

Organizational Self-Design in Semi-dynamic Environments

Sachin Kamboj and Keith S. Decker
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19716
{kamboj, decker}@cis.udel.edu

ABSTRACT

In this paper we propose a run-time approach to organization that is contingent on the task structure of the problem being solved and the environmental conditions under which it is being solved. We use TÆMS as the underlying representation for our problems and describe a framework that uses Organizational Self-Design (OSD) to allocate tasks and resources to the agents and coordinate their activities.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent systems

General Terms

Algorithms, Design, Performance

Keywords

Organizational-Self Design, Task and Resource Allocation

1. INTRODUCTION

In this paper, we are primarily interested in the organizational design of a multiagent system — the roles enacted by the agents, the coordination between the roles and the number and assignment of roles and resources to the individual agents. The organizational design is complicated by the fact that there is no best way to organize and all ways of organizing are not equally effective[1]. Instead, the optimal organizational structure depends both on the problem at hand and the environmental conditions under which the problem needs to be solved. The environmental conditions may not be known *a priori* or may change over time, which would preclude the use of a static organizational structure. On the other hand, all problem instances and environmental conditions are not always unique which would rule out the use of a new, bespoke organizational structure for every problem instance.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.
Copyright 2006 ACM 1-59593-303-4/06/0005 ...\$5.00.

In our approach we use Organizational Self-Design (OSD) to dynamically alter the organizational structure of the agents. We define two operators for OSD — agent spawning and composition — when an agent becomes overloaded, it spawns off a new agent to handle part of its task load/responsibility; when an agent lies idle for an extended period of time, it may decide to compose with another (underloaded) agent.

Our work builds on the work by [2]. The primary difference between their work and our work is that we use TÆMS [3] as the underlying representation for our problems. TÆMS is a computational framework that uses annotated hierarchical task networks (HTNs) to allow quantitative reasoning over the task structures. TÆMS allows us to explicitly reason about alternative ways of doing a task, arbitrary ways of combining subtasks, uncertainties, quality/cost tradeoffs, and non-local effects and is hence more general than the approach used by [2].

2. ORGANIZATIONAL SELF DESIGN

2.1 Agent Roles and Relationships

As explained in Section 1, the organizational structure is primarily composed of roles and the relationships between the roles. One or more agents may enact a particular role and one or more roles must be enacted by every agent. The roles may be thought of as the parts played by the agents enacting the roles in the solution to the problem and reflect the long-term commitments made by the agents in question to a certain course of action (that includes task responsibility, authority, and mechanisms for coordination). The relationships between the roles are the coordination relationships that exist between the subparts of a problem.

In our approach, the organizational design is directly contingent on the task structure and the environmental conditions under which the problems need to be solved. We define a role as a TÆMS subtree rooted at a particular node. Note, by definition, a role may consist of one or more other (sub-) roles as a particular TÆMS node may itself be made up of one or more subtrees. Hence, we will use the terms role, task node and task interchangeably.

We, also, differentiate between local and managed (non-local) roles. Local roles are roles that are the sole responsibility of a single agent, that is, the agent concerned is responsible for solving all the subproblems of the tree rooted at that node. For such roles, the agent concerned can do one or more subtasks, solely at its discretion and without consultation with any other agent. Managed roles, on the other hand, must be coordinated between two or more agents

as such roles will have two or more descendent local roles that are the responsibility of two or more separate agents. We achieve this coordination by assigning one of the agents as the manager responsible for enacting the non-local role. This manager is responsible for making the coordination decisions, often in consultation with the agents enacting the descendent sub-roles of that particular non-local role.

2.2 Organization Formation and Adaptation

To form or adapt their organizational structure, the agents use two organizational primitives: agent spawning and composition. These two primitives result in a change in the assignment of roles to the agents. Agent spawning is the generation of a new agent to handle a subset of the roles of the spawning agent. Agent composition, on the other hand, is orthogonal to agent spawning and involves the merging of two or more agents together — the combined agent is responsible for enacting all the roles of the agents being merged. Hence, OSD can be thought of as a search in the space of all the role assignments for a suitable role assignment that minimizes or maximizes a performance measure.

In order to participate in the formation and adaption of an organization, the agents need to explicitly represent and reason about the role assignments. Hence, as a part of its organizational knowledge, each agent keeps a list of the local roles that it is enacting and the non-local roles that it is managing. Note that each agent only has limited organizational knowledge and is individually responsible for spawning off or combining with another agent, as needed, based on its estimate of its performance so far.

To see how the organizational primitives work, we first describe four rules that can be thought of as the organizational invariants which will always hold before and after any organizational change:

1. For a local role, all the descendent nodes of that role will be local.
2. Similarly, for a managed (non-local) role, all the ascendent nodes of that role will be managed.
3. If two local roles that are enacted by two different agents share a common ancestor, that ancestor will be a managed role.
4. If all the direct descendants of a role are local and the sole responsibility of a single agent, that role will be a local role.

When a new agent is spawned, the agent doing the spawning will assign one or more of its local roles to the newly spawned agent. To preserve invariant rules 2 and 3, the spawning agent will change the type of all the ascendent roles of the nodes assigned to the newly spawned agent from local to managed. Note that the spawning agent is only changing its local organizational knowledge and not the global organizational knowledge. At the same time, the spawning agent is taking on the task of managing the previously local roles. Similarly, the newly spawned agent will only know of its just assigned local roles.

When an agent (the *composing* agent) decides to compose with another agent (the *composed* agent), the organizational knowledge of the composing agent is merged with the organizational knowledge of the composed agent. To do this, the composed agent takes on the roles of all the local and managed tasks of the composing agent. Care is taken to preserve the organizational invariant rules 1 and 4.

2.3 Reasons for Organizational Change

As organizational change is expensive (requiring clock cycles, allocation/deallocation of resources, etc.) we want a stable organizational structure that is suited to the task and environmental conditions at hand. Hence, we wish to change the organizational structure only if the task structure and/or environmental conditions change. Also to allow temporary changes to the environmental conditions to be overlooked, we want the probability of an organizational change to be inversely proportional to the time since the last organizational change. If this time is relatively short, the agents are still adjusting to the changes in the environment - hence the probability of an agent initiating an organizational change should be high. Similarly, if the time since the last organizational change is relatively large, we wish to have a low probability of organizational change.

To allow this variation in probability of organizational change, we use simulated annealing to determine the probability of keeping an existing organizational structure. This probability is calculated using the annealing formula: $p = e^{-\frac{\Delta E}{kT}}$ where ΔE is the “amount” of overload/underload, T is the time since the last organizational change and k is a constant. The mechanism of computing ΔE is different for agent spawning than for agent composition and is described below. From this formula, if T is large, p , or the probability of keeping the existing organizational structure is large.

Agent spawning only occurs when the agent doing the spawning is too overloaded and cannot complete all the tasks in its task queue by the given deadlines of the tasks. To compute if spawning is necessary, we use the annealing equation with $\Delta E = \frac{1}{\alpha * Slack}$ where α is a constant and *Slack* is the difference between the total time available for completion of the outstanding tasks and the sum of the expected time required for completion of each task on the task queue.

Agent composition, on the other hand, is exactly orthogonal to agent spawning as agent composition only occurs when the agents are underloaded. In such a situation, some of the agents will be sitting idle waiting for tasks to arrive. These idle agents will either be utilizing resources while waiting, or more likely, will have resources allocated to them that could be used elsewhere in the system. In either case, it makes sense to combine some of the idle agents with other agents freeing precious resources.

To calculate if agent composition is necessary, we again use the simulated annealing equation. However, in this case, $\Delta E = \beta * Idle_Time$, where β is a constant and *Idle_Time* is the amount of time for which the agent was idle. If the agent has been sitting idle for a long period of time, ΔE is large, which implies that p , the probability of keeping the existing organizational structure, is low.

3. EVALUATION

To evaluate our approach, we ran a series of experiments that simulated the operation of both the OSD agents and the Contract Net agents on various task structures with varied arrival rates and deadlines. At the start of each experiment, a random TÆMS task structure was generated with a specified depth and branching factor. During the course of the experiment, a series of task instances arrive at the organization and must be completed by the agents before their specified deadlines.

To directly compare the OSD approach with the Contract

Net approach, each experiment was repeated several times — using OSD agents on the first run and a different number of Contract Net agents on each subsequent run. We were careful to use the same task structure, task arrival times, task deadlines and random numbers for each of these trials.

We divided the experiments into two groups: experiments in which the environment was static (fixed task arrival rates and deadlines) and experiments in which the environment was dynamic (varying arrival rates and/or deadlines).

The two graphs in Figure 1, show the average performance of the OSD organization against the Contract Net organizations with 8, 10, 12 and 14 agents. The results shown are the averages of running 40 experiments. 20 of those experiments had a static environment with a fixed task arrival time of 15 cycles and a deadline window of 20 cycles. The remaining 20 experiments had a varying task arrival rate - the task arrival rate was changed from 15 cycles to 30 cycles and back to 15 cycles after every 20 tasks. In all the experiments, the task structures were randomly generated with a maximum depth of 4 and a maximum branching factor of 3. The runtime of all the experiments was 2500 cycles.

We tested several hypotheses relating to the comparative performance of our OSD approach using the *Wilcoxon Matched-Pair Signed-Rank* tests. *Matched-Pair* signifies that we are comparing the performance of each system on precisely the same randomized task set within each separate experiment. The tested hypothesis are:

The OSD organization requires fewer agents to complete an equal or larger number of tasks when compared to the Contract Net organization: To test this hypothesis, we tested the stronger null hypothesis that states that the contract net agents complete more tasks. This null hypothesis is rejected for all contract net organizations with less than 14 agents (static: $p < 0.0003$; dynamic: $p < 0.03$). For large contract net organizations, the number of tasks completed is statistically equivalent to the number completed by the OSD agents, however the *number of agents* used by the OSD organization is *smaller*: 9.59 agents (in the static case) and 7.38 agents (in the dynamic case) versus 14 contract net agents. Thus the original hypothesis, that OSD requires fewer agents to complete an equal or larger number of tasks, is upheld.

The OSD organizations achieve an equal or greater average quality than the Contract Net organizations: The null hypothesis is that the Contract Net agents achieve a greater average quality. We can reject the null hypothesis for contract net organizations with less than 12 agents (static: $p < 0.01$; dynamic: $p < 0.05$). For larger contract net organizations, the average quality is statistically equivalent to that achieved by OSD.

The OSD agents have a lower average response time as compared to the Contract Net agents: The null hypothesis that OSD has the same or higher response time is rejected for all contract net organizations (static: $p < 0.0002$; dynamic: $p < 0.0004$).

The OSD agents send less messages than the Contract Net Agents: The null hypothesis that OSD sends the same or more messages is rejected for all contract net organizations ($p < .0003$ in all cases except 8 contract net agents in a static environment where $p < 0.02$)

Hence, as demonstrated by the above tests, our agents perform better than the contract net agents as they complete a larger number of tasks, achieve a greater quality and also

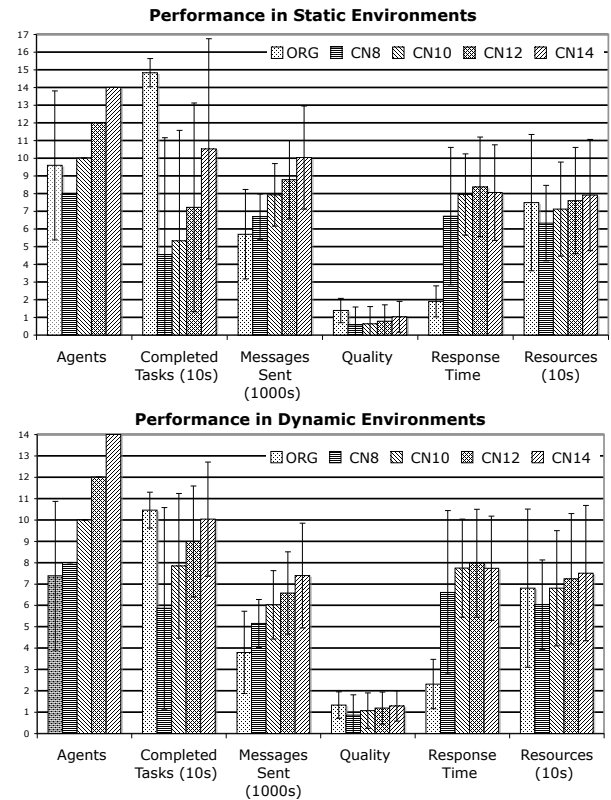


Figure 1: Graph comparing the average performance of the OSD organization with the Contract Net organizations (with 8, 10, 12 and 14 agents). The error bars show the standard deviations.

have a lower response time and communication overhead. These results make intuitive sense given our goals for the OSD approach. We expected the OSD organizations to have a faster average response time and to send less messages because the agents in the OSD organization are not wasting time and messages sending bid requests and replying to bids. The quality gained on the tasks is directly dependent on the number of tasks completed, hence the more the number of tasks completed, the greater average quality. The results of testing the first hypothesis were slightly more surprising. It appears that due to the inherent inefficiency of the contract net protocol in bidding for each and every task instance, a greater number of agents are needed to complete an equal number of tasks.

4. REFERENCES

- [1] K. M. Carley and L. Gasser. Computational organization theory. In G. Wiess, editor, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, pages 299–330, MIT Press, 1999.
- [2] L. Gasser and T. Ishida. A Dynamic Organizational Architecture for Adaptive Problem Solving. In *Ninth National Conference on Artificial Intelligence*, pp. 185–190, Menlo Park, California, 1991. AAAI Press.
- [3] V. Lesser, et. al. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *AAMAS '02 (Plenary Lecture)*, pp. 1–2, 2002.