A Dynamic Organizational Architecture for Adaptive Problem Solving

Les Gasser
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0782 USA
(213) 740-4510 gasser@usc.edu

Toru Ishida
NTT Communications and
Information Processing Laboratories
1-2356, Take, Yokosuka-shi, 238-03, Japan
ishida%nttkb.ntt.jp@relay.cs.net

1 Introduction

There is an essential correspondence between the architecture of a distributed problem-solving system, the structure of the problems it solves, and the environmental conditions under which it solves them. In a dynamic world, such as one populated by multiple agents in a changing environment, this correspondence must be maintained by dynamic adaptation. There are four ways to disrupt or to maintain this correspondence: alter the structure of problems, the environmental conditions, the problem-solving architecture, or the goal-knowledge-action relationships (e.g., task and skill allocations or types of knowledge).

A well-known AI approach to adaptive problem-solving systems has been to use a *fixed problem-solving architecture* which responds to environmental change by *restructuring problems* (e.g. by relaxing problem constraints, abstracting search spaces, or changing decision criteria dynamically, as in several resource-bounded problem-solving AI systems [Lesser 88, Schwuttke 91]) or by long term *adaptation of problem-solving knowledge* (learning). Distributed AI researchers, in contrast, have begun to investigate problem-solving systems that *restructure their own macroarchitecture*, to add to the repertoire of adaptive responses. In general, these systems have comprised a fixed collection of problem-solving *agents*, each of which has a stable microarchitecture. The set of interagent relationships or *organization* of agents is changed, yielding a dynamically adaptive macroarchitecture. The best-known early dynamic macroarchitecture is the Contract Net system [Davis 83] in which manager-worker relationships evolved opportunistically based on the structure of the given problem decomposition, the availability of free agents with required capabilities, and the outcomes of mutual selection processes governed by a bidding-contracting protocol. Later, researchers using the DVMT system [Corkill 82, Durfee 87a,b] proposed mechanisms such as *metalevel control*, *partial global planning*, and *organization self-design* (OSD) as means for dynamically structuring the macroarchitectural relationships of a problem-solving system.

In earlier OSD research, we introduced *organizational knowledge*, used it as a basis for 2 *reorganization primitives* (composition and decomposition), and demonstrated their value as adaptive mechanisms for problems with time constraints [Ishida 90a]. Our reorganization primitives dynamically vary a system macroarchitecture by adjusting inter-agent relationships, the knowledge agents have about one another, the size of the agent population, and the resources allocated to each agent. In our prior work, reorganization was triggered by reorganization request messages from an outside observer, not by the organization itself. We now give agents the knowledge they need to reorganize themselves. We introduce additional organizational knowledge called *agent-organization relationships*, and a new agent microarchitecture, the *self-organizable, distributed production-system-based agent* (SDPSA). With these new concepts, we can examine some novel ideas about the nature and representation of organization. We also report our study of the relationships between organizational knowledge and communication-coordination overheads. As before, our formal problem-solving model and OSD approach are based on production systems, but our OSD approach is, in general, not limited to production-rule systems. We use production rules as a general model of individual problem-solving actions, because they have been shown to be useful as abstractions of organizational and problem-solving processes of many kinds (cf. [Zisman 80]). For this reason, we use the terms "production rule" and "problem-solving action" interchangeably below.

2 Organization Self-Design (OSD)

We are interested in OSD for *problem-solving organizations* (PSOs), embedded in an *environment*. The products of a PSO are *solutions* to *individual problem-solving requests* issued from the environment. Changes in the relationship between a PSO and its environment can create pressure for reorganization in the PSO. Pos-

sible pressures include 1) demands for change in the organizational performance level (e.g., manifested in new quality levels or shorter/longer response time requirements), 2) change in the level of demand per solution type (e.g., manifested in more or fewer problem-solving requests per unit time, or changes the mix of problem types), 3) changes in the level of demand for resources that the organization shares with others in its environment.

In our model, problem-solving requests issued from the environment arrive at the organization continuously, at variable rates. To respond, the organization must supply meaningful results within quality and performance standards, which are also set by the environment and which also may vary. These variations provide the changing conditions to which the organization must adapt, using organizational knowledge and OSD primitives.

2.1 Architecture and Reorganization Operations

Figure 1 shows the basic agent microarchitecture for SDPSAs. Each agent repeatedly executes a modified *match-select-act* problem-solving cycle, in which production rules model domain problem-solving actions. Reorganization decisions are made using organizational knowledge, detailed below. Each reorganization decision invokes one of two reorganization primitives, which work by changing the relationships between knowledge (rules and working memory elements) and action (match-select-act processes) in the organization (i.e. by changing the definitions of agents). Figure 2 illustrates the process of OSD.

Decomposition breaks the correspondence between a collection of problem-solving rules and their interpreter, by creating a second interpreter (agent) for some rules, inserting communication actions among agents, and enforcing synchronization among dependent problem-solving actions using a specialized deadlock-free protocol (see [Ishida 90a]). The extra resources increase intra-problem parallelism and may improve performance, but coordination overhead is also increased. Decomposition can also increase organizational throughput when multiple problem requests can be processed in a pipeline (increasing *inter*-problem parallelism).

Composition combines two interdependent *neighbors* (agents with action interdependencies) into one, creating a "closer" relationship among groups of problem-solving knowledge by removing agents and interagent messages and freeing both computation and communication resources. Maximum decomposition does not necessarily yield the best response time or throughput, due to coordination and communication overheads. Thus composition may also actually improve performance where coordination overhead is high.

Since the aims of composition and decomposition are independent, both kinds of reorganization can be performed simultaneously in different parts of the organization—both problem-solving and organization self-design are treated as decentralized processes. During the reorganization process, deadlock never occurs, because reorganization does not block other agents' domain problem solving or reorganization (see [Ishida 90b] for details).

2.3 Organizational Knowledge

We have established two types of organizational knowledge. *Agent-agent relationships* represent the dynamic state of dependencies and interferences among knowledge in agents throughout the organization, and have been detailed elsewhere [Ishida 90a, Ishida 90b]. *Agent-organization* relationships comprise *local statistics*, *organizational statistics* and *reorganization rules*, which we now define.

First, we restrict ourselves to problem spaces in which there is a monotonically decreasing relationship between the solution quality of the goals and the probability of finding a goal using a random search (i.e., higher-quality goals always imply lower probability). Thus raising the required solution quality for a problem space always increases the average amount of search—i.e., the number of problem-solving actions (rule firings)—needed to find a goal in any problem instance[1]. Let the *goal density* D of a problem space be the prior probability of reaching a goal with a fixed level of effort in a random search of the problem space. Thus, problem spaces with lower D require greater search effort on the average.

Let $T_{deadline}$ be a time constraint placed on the problem-solving process for any problem instance (measured in problem-solving cycles). Let:

$$PERF_{required} = k \cdot D \cdot T_{deadline}$$

Of two problem solvers in the same space, one with better control heuristics will expend less search effort for a given D. k accounts for the size of the problem space, and for the heuristic performance of the ruleset in focusing search, and allows us to normalize performance. A particular level of required performance is a function of desired quality and time constraints. By our definition, a *lower* value of $PERF_{required}$ is *more difficult* to achieve. "Better" needed performance (i.e. a *lower* value for $PERF_{required}$) can be specified by raising the solution quality (*reducing* the goal density D) or by increasing the time pressure (by *lowering* $T_{deadline}$). In

[1]There are other ways of manipulating the goal density and thus the required search effort, e.g., by changing the abstraction level of the problem space, as in the *approximate processing* approach of [Lesser 88].

general, by our definitions, greater concurrency allows higher solution quality given a particular $T_{deadline}$, because the problem-solvers can do more search. Thus our reorganization approach of composition and decomposition can adapt *both* the quality and timeliness of the organization's response.

Finally, let:

$$F_{i+1} = t_{problem-instance_{i+1}} - t_{problem-instance_i}$$

be the *arrival interval* of problem request $i+1$ (where $t_{problem-instance_i}$ is the arrival time of the i^{th} problem instance. Clearly, *higher* values of F place *lower* demands on the organization. Changes in environmental performance demands are stated as changes in $PERF_{required}$ or as changes in F, so we can characterize overall environmental demand E as:

$$E = F \cdot PERF_{required}$$

Higher values of E are easier to achieve, while lower values are harder to achieve.

To adapt to changes in E, agents in the organization invoke composition and decomposition operators. To reason about when and how to reorganize, we define two types of organizational knowledge, as follows.

Local Statistics: We introduce an *activity ratio* R that represents how busy each agent is. Let S be a predefined period (normalized by problem-solving cycles) for measuring statistics, and N be a number of problem-solving actions (e.g., rule firings) during S. Then the R can be defined by N/S. When $R = 1.0$ (i.e., there are no idle problem-solving cycles over the measurement interval S), an agent is called *busy*, while when $R < 1.0$, an agent has excess capacity.

Organizational Statistics: We assume each agent can know (by periodic reports) whether the organization as a whole is currently meeting the performance criterion $PERF_{required}$. (This can be done without a global clock—see [Ishida 90b].) Let:

$$PERF_{actual} = k \cdot D \cdot T_{response}$$

be the most recently measured performance, where $T_{response}$ is the actual measured organizational response time (also in problem-solving cycles). When $PERF_{actual} > PERF_{required}$, the performance of the organization should be improved[2]. When $PERF_{actual} < PERF_{required}$, the organization can afford to release extra resources.

Reorganization Rules: The following rules use local and organizational statistics to select appropriate reorganization primitives as necessary.

R1: Decompose if $PERF_{required} < PERF_{actual}$ and $R = 1.0$

R2: Compose if $PERF_{required} > PERF_{actual}$ and $2R < PERF_{required}/PERF_{actual}$

R3: Compose if $R < 0.5$

R1 initiates decomposition of busy agents when the organization cannot meet performance requirements. **R2** initiates composition when the organization beats performance requirements, to release excess resources. Composition is performed even if agents are fully busy, when $PERF_{required}$ is sufficiently greater than $PERF_{actual}$, again to release resources. **R3** is introduced to account for communication overhead. Suppose environmental demand is initially high, and later decreases. Initially, **R1** is repeatedly applied, maximizing pipeline parallelism to improve organizational response. Later, even though the frequency of requests decreases, **R2** may not have been applied because the communication overhead may not allow agents to meet performance requirements. Thus, **R3** is necessary to merge lightly-loaded agents even when $PERF_{actual}$ exceeds $PERF_{required}$. This merging can lower communication and coordination cost in the overall problem pipeline, improving performance.

Reorganization decisions are made during the SDPSA reasoning cycle, in the same way as domain-level problem-solving decisions (both are modeled as production rules). In this way, OSD and domain problem-solving actions are arbitrarily interleaved. In our implementation we assume higher priority is given to the reorganization decisions during the Select phase of the SDPSA problem-solving cycle. This mechanism is analogous to the integration of control and domain knowledge source activations in systems such as BB1 [Hayes-Roth 85], or to integrated metalevel reasoning in the DVMT [Durfee 87a].

3 Experimental Evaluation

We have evaluated the effectiveness of our approach, using a simulation solving the *Waltz labeling program*: 36 rules solve the problem that appears in Figure 3-17 in [Winston 77] with 80 rule firings (for details see [Ishida 90b].) Our experiments begin with one agent containing all problem-solving knowledge. Organizational knowledge (e.g., dependencies and interferences among problem-solving actions) for the initial agent is prepared by analyzing its domain knowledge before execution. Some of its initial organizational knowledge is trivial—with no interdependent neighbors, all qualifications by agent are references to itself. All dynamic organizational knowledge in the organization is developed by the organization itself over time.

[2]The reader is again cautioned that, in the discussions below, *lower* PERF values are more difficult to achieve; this is somewhat counterintuitive.

Figures 3 and 4 show our simulation results, with communication and reorganization costs ignored. The line chart indicates response times normalized by problem-solving cycles. The step chart shows changes in the number of agents in the organization. $PERF_{required}$ is set at 20 and (S) is set at 10 problem-solving cycles. In Figure 3, requests arrive at constant intervals, while in Figure 4, F changes with time. Overall, in these cases, our autonomous reorganization has achieved approximately, though not exactly, the same performance as the non-autonomous reorganization reported in [Ishida 90a]. As in our previous work, in these figures the organization demonstrates three properties (cf. [Ishida 90a]). First, it is still adaptive over time, with stabilization of the number of agents and R almost equalized across agents in Figure 3, and with long-term temporal decrease in the number of agents at the busiest peaks (28, 24, and 22 agents) in Figure 4. Second, using Figure 4, it again shows real-time responsiveness, by comparison to a permanent-agent system with the same average number of agents (9). Third, it still exhibits efficient resource utilization, using 9 agents on average, compared to the conventional statically-decomposed approach which requires 17 processors to meet performance requirements under dynamic conditions.

Figures 5 and 6 describe the same situation as Figure 3, but include communication and reorganization overheads (O_c and O_r, respectively, measured in problem-solving cycles). We have simulated cases where O_c is equivalent to 1, 3 or 5 cycles. O_c is generally not critical on message passing machines because it approximates one problem-solving cycle [Ishida 90b]. But it may not be possible to ignore O_r even in fast message passing machines, depending on how much knowledge must be transferred during reorganization. O_r never exceeds 10 in our example, but we simulated cases with O_r of 10, 30 or 50 cycles to observe its general influences. The major results are as follows.

Communication overhead: Figure 5 shows O_c only. When $O_c = 1$, the organization can meet $PERF_{required}$. When $O_c = 3$ or more, the organization fails to meet $PERF_{required}$, because communication overhead causes a delay that affects the stable state of the organization. The organization can fluctuate in two ways. When agents decompose themselves rapidly so that $PERF_{actual}$ becomes much less than $PERF_{required}$, **R2** is triggered, and agents start composition. The organization can also fluctuate even if $PERF_{actual}$ exceeds $PERF_{required}$, if communication delays significantly lower agents' activity ratios. In this case, **R3** becomes satisfied, causing re-composition.

Reorganization overhead: Figure 6 shows O_r only. O_r is temporary and we would not expect it to affect the organization's stability. When $O_r = 10$, the organization does quickly reach stability. But when O_r becomes larger (e.g., 30 or more), the organization oscillates, for the following reason. Since reorganizing agents cannot fire rules during the decomposition process, their R values decrease. R values of neighboring agents also decrease because no new data are transferred from the reorganizing agents. Neighbors fire **R3** and compose, causing oscillation. A damping constant would inhibit $R3$, preventing early composition, but would also impede responsiveness to an increase in F. The reorganization sensitivity could also be decreased by enlarging S. This area merits further study (see [Hogg 90, Ishida 90b]).

4 Conclusions

We have presented a general, conceptually simple, and formally analyzable distributed problem-solving model which can reorganize its macroarchitecture to flexibly adapt to changing performance requirements. We have also presented the organizational knowledge necessary to allow the system to make certain types of autonomous reorganization decision, and have studied the relationships between organizational knowledge and communication-coordination overheads. Used as an organization control structure, this approach has promise for a range of distributed problem-solving structures. For example, our model can be used as a simulation tool to derive good (if not ideal) static configurations for particular problems and environmental performance demand configurations.

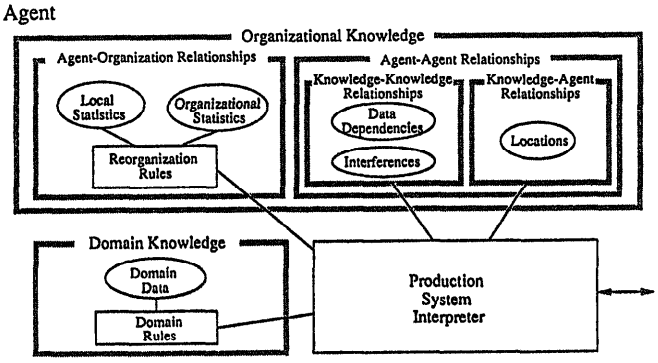
There are several useful extensions to this work. First, we would like to incorporate the cost and expected benefit of composition and decomposition decisions in the reorganization rules. Second, we would like to make the architecture more general, reorganizing by redistributing knowledge and goals, as well as by creating and removing agents, and creating metaknowledge with which to choose among these options. For example, extra resources can come from underused capacity of existing agents and from resources wasted in poorly organized communication and interaction structures, as well as from new agents. Similarly, underutilized resources can be returned to the organization itself via improved structure, rather than to the environment by removing agents. These 2 new forms can be implemented using *location knowledge* [Ishida 90a] tp provide the dynamic extension of the static approach organization based on constrained interest areas of an agent used in [Corkill 82, Durfee 87a].

Finally, there are many conceptual approaches to organization in the literature [Bond 88, Gasser 91b, Ishida 90b]. In most DAI literature, organizations comprise a fixed collection of agents each of which has a stable internal architecture, and whose boundaries are fixed. Our OSD scheme creates and destroys agents, as

well as transferring organizational and problem-solving knowledge among agents. In effect, our system can be seen as a fabric of knowledge, resources, and action, *out of which agents actively and flexibly construct and reconstruct themselves* by adding and subtracting resources and by changing agent-knowledge boundaries. It is the overall collection of problem-solving knowledge that is fixed—not the definition of agents. This represents a new, "social" approach to nature of both agents and organization [Gasser 90, Gasser 91a,b]. It appears to offer the promise of a wider degree of organizational flexibility.

Acknowledgements

The authors wish to thank Kunio Murakami, Tsukasa Kawaoka and Ryohei Nakano for supporting our joint research, and Makoto Yokoo for his contributions to the architecture and simulations. We also appreciate comments of Alan Bond, Jean-Pierre Briot, and Ken Goldberg.



Self-Organizable Distributed Production System-Based Agent

Figure 1 Agent Architecture

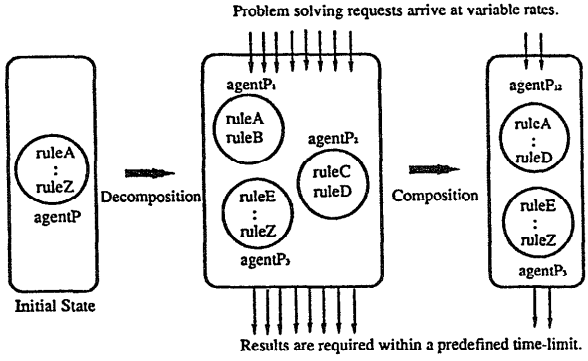


Figure 2 Composition and Decomposition

References

[Bond 88] A. Bond and L. Gasser, *Readings in Distributed Artificial Intelligence*, San Mateo, CA: Morgan Kaufman, 1988.

[Corkill 82] D. D. Corkill, *A Framework for Organizational Self-Design in Distributed Problem Solving Networks*, PhD Dissertation, COINS-TR-82-33, University of Massachusetts, 1982.

[Davis 83] R. Davis and R. G. Smith, "Negotiation as a Metaphor for Distributed Problem Solving," *Artificial Intelligence*, Vol. 20, pp. 63-109, 1983.

[Durfee 87a] E. H. Durfee, V. R. Lesser, and D.D., Corkill, "Coherent Cooperation among Communicating Problem Solvers," *IEEE Transactions on Computers*, Volume C-36, pp. 1275-1291, 1987.

[Durfee 87b] E. H. Durfee and V. R. Lesser, ""Using Partial Global Plans to Coordinate Distributed Problem Solvers" (*IJCAI-87*, pp. 875–883, 1987.

[Gasser 90] L. Gasser, "Conceptual Modeling in Distributed Artificial Intelligence," *Journal of the Japanese Society for Artificial Intelligence*, Vol. 5, No. 4, July, 1990.

[Gasser 91a] L. Gasser, "Social Conceptions of Knowledge and Action," *Artificial Intelligence*, January, 1991 (in press).

[Gasser 91b] L. Gasser, "Organization Theory from the Perspective of Distributed Artificial Intellligence" in Michael Masuch and Massimo Warglien, eds., *AI in Organization and Management Theory*, Elsevier, 1991 (in press).

[Hayes-Roth 85] B. Hayes-Roth, "A Blackboard Architecture for Control," *Artificial Intelligence*, Vol. 26, pp. 251-321, 1985.

[Hogg 90] Tadd Hogg and Bernardo A. Huberman, "Controlling Chaos in Distributed Systems," Technical Report SSL-90-52, Dynamics of Computation Group, Xerox Palo Alto Research Center, Palo Alto, CA, 1990.

[Ishida 90a] T. Ishida, M. Yokoo and L. Gasser, "An Organizational Approach to Adaptive Production Systems," *AAAI-90*, pp. 52-58, 1990.

[Ishida 90b] T. Ishida, L. Gasser, and M. Yokoo, "Organization Self-Design of Distributed Production System-Based Agents" USC DAI Group Research Note 68, Dept. of Computer Science, USC, 1990.

[Lesser 88] V. R. Lesser, J. Pavlin and E. H. Durfee, "Approximate Processing for Real Time Problem Solving," *AI Magazine*, Vol. 9, No. 1, pp. 49-61, 1988.

[Schwuttke 91] Ursula Schwuttke and Les Gasser, "Dynamic Tradeoff Evaluation for Real-Time AI." USC DAI Group Research Note 83, Dept. of Computer Science, USC, January, 1991.

[Winston 77] P. H. Winston, *Artificial Intelligence*, Addison-Wesley, 1977.

[Zisman 80] M.D. Zisman, "Using Production Systems for Modeling Asynchronous Concurrent Processes," in D. Waterman, *Pattern-Directed Inference Systems*, Academic Press, New York, 1978.

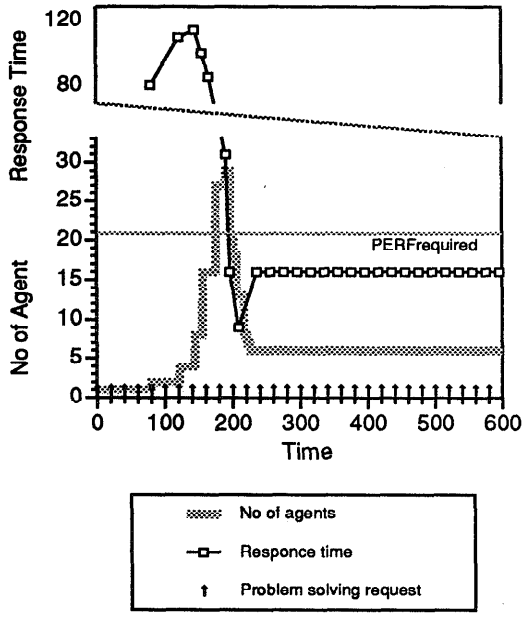


Figure 3 Simulation Results (Constant Interval)

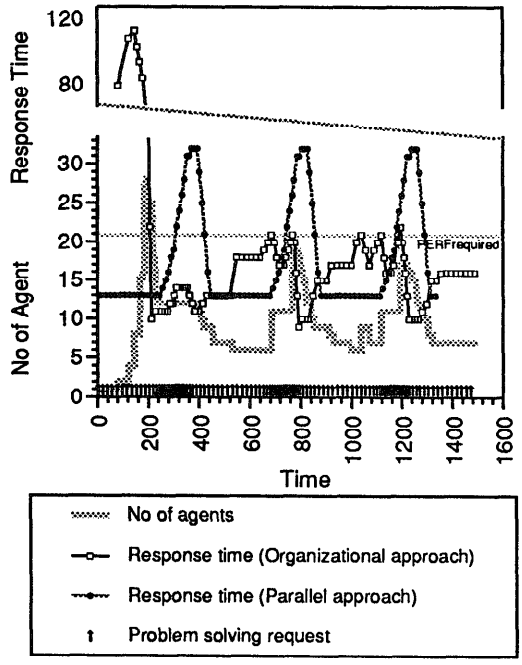


Figure4 Simulation Results (Changed Interval)

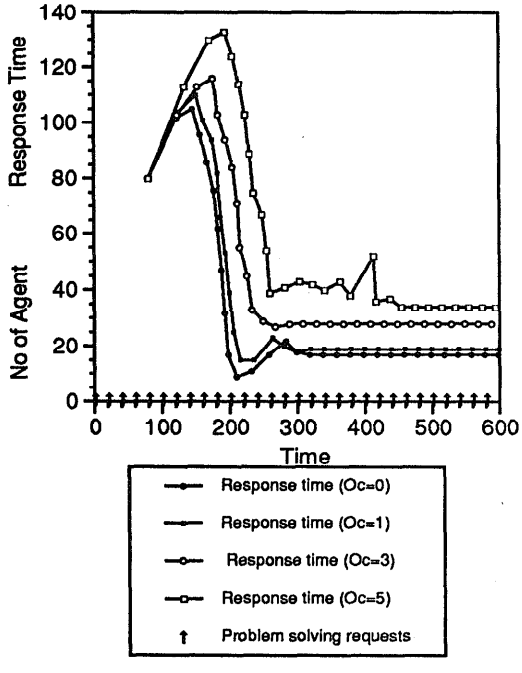


Figure 5 Simulation Results (Communication Overheads)

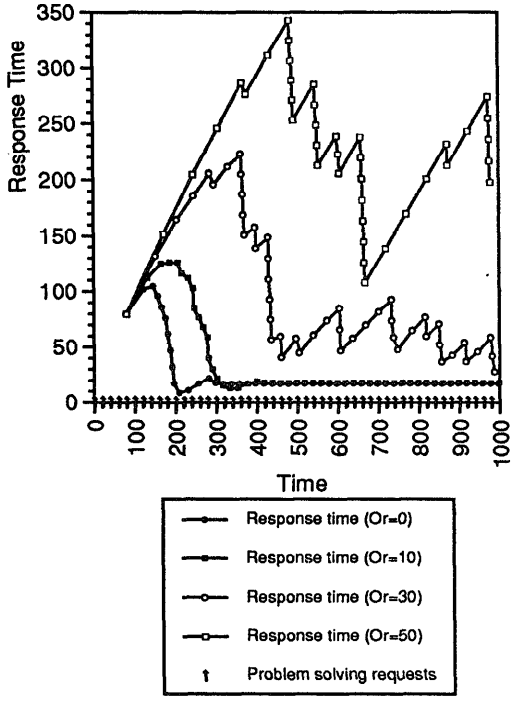


Figure 6 Simulation Results (Reorganization Overheads)