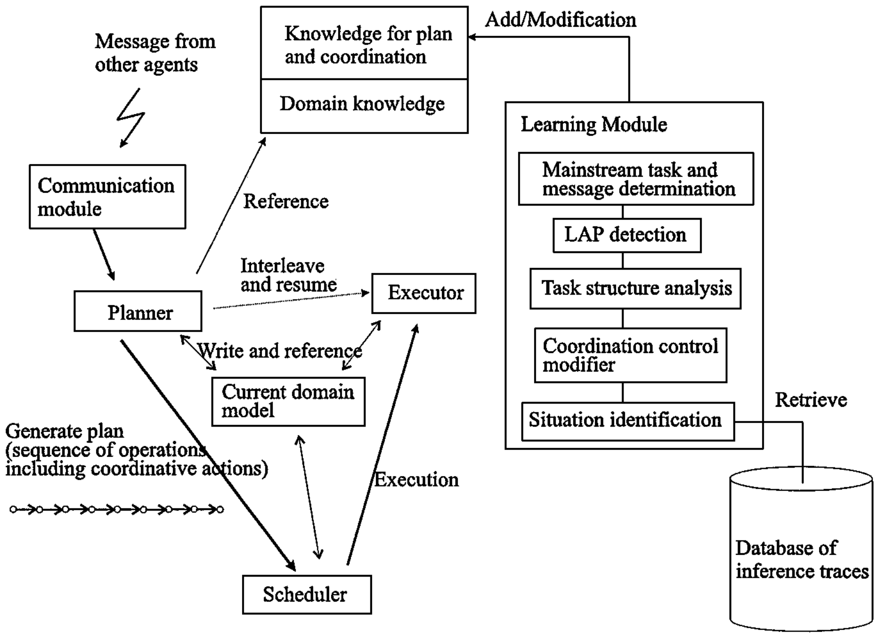


Figure 2. System architecture.

3.3. *The learning framework*

The knowledge used by our learning framework includes a collection of heuristic rules and procedures for: recognizing situations where there is costly incoherent behavior,[4] identifying control decisions that lead to this behavior, and modifying these control decisions or replacing them with new decision processes that rectify the inappropriate control. When an undesirable situation, called a *learning analysis problem* (*LAP*), is detected in an agent, the learning component of that agent takes the following steps based on recorded traces of activity stored in that agent and in other agents:

(1) *Mainstream Task and Message Determination* identifies the tasks and messages that contributed to achieving the final result of the LODES diagnostic process (we call them mainstream tasks and messages).
(2) *LAP Detection* locates, based on the results of Mainstream Tasks and Message Determination, control decisions in the trace that resulted in the execution of the subtask operations which induced the observed undesirable situation.
(3) *Task Structure Analysis* builds the local view of the agent's task structures and the network models that the agent had locally when the tasks that contributed to the LAP were selected. These views are fully exchanged among agents involved in the LAP so as to generate a more comprehensive view about that situation. Agents then reproduce their inference process based on all the information possible in the situation. Both of

these traces are then analyzed in order to characterize the exact cause of the LAP (i.e., was there information present in the more comprehensive view that was absent from the local view, which could be used to make an alternative control decision that would lead to a resolution of the problem).

(4) *Coordination Control Modification* adds context-sensitive rules to the local planner based on the analysis in the previous step; these rules may require the gathering of appropriate nonlocal information prior to their execution for use in selecting and prioritizing tasks so as to avoid recurrence of the LAP in this situation.[5]

(5) *Situation Identification* determines how to best characterize the context-sensitive situation in which the rules developed in the previous step should be applied. This step uses previous instances as well as the currently analyzed instance.

This basic paradigm of monitoring, detection, analysis and adaptation is similar to the approach laid out but never implemented for use in organizational self-design (Corkill & Lesser, 1983); this basic paradigm was later partially implemented as an approach to metalevel control for a single-agent interpretation system (Hudlická & Lesser, 1984, 1987), and more recently completely implemented in (Lesser, Nawab, & Klassner, 1995).

This approach has strong similarities to the case-based approach developed by Hammond (1989) for modifying cases in his casebase of recipes based on failures detected when the recipe was used with substitute ingredients. Another case-based approach in multiagent systems is PASUADER by Sycara (1989), where, in negotiation processes, a number of compromising results are developed from similar cases handled in the past. In general, a case-based approach is utilized in a domain that has the weak domain theory and where a previous case that is similar provides strong information about how to handle the current case. In our approach, prior instances are used to generate and improve learned rules, which are used to identify what information and operations should be included in the local model for coordinative actions in each situation. Moreover, our approach assumes relatively strong domain theories. However, the inference based on prior instances is necessary in multiagent domains, because some uncertainty is associated due to insufficient and out-of-date information.

Our approach also has similarities to EBL (DeJong, 1981; Mitchell, Keller, & Kedar-Cabelli, 1986) in the sense that the records of inference are analyzed based on dependency-relations provided by the TÆMS framework and rich domain knowledge. In single-agent systems, the major purpose of EBL is speed-up. The analysis in our learning, however, identifies which nonlocal information is required for correct decisions (at least, the decisions that do not cause the same LAP to reappear in the current situation) and which ratings of local tasks and sent/received messages are not appropriate. Based on this analysis, rules are added for gathering/requesting the necessary nonlocal information, or for adjusting the ratings of tasks and messages.

4. Steps of the learning process

This section details the steps of the learning process using the example problem discussed in Section 2. It should be noted that this process has been completely implemented in a complex multiagent system; furthermore, the learning is fully distributed and is agent-centered in that an agent only learns rules for itself.

4.1. Inference traces

We assume that an agent is able to record traces of reasoning for analysis by the learning component. This recorded trace is a sequence of action units (AUs) with associated data which are generated by an agent during its diagnostic reasoning.

The action unit is one of:

- A decision: the selection of either a goal, hypothesis, or a plan, and the scheduling of a plan.[6]
- An execution of an operation: an operation as described above is recognized as the smallest program, command, etc., that cannot be stopped or interrupted by the scheduling mechanism.
- An (external) event: message arrival or user's interrupt.

The trace expresses the history of inference, thus this is called an *inference trace*. The associated data are the reasons, the results and the messages: why the AU is selected or executed, which data is referenced, what data is generated, and what messages are sent.[7] We can assume that all internal information such as models of the domain, other agents, and the local inference state are expressed by corresponding variables and are part of the associated data. The examples of inference traces are described in figures 4 and 6.

Thus, an inference trace describes the following aspects of local problem solving: (1) executed tasks and operations; (2) task relationships; (3) information communicated; (4) resource usage; (5) domain data and knowledge used; and (6) control knowledge used. An agent must, furthermore, be able to reproduce the same decisions and reasoning from the recorded inference trace in order to do the task analysis step in the learning algorithm.[8]

4.2. Mainstream task and message determination

The first step in the learning process is to identify the mainstream tasks and messages. These can be obtained by tracing backwards through task enablement relations (that is, enable- and support-relations) provided by TÆMS from the final result in the problem-solving trace as follows. The task that directly led to the final result is a mainstream task. A task that *enables* (required precondition of) a mainstream task is a mainstream task. A task that *supports* a mainstream task (that is, raises its rating) is a mainstream task. A message that produced a mainstream task is a mainstream message. A task that produced a mainstream message is a mainstream task. A task that produced the content of a mainstream message is a mainstream task. A task that produced a mainstream task as a subtask is a mainstream task. This process of computing the mainstream extends beyond a single agent's local problem solving to incorporate activities of agents distributed throughout the network.[9]

4.3. LAP detection

The second step of the learning process is to isolate situations in which any of the following inappropriate actions occurred:

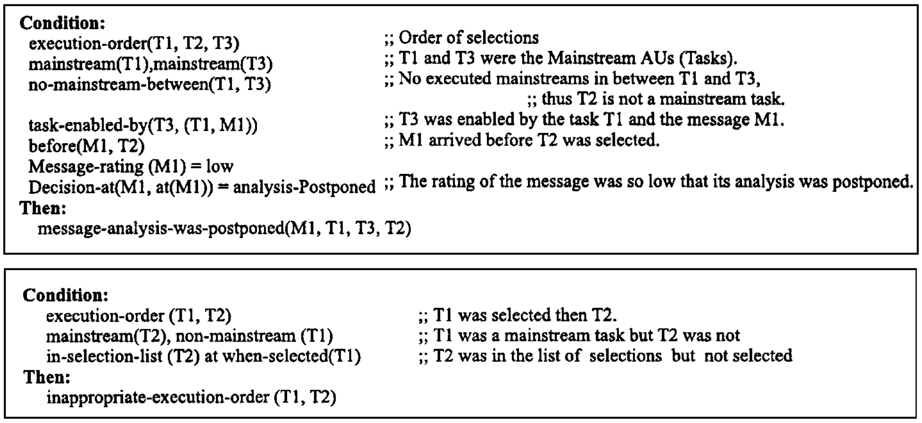
(D1) Blocking of mainstream tasks and messages: nonmainstream tasks that are unnecessary for achieving the goal of problem solving for the current episode and are selected for execution before mainstream tasks that are executable.
(D2) Redundant mainstream task: for example, a variable is defined twice or its value is redundantly sent from other agents.
(D3) Inappropriate scheduling: other agents wait for the result of a task that is executable but its execution is delayed by the scheduler.
(D4) Inappropriate task allocation: some agents were allocated a number of tasks all of which are time-consuming or which use resources beyond their capabilities.
(D5) Problematic external actions: tasks that cause, for example, resource overload problems in shared resources (see the example problem).

These problem-solving situations are LAPs that indicate situations where noncoherent behavior has taken place.

Note that situations D1 to D3 can be specified in a domain-independent manner and are detected using formal heuristic rules. However, the detection of situation D4 is more domain- or system-dependent, because it requires knowledge about how long certain tasks should normally be expected to take. This knowledge could be either provided by the system designer (which was the approach we used) or learned as a result of the ongoing operation of the system (that is, estimating from the average value of prior durations, or the average amount of resources used previously). Further, this knowledge could also be context-sensitive based on characteristics of the current operating environment. The detection of situation D5 also requires the resource monitoring module (a LODES agent has this module) that can detect the state and the time of resource overloads.

Examples of heuristics for recognizing these situations in an inference trace, in this case D1, are shown in figure 3. For domain- or system-dependent LAPs such as D4 and D5, we assume that there are heuristic rules or functions for locating which operations cause these situations. For example, LODES has the `find-task` function that locates the AU (operations) that cause the problematic situation detected by its monitoring module.

In the example, the problem triggering diagnosis—which is the overloading of a narrow communication bandwidth line caused by the generation of a large number of test packets—is reported by the network resource monitoring module. The LAP detection analysis identifies the task(s) in which the diagnostic test packets were sent by analyzing the trace of the first diagnostic problem solving, e.g., an example of problematic external action (a D5 situation). Figure 4 illustrates the mainstream tasks. Agents can identify that the observed test packets were sent in the plan P3 named “Get-RTT-between-Both-Ends” in this step.



```
Condition:
  execution-order(T1, T2, T3)              ;; Order of selections
  mainstream(T1),mainstream(T3)            ;; T1 and T3 were the Mainstream AUs (Tasks).
  no-mainstream-between(T1, T3)            ;; No executed mainstreams in between T1 and T3,
                                                        ;; thus T2 is not a mainstream task.
  task-enabled-by(T3, (T1, M1))            ;; T3 was enabled by the task T1 and the message M1.
  before(M1, T2)                           ;; M1 arrived before T2 was selected.
  Message-rating (M1) = low
  Decision-at(M1, at(M1)) = analysis-Postponed ;; The rating of the message was so low that its analysis was postponed.
Then:
  message-analysis-was-postponed(M1, T1, T3, T2)
```

```
Condition:
   execution-order (T1, T2)                   ;; T1 was selected then T2.
   mainstream(T2), non-mainstream (T1)        ;; T1 was a mainstream task but T2 was not
   in-selection-list (T2) at when-selected(T1) ;; T2 was in the list of selections but not selected
Then:
   inappropriate-execution-order (T1, T2)
```

Figure 3. An example of detection heuristics. The first rule for D1 describes the situation where the execution of a mainstream task (T_3) is delayed (task T_2 that does not contribute to the final result is executed before T_3) because an important message (M_1) has a low rating. The second one describes the situation where that task T_2 is ready to be selected but instead a nonmainstream task T_1 is selected for execution. In LODES, the system-dependent predicate `find-tasks(T-list, <condition-provided-by-monitor>)` is provided for situation D5 to locate the task(s) that cause the problematic situation detected by the monitoring module.

4.4. *Task structure analysis*

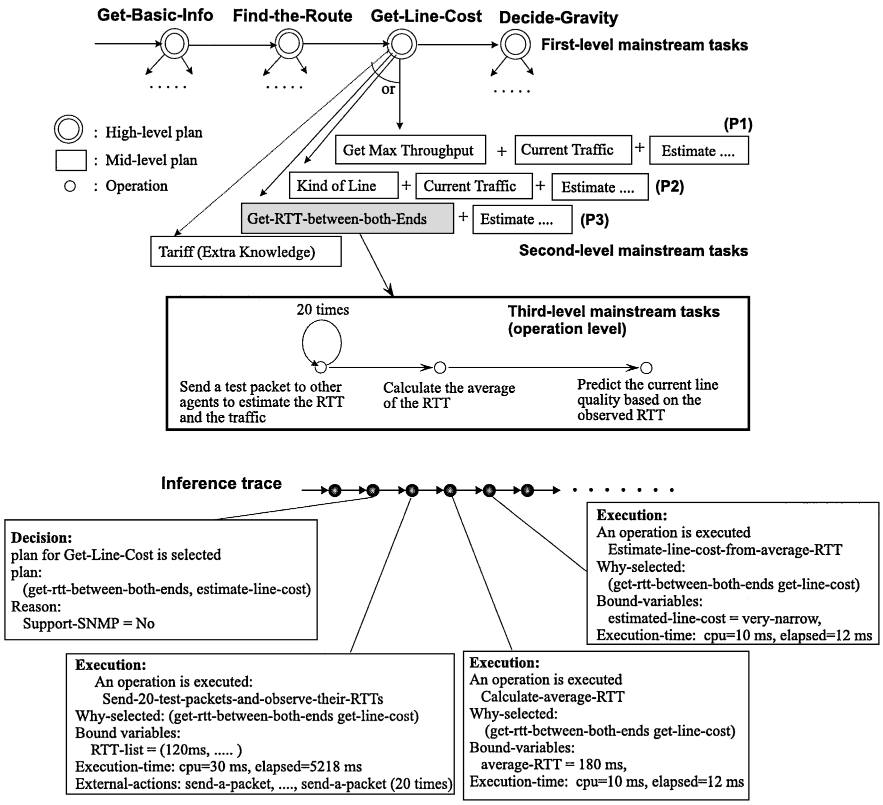
The third step of the learning process involves agents building a comprehensive view to describe the detected LAP. This step is performed as follows:

(A1) The agent identifies the reasons why the agent(s) decided to select the task causing the LAP—that is, it finds the node in the inference trace corresponding to this decision and the reasons (set of variables expressing enable- and support-relations).

(A2) The agent recreates, using the inference trace, the view of the task structures and the domain model at the point when the detected decision was made. Other agents which did not make this decision also create their subjective view of the situation at the point when the LAP occurred.[10] This is a subjective view because it is based on the information an agent used in its deliberations—not what information was actually available.

(A3) All involved agents exchange their subjective views of their state at the time when the problematic decision was made so as to generate a more comprehensive view. This extended view includes goals, scheduled tasks, inference states and resource usage of all agents, as well as other observed domain data. This view is further extended by adding the results of mainstream tasks that were executed in the inference or were ready to execute when the detected decision was made.

(A4) The agent that executed the task causing the LAP reproduces the local planning and scheduling decisions using the generated comprehensive view. The agent marks those pieces of the comprehensive view that are needed to select the appropriate task. This marking process is identical to the one used to identify mainstream tasks.


Get-Basic-Info
Find-the-Route
Get-Line-Cost
Decide-Gravity
First-level mainstream tasks
or
: High-level plan
: Mid-level plan
: Operation
(P1)
Get Max Throughput + Current Traffic + Estimate ....
Kind of Line + Current Traffic + Estimate .... (P2)
Get-RTT-between-both-Ends + Estimate .... (P3)
Tariff (Extra Knowledge)
Second-level mainstream tasks
20 times
Third-level mainstream tasks (operation level)
Send a test packet to other agents to estimate the RTT and the traffic
Calculate the average of the RTT
Predict the current line quality based on the observed RTT
Inference trace

Decision:
plan for Get-Line-Cost is selected
plan:
(get-rtt-between-both-ends, estimate-line-cost)
Reason:
Support-SNMP = No

Execution:
An operation is executed
Estimate-line-cost-from-average-RTT
Why-selected:
(get-rtt-between-both-ends get-line-cost)
Bound-variables:
estimated-line-cost = very-narrow,
Execution-time: cpu=10 ms, elapsed=12 ms

Execution:
An operation is executed:
Send-20-test-packets-and-observe-their-RTTs
Why-selected: (get-rtt-between-both-ends get-line-cost)
Bound variables:
RTT-list = (120ms, ...)
Execution-time: cpu=30 ms, elapsed=5218 ms
External-actions: send-a-packet, ..., send-a-packet (20 times)

Execution:
An operation is executed
Calculate-average-RTT
Why-selected:
(get-rtt-between-both-ends get-line-cost)
Bound-variables:
average-RTT = 180 ms,
Execution-time: cpu=10 ms, elapsed=12 ms

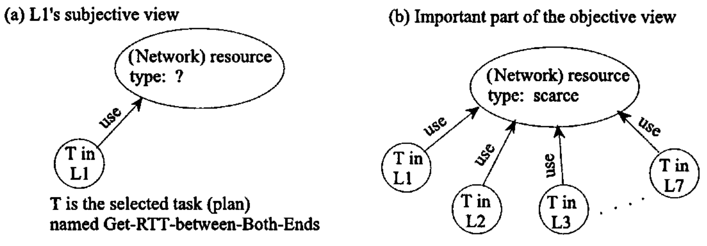
Figure 4. Plans and inference trace created during diagnosis of the example.

Based on this reproduction of the agent's problem-solving process from both the perspective of a local view and of a more comprehensive view, there are three different problematic situations that we recognize. The first problem, called a *lack-of-information problem*, occurs when an agent chooses the wrong alternative high-level task to solve the given diagnostic goal because of the lack of nonlocal information. By "wrong," we do not mean that the task would not have achieved the desired goal if there were appropriate time and resources available but rather its execution was responsible for the observed LAP; we are not detecting incorrect domain knowledge. The detected incoherent behavior can be resolved by just choosing a different high-level task in this specific situation. What the learning component does in this case is to add rules to an agent to acquire the appropriate nonlocal information to identify this specific situation and to use this information to choose the appropriate high-level task that eliminates the incoherent behavior in this specific situation.

The second problem, called an *inappropriate-rating problem*, occurs when the correct plan cannot be selected because of an inappropriate rating function, though all the required data is available in the agent. In this case, because of an inappropriate rating, a nonmainstream plan is selected initially. This situation can be recognized because the system, after failing to solve the problem with the first plan, had tried alternative plans until it succeeded in the diagnosis. Also, no additional nonlocal information that supports the choice of the successful plan arrived between the time the first choice was made and the time that the correct plan was chosen. Thus, we can conclude that the rating of the plan is not appropriate in this specific situation. To solve this problem, it is necessary to add a control rule keyed to this situation which will either directly select this appropriate plan or change the plan's rating function.

The final problem, called an *unnecessary-incoming-message problem*, occurs when the appropriate plan is selected but its execution is interrupted because of the analysis of either a redundant or nonmainstream message from another agent. This distracting message not only affects the scheduler but it is possible that the result of its analysis may induce the execution of an unnecessary plan. To solve this problem, more context-sensitive rules need to be added to decide the importance of processing a received message of a certain type in the current context. This type of problem is discussed in more detail in Section 5.

In the example problem, agents create a comprehensive view of the situation in which the plan "Get-RTT-between-Both-Ends" was selected. Agents select this plan to estimate the network load and bandwidth by gathering statistics on the round-trip time (RTT) of a number of test packets sent into the network. This plan, which is not the optimal way to gather this statistic, produces only an approximate value for RTT. A better plan would have been to use the Simple Network Management Protocol (SNMP), which is designed for acquiring network management data (Case et al., 1990), but this was not possible in the example since the adjacent network routers did not implement it. An agent, in order to understand what tasks other agents are executing, must know that local routers do not use the SNMP, and that all agents along the route from Net7 to Net1 will be performing the same diagnoses. Agents can then understand that an identical task was selected in other agents. This view of which tasks were to be executed in other agents was not present and thus responsible for the LAP occurring; this is one of the important parts of the enhanced subjective view necessary for achieving more coordinated activities (i.e., *lack-of-information problem* as shown in figure 5(a) and (b)). In figure 5(c) we show that the learned rule modifies the planner's choice so as to induce situation-specific coordination with other diagnostic agents. The action part of the rule in this case gathers the information about the selected plans in other agents; with this information in hand, a predefined coordination rule will be triggered that invokes a coordination strategy in which one agent in the group of agents that are performing identical diagnoses is chosen to perform the diagnosis, and then sends the results of the diagnosis to the other agents. In this way, if the implicit coordination achievable through situation-specific selection of a task and the setting priorities of tasks and messages used by the simple priority scheduler is not sufficiently powerful to solve the coordination problem, then more explicit coordination among agent activities can be introduced in a situation-specific manner. It should be noted that the use of the implicit coordination approach, where appropriate, is advantageous because of the low overhead involved in its application. Another important point is that an explicit coordination strategy does not override the basic priority task scheduling process of the agent but rather uses an explicit metalevel dialogue with other agents to decide on which tasks to select locally, and their


(a) L1's subjective view
(Network) resource type: ?
use
T in L1
T is the selected task (plan) named Get-RTT-between-Both-Ends
(b) Important part of the objective view
(Network) resource type: scarce
use
use
use
use
T in L1
T in L2
T in L3
T in L7
. . .

(c) The learned rule

```
IF Initial-Symptom = ICMP-unreachable-storm
   and the plan "Get-RTT-between-Both-Ends" is selected   ;; that is, the local agent is going to execute
                                                             an operation that consumes network resources
   and Min-of-Max-Thrput < 64Kb                           ;; that is, the type of resource is scarce
Then
   Gather the scheduled plans of other agents
```

Figure 5. Views for problem solving. (a) is the task structure created in L1 before learning. This view does not include the resource type (which is included in the nonlocal domain model) and other agents' intended tasks (which are included in the nonlocal task structure). (b) is the task structure that will be created in L1, . . . , L7 after learning in the example problem, and (c) is the learned context-sensitive rule that may create (b). The learned coordination control takes place when the acquired view is identical to the view (b).

priorities. We will not discuss how this action part is chosen in this situation from a library of preexisting explicit coordination strategies, because discussion on such a coordination strategy library is beyond the scope of this paper (for more details see (Sugawara & Lesser, 1993).

There are two additional categories of problems that can be recognized as a result of the comparison of agent problem solving with a comprehensive view and with only a local view. We mention them only briefly because they cannot be solved by adding situation-specific control rules to the planner. One, called a *lack-of-planning-time problem*, occurs when there is not enough time to generate all possible plans and/or to decide which one is appropriate among them. This situation occurs when there is not sufficient time to complete the diagnosis. In this case, if during the reproduction process an appropriate plan based on only local information that was available to the agent is generated, then the LAP may be avoided by just allowing more time in this situation for solving the diagnostic goal before recognizing that there has been an error in diagnostic processing or by speeding up the planning process. Though not implemented, we believe that some aspects of the lack-of-planning-time problem can be solved using EBL or statistical learning techniques to generate rules that directly connect obtained information to the appropriate plan without complex analysis so as to speed up the plan generation and selection process. The second, called a *lack-of-domain-knowledge problem*, occurs when an appropriate domain task for solving the diagnostic goal is not generated. The cause of this type of problem is either a lack of domain or planning-level knowledge, both of which should be provided a priori. Our focus in this paper is on identifying required nonlocal information for effective coordination; thus, the lack-of-domain-rules problem is beyond the scope of this paper since it is not solved solely by making existing rules more context-sensitive with respect to the external agent environment.

4.5. Coordination control modification

The fourth step involves adding or modifying control rules for coordination based on the analysis of the view developed in the previous step. This new control is the consequence part of the learned coordination rules. The approach to modification depends on the types of problems identified in the previous step. In the case of lack-of-information problems where agents cannot generate or select the appropriate plan because of a lack of some nonlocal information, two possible causes are also considered:

(1) The required data was generated in other agents but not transferred to the local agent because other agents were not requested to send it, or they thought that the data was not important.
(2) The required data was not generated in other agents because the tasks for generating the data were not scheduled yet or had low priority.

In both cases, the problem can be solved by sending a request with higher priority to the appropriate agent for the needed data.[11] To accomplish this, a new situation-specific control rule needs to be added to the planner at the point where it decides which plan to select. What data is required from another agent is identified in step (A4). In the example problem, it is necessary to gather information about the selected plans in other agents, so that the local agent can recognize the details of resources and their near-term expected usage patterns that will be used by the chosen plan (see figure 5(c)).

Inappropriate-rating problems can also be solved by adding the following new control rules. In these problems, the required data was already in the local agent but was not fully analyzed because the data arrived with low ratings or the local agent decided that the data was not important. In other situations, the mainstream plan was in the candidate list for the selection but was not selected because it has a relatively lower rating. Both situations can be solved by lowering the rating of nonmainstream tasks and messages that blocked the mainstream ones, and raising the ratings of the blocked mainstream tasks and messages. Likewise, the unnecessary-incoming-message problem (which explains that the processing of a distracting message caused the execution of the appropriate plan to be delayed) can be similarly solved; in this case, control rules need to be added to cause the scheduler to either ignore or postpone the analysis of the nonmainstream messages in this specific situation (see Section 5.1 and figure 8 for more details).

4.6. Situation identification

Since the learned rules are situation-specific, agents have to identify the situation in which the learned rule should be applied. The purpose of this step is to create the premise part of the rule for identifying the situation. For the first problem, the situation must be identified based on the analysis of the subjective view and the comprehensive view constructed in step (A4). In the case of a lack-of-information or an unnecessary-incoming-message problem, all of the subjective view in the local agent is the premise part of the rule, that is, the conjunction of all variable-value pairs describing the view. In the case of an inappropriate-rating problem, all of the subjective view and the nonlocal information in the comprehensive view that was marked in the reproduction process in step (A4) is the premise part of the rule. We assume that the marked nonlocal information is requested by other agents for verifying the premise part of the learned rule. However, such a complete version of the view is usually over-specific; we think that only a part of this view is essential to characterize the situation where the derived rule should be applied. The question we then face is what parts of the subjective and comprehensive views are necessary to uniquely identify the problematic situation.

A simple inductive method (that has so far been adequate for our needs) is introduced to solve this identification problem. First, previously stored traces (and other agents' traces, if possible; see the example below) containing the same LAP as well as the last reasoning trace are gathered,[12] then these traces are divided into two types of instances—positive and negative instances (Sugawara & Lesser, 1993). When a variable used in the different types of instances has the same value, or when a variable used in the same type of instances has different values, we can conclude that the variable is not necessary for distinguishing the situation from others. This process of comparison is called comparative analysis (CA) (Hudlická & Lesser, 1984). Note that only the network model and selected plans (in each level and in each agent) are compared.[13]

In this example, agents can identify the situation by comparing their traces of reasoning about the secondary problem with all the other agents diagnosing the same secondary problem. Although L1, . . . , L7 observe the secondary problem, only L5 and L6 concluded that the problem was not tolerable, because they know the existence of a narrow-bandwidth line. This difference is expressed by the variable Min-of-MaxThrput; other variables are eliminated by the CA (cf. Table 1). The results finally obtained through learning are illustrated in figure 5. Note that the variable Min-of-Max Thrput is, as a result of the learning process, identified as the mainstream variable that was not included in the original subjective view.

Note that after this learning, an agent's action may still cause the same or other problems. For example, it is possible that the situation where the learned rules should be applied cannot correctly be identified because of insufficient instances. Alternatively, the appropriate inference strategy may need to change over time in a gradually evolving environment. In both cases, agents can incrementally converge on the correct rules by applying this learning iteratively.

4.7. Empirical results

All aspects of the learning component necessary for doing the examples described in the paper have been implemented in the LODES system. The learned control plan for the example problem (see figure 5) described in this paper represents the output of our implemented learning component. The CPU time and the elapsed time of the learning process for this example are approximately 5.12 s and 102.5 s in C and LISP (interpreter), respectively, on Sun machines (SparcStation 1 (SS1), SS1+ and SS2).[14] The difference between the CPU time and the elapsed time is caused by communications and synchronization. The number

Table 1. Comparative analysis with (a) positive instances (L5 and L6) and (b) negative and positive instances (L4 and L5).

(a) Positive instances (L5 and L6)

Variables	Values in L5	Values in L6	
Adjacent networks	Net4, Net6	Net5, Net7	Eliminated
Adjacent routers	R4, R5	R6, R7	Eliminated
End-nodes	(L1, L7)	(L1, L7)	
Src-MAC	xx:xx:xx:f:2a:3b	xx:xx:xx:0:12:8c	Eliminated
Dst-MAC	xx:xx:xx:f:2a:3a	xx:xx:xx:0:2f:7e	Eliminated
Type-of-Storm	ICMP-Echoes	ICMP-Echoes	
MaxThrput1	10,000,000	64,000	Eliminated
MaxThrput2	64,000	10,000,000	Eliminated
Min-of-MaxThrput (Quantitative measure)	64,000	64,000	64,000 (max of L5 and L6's values)
Observed-Number-of-Echoes (Quantitative measure)	68	64	64 (almost identical) (min of L5 and L6's values)
Current-Traffic	low	low	

(b) Negative and positive instances (L4 and L5)

Variables	Values in L4	Values in L5	
End-nodes	(L1, L7)	(L1, L7)	Eliminated
Type-of-Storm	ICMP-Echoes	ICMP-Echoes	Eliminated
Min-of-MaxThrput (Quantitative measure)	10,000,000	64,000	64,000 (min of L4 and L5's values)
Observed-Number-of-Echoes (Quantitative measure)	68	61	Eliminated (almost identical)
Current-Traffic	Low	Low	Eliminated

of messages during the learning process is 148. These messages include those for starting the learning process and synchronizing each step of the learning, as well as those for information exchange required in certain learning steps (e.g., mainstream task determination). The modified agent control resulted in an increase of approximately 10% in the time for all agents to arrive at a diagnosis of the first problem because of additional coordination activities that centralized some of the diagnostic activity. The inference traces are also longer. However, the total number of communication packets sent among agents is approximately half the amount prior to the learning. Thus, in this case, the learning resulted in a trade-off between additional time to perform domain problem solving and use of a scarce resource. Before learning, agents created their plans based on the very limited view of the network and other agents as illustrated in figure 5(a). We want to emphasize that learning enables the agents to create their plans based on more accurate and appropriate views about other agents and the status of network resources as illustrated in figure 5(b).

5. Another example—Identifying important messages

In order to indicate the generality of our approach, we next discuss an additional example coordination problem. This problem arises when the priority for processing messages of a certain type is incorrectly set in a particular situation, that is, an inappropriate-rating problem.

5.1. Example description and learned rules

Suppose that, in the environment shown in figure 1, an H_A's user in Net1 tried to telnet to H_B in Net7 and there is no response from H_B (this is the symptom of the problem). Let us consider the following three possible causes for this problem:

(P1) H_B does not have the routing information to send the packets to network Net1.
(P2) H_B is off-line.
(P3) H_B is running with no network services, but the network driver is active (e.g., the processor is in single-user mode in order to back up the contents of a disk).

Any of these problem explanations can be used to explain the symptom, from the viewpoint of H_A. After the symptom is reported to L1 (diagnostic agent for Net1) by the user or the network manager, L1 initially believes that this is not a local problem and thus invokes L7 (diagnostic agent for Net7) for coordinated diagnosis. Note that any observational result in L1 for these three problem causes is identical (i.e., there is no H_B's response to any test and request packet) and thus L1 cannot distinguish among these problems. To identify the cause, coordinated actions among the diagnostic agents is essential. The initial LODES agents (that is, before learning) can diagnose the problem, but not efficiently, since redundant and unnecessary tasks are performed. For example, the cause of the problem P1 cannot be isolated solely by L7; L7 believes that the cause must be on L1's side because L7's observations imply that all H_B's actions are correct. L7 can only change its mind when it receives L1's message indicating L1 has observed that the communication it expected from H_B has not occurred (e.g., the ICMP echo reply from H_B cannot be observed by L1. This result is stored into the variable "Echo-Reply-from-the-Remote-Host" in L1). L7 can, however, diagnose the problem itself if it is P2, and no information from L1 is necessary in this case. An unnecessary message from L1 may lead to unnecessary analysis of this message and/or execution of redundant tasks. If the problem is P3, L7 can perform a diagnosis without the value of "Echo-Reply-from-the-Remote-Host" in L1, but this value can eliminate the executions of some of its diagnostic tasks. However, since L1 initially cannot distinguish among these problem causes, it cannot decide whether or not the local result "Echo-Reply-from-the-Remote-Host" is really important. Initially, L7 also cannot determine the importance of processing received messages from L1, and this results in either executing unnecessary tasks or delaying important tasks.

Let us now look at how L1 and L7 in this diagnosis situation can learn a more effective coordinated diagnosis strategy. Consider the problem P2. L7 can isolate the cause without information from L1, but performs a number of unnecessary tasks, some of which are induced by the message from L1 containing the variable value of "Echo-Reply-from-the-Remote-Host."[15] As a result of identifying mainstream tasks and messages in the learning process, L7 can recognize that L1's message about the value of "Echo-Reply-from-the-Remote-Host" is not part of the mainstream. (All messages from L1 are not mainstream except the ones containing the data about the symptom reported by the user.) Therefore, the activities for reading and analyzing this message and subsequent activities based on this message are unnecessary. These unnecessary activities are detected as the problem D1; more concretely, the nonmainstream message from L1 can be detected by the first rule described in figure 3. If we assume that this problem is not a lack-of-information type,[16] the goal of the learning method is to generate, in L7, the rule that lowers the rating of the message about the value of "Echo-Reply-from-the-Remote-Host" in the appropriate situation as shown in figure 7.

Next, suppose that the problem is P1. L7 can also isolate the cause through coordinated actions with L1. In this case, the message containing L1's result of "Echo-Reply-from-the-Remote-Host" is mainstream. That is, L7 believes that the host H_B has no problems, but L1's result indicates the conflict between L7's domain model for H_B and L1's. This conflicting information induces another important task and L7 can then reach the required conclusion. In this case, L7 may perform an unnecessary task, because L7 cannot understand that "Echo-Reply-from-the-Remote-Host" is important, so its analysis may be delayed as shown in figure 6. In this situation, through learning, L7 will generate the rule for P1 that raises the rating of the message about the value of "Echo-Reply-from-the-Remote-Host" in

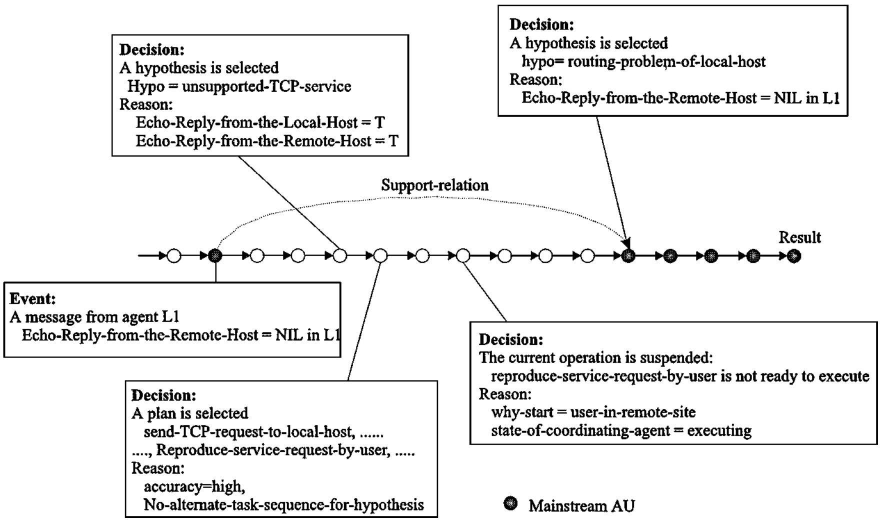
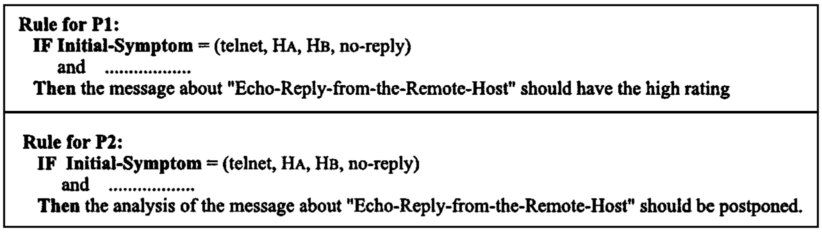

Decision:
A hypothesis is selected
Hypo = unsupported-TCP-service
Reason:
Echo-Reply-from-the-Local-Host = T
Echo-Reply-from-the-Remote-Host = T
Decision:
A hypothesis is selected
hypo= routing-problem-of-local-host
Reason:
Echo-Reply-from-the-Remote-Host = NIL in L1
Support-relation
Result
Event:
A message from agent L1
Echo-Reply-from-the-Remote-Host = NIL in L1
Decision:
The current operation is suspended:
reproduce-service-request-by-user is not ready to execute
Reason:
why-start = user-in-remote-site
state-of-coordinating-agent = executing
Decision:
A plan is selected
send-TCP-request-to-local-host, ...
...., Reproduce-service-request-by-user, ...
Reason:
accuracy=high,
No-alternate-task-sequence-for-hypothesis
● Mainstream AU

Figure 6. Inference trace for P2.



Rule for P1:
IF Initial-Symptom = (telnet, H_A, H_B, no-reply)
and ....
Then the message about "Echo-Reply-from-the-Remote-Host" should have the high rating

Rule for P2:
IF Initial-Symptom = (telnet, H_A, H_B, no-reply)
and ....
Then the analysis of the message about "Echo-Reply-from-the-Remote-Host" should be postponed.

Figure 7. The rules generated to appropriately rate received messages.

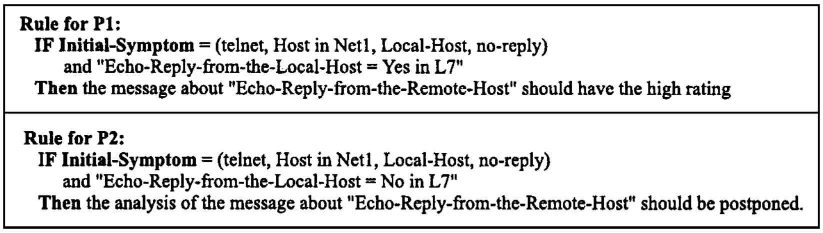
the same situation as shown in figure 7. Note that both rules for P1 and P2 are overspecific because the conjunction of all variables embedded into the premise part of the rules includes L7's subjective view at the point when the message arrived.

However, after diagnosing the P1 problem, L7 can recognize that the variable value of "Echo-Reply-from-the-Remote-Host" plays an important role as a result of the situation identification step. Based on a previous diagnosis for the problem P2, L7 decided that the nonlocal data "Echo-Reply-from-the-Remote-Host = No" (meaning that no replies were observed) in L1 is not part of the mainstream. The new inference trace for the problem P1 is the negative instance of this, since its initial (reported) symptom is identical, but the message containing the variable "Echo-Reply-from-the-Remote-Host" is part of the mainstream.

Table 2 indicates the subset of variables that appear in both P1 and P2.[17] After this step, "Echo-Reply-from-the-Local-Host = Yes in L7" is added in the premise part of the rule for P1 and "Echo-Reply-from-the-Local-Host = No in L7" is also added to the premise part of the rule for P2 in the L7 agent. Note that, with additional positive instances, the H_A and H_B in these rules could be generalized. For example, if the problem P2 occurs between another host in Net1 and another local host (from the viewpoint of L7), then H_A and H_B will be replaced by the more general predicate "Host in Net1" and "Local-Host," respectively, as shown in figure 8. We must say that L1 cannot generate the rule by itself because, after more instances, L1 generates conflicting rules: one indicates that the message "Echo-Reply-from-the-Remote-Host" is important, but another indicates it is not. In this case, the agent gives up on trying to generate a rule to improve performance.

Table 2. Comparative analysis for the problems P1 and P2.

Variable comparison in L1	Problem P1	Problem P2	
Echo-Reply-from-the-Local-Host in L1	Yes	Yes	Eliminated
Echo-Reply-from-the-Remote-Host in L1	No	No	Eliminated
Variable comparison in L7			
Echo-Reply-from-the-Local-Host in L7	Yes	No	
Echo-Reply-from-the-Remote-Host in L7	Yes	Yes	Eliminated



Rule for P1:
IF Initial-Symptom = (telnet, Host in Net1, Local-Host, no-reply)
and "Echo-Reply-from-the-Local-Host = Yes in L7"
Then the message about "Echo-Reply-from-the-Remote-Host" should have the high rating

Rule for P2:
IF Initial-Symptom = (telnet, Host in Net1, Local-Host, no-reply)
and "Echo-Reply-from-the-Local-Host = No in L7"
Then the analysis of the message about "Echo-Reply-from-the-Remote-Host" should be postponed.

Figure 8. The generated rules in L7.

5.2. Experimental result

The experimental result is shown in Table 3. In both cases, a number of unnecessary tasks are no longer executed after L7's control has been modified by the learned rules. In the case of the problem P1, for example, L7 requested L1 to perform the task of reconfirming that no ICMP-echo-reply packets were observed (it is possible that L1 could not observe the ICMP-echo-reply packets because one of the intermediate network segments was so busy that the reply packet was delayed or discarded). After the rule for P1 is learned, this task is not requested since the message containing the variable and value of "Echo-Reply-from-the-Remote-Host" has a higher rating; thus, this analysis and the induced tasks are executed before other tasks. One of the induced tasks is to request a task of L1. The requested task is immediately done since it has the highest rating. Thus another unnecessary task is also eliminated in L1. In the case of the problem P2, the analysis of the received message "Echo-Reply-from-the-Remote-Host = No in L1" is delayed because L7 knows that it is not important. The details are described in (Sugawara & Lesser, 1993). In both P1 and P2, only one or two operations are eliminated, but the inference traces are much shorter. This is because not only operations but also unnecessary decisions are eliminated. Note that, for the problem P2, L7 also learns other rules which allow it to ignore any messages from L1, except for the initial messages involved in starting coordinated diagnosis and reporting the symptom, thus eliminating additional processing tasks in L7. The above experimental result assumes that L7 has only the rules in figure 8.

Table 3. The number of tasks (operations) executed and the length of the recorded inference traces in agent L7.

	In L7 (before learning)	In L7 (after learning)
Number of operations executed for P1	8 (or 9)	7
Length of inference trace	33 (or 34)	27
Number of operations executed for P2	6	5
Length of inference trace	29	21

6. Discussion and conclusion

Although coordination is an essential technique for cooperative distributed problem solving, a trade-off exists between the amount of effort to implement an effective coordination strategy and the savings accrued as a result of more effective coordination. Both too much or too little coordination can degrade overall system performance. A balance can be achieved if situation-specific models of the network state can be created and used to arrive at an acceptable control decision. We feel it is an impossible task for the system designers, at design time, to choose this appropriate balance in all situations, especially for systems operating in open and evolving environments. This paper has discussed an approach to learning situation-specific coordination control rules. The method enables the system to avoid a previously recognized, noncoherent situation by modifying or extending local control strategies to be more situation-specific so as to have a more enhanced view of the relationships of activities in other agents, the status of other agents' deliberations and the environment. In this way, more overhead for coordination will be introduced only in situations where the system observes that current coordination strategies are seriously ineffective.

Our approach is fully distributed, agent-centered, and has been implemented in a real multiagent system for network diagnosis. This contrasts with most research done on learning coordination strategies in multiagent systems which have been done in artificial or synthetic domains with agents having simple coordination patterns, limited computational state, and whose inference processes involve, at most, a few steps (Grefenstette, 1992; Kinney & Tsatsoulis, 1993; Sen, Sekaran, & Hale, 1994; Shoham & Tennenholtz, 1992; Sandholm & Crites, 1995; Tan, 1993; Weiss, 1994).

As part of this discussion, it is appropriate to contrast some of our design decisions with alternative options. We start out with agents having a simple local control architecture as described in Section 3.1 and no explicit protocols for coordination of their activities with other agents; agents only have the ability to send a request to another agent for a prioritized task to be executed. The result of learning is to augment this framework by making certain local control decisions based on nonlocal information, and where necessary, using an explicit agent coordination protocol from a preexisting library. An alternative choice is to use an agent that has a more complex local control architecture and a parameterized family of coordination protocols such as GPGP (Decker & Lesser, 1995) that interact with the local control architecture to achieve the desired coordination; each family member of a protocol implies different overheads in terms of metalevel communication and computation. In the GPGP case, learning involves choosing for specific situations which family member of the protocol is most cost effective.

The use of an agent with a simple local control architecture is advantageous when often there is no need for coordination, or coordination can be achieved by the exchange of a few pieces of information; in this situation, the simplicity of the local control architecture allows the overhead for supporting coordination to be kept to a minimum. The potential disadvantage of basing learning on a simple agent occurs where an agent is pursuing multiple goals simultaneously, each goal possibly requiring different explicit coordination protocols.

In this case, it may be difficult without a more sophisticated local agent control architecture underlying these protocols for them not to adversely interfere with each other when simultaneously active. Although we have chosen here to use a simple agent local control architecture, we feel that the basic approach to learning situation-specific coordination we have laid out is equally applicable to a more sophisticated agent architecture and coordination framework (see (Nagendra Prasad & Lesser, 1996) for an alternative way of learning situation-specific coordination strategies for GPGP).

Another important aspect of this work is its emphasis on knowledge-intensive learning based on a single coordination episode. The aspect of our learning that is most dependent on domain knowledge is the situation identification process detailed in Section 4.6. The use of domain knowledge in this process allows us to characterize the coordination situation so that its description is not overly general nor overly specific. However, we feel a more statistical learning approach which would require repeated occurrences of a similar LAP could be substituted to avoid or limit the amount of domain-specific knowledge that is required without changing our basic approach.

In summary, the most important contribution of this paper is that distributed learning of situation-specific coordination strategies is feasible in real multiagent systems. We have not, however, addressed here a number of important questions to understand the full potential of our approach. For example: What is the entire range of coordination problems that can be solved by this approach? What level of detail is needed to record agent activities to enable postprocessing of this trace for use in diagnosis? How much domain-dependent knowledge is needed to use this approach in other applications? How should nonhomogeneous agents be handled? With respect to the last two points, one of the directions we hope to pursue is making our approach to learning less domain-dependent. We are currently examining how our approach could be applied to the GPGP domain-independent coordination framework with its associated TÆMS (Decker & Lesser, 1993) representation that has been extended to include resource relationships as depicted in figure 5(a) and (b). The idea is to use TÆMS to represent agent activities and the operating environment and develop a version of the learning component that could work from this representation. Our preliminary work in this direction, reported in (Bazzan, Lesser, & Xuan, 1998), is encouraging.

Acknowledgment

We would like to acknowledge the thoughtful readings of this paper by David Jensen, Daniel Neiman, Maram Nagendra Prasad and Thomas Wagner which helped us in significantly improving its presentation.

This research was supported in part by the National Science Foundation (under Grant No. IRI-9523419), by the Dept. of the Navy, Office of the Chief of Naval Research (under Grant No. N00014-95-1-1198), by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF (under Agreement No. F30602-97-1-0249), and by a grant from Network General Corporation (now part of Network Associates). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the organizations supporting this research.

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Notes

1. The behavior of cooperative agents is not coherent when agents transmit information that is not relevant or not timely, derive information that had already been generated by another agent, or cause the overloading of scarce or expensive shared resources.
2. Support-relation is a new relationship that was not in the original formulation discussed and relates to how a task in one agent can affect the subjective view of its own and another agent's task structure by changing the importance rating of its tasks. This rating change can, in turn, cause the agent to choose one task over another for execution.
3. When it sends a request to another agent to perform a diagnostic task, an agent also attaches an importance rating derived from its local perspective. The receiving agent uses this rating as a factor in its decision about when to execute the requested task.
4. This is done by the metalevel controller within an agent or by an external monitor for specific (shared) resources (such as network resources or database). The kinds of events that are detected are described in (Sugawara & Lesser, 1993). It is also assumed that an agent locally records an abstracted trace of its recent problem-solving actions which can be reviewed on-line by the learning component.
5. In future work, we hope to add in an additional phase which takes past rules in conjunction with new rules that we have just constructed, all of which have been based on single instance failure analysis, and develops more general forms of these control rules when appropriate.
6. The problem-solving activities of a LODES agent could be represented at a finer level of detail. However, we have not found that necessary in the types of analysis we need for recognizing the cause of incoherent behavior.
7. Time information can be added for estimating task durations, if possible and required.
8. In LODES, the amount of data that each agent needs to record in order to faithfully reproduce the trace was not significant. However, in other domains, where agents have large knowledge bases or require a large number of actions to carry out problem-solving activities, the recording of all information pertinent to reproducing agent decisions may not be feasible. In such domains, one approach is to only record selected aspects of agents' decisions and then, based on this information, narrow down the cause of the problem and what additional information would be necessary to resolve the problem. The agents would then be instructed to collect certain additional information for post-analysis if they again incur the specific high-level situation that may have caused the problem.
9. In TÆMS, some task relations may affect the duration and the quality of other tasks quantitatively (such as facilitate-relation) rather than qualitatively (such as enable-relation). This mainstream determination step can be extended to handle these quantitative relations (Bazzan, Lesser, & Xuan, 1998).
10. This view is identical to the distributed snapshots introduced by Chandy and Lamport (Chandy & Lamport, 1985).
11. Another way of solving this problem would be to add rules to the appropriate agent to automatically generate and send the needed information in the specific situation. In our approach, the agent doing the learning makes local changes to its control plan (i.e., agent-centered) that solve the problem rather than forcing other agents to change their control plans.
12. This approach assumes that we have stored some amount of history in the agent about previous problem-solving experiences in a way to facilitate such analysis, or that we will observe more carefully the next time this problematic situation occurs (Mitchell, Keller, & Kedar-Cabelli, 1986).
13. CA is not general method for inductive learning but is sufficient for our current requirement. Of course, we think that other more inductive methods can be used for more complicated situations such as a situation described by OR conditions.
14. The CPU time can be improved by compiling and/or optimizing the LISP program.
15. Initially, L7 cannot ignore this message because of the possibilities of P1 and P3.
16. When it is a lack-of-information problem, L7 can select correct actions if appropriate nonlocal information is provided. The rule is generated in the same way as in the previous example for gathering needed data.

17. Of course, there are other bound variables. For example, the variable "ARP-reply," indicating whether or not there is an ARP reply from the local host (that is, H_B), will be bounded in L7 for the problem P2. However, we assume that they can be ignored in the current learning process because they are also eliminated in the same way.

References

Bazzan, A.L.C., Lesser, V.R., & Xuan, P. (1998). *Adapting an organizational design through domain-independent diagnosis*. (Computer Science Technical Report 98–14). Amherst: University of Massachusetts.

Case, J., Fedor, M., Schoffstall, M., & Davin, J. (1990). A simple network management protocol (SNMP). RFC1157.

Chandy, K.M., & Lamport, L. (1985). Distributed snapshots: Determining global states of distributed systems. *ACM Trans. of Computer Systems*, *3*(1), 63–75.

Corkill, D.D. & Lesser, V.R. (1983). The use of meta-level control for coordination in a distributed problem solving network (long paper). *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (pp. 748–756). Karlsruhe, FRG. (Also published in (1986) Benjamin W. Wah & G.-J. Li (Eds.), *Computer architectures for artificial intelligence applications*. IEEE Computer Society Press.)

Decker, K., & Lesser, V.R. (1993). Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance and Management*, special issue on Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior, *2*, 215–234.

Decker, K.S., & Lesser, V.R. (1995). Designing a family of coordination algorithms. *Proceedings of the First International Conference on Multi-Agent Systems*. San Francisco: AAAI Press.

DeJong, G. (1981). Generalizations based on explanations. *Proceedings of Seventh International Joint Conference on Artificial Intelligence* (pp. 67–69).

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

Durfee, E.H., Lesser, V.R., & Corkill, D.D. (1987). Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, *36*(11), 1275–1291. (Also published in (1988) A. Bond & L. Gasser (Eds.), *Readings in distributed artificial intelligence*. CA: Morgan Kaufmann Publishers.)

Grefenstette, J. (1992). The evolution of strategies for multi-agent environments. *Adaptive Behavior*, *1*(1), 65–89.

Hammond, K.J. (1989). *Casebased planning: Viewing planning as a memory task*. Academic Press.

Hudlická, E., & Lesser, V.R. (1984). Meta-level control through fault detection and diagnosis. *Proceedings of the 1984 National Conference on AI* (pp. 153–161).

Hudlickà, E., & Lesser, V.R. (1987). Modeling and diagnosing problem-solving system behavior. *IEEE Transactions on Systems, Man, and Cybernetics*, *17*(3), 407–419. (Also published in (1988) A. Bond & L. Gasser (Eds.), *Readings in distributed artificial intelligence*. CA: Morgan Kaufmann Publishers.)

Kinney, M., & Tsatsoulis, C. (1993). *Learning communication strategies in distributed agent environments*. (Working Paper CECASE-WP-93-4). Lawrence: Center for Excellence in Computer-Aided Systems Engineering, The University of Kansas.

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

Lesser, V.R. (1998). Reflections of the nature of multi-agent coordination and its implications for an agent architecture. *Autonomous Agents and Multi-Agent Systems*, *1*, 89–111.

Lesser, V.R., Decker, K., Carver, N., Garvey, A., Neiman, D., Nagendra Prasad, M., & Wagner, T. (1998). *Evolution of the GPGP domain-independent coordination framework*. (Computer Science Technical Report 98-05). Amherst: University of Massachusetts.

Lesser, V., Nawab, H., & Klassner, F. (1995). IPUS: An architecture for the integrated processing and understanding of signals. *Artificial Intelligence*, *77*, 129–171.

Nagendra Prasad, M.V., & Lesser, V. (1996). Off-line learning of coordination in functionally structured agents for distributed data processing. *Proceedings of the ICMAS-96 Workshop on LIOME*. (An extended version of the paper appears as: (1997). *Learning situation-specific coordination in cooperative multi-agent systems*. (Computer Science Technical Report 97-12). Amherst: University of Massachusetts.

Nagendra Prasad, M.V., Lesser, V.R., & Lander, S. (1998). Learning organizational roles for negotiated search in a multi-agent system. *International Journal of Human-Computer Studies (IJHCS)*, special issue on Evolution and Learning in Multi-Agent Systems, *48*, 51–67.

Mitchell, T.M., Keller, R.M., & Kedar-Cabelli, S.T. (1986). Explanation-based generalizations: A unifying view. *Machine Learning*, *1*, 47–80.

Sandholm, T., & Crites, R. (1995). Multi-agent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, *37*, 147–166, special issue on the Prisoner's Dilemma.

Sen, S., Sekaran, M., & Hale, J. (1994). Learning to coordinate without sharing information. *Proceedings of the 1994 National Conference on AI* (pp. 426–431).

Shoham, Y., & Tennenholtz, M. (1992). Emergent conventions in multi-agent systems: Initial experimental results and observations. *Proceedings of KR-92*.

Sugawara, T. (1990). A cooperative LAN diagnostic and observation expert system. *Proceedings of IEEE Phoenix Conference on Comp. and Comm.* (pp. 667–674).

Sugawara, T., & Lesser, V.R. (1993). *On-line learning of coordination plans*. (Computer Science Technical Report 93-27). Amherst: University of Massachusetts. (A shorter version of this paper was also published in (1993) *Proc. of the 12th Intl. Workshop on Distributed AI.*)

Sugawara, T., & Murakami, K. (1992). A multiagent diagnostic system for internetwork problems. *Proceedings of INET'92*.

Sycara, K. (1989). Multiagent compromise via negotiation. In L. Gasser & M.N. Huhns (Eds.), *Distributed artificial intelligence II*. CA: Morgan Kaufmann Publishers.

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

Weiss, G. (1994). Some studies in distributed machine learning and organizational design. (Technical Report FKI-189-94). TU, Mn̈chen: Institut für Informatik.

Received February 12, 1997
Accepted
Final manuscript March 11, 1998