Introspection as Control in Result-Sharing Assumption-Based Reasoning Agents

Cindy L. Mason [1]

Artificial Intelligence Research Branch
NASA Ames Research Center, Mail Stop 269-2, Moffett Field, California 94305 USA

Email: mason@ptolemy.arc.nasa.gov
Phone: (415) 604-0305
Fax: (415) 604-3594

Abstract

Introspection refers to an agent's knowledge of its own beliefs. In this paper we explore the use of introspection as a mechanism for control in result-sharing cooperative assumption-based reasoning systems. Contrary to the current trend towards building agents with mechanisms for selective communication, our experiments indicate there exists an interesting class of cooperative distributed problem solving systems in which agents may broadcast results without regard to selection. Recipient agents avoid penalties typically associated with such broadcast strategies using introspection and a history of knowledge state to focus problem solving.

1 Introduction

Over the past decade, cooperative distributed problem solving (CDPS) has proven to be a powerful paradigm in which high-level problem-solving agents are brought together to cooperatively solve problems on loosely-coupled computer architectures. Examples of such problems include scheduling for networks of telescopes [14], cooperative robotics [6], and network monitoring systems [4,13]. A variety of applications are discussed in [1,7,10].

Cooperation among agents in a CDPS system may be coarsely divided according to the two forms of cooperation that they model: task-sharing and result-sharing. In the former, agents cooperate by sharing the decomposition, assignment and solution of subproblems. In the latter, agents must not only provide solutions to subproblems, but must also reason about when subproblem solutions should be sent, which subproblem solutions should be sent, and to whom they should be sent. Here we focus on the result-sharing class of CDPS systems.

[1]This research was supported by a University of California-Lawrence Livermore National Laboratory graduate fellowship and a National Research Council Associateship from the Academy of Science.

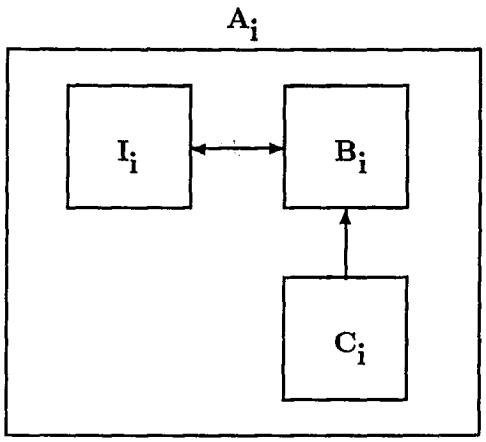


Figure 1:

As developers of result-sharing CDPS systems, we are concerned primarily with communication between agents. Effective coordination of problem solver agents cannot be considered by single message exchange. Instead, communication occurs through protocols for informing, requesting, and convincing. Cost of communication can be expensive - not just latency and transmission and reception overhead, but an arriving message may cause a flurry of computation to ensue. Result-sharing agents are subject to distraction and may waste valuable resources exploring incoming results that may turn out to be useless or repetitive. Hence DAI researchers strive to limit communication - designing agents to make reasoned decisions about which agents to transmit results, what results are relevant, and when they should be sent. So far, research towards controlling communication among result-sharing agents has largely focused on techniques for restricting the transmission of messages - organization structuring, meta-plans, planning, etc. From these trends one might conclude that the best form of communication in result-sharing networks is a narrow casting of results towards agents in need. To our surprise, we found that if the recipient agents used introspection, we were able to obtain the increase in solution quality payoff that comes with broadcast communication strategies, while paying little cost in the way of wasted or repetitive problem-solving.

The paper is organized as follows. Sections 2 and 3 describe a model of belief for our introspective agent. Section 4 details the corresponding implementation of the agent. Sections 5 and 6 describes our communication experiments using 5, 7, 11, and 15 agents.

2 The Introspective Agent Model

In this section we present an informal conceptual model of our introspective agents, focusing on the concept of belief. It has been reasonably argued by Konolige that an agent's belief system should be portrayed as a separate component of an agent's cognitive architecture.

As shown in figure 1, $\mathbf{A_i}$ is an agent composed of: $\mathbf{B_i}$, a belief subsystem, $\mathbf{I_i}$, an inferencing

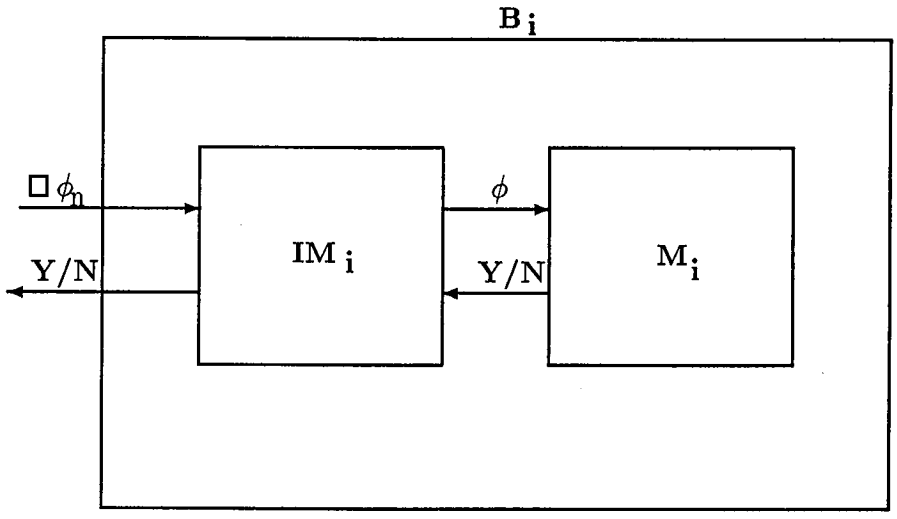


Figure 2:

subsystem for performing inferences, and $\mathbf{C_i}$, a communications interface for interacting with other agents. The belief system $\mathbf{B_i}$ consists of a finite list of facts the agent initially believes and a belief updating mechanism. The belief subsystem $\mathbf{B_i}$ interacts with the other subsystems of $\mathbf{A_i}$ as a fact repository, accepting propositions from the inferencing subsystem $\mathbf{I_i}$ and the communications interface $\mathbf{C_i}$, and as a query-answering device for $\mathbf{I_i}$.

As shown in figure 2, the belief subsystem of agent $\mathbf{A_i}$, can be described as a two-level introspective machine, with an introspective component $\mathbf{IM_i}$ and the heart of the belief machinery, $\mathbf{M_i}$, that maintains consistency and determines proposition membership. This model is similar in design to the conceptual models used by Konolige [12] to describe introspection in single agents. The queries presented to the belief subsystem are posed in some language L whose exact form is inconsequential except that there must be an explicit reference to an agent's own state of belief. Adapting the notation of Konolige, we represent these expressions by the form $\Box\phi$, which means agent $\mathbf{A_i}$ believes ϕ to be one of its beliefs. In general, ϕ may represent conclusions drawn by $\mathbf{A_i}$ or another agent, $\mathbf{A_j}$, as when ϕ has been communicated. We use the notation ϕ_n to represent a proposition originating from agent $\mathbf{A_n}$ when the origin of ϕ bears relevance to the discussion.

The response of the introspective belief machine $\mathbf{IM_i}$ is based on matching a query against the current fact list[2]. Queries of the form $\Box\phi$ presented to $\mathbf{IM_i}$ can be answered by presenting ϕ to the machine $\mathbf{M_i}$. This formulation of computational introspection is intuitively appealing as it appears to mimic human introspection[8]. While multiple levels of introspection are possible, for our purposes a single level of introspection suffices.

[2]Alternatively, a belief machine may try to derive ϕ, however, without a mechanism to incorporate some notion of limited resources, such machines are unimplementable as they are undecidable.

$$\Box\phi \equiv IM(M(\phi))$$

$$\Box\phi \begin{cases} M(\phi) : Y \rightarrow IM(\Box\phi) : Y \\ M(\phi) : N \rightarrow IM(\Box\phi) : N \end{cases}$$

$$\neg\Box\phi \begin{cases} M(\phi) : Y \rightarrow IM(\neg\Box\phi) : N \\ M(\phi) : N \rightarrow IM(\neg\Box\phi) : Y \end{cases}$$

When faced with the query $\Box\phi$, IM poses the query ϕ to M, and simply returns yes if M says yes, and no if M says no. From the cognitive perspective of the agent, "yes" means that the agent has considered its set of beliefs and has concluded that it believes ϕ, and therefore believes that it believes ϕ. In other words, $\Box\phi$ is one of the agent's beliefs. On the other hand, when presented with $\neg\Box\phi$, IM will respond no if M says yes, and yes if M says no. Now, "yes" means that the agent believes that it disbelieves ϕ. The agent doesn't believe that ϕ is a belief, so $\neg\Box\phi$ *is* a belief.

We have now defined an agent's cognitive outlook in terms of its state of belief or disbelief in the proposition ϕ. Together the set of believed facts and the set of disbelieved facts constitute what is *known* by the agent. We may define the set of known facts as the set of ϕ that satisfy a query in L of the form $\Box\phi \vee \neg\Box\phi$. We define the "known" modal operator $\Diamond$ as follows:

$$\Diamond\phi \equiv \Box\phi \vee \neg\Box\phi$$

By definition then, the set of *unknown* facts is the set of ϕ that satisfy a query in L of the form $\neg(\Box\phi \vee \neg\Box\phi)$, that is, the set of all ϕ that are in the agent's fact list regardless of state of belief. It follows that $\neg\Diamond$ can be used to describe what an agent does not know.

In a result-sharing system, propositions in the belief system may occur not only as a result of local inferencing but also as a result of communication. It is the case that $\mathbf{A_i}$ believes $\Box\phi_j$ as a result of a previous communication of ϕ by $\mathbf{A_j}$. The set of propositions an agent believes includes not only locally generated propositions but also propositions generated by other agents. It follows that agents may also reason about the state of belief of the propositions of another agent. If agent $\mathbf{A_i}$ believes $\Box\phi_j$ where $j \neq i$, then agent $\mathbf{A_i}$ believes that agent $\mathbf{A_j}$ believes ϕ_j. When belief-based agents reason with assumptions or use defaults, the conclusions that an agent holds may be retracted subsequent to the communication of those beliefs.

Using our model the situation may be described as

$$\begin{array}{lll} \text{For } \mathbf{A_i}, & \mathbf{IM_i}(\Box\phi_i) : N & \text{or } \neg\Box\phi_i \\ \text{For } \mathbf{A_j}, & \mathbf{IM_j}(\Box\phi_i) : Y & \text{or } \Box\phi_i \end{array}$$

where $\mathbf{A_j}$ believes $\mathbf{A_i}$ believes ϕ_i but $\mathbf{A_i}$ does not believe it believes ϕ_i.

This form of incoherency in result-sharing assumption-based reasoning systems is a result of physical inconsistency, where two agents share copies of the same belief (here originating with agent $\mathbf{A_i}$). In general, this type of incoherency may be addressed by using a global truth maintenance system that guarantees physical global consistency[2]. The distributed TMS systems of [2] and [19] both address this form of inconsistency.

Incoherency among result-sharing assumption-based reasoners may also take the form of logical inconsistency. Logical inconsistency occurs when two agents disagree about a particular conclusion or solution to a subproblem. For example, when agent $\mathbf{A_i}$ believes ϕ, which is counterfactual to agent $\mathbf{A_j}$'s belief in $\neg\phi$.

$$\begin{array}{lll} \text{For } \mathbf{A_i}, & \mathbf{IM_i}(\Box\phi) : Y & \text{or } \Box\phi \\ \text{For } \mathbf{A_j}, & \mathbf{IM_j}(\Box\neg\phi) : Y & \text{or } \Box\neg\phi \end{array}$$

More generally, logical incoherency may be described as $A_i : \Box\phi$ and $A_j : \Box\gamma$, where ϕ and γ are incompatible according to domain knowledge. The logical inconsistency may be detected when one of agents $\mathbf{A_i}$ or $\mathbf{A_j}$ communicate a counterfactual (e.g., $\mathbf{A_i}$ sends ϕ to $\mathbf{A_j}$) or when each of $\mathbf{A_i}$ and $\mathbf{A_j}$ report beliefs ϕ and γ to a shared belief system[3].

Whether or not the disagreement should be permitted, that is, whether or not the logical inconsistency can be characterized as a form of incoherency, depends upon the characteristics of the problem domain. An agent designer's demands for inter-agent consistency are based on overall problem decomposition, how individual subproblems are solved, and how subproblem solutions are synthesized to form the final network solution. These aspects of distributed problem solving differ greatly from one application to the next. Where problem decompositions give rise to subproblems requiring agreement, the inconsistency among beliefs of agents assigned to the subproblems can be viewed as incoherency. However, in many cases, it is important to preserve a "difference of opinion", as when several agents bring complimentary perspectives to bear on a problem.

If $\mathbf{A_i}$ holds a belief ϕ_i that is incompatible to the belief γ_j held by $\mathbf{A_j}$, then either (a) one of $\mathbf{A_i}$ or $\mathbf{A_j}$ must revise its beliefs or (b) there must be a representation of two distinct belief spaces - one representing $\mathbf{A_i}$'s assumptions and beliefs, and the other representing $\mathbf{A_j}$'s assumptions and beliefs. If $\mathbf{A_i}$ or $\mathbf{A_j}$ revises its beliefs, then we may effectively consider the agents to be reasoning in the same belief space. Justification-based global truth maintenance systems, such as the DTMS [2], enforce logical consistency among agents sharing the same belief space. That is, a global JTMS system guarantees:

$$\text{If } \mathbf{A_i} : \Box\phi \text{ then } \forall \mathbf{A_j}(j \neq i)\mathbf{A_j} : \Diamond\phi \Rightarrow \mathbf{IM_j}(\Box\phi) : Y$$

If agent $\mathbf{A_i}$ believes that it believes ϕ then every agent that knows ϕ also believes that it believes ϕ.

[3]As a result of agent distribution, it is possible that incompatible beliefs may be derived by two agents but not be detected unless an incompatible fact is communicated and therefore observed by one of the agents.

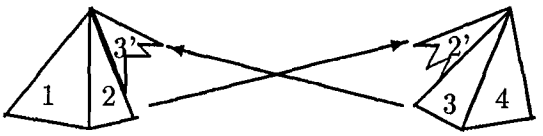


Figure 3:

Assumption-based global truth maintenance systems, such as the DATMS[19], allow agents to reason about multiple, possibly conflicting, sets of beliefs at once. The mechanisms of a global ATMS system guarantee only that conflicting beliefs may not be combined to create new beliefs. Each agent may have a number of belief spaces, the union of which may be inconsistent. These belief spaces may represent several alternative views the agent itself is considering or the alternative views of its fellow agents. This idea is illustrated in figure 3. The agent on the left currently maintains two belief spaces of its own, numbered 1 and 2, and one, numbered 3', that was communicated from the agent on the right. The agent on the right has 3 belief spaces as well; 3 and 4 are its own, while 2' was communicated.

3 Introspection as Control

Coherency and performance in a result-sharing network are threatened by a number of problems related to the fact that an agent unwittingly responds to facts placed in its working memory by another agent. For example, as a result of unnecessary inferencing on facts that have been communicated, agents may overload their working memory, increasing the cost of belief revision and pattern matching operations. Methods directed at curtailing incoherency in result-sharing systems focus on equipping the sending agent with representation and reasoning structures to make intelligent decisions about what results to send, to whom they should be sent, and when. Techniques for guiding transmission of results range from organizational structures, communication of metaplans, and planning, to actually simulating the reasoning of another agent to predict the problem solving relevancy of results. While the creation of an agent that can intelligently transmit results is a worthwhile goal, our experimentation indicates increases in coherency may be achieved in certain classes of result-sharing systems using mechanisms designed for the receiving agent.

By making states of belief and states of knowledge explicit and available to the agent, we provide it with the means to not only reason about the propositions it believes, but also about the state of beliefs concerning the propositions and its own state of knowledge. This is a powerful mechanism for accessing the full history of what an agent has known during the course of problem-solving. When implementing a result-sharing computational agent, the developer may use this mechanism to guide the recipient agent's selection of incoming results to explore thereby reducing the incoherency in result-sharing systems. In the next sections, we describe our implementation of result-sharing assumption-based reasoning agents and explore the relative performance of the network using various communication strategies with

and without introspection using 5, 7, 11, and 15 agents.

4 The Computational Agent Architecture

The agent model has been implemented using ROO, a DAI toolkit for building Rule-oriented cOOperative assumption-based reasoning systems[17]. The agent architecture consists of a social component, a problem-solving component, and a belief revision system, corresponding to the three components, $\mathbf{C_i}$, $\mathbf{I_i}$, and $\mathbf{B_i}$, of the agent model. Each agent is a rule-based deduction system, where knowledge is expressed in terms of rules and beliefs. Each agent's knowledge is divided into (1) social knowledge about communication and use of incoming results (2) problem-solving knowledge about the task domain - premises, assumptions, and inferences; and (3) knowledge about belief revision - the combinations of beliefs that are incompatible. Each agent has a DATMS to support belief revision in the distributed problem-solving environment, and message-passing software. As a result of message passing, externally deduced facts are added to an agent's belief database. Thus, an agent's interpreter component will match communication, problem-solving, and belief-revision rules against internally and externally generated inferences. As a result of the production system representation, the problem of choosing among competing defaults becomes a matter of conflict resolution[4].

Cooperation among production systems is realized using result-sharing. Individuals assist one another by exchanging observations and information based on different views of the problem. The different views or perspectives are a result of the different knowledge each agent has and from seeing a problem instance from different perspectives.

The agents are implemented using the Belief-Based Rule Language (BBRL), a production system language built on top of LISP[17]. Two problem solver interfaces to the belief system, or the DATMS, were used, one that allowed introspective fetch operations and one that did not. The introspective fetch operators consisted of 'BELIEVED', 'DISBELIEVED', 'KNOWN', 'UNKNOWN', and 'DISBELIEVED-OR-UNKNOWN'; a 'CONTRADICT' operator was available in both the interfaces for the agent to communicate a contradiction it has detected to the belief system. Like the belief revision systems of ART[5] and KEE[6], the DATMS represents each instance of a fact derivation in a separate node. Although we originally developed our fact labels this way for efficiency reasons, it has allowed us to maintain the full history of a fact's belief status, and therefore allows the agent to refer to a history of its problem solving. Hence the agent may not only refer to the set of currently believed facts, but also to those which are no longer believed, or which are not known. A prototypical CDPS system for global seismic monitoring was developed using BBRL and is discussed in subsequent sections along with a description of experiments with three communication strategies.

[4]This was first observed by Jon Doyle

[5]ART is a registered trademark of Inference Corporation.

[6]KEE is a registered trademark of Intellicorp, Inc.

Empirical Studies

5 Problem Domain - Global Seismic Monitoring

Seismic monitoring is one of the principle means available to verify compliance with treaties regulating underground nuclear weapons testing[7]. Seismic monitoring involves interpreting signals from seismometers placed on the surface of the earth to measure and record earth movement. A major challenge to seismic monitoring is the threat of evasion. One party may deliberately mask or muffle an illicit nuclear test to conceal it from the seismic network. In general, detection of an explosion may be impaired by devising a test whose signal level is below background noise level or whose signal occurs in conjunction with other seismic events.

To address the problem of evasion researchers are developing in-country networks and advanced sensor technologies, such as seismic arrays, to lower detection thresholds, and are exploring new methods of signal processing that are largely computationally intensive. Seismic monitoring techniques for a Low Yield Threshold Test Ban Treaty (LYTTBT)and a Comprehensive Test Ban Treaty (CTBT) will require the capability to detect and identify events having small magnitudes and occurring at distances less than 2000km from seismic stations [3].

However, lowering the detection threshold increases the number of events that must be analyzed. As the magnitude of the events we are interested in decreases, the number of seismic detections increases exponentially. For a Low-Yield Threshold Test Ban Treaty, where magnitudes may be close to 2.2, the figure reaches into the tens of thousands, just for the former Soviet Union region[18]. In a network with 30 stations, and if we receive on average 60 detections per day at each station, there will be 30*60*365, or 657,000, events per year. These seismic detections include a number of cultural "seismic" events (such as mining blasts, saw mills, power machinery, and airports), and natural seismic events (such as ocean waves, wind, and river ice breaking up in spring), as well as possible nuclear explosions. Each event must be analyzed to determine if it contains a clandestine nuclear test. The amount of data generated by such monitoring networks is overwhelming for the limited number of experts capable of making such interpretations. Therefore, reliable verification of a CTBT will require an automated system for interpreting and classifying seismic events. Here we focus on the problem of interpreting (locating) an event, although to a large extent the location may often aid in classification.

5.1 Why Seismic Networks

Networks of seismic stations are necessary to provide coverage and increase confidence in the detection capabilities. The seismic data a sensor station receives depends on the location of

[7]The ability to verify 100% compliance is a subject of debate, principally due to the problem of potential evasion. Verification goals for a CTBT are thus described as the following[9] "1) deter militarily significant testing programs 2) ensure significant attempts to continue underground testing and evade detection are identified in time to respond appropriately and 3) build confidence by minimizing the number of natural or man-made non-nuclear events that are misidentified as nuclear explosions or remain unidentified."

the energy source and the geological characteristics of the region on which the sensor site is located. It is not known *a priori* what site will receive what data, nor can the quality of data received at a site always be predicted. Using networks of sensor stations decreases the probability that a seismic event will go undetected.

Stated more formally, the signal interpretation problem can be described in terms of a set of events E_1, ..., E_N and a set of sensors S_1, ..., S_M[8]. Each event E_i is a point energy source that produces a signal e_i. For each combination of event E_i and sensor S_j there is a set of propagation paths P_{ij1}, P_{ij2}, ..., P_{ijk} along which energy from e_i can reach S_j. Each path P_{ijk} has a particular propagation-delay time t_{ijk} and transfer function h_{ijk}. Thus the signal at S_j is

$$s_j(t) = \sum_{i=1}^{N} \sum_{l=1}^{k} h_{ijl}(t) * e_i(t)$$

Another reason that networks of sensors are used is because the signal interpretation of individual stations may contain errors, and can only be used as an estima te. If more than one sensor site independently detects the same event, confidence in the detection is increased. However, conclusions about a particular seismic event are not made merely by counting the number of stations that witnessed an event, but by the strength of the seismic evidence as well. It should be noted that two seismic stations may actually witness different events.

5.2 Problem Decomposition

The studies in this section are based on a prototype system designed for the distributed interpretation of global networks of seismic stations with characteristics similar to those proposed by the U.N. Ad Hoc Group of Scientific Experts (GSE) [21,22] for verification of a Comprehensive Test Ban Treaty. The basic philosophy of the design is to build a distributed interpretation system that follows the naturally distributed interpretation by seismologists.

The seismologists' interpretation network consists of a number of individuals who interpret data for a particular geological or political region or sensor site. The knowledge to interpret the sensor data is naturally decomposed, with each seismologist reasoning about the data using specialized knowledge of a particular geological region and/or sensor technology.

To form an accurate interpretation of their sensor data, seismologists can exchange partial or full interpretations, and verify what they have seen or give pointers to other seismologists as to what they should be seeing. In no case does a seismologist conclude the presence or absence of an event unless it is supported by his/her own sensor data. This aspect of the network solution comes from the fact that seismologists may be viewing different seismic events. This is particularly important in the seismic treaty verification problem, where multiple artificially-induced seismic events, such as would occur in jamming or other evasion scenarios, might be used to evade treaty provisions. A single uncompromised station may be the only witness to a clandestine event.

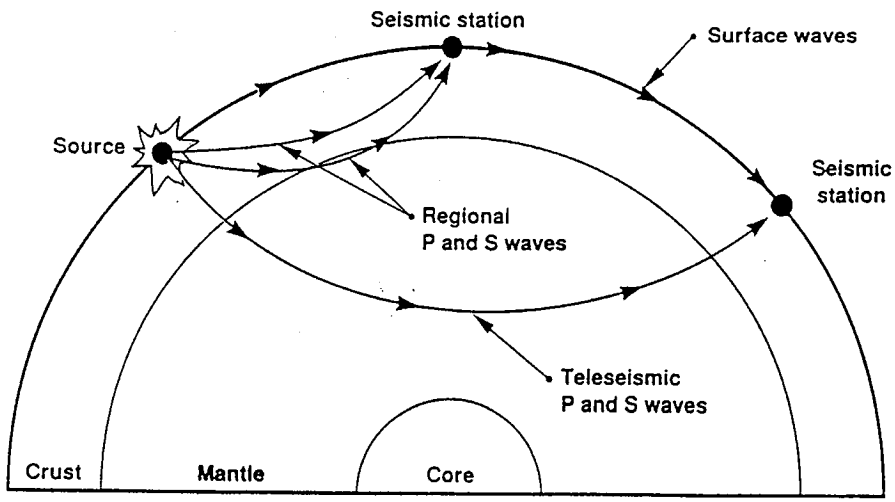
In essence, the architecture of our interpretation system maps the knowledge of a group of seismic experts onto a collection of interconnected workstations, one seismologist per

[8]The description assumes a homogeneous medium and a linear system.

workstation. Each agent is assigned the task of interpreting data for a particular station or group of stations within a geologically distinct area. In terms of cooperative behavior, agents work together towards the common goal of identifying any and all seismic events, and by exchanging partial results to improve solutions derived by each agent, thus improving the solution derived by the network. Individually, agents work to maximally improve their own interpretations. The form of the final solution consists of a bulletin listing each agent's interpretation by sensor station or geological region. The bulletin is then analyzed by a human agent for comparison with other intelligence information.

6 Problem Solving Strategy

The goal of seismic data interpretation is to create a hypothesis about the number and location of events which would generate the waveform features present in a seismogram. To understand how an interpretation is formed, it helps to visualize what happens during a seismic event.

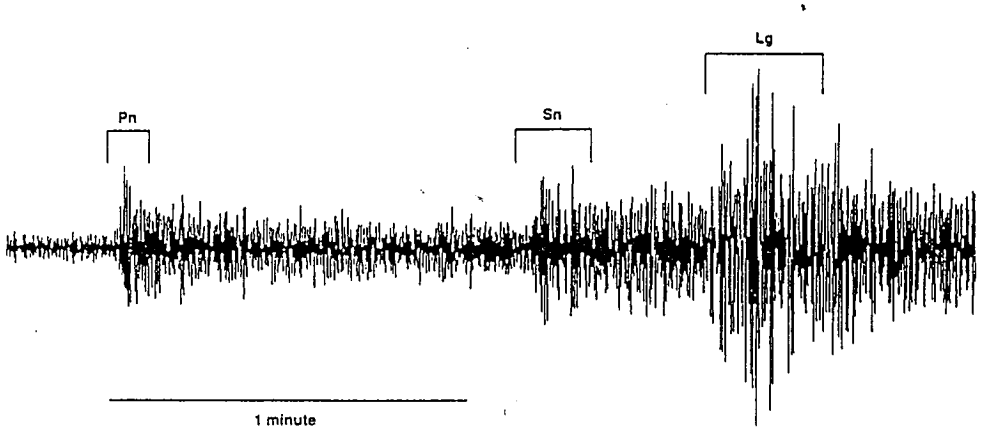


Seismic waves separate and take distinct paths through the earth

Figure 4:

As shown in figure 4, seismic monitoring looks at earthquakes and underground explosions using seismometers placed on the surface of the earth to measure and record earth movement. A seismic event is a lot like dropping a pebble in a pond in that it sets up a number of waves that propagate out from the point of the energy source. A seismic event produces a number of different kinds of waves that take different paths through the earth and travel at different rates, thus arriving at the seismic station at different times. As a result, the waves appear as

distinct features in the seismic signal. Figure 5 shows a seismogram recorded at the NORESS seismic array in Norway produced by a mining explosion in the former Soviet Union. Three distinct wave arrivals are shown.



This seismogram recorded at the NORESS seismic array in Norway was produced by a mining explosion in the former Soviet Union. Three distinct wave arrivals are shown: (1) a compressional wave (Pn) that propagated deep into the crust, (2) a following shear wave (Sn) that also propagated deep into the crust, and (3) a type of surface wave (Lg) immediately following.

Figure 5:

Longitudinal waves or *phases* arrive first and are called P waves or phases (P stands for the latin primus). These are followed first by transverse waves (phases) called S waves (S stands for secundus), and then by surface waves (phases) called L waves (phases) (named after their discoverers Love and Rayleigh). It is this difference in arrival time which enables a seismologist to estimate the distance to an event. The further apart the phases are, the farther they had to travel to reach the station. Direction is determined using a number of phase characteristics calculated with signal processing tools. The location of a seismic source is then estimated using distance and direction.

The process used to interpret seismic data, consists of 4 steps :

1. Segmentation - The seismogram is partitioned into intervals called "segments," containing detectable features. Each segment in the seismogram is defined by a begin and end time which bracket the interval.

2. Identification - Identify the phase type for each segment in the seismogram using measurable features. Measurable waveform properties such as propagation velocity, polarization and amplitude provide clues as to phase type.

3. Association - Identified phases are grouped together as coming from the same event. A complete interpretation consists of a number of events which together account for all detected segments.

4. Location - A location for a seismic event is estimated using distance and direction. Distance is estimated using associated phase arrival times. Direction is determined using a number of calculated phase characteristics.

These 4 steps may proceed sequentially for simple events. However, formation of an interpretation is further complicated when signals from multiple events arrive at the seismic recording station. Not all phases from the events will arrive, and those that do arrive can be of poor quality. In order to form an interpretation, the seismologist needs to identify phase type and associate phases as belonging to the same event.

Building the Solution

In essence, the seismologist forms an interpretation using a data-driven form of hypothesis and test. The general problem-solving strategy in generating a solution is to build and extend partial hypotheses using assumptions and computations (both symbolic and numeric) until either an inconsistency is derived or a complete hypothesis is formed. An initial hypothesis is formed based on the agent's data. To extend the hypotheses, individual agents rely on the use of assumptions to derive more information about their data, possibly using resource intensive numerical computations. In general, the strategy of an agent in evaluating its partial hypotheses is to make all possible inferences about the partial hypothesis trying to find a reason to discard it. Once an agent has attempted to find consistent solutions on its own, it considers information from other agents and continues to derive enough information (symbolic and numeric) to create a consistent solution. The problem is solved when the agent has found all complete and consistent solutions.

The process whereby a seismologist hypothesizes a partial interpretation and determines its validity can be described with assumption-based reasoning using a belief revision system. A partial interpretation is disproved if the belief in its validity leads to a contradiction. For example, it is known that a Pn phase must have a velocity of greater than 6Km/sec. Suppose that it is assumed that a phase is Pn in a partial interpretation and that gives rise to an inference that applies numerical processing and reveals the fact that the phase has a velocity less than 6Km/sec. Then it must be the case that the belief in the assumption is incorrect and the partial interpretation is discarded.

Agent Interaction

Each agent is assigned the task of interpreting data for a particular station or group of stations within a geologically distinct area. In terms of cooperative behavior, agents work together towards the common goal of identifying any and all seismic events and by exchanging partial results to improve solutions derived by each agent, thus improving the solution derived by the network. Individually, agents work to maximally improve their own interpretations.

Each agent first identifies the phases which it can *easily* discern. Some agents will identify one phase, while others will see multiple phases, or no phases at all. Any agent which has identified and associated phases makes an approximate source location and communicates

the interpretation including begin-time, phase-id, and source location to the other agents.[9] On the basis of the information received from other agents each agent either finds initially missed phases, or rejects mis-identified phases. Agents will add or reject phases on the basis of supporting evidence found in their data.

The advantages of communicating are twofold. First, results from other agents may be used to direct the use of signal processing techniques on their own data, thereby decreasing the amount of time and computation needed to detect the presence of an event. Second, results from other agents may be used to derive solutions which otherwise would not have occurred, thus improving the strength of the network solution.

The interactions of the agents in a distributed problem-solving network are made clear in the following example. For simplicity, we consider a 2 station network (see figure 7) where agent A1 is receiving data from a seismic array station and agent A2 is a standard 3-component seismic sensor station. The signal received by agent A2 (at station 2) contains substantial noise and requires extensive filtering to detect features actually present in the data. A2 has identified 1 phase. A1 has a strong clear signal and is able to identify and associate two phases. After receiving A1's tenative results, A2 uses filtering in directed fashion and is able to recognize and identify a second phase, thus establishing its own estimate of the location of the seismic event.

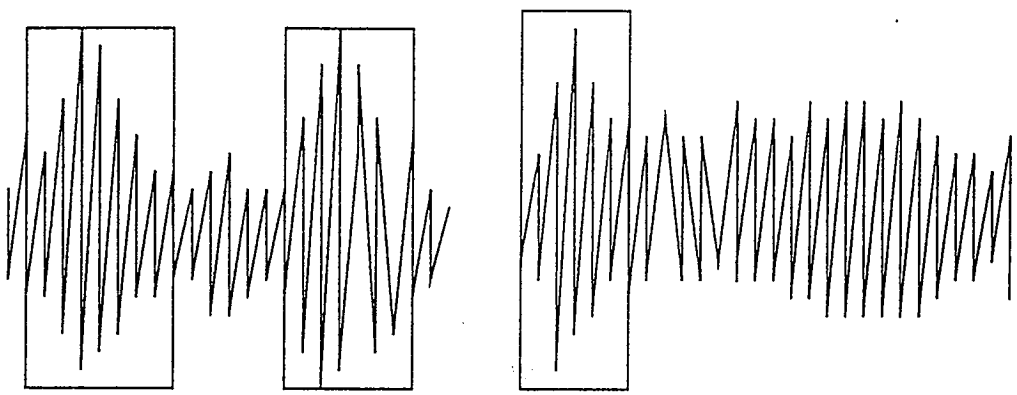


Figure 6:

7 Communication Experiments

In the following section we examine results of experiments that measure the effects of various communication strategies on the performance of our CDPS system using 5,7,11, and 15 agent (workstation) systems. The metrics for performance were the average CPU time required by each agent to generate an interpretation, size of working memory, CPU time for

[9]The decision to transmit the partial interpretations and to whom is varied according to the communication strategy being used.

pattern-matching and manipulation of assumption sets, number of inferences, and number of assumption sets.

Three types of communication strategies were explored: broadcast, directed-request, and undirected-request. In the broadcast communication strategy, results are exchanged on a volunteer basis, without solicitation on the part of the receiving agents. By contrast, the directed-request and undirected-request communication strategies involve making requests for information. Directed-request involves making requests to a dynamically determined list of agents. In the undirected-request strategy, agents simply broadcast a request for information.

7.1 Experiment Design and Selection of Test Data

To build agent knowledge bases, simplified seismic interpretation rules were adapted from the single-agent seismic event analyzer SEA [18], and from conversations with seismic treaty verification experts at Lawrence Livermore National Labs and at the U.N. in Geneva, Switzerland. Results from signal processing routines were simulated and I/O was avoided, except for inter-agent messages. Using a simplified knowledge base reduces the knowledge engineering effort required to construct the CDPS, without sacrificing the basic characteristics of the distributed seismic monitoring task. The data set was chosen to contain seismic events from which some agents make accurate identification and location (those without contradictions) and some may not (those with contradictions). For simplification, the data in each trial has a single correct interpretation.

The group of agents has no overall control hierarchy and each agent is data driven. Each agent's primary responsibility is to interpret its local data as best it can, and secondarily, to supply information that can be used by other agents to interpret their data. The decisions that determine which agents transmit results and to whom are made both on-line, according to the data an agent receives, and off-line, according to the communication policy.

The goal of our experimentation was that the results should be representative of the wide range of behavior we would expect to encounter in an operational seismic monitoring network. In a CDPS system composed of result-sharing non-monotonic reasoning agents, three classes of agents can be identified in terms of communication behavior: 1) agents that rely on the use of incoming information (partial hypotheses) to solve their subproblems, and do not transmit results, 2) agents that have enough information to solve their own subproblem and are capable of providing information to others, but encounter contradictions after a message has been transmitted, and 3) agents that have enough information to solve their subproblem, are capable of providing information to others, but do not encounter subsequent contradictions.

A fourth class of agent is possible. If a type 1 agent creates a subproblem solution using a shared result, it can subsequently supply help. However, this help may cause an indefinite loop of message sharing and problem solving when agents must find all solutions. For the general problem of seismic interpretation, we are interested in finding all solutions. We therefore ran our experiments with three classes of agents, rather than four.

When agents need only find a single, or few solutions, the cost of subsequently sharing a solution may include more expensive global truth maintenance, depending on how agents use the incoming messages to create their solution. Suppose the agent combines locally and non-locally derived facts, and subsequently communicates that solution. Then the agent that originally derived the non-local component of this solution no longer has knowledge of which agents have copies of its derivations. When an agent relies on the use of a communication history to guarantee physical consistency, we can no longer guarantee global consistency. The solution to this is simply to ensure global consistency by broadcasting TMS-related messages, rather than multi-cast using a communication history.

To implement the three classes of agents in the seismic domain, we vary the distribution of noise among the signals that agents receive (this is equivalent to having varied locations, some agents will encounter more noise than others, depending the path through the earth the signal took and the relative noise propagation variations.) This, in turn, varies an agent's ability to 1) provide other agents with seismic event information, and 2) to form a solution without cooperation.

For any given problem instance it is not known a priori which agents will need help, and which will encounter contradictions. We assume each type of agent occurs with a fixed probability. Each experimental trial models a particular seismic network scenario. In general, each trial consists of an N agent network, where X agents need help, Y agents have contradictions, and $N - (X + Y)$ have no contradictions. The experimental trials cover all possible combinations of numbers of the three types of agents for a given total number of agents.

The computational capabilities of each node in these experiments are equal, so the assignment of agents to nodes in the network is not significant. The results from a particular combination of agent types are independent of the particular assignment of agents to nodes in a network. This allows us to calculate the distributions over the complete set of all possible permutations of N agents selected from the three classes by performing only one experimental trial for each combination of N agents selected from three classes. The results taken from each distinct combination are weighted by the number of permutations it represents and the probabilities of the individual agent classes selected.

When plotting the data from the experimental trials, each possible combination of agents is represented by one trial. Although our model allows us to assign differing probabilities to the occurrence of each type of agent, the graphs assume each of the three types of agents are equally likely to occur. The value from each trial is weighted by the probability that this particular combination will occur, given the probabilities of the individual agents. For example, values for variables taken from an experimental trial involving five agents, where three agents encounter contradictions, and one agent needs help, is weighted less than values for variables taken from the trial where only two agents have contradictions, and two agents need help. This is because the first configuration represents $20 = 5!/(3!1!1!)$, possible agent to node assignments (or permutations), while the second configuration represents $30 = 5!/(2!2!1!)$, possible agent to node assignments, each assignment being equally likely.

In plotting the expected values (or averages) for an n agent network, we used the formula:

$$Expected\text{-}Value(Agent\text{-}CPU\text{-}Time) = \sum_{i=1}^{C} (Value\text{-}from\text{-}trial) * PR(trial)$$

where *Value-from-trial* for an n agent trial is defined as $\sum_{i=1}^{n} CPU_{Agent_i} * \frac{1}{n}$, the probability of a particular network configuration occuring, $PR(trial) = \frac{Number-of-permutations}{Total-Permutations}$, and C is the number of combinations or experimental trials for a network of size n. The value of C, or the number of trials, for each network of size n is

$$\frac{1}{(X-1)!} * \prod_{i=1}^{X-1} (Y+i)$$

where X is the number of agents that have noise (need help), and Y is the number of agents that encounter a contradiction after transmitting a message. Our graphs indicate an expected value taken over C trials for each of 4 network sizes: 5, 7, 11, and 15.

Experimental Trials

Number-of-Agents	Number-of-trials
n	C
5	21
7	36
11	78
15	136

The experiments were conducted using MATE, the Multi-Agent Test Environment[15]. Using MATE, a suite of experiments may be run without human intervention. MATE facilities include automatic network resource allocation, monitoring and executing sequences of experiments, and automatic archival of experiment results and messages.

7.2 Performance and Solution Quality

We define the performance of a problem-solving system in terms of the computational resources, such as time or memory, required to produce solutions of a given quality. In distributed systems composed of autonomous collaborating agents, performance is determined both by local problem solving and communication strategies. In a distributed system, however, bandwith limitations make it impossible for any one agent to know the entire state of the system at any point in time. As a result, agents must operate with uncertainty. The performance of the system as a whole will necessarily be suboptimal from the perspective of an imaginary, omniscient agent[20].

The challenge is to construct local problem solving and communication strategies that can achieve a desired solution quality, while minimizing the costs of cooperation in the face of

limited knowledge. DAI researchers address this problem by increasing the sophistication of their agents, employing methods such as organizational structures, meta-plans, and reasoning about communication. The goal of our efforts is the realization of a new level of abstraction in the knowledge required to design such problem solving agents, the *social level*[11]. The success of this social level depends on striking a balance between the cost of computations to support it, and the overall performance of the system.

As described in the previous section, our motivation for bringing together autonomous seismic interpretation systems is that through cooperation there is the promise of improved solution quality. Solution quality can be expressed in terms of *accuracy* and *completeness*. The accuracy of a solution involves both the correct identification of phase or feature types, and the correct location determination. Completeness refers to the number of waveform features present in the raw data that are accounted for, or predicted by, the agent's final hypothesis.

7.3 How the Communication Strategies Fared

The purpose of this experiment is to determine the effect of communication strategies on the performance of cooperative assumption-based reasoning networks. Three different strategies are evaluated: a) broadcast, b) undirected-request and c) directed-request. The protocol for each communication policy is the following:

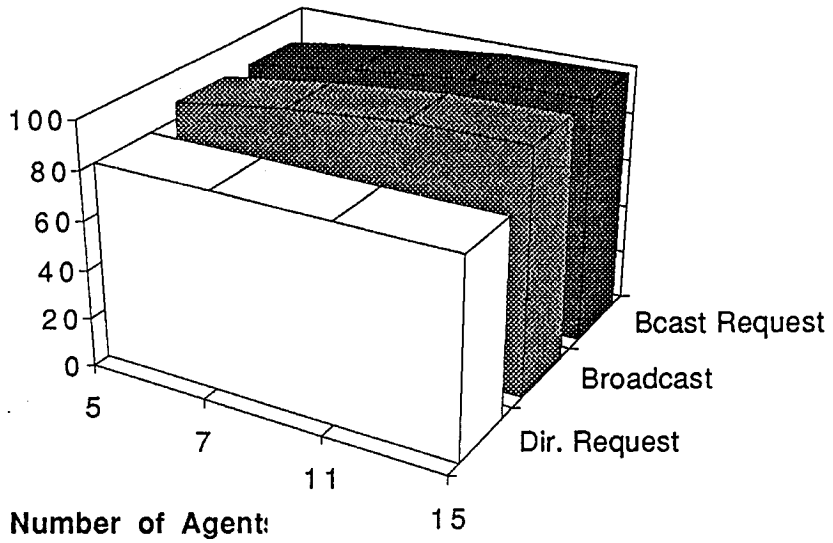
broadcast Agents broadcast results. Agents receive results without requesting them (i.e., a junkmail approach to communication, agents advertise results)

undirected-request An agent broadcasts a request for results to all other agents. Agents send results only to agents making requests for information.

directed-request Send information only to agents making requests, where requests are directed to a specific list of agents. The list of agents may be determined at run-time or may be static. Agents are programmed to consult at most 3 other agents.

Figure 7 shows solution quality for each of the communication schemes as measured by the percentage of agents with complete solutions. Predictably, the broadcast-based schemes had more complete solutions than the directed request scheme. This is largely due to the uncertainty in determining either a priori or on-line which agents can provide help. Examining the graph, we can see that for the broadcast-based strategies, as the number of agents increase, the percentage of agents with solutions increases. This is because the more agents there are, the more likely it is one of them has a correct and complete solution to share. By contrast, the number of agents has no effect on the improvement in solution quality for a directed request communication scheme. It is related, however, to the number of requests an agent makes.

Figure 8 gives various resource measurements for each of the three communication schemes run on a series of trials for 5,7,11, and 15 agents. The graphs display the performance of an



Solution quality as measured in percent of agents with complete solutions.

Figure 7:

individual agent averaged over each of the different network configuration for each different size network.

Examining the shapes of the distributions of the various performance indicators for each communication strategy suggests that the number of result messages received has the most significant effect on agent performance. This explains why the directed-request systems have superior performance. However, whenever the agents broadcast, solution quality is maximized in terms of the number of features that agents are successfully extracting from the seismic data. Although broadcast and undirected-request had fewer agents with incomplete solutions than in the directed-request system, these systems pay a penalty in the total number of inferences each agent must perform. The undirected-request system requires fewer inferences per agent (figure 8a), and as a result requires fewer working memory elements (figures 8b) assumption sets (figure 8c) and total CPU time[10] (figure 8d). In terms of resource metrics, directed-request has the advantage of being much more efficient. We were therefore driven to find a mechanism to reduce the costs associated with broadcast-based schemes.

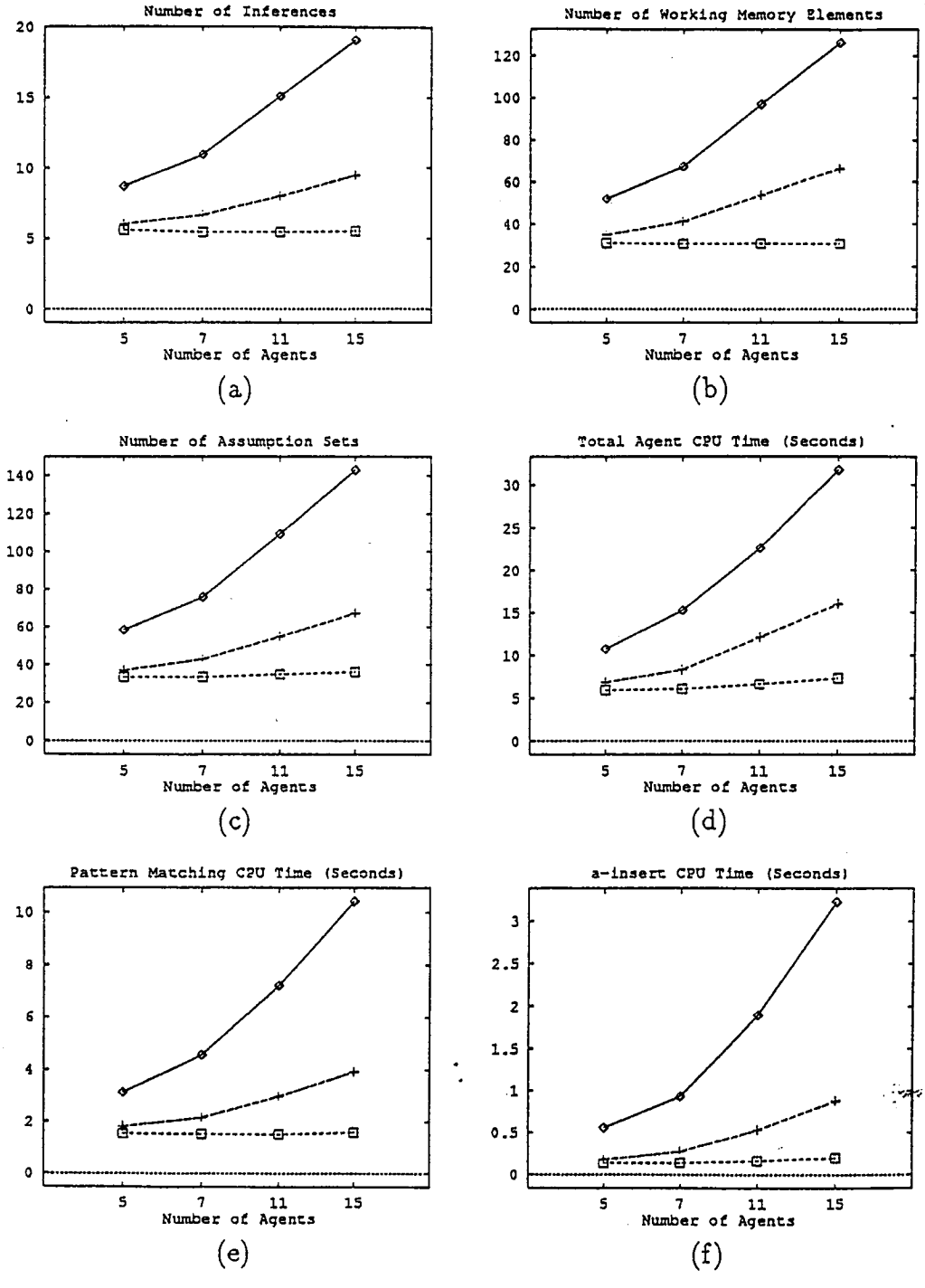
7.4 Introspection and the Problem Solver Interface

In the previous experiments we saw that reducing the number of result messages can cause dramatic improvement in efficiency, at a cost in solution quality. In this section we explore the use of an extended DATMS interface, allowing agents to reason not only about the currently believed facts in working memory, but about the state of facts in working memory, to control inferencing and thereby improve performance.

In a directed-request communication scheme, agents rely on domain-specific knowledge to direct requests for information. [11] This approach allows agents in need of help to receive

[10]Agents ran uniformly on Sparc-1 processors.

[11]Although in our experiments the list of agents to whom an agent direct requests is created based on



Performance measures of three communication policies plotted as the number of agents varies. Broadcast is shown with solid lines and diamonds, Undirected-Request is show with long dashes and crosses, and Directed-Request is shown with short dashes and squares.

Figure 8

useful result messages most of the time, while broadcast and undirected-request communication strategies guarantee agents will receive useful results all of the time (provided there is at least one agent with a useful result to share). However, undirected-request and broadcast strategies lead to inefficiencies because all incoming results must be explored when there may be only a few solutions. Agents rediscover inferences and contradictions when incoming results are co-referential (two beliefs or sets of beliefs are co-referent if they refer to the same object or concept in reality.)

The general problem of co-referentiality occurs in multi-agent problem-solvers when agents represent both the beliefs of other agents and their own beliefs in working memory. The simultaneous representation of multiple "views" requires a representation that distinguishes one agent's beliefs from other agents. However, a belief representation scheme that maintains agent distinctions will by definition have no syntactic means to determine whether beliefs are co-referent.

This aspect of non-probabilistic multi-agent problem-solving is particularly devastating when agents each rely on the use of an ATMS-like belief subsystem and must find all solutions. When an agent receives an incoming result message, new beliefs (imported beliefs) are established in working memory, and the DATMS responds by inserting the accompanying new assumption sets into the assumptions data base (using the ATMS terminology [5], a new node is established, with a label containing a single environment that was imported.) This has the effect of forcing the problem-solver to explore new contexts and determine whether or not the consequences of the new beliefs lead to a consistent conclusion. This is regardless of whether or not these beliefs may in actuality be equivalent to beliefs that were previously considered. As we saw in the first experiment, the efficiency of the broadcast and undirected-request communication strategies is related to the number of incoming results they are forced to consider.

One way to address this problem is to use an "EQUAL" relation between beliefs or sets of beliefs that are viewed by the agent as equivalent. The idea is when a belief or belief set is found to be co-referent, it is bound to the equivalent beliefs in memory by the EQUAL relation. This has the effect of establishing a single view by "merging" equivalent beliefs in working memory. This approach, while conceptually appealing, is complicated to implement. It is unclear how difficult it may be to determine in a domain independent fashion that two beliefs are co-referent. An elaborate scheme is necessary to compute new assumption sets for beliefs bound by an EQUAL to prevent work on equivalent beliefs when contradictions involving the beliefs are discovered. A simpler way to reduce these inefficiencies is to extend the DATMS interface to allow agents more control over the contexts they explore.

Traditionally, a TMS provides the problem-solving component with access to the currently believed set of facts. This is accomplished through the BBRL LHS clause. This simple interface allows problem-solvers to reason only about what is currently believed. Giving agents a means to reason about what is disbelieved, what is known (believed or not),

---

domain knowledge, the communication rule syntax accepts the invocation of any lisp function returning a list of agent names. It is conceivable the list of agents may be determined by historical records indicating previous success rates of agent communication, or a simple list of predetermined agents.

what is unknown and so on, allows agents to base problem-solving decisions on their own state of knowledge, as opposed to simply what is believed in working memory. Agents can determine not to pursue an incoming result based on failing to generate a consistent solution with a similar set of beliefs. We extend the DATMS interface for each agent to include

Believed - the facts that have an assumption set that is consistent

Disbelieved - the facts do not have an assumption set that is consistent

Known - the facts are either believed or disbelieved

Unknown - the facts do not yet exist in working memory

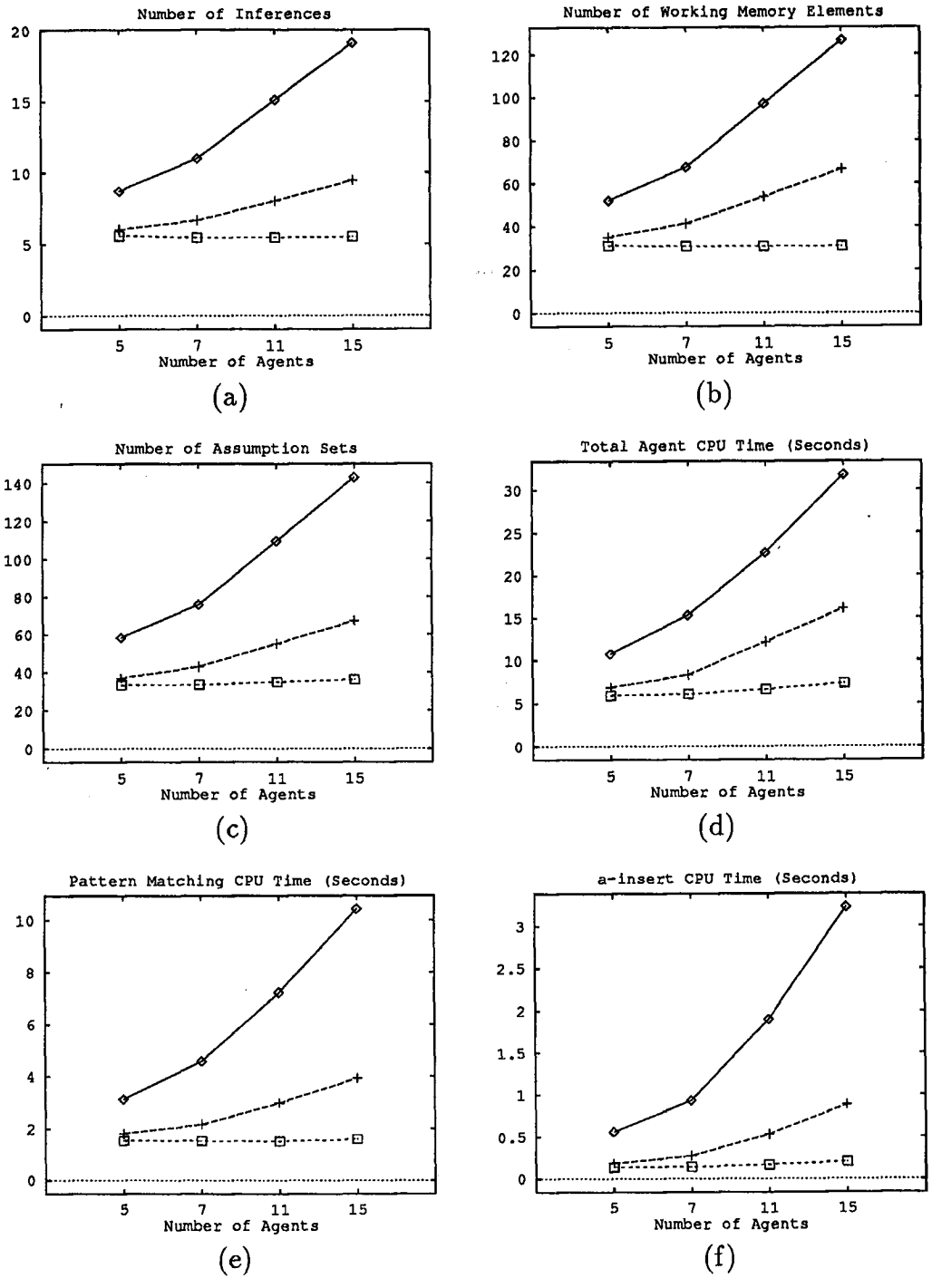
UnknownOrDisbelieved - the facts either do not exist or are disbelieved

These additional functions appear as predicates that can occur in the condition part of an inference rule. The additional predicates are implemented as a filter that is applied to rule instantiations once all variables in the Believed clause have been bound. As a result the justification for any fact is still monotonic and includes only the facts in a Believed clause. These concepts are best illustrated by example.

```
(INF-RULE LOOK-FOR-EVENT
     ((UNKNOWN  ((EVENT  instance    $E1)
                 ($E1    has-loc     $L1))
      (BELIEVE  ((FOREIGN-EVENT instance    $E2)
                 ($E2    has-loc     $L2)
                 ($E2    begin-time  $B2)
                 ($E2    end-time    $END2)
                 ($E2    origin      $AGENT)
                 ($AGENT sensor-type 'Array))))
     (ASSUME (make-event ((begin-time $B2)
                          (end-time   $END2)))))
```

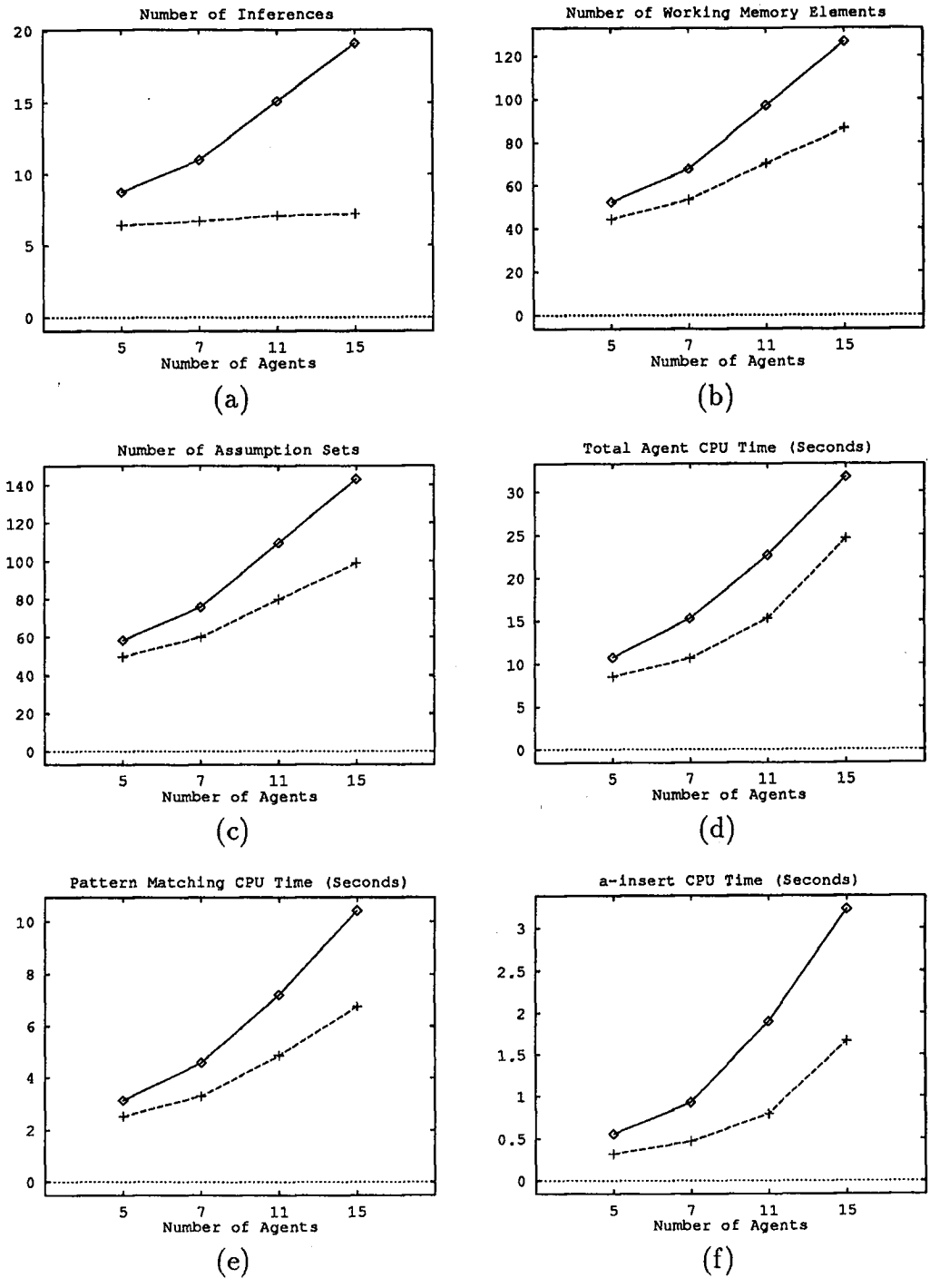
The rule LOOK-FOR-EVENT indicates the condition that the agent believes there is an event, but it does not know its location. The agent tries to use information about an event (FOREIGN-EVENT), seen by another agent using an array of sensors, to derive enough information to build its own interpretation of the event. Using the begin and end times as assumptions, the agent narrows the window of seismic data on which to apply signal processing in an attempt to complete its partial interpretation about the seismic event.

Figures 9 through 12 show results of experimental trials in each of broadcast, undirected-request and directed-request communication strategies. Graphs indicate average values per agent.



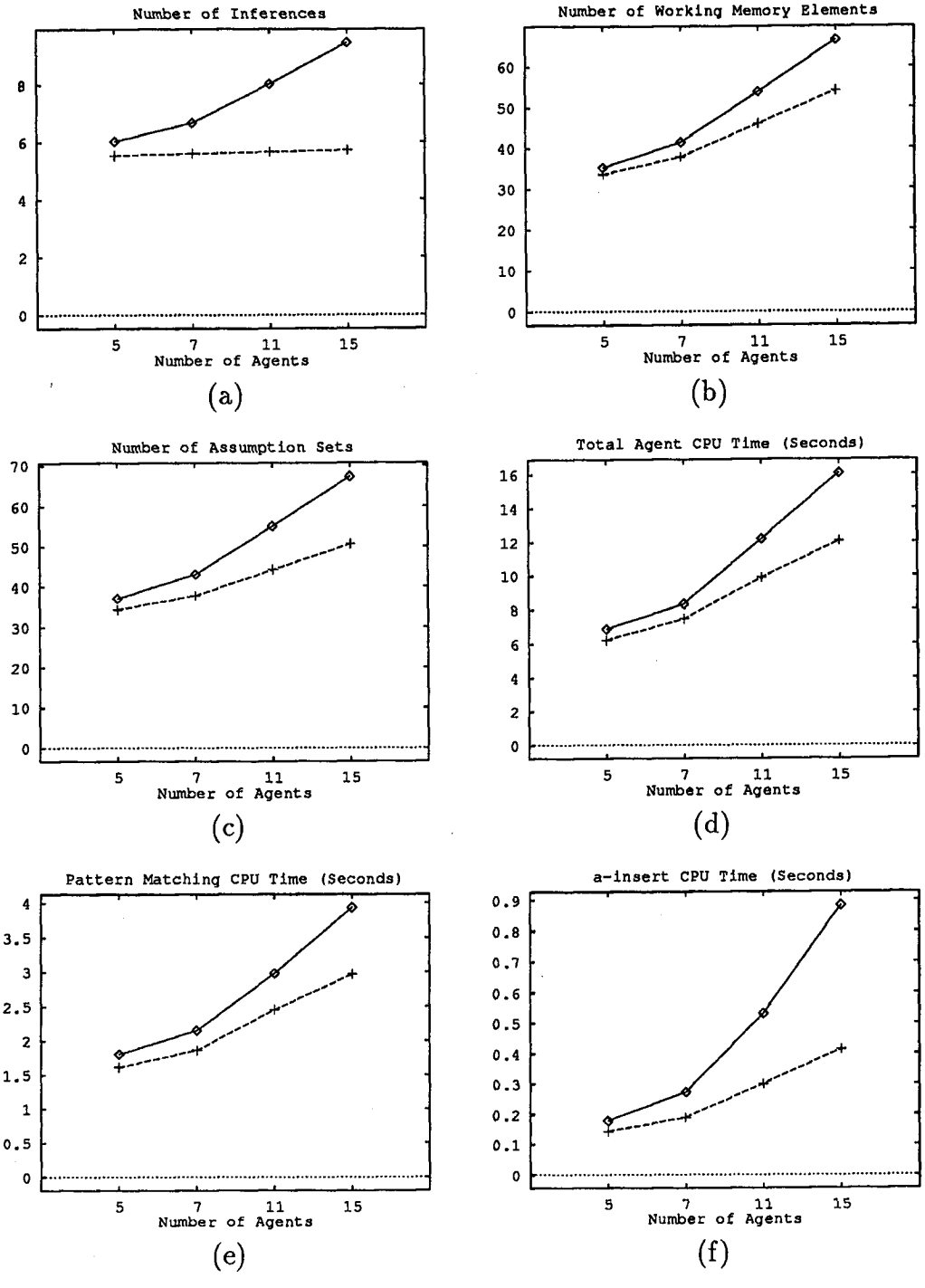
Performace plots for all three communication policies with introspection Broadcast is shown with solid lines and diamonds, Undirected-Request is show with long dashes and crosses, and Directed-Request is shown with short dashes and squares.

Figure 9



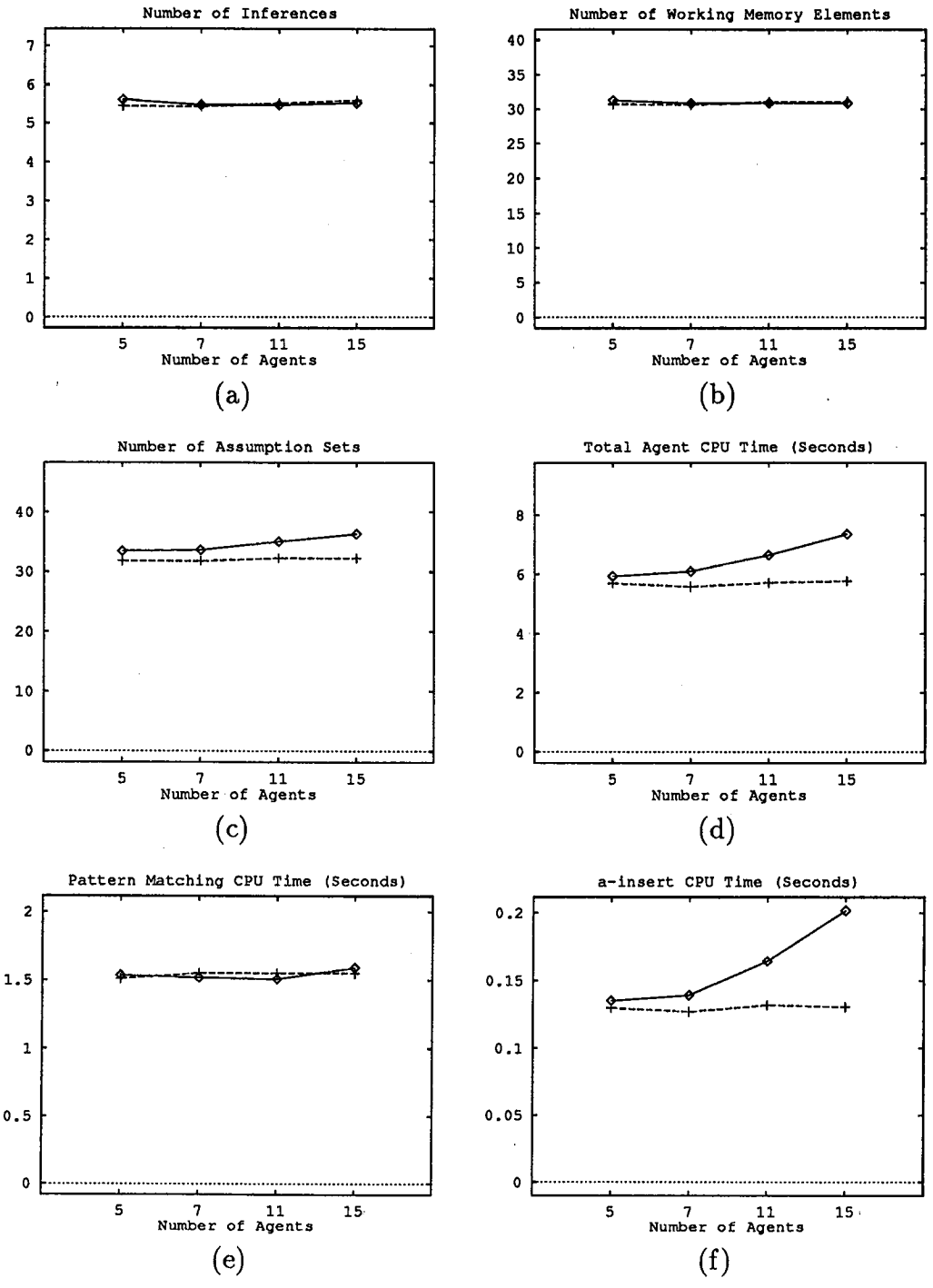
Dotted and solid lines show performace plots for the "broadcast" communication policy with and without (respectively) introspection.

Figure 10



Dotted and solid lines show performace plots for the "Undirected-Request" communication policy with and without (respectively) introspection.

Figure 11



Dotted and solid lines show performace plots for the "Directed-Request" communication policy with and without (respectively) introspection.

Figure 12

The extended DATMS interface succeeds by slashing total agent cpu time for the broadcast system agents (figure d) by almost one third, as compared to trials without it. It also succeeds by greatly reducing the number of inferences (figure 10a), the number of assumption sets and working memory elements(figures 10b and 10c) and the amount of CPU time spent on assumption sets and pattern matching (figures 10e and 10f). The extended DATMS interface has a similar effect on efficiency in the undirected-request system (figure 11). However, the introspection mechanism has little effect on directed-request since agents are already operating at near optimal levels (figure 12).

In both broadcast and undirected-request systems we avoid needless inefficiency because agents can ignore transmitted beliefs that would lead to rediscovering solutions or contradictions. Conventional TMS interfaces avoid only those beliefs that have been shown to be contradictory by conflicts in assumption sets. This finding supports the observation of [5] on the need to extend problem-solver control for single-agent systems "if there are a large number of solutions, of which only a few are required...." In the multi-agent case, the more results an agent receives, the more effective the extended TMS interface becomes.

Examining figures 9 through 12, these experiments indicate the interface has a significant effect on efficiency in both broadcast and undirected-request strategies, with little improvement in the directed-request strategy. Despite this performance improvement in the broadcast and undirected-request strategies, directed-request remains superior to both systems in terms of resources consumed. The remaining source of inefficiency in the other two systems is that although it is possible to avoid working on uninteresting solutions, it is not possible to avoid receiving an uninteresting solution. This is particularly relevant in the broadcast system. If there is a large number of agents (say greater than fifty) then working memory may be consumed by incoming messages, with a prohibitive cost of cooperation. To improve solution quality while maintaining efficient performance, a fourth strategy for communication would be to first try to find useful results using directed-request, followed by undirected-request in case of failure.

8 Summary and Conclusions

We explored the use of introspection as a means for control in result-sharing assumption-based reasoning systems. We presented an informal conceptual model of the introspective agent and examined broadcast, undirected-request, and directed-request communication strategies. Using broadcast schemes, result-sharing agents can maximize the number of agents that will find solutions. The percentage of agents that find solutions increases as we add more agents. We found recipient agents can avoid penalties usually incurred by broadcast strategies through the use of introspection and a history of knowledge state to focus problem-solving. The effectiveness of this mechanism improves as the number of agents increases. We found that while directed-request was the most efficient, reasonable efficiency can be achieved with broadcast or undirected-request communication, which maximize solution quality using an introspective interface to an agent's belief system. This mechanism represents a shift in thinking about the design of result-sharing agents by placing the burden of communication on the receiving agent rather than the sending agent.

References

[1] A. Bond and L. Gasser (Eds.), *Readings in Distributed Artificial Intelligence*, Morgan Kaufman Publishers, Inc., San Mateo, CA, 1988.

[2] D. M. Bridgeland, and M. N. Huhns, "Distributed Truth maintenance," *Proceedings AAAI-90*, Vol. 1, pp. 72-77, 1990.

[3] O. Dahlman and H. Israelson, Monitoring Under- ground Explosions, Elsevier, New York, NY, 1977.

[4] R. Davis and R. G. Smith, "Negotiation as a Metaphor for Distributed Problem Solving," *Artificial Intelligence* **20** (1983), 63-109.

[5] J. deKleer, "An Assumption Based TMS," *Artificial Intelligence* **28** (1986), 197-224.

[6] E. H. Durfee and T. A. Montgomery, "Coordination as Distributed Search in a Hierarchical Behavior Space," IEEE Trans. on Systems, Man, and Cybernetics, 21(6), November 1991.

[7] L. Gasser, and M. N. Huhns, *Distributed Artificial Intelligence, Volume II*, Pitman Publishing, London, 1989.

[8] J. Hintikka, *Knowledge and Belief*, Cornell University Press, Ithaca, New York, 1962.

[9] W. J. Hannon, "Seismic Verification of a Comprehensive Test Ban," *Science*, Vol. 227, January, 251-257, 1985.

[10] M. N. Huhns, *Distributed Artifical Intelligence*, Pitman Publishing, London, 1987

[11] N. Jennings, "Towards A Cooperative Knowledge Level For Collaborative Problem Solving," Proc. 10th European Conference on Artificial Intelligence, Vienna, Austria, 1992.

[12] K. Konolige, "A Computational Theory of Belief Introspection", *Proceedings IJCAI-85*, pp. 502-508, 1985.

[13] V. R. Lesser and D. D. Corkill, "The Distributed Vehicle Monitoring Testbed," *AI Magazine*, Fall 1983, 15-33.

[14] C. Mason, "Collaborative Networks of Independent Automatic Telescopes," *Astronomy for the Earth and Moon*, D. Pyper-Smith, Ed., Astronomical Society of the Pacific, 1993.

[15] C. Mason, "MATE: An Experiment While You Sleep Test Bed", *Proceedings AAAI-93 Workshop on Collaborative Design*, AAAI Press, 1993.

[16] C. Mason, "Cooperative Distributed Problem-Solving with the DATMS:First Results," NASA Ames Technical Report, FIA92-X, 1992.

[17] C. Mason, "ROO: A DAI Toolkit for Belief-Based Reasoning Agents", Proc. Intl. Conference on Cooperative Knowledge-Based Systems, Keele, England, 1994.

[18] C. Mason, R. Johnson, R. Searfus, D. Lager, and T. Canales, "A Seismic Event Analyzer for Nuclear Test Ban Treaty Verification" *Proc. Third Intl. Conf on Applications of Artificial Intelligence in Engineering*, Stanford, CA, August 1988.

[19] C. Mason, and R. Johnson, "DATMS: A Framework For Distributed Assumption Based Reasoning," in *Distributed Artificial Intelligence Volume II*, L. Gasser, and M. N. Huhns, Eds., Pitman Publishing, London, 1989.

[20] J. Pasquale, "Intelligent Decentralized Control in Large Distributed Computer Systems," Ph.D. Thesis, Dept. of Computer Science, University of California, Berkeley, 1988.

[21] United States Delegation, "United States Plans For Participation In The Second GSE Technical Test," Report GSE/US/53 to the Group of Scientific Experts for the United Nations Conference on Disarmament, March 1989.

[22] United States Delegation, " Concepts for Communications in the Global System," Report GSE/US/54 to the Group of Scientific Experts for the United Nations Conference on Disarmament, March 1989.