An Adaptive Organizational Policy for Multi Agent Systems — AASMAN

S. Shaheen Fatima
Dept of Computer Science and Engineering
M.J. College of Engineering and Technology
No 3 Banjara Hills
Hyderabad 500034 India

G. Uma
D.E. Shaw India Software P.L.
G. Pulla Reddy Buildings
Begumpet
Hyderabad 500016 India

Abstract

Several formalisms for implementing organizational policies that assign specific roles to agents in a MAS have been proposed like the contract net protocol (CNP), the social reasoning mechanism (SRM), and the distributed computational economy (DCE). Organization self design (OSD) had been proposed to achieve load balancing in a distributed environment. In this paper we propose an adaptive organizational framework that exploits the essential features of CNP, SRM and DCE and integrates these with OSD. The resulting framework not only meets the needs of time constrained non-critical applications where computational load on the organization cannot be predicted ahead of time but also utilises the available system resources efficiently.

Keywords

Multi-Agent System, Contract Net Protocol, Social Reasoning Mechanism, Distributed Computational Economy, Organization Self Design

I. Introduction

The key issue in Multi-Agent System research is that of developing effective and adaptive formalisms for organizational policies that determine how goals are to be distributed among agents. There must be a means whereby agents with goals to be achieved can find the most appropriate idle agents to achieve those goals [3,10,11]. Our aim is to present an adaptive organizational policy for MAS that operate in time constrained domains. Several formalisms for implementing organizational policies have been proposed. These existing formalisms namely CNP, SRM, and DCE have each taken one aspect of problem solving in human societies and put forward a computational model for that. OSD is targeted to achieve load balancing in a distributed environment.

But since co-ordination in human societies is based on many aspects including dependence, material benefits and social laws we believe that we could build open multi-agent systems that are closer to human societies by developing a mechanism to implement all these characteristics. We therefore exploit the negotiation mechanism of CNP, social laws of SRM, market economy feature of DCE and integrate these with OSD to obtain an adaptive organizational framework that can operate under time constraints. This framework can therefore be applied to problem domains that require tasks like monitoring, fault detection, diagnosis and treatment to be performed within a stipulated time. Section II briefly examines the existing formalisms. In Section III we explain the proposed framework. Section IV gives the results of simulation of this model. Finally we show empirically that the proposed model has the characteristics of adaptiveness, meeting deadlines and efficient resource utilisation and highlight the direction of our future work.

II. Related Work

The following organizational policies were developed for task allocation in MAS. The primary goal of CNP [1,2,4,7] is opportunistic and adaptive task allocation using a framework called negotiation. Originally agents were assumed to provide their services without any motivation. However, later some modifications were made by having the manager announce some rewards at the time of task announcement which serve as a motivation for coalition formation. In the SRM model [6] dynamic coalitions are formed on the basis of motivation in the form of social dependencies. Agents provide services to requesting agents only if they believe that they can obtain some service in return from the requesting agent, i.e. they have a social dependence relation with each other. Social laws
are used to resolve conflicts that arise when more than one goal execution request arrives at an agent. The similarity between resource allocation problems of distributed computing and economics resulted in the application of markets to computation. Numerous projects have applied market mechanisms to problems in distributed task allocation [9]. In this approach agents are participants in a computational economy, interacting in the market to further their own interests. Agents are designed to choose strategies for buying and selling based on their own capabilities and preferences and the going market price for services. Such environments are expected to become increasingly common for MAS because the “market” is the world standard default interface for interacting entities.

OSD was proposed to allow an organization of agents to adapt itself to dynamically changing situations [8, 12]. Toru Ishida et al [8] present an approach that relies on the reorganization of a collection of agents to track changes in requirements. Their approach exploits an adaptive allocation of resources and organization form to satisfy performance constraints. Agents are created and released and domain knowledge is continually reallocated. To extend the agent architecture for OSD they introduce composition and decomposition of agents as new reorganization primitives. They also formalized organizational knowledge to represent interactions among agents and their organization. Overall, each of these developments provides a rich ground for the development of concepts and implementation of one aspect of human organization in MAS. But in human societies, each of these aspects is not considered in isolation. During problem solving, one or more of them may play a role in reasoning. Towards this end we present a protocol that exploits and integrates the negotiation mechanism of the CNP, social laws of the SRM, market economy feature of the DCE and OSD.

The protocol also meets the needs of time constrained domains where computational load on the organization cannot be predicted ahead of time.

III. An Adaptive Organizational Policy for Multi-Agent Systems

In this framework multiple agents solve a single or multiple problems. At a high level of description we have interacting agents that abide by social and market laws. The skills or services of agents are quantified using the notion of funds. These agents play the role of buyers, sellers, managers or contractors. An agent can take on any number of roles simultaneously and the roles that agents play change dynamically.

The basic architecture comprises of agents with manager and contractor roles. The manager is initially assigned a task with a deadline and is also allocated some funds. The goal of the manager is to accomplish the task within the allocated funds and deadline. It can do this either individually or by negotiating with other agents, forming a coalition with those agents and allocating subtasks to them. In the case of coalition formation the manager assumes the role of a buyer and the other agents in the coalition become sellers for the assigned subtasks and also play the role of contractors.

Contractors for coalition formation are chosen on the basis of goal deadline, funds allocated to the manager and the current market price for buying a service. There could be two or more agents in the MAS that possess the required skill to take on a subtask of the manager. If one of these agents possesses the capability to complete this subtask in time T by charging a price P and another agent possesses the capability to complete the same subtask in a time less than T by charging a price more than P, then one of these two will be selected by the manager on the basis of goal deadline and funds allocated to it. The manager thus efficiently distributes its funds among its contractors to accomplish its goal.

The process of coalition formation takes place as follows. Idle agents that have no goals at hand make auctions for selling their skills. These auctions specify the available skills, their time requirements and associated prices. As agents are assumed to have differing capabilities, an agent fixes the price for a service depending on its capability, i.e., its speed, reliability, etc. The manager in need of forming a coalition listens to these auctions and sends a bid to the agent providing the most suitable auction that is determined on the basis of goal deadline and available funds. A seller may receive more than one bid in response to the auctions but the bid for the highest priority goal (all goals are assumed to be initially prioritised) is selected by the seller. In case a tie occurs, social laws are used to resolve the conflict. According to social laws an agent that had provided some service in the past or possesses the capability to provide some service in future is given priority over other agents. Based on these factors the seller selects the bid with the highest priority and sends a commitment message to the manager which in turn awards the contract to it by sending a goal award message. The seller now becomes the contractor for the awarded goal and the manager and contractors are said to form a coalition.
However if the manager does not succeed in forming a coalition even after trying all possibilities, it starts reorganization by doing a decomposition (explained in section IIIIC).

Finally if reorganization also does not allow the system to meet the deadline, and if the priority of the incoming request is higher than the priority of any of its already committed requests, then the least priority goal is selected for recommitment. This repeats till the high priority incoming request can be accommodated. For every decommitment, the agent pays some funds as penalty to the manager of that goal. This penalty is the difference of the amount of funds that were committed and received initially for that goal and the amount of processing resources used by that goal till the point of decommitment. The contracts are therefore levelled commitment contracts (LCC) and not full commitment contracts (FCC). Sandholm and Lesser [13] prove that FCC are a subset of LCC. They also show that LCC provide a better alternative to FCC in dynamic environments where the demand on the system varies with time.

In MAS, in addition to the domain knowledge, each agent needs organizational knowledge that represents necessary interactions among agents. The agent architecture describes all the knowledge that agents need to possess in order to work together as a group.

A. Agent Architecture

Figure 1 describes the agent architecture. We view an agent as having five major functional components: a sensor for sensing the environment, a communication processor that handles message traffic with other agents, a goal processor that carries out the computation associated with goals, a contract processor that actually processes different types of messages and manages agent resources and a local knowledge base comprising the following:

Domain knowledge: This is the knowledge required for problem solving. It consists of goals and the associated plans for achieving those goals and the time and funds required to achieve the goal.

Organizational knowledge: In MAS multiple agents solve a single problem or multiple problems. Dependence exists among goals when the result of parallel execution of the goals is different from the result of sequential execution applied in any order. If two dependent goals are distributed to different agents, the agents have to locally synchronize to prevent the goals from being executed in parallel and thus maintain consistency. This inter-agent inconsistency is handled locally by using temporary synchronization via goal-deactivation. Organizational knowledge represents agent-agent relationships, that is, the goal dependence relationships among agents. An agent that has such a relationship with another agent is called the agent's neighbour and the set of goals having one or more dependence relations with other goals is denoted as GD. Organizational knowledge also includes agent-organization relationships (as explained in section IIIIC).

B. Task Execution Protocol

We first present an organizational policy without using reorganization. Then we introduce composition and decomposition primitives into it and show how this improves the performance of the protocol and makes it suitable for building open MAS. The goal or task execution protocol without reorganization therefore becomes

1. Process messages: The processing of different messages that an agent can receive is done as follows:
a) Goal execution request: If the agent can meet the deadline of the request, either individually or as a coalition with other agents, a commitment message is sent to the requesting agent. Otherwise, the agent selects the least priority goal for decommitment, sends a decommitment message to the manager of that goal, and also pays penalty to the manager of the decommitted goal. Decommitment repeats till the incoming request can be accommodated. Then a commitment message is sent to the requesting agent.

b) Synchronization request message: Synchronization acknowledgement message is sent immediately if the priority of the goal currently under execution is less than the priority of the goal for which the synchronization request message was received. Otherwise, the acknowledgement is sent after completing the current goal. In both cases, the agent deactivates all the dependent goals before sending the synchronization acknowledgement message. As a result of this deactivation, the deadlines of some of the already committed requests may get violated. Such requests are decommitted.

c) Synchronization release message: Goals that were earlier deactivated due to synchronization request message are now reactivated.

d) Partial results from contractors: Integration of partial results takes place.

e) Auctions: The most appropriate agent for coalition formation is found and a bid is sent to it.

f) Bids: Bids are prioritized and a commitment message is sent to the highest priority bidding agent.

g) Commitment: A goal award message is sent to the agent from whom the commitment message is received and the required funds are transferred to it.

h) Award goal: The agent accepts the goal request with its associated deadline and funds.

i) Decommit: The decommitted goal is treated as any other goal execution request message.

2. Select task for execution: Highest priority task is selected for execution. (Reorganization task > Problem solving task)

3. Request synchronization: Using knowledge from goal dependence relations, synchronization request messages are sent to neighboring agents. An acknowledgement is then awaited from all these agents.

4. Execute task: The selected task is executed and results are reported to the manager.

5. Release synchronization: A synchronization release message is sent to all synchronized agents.

The limitation of the above protocol is that it is suited only to those problem domains where the demand on the MAS does not vary with time. We therefore introduce OSD into the protocol to make it adapt itself to varying problem solving demands. We assume that problem solving requests arrive at the organization at variable rates. To respond, the organization must adapt to changing conditions and supply meaningful results within pre-specified time limits. The OSD approach relies on the reorganization of a collection of agents and an adaptive allocation of resources and organization form to satisfy performance constraints. Agents are created and released and domain knowledge is continually reallocated. In order to achieve this, the organizational knowledge is extended to include agent-organization relationships that represent how agents’ local decisions affect the behavior of the entire organization. This is shown in Figure 1. Two reorganization primitives, composition and decomposition, that create and release agents are also introduced.

Both composition and decomposition dynamically change the size of the agent population, the resources allocated to each agent and the distribution of problem solving and organizational knowledge in the organization. In general, decomposition increases the overall level of resources used while composition decreases the level of resources used.

C. Reorganization primitives

Effective reorganization agents invoke the following reorganization primitives.

- **Decomposition**: Divides one agent into two. This is done to increase parallelism.

- **Composition**: Combines two agents into one. This is done either to adaptively free up computing resources for cost-effective problem solving or to reduce response time. Composition may actually reduce response time even though parallelism decreases where co-ordination overhead (i.e., communication and synchronization overhead) is high.

Decomposition is triggered when environmental conditions (problem solving demand on the organization and response time) exceed the organization's ability to respond given the current form and resource level. Excessive demand at the organization level is translated into excessive local demand in particular regions of the organization that is measured using local and organizational statistics. At this point, agents with excessive local demand are divided into multiple agents and additional computational resources are assigned to them. Decomposition continues until parallelism increases and response time improves [8]. Composition is performed when under-utilized resources can be released for use by other agents so to improve local performance by reducing co-ordination overhead. When two agents taken together contain an excess of resources, they are combined into one agent via composition. Composition repeats until no more composition is possible under the conditions of meeting deadlines [8].

123
Since the aim of composition and decomposition are independent, both kinds of reorganization can be performed simultaneously in different parts of the organization.

For effective invocation of these primitives, the organizational knowledge is extended to include agent-organization relationships. In order to obtain better decisions with maximal locality, agent-organization relationships are maintained in the form of local and organizational statistics. Reorganization rules that use these statistics are also provided to select an appropriate reorganization primitive when necessary.

**Local Statistics.** Counters are maintained for counting the number of requests arriving for every goal (Cgi). The rate of arrival of problem solving requests, the number of problem solving requests that are not honoured and the rate of arrival of synchronization requests are also maintained as local statistics.

The rate of arrival of problem solving requests represents the level of activity of each agent. When this rate exceeds a certain threshold value it means that the problem solving demand on the agent is becoming very high. Thus one of the conditions that triggers decomposition is the rate of arrival of problem solving requests (C1). A second condition for triggering decomposition is the number of problem solving requests that are not honoured (because deadlines associated with these requests cannot be met) during a certain time period (C2). The rate of arrival of synchronization requests at an agent which measures the synchronization overhead is used as the third condition for triggering decomposition (C3). As a result of synchronizing, the deadlines of some of the already committed goals may get violated and such goals will therefore have to be decommissioned. Thus if the synchronization overhead increases, the number of decommissions also increases. To avoid this, a new agent is created via decomposition which is allocated all domain knowledge for performing goals in the set GD and this part of the knowledge is deleted from existing agents. Now there is only one agent in the MAS with all the knowledge required for achieving goals in GD. This eliminates synchronization overhead and the number of decommissions as all requests for achieving a goal in the set GD arrive at this single agent.

When the rate of arrival of problem solving requests falls below a threshold value then it means that the agent has a very low level of activity and it therefore starts making Tomplete be the expected completion time of the most recently made request and Tdeadline the pre-defined time limit of that goal. When Tcomplete > Tdeadline, the performance of the organization should be improved by doing a decomposition, while when Tcomplete < Tdeadline, the organization can release resources by doing a composition.

**Reorganization rules.** The reorganization rules that use these statistics therefore become:

1. **Perform decomposition** if
   - R1) C1 or C2 or C3 or Tcomplete > Tdeadline is satisfied.
   - R2) C4 or Tcomplete < Tdeadline is satisfied.

**D. Reorganization protocols**

Reorganization is done by the execution of reorganization tasks in step 2 of the normal task execution cycle. Reorganization task is given higher priority over domain problem solving task. If any of the conditions for rules R1 or R2 is met then the task execution cycle selects the corresponding reorganization task. The protocol given here describes how one agent, say A, decomposes itself into two agents, say A and B. During reorganization domain knowledge and organizational knowledge are transferred from agent A to agent B. Information about the creation of the new agent is broadcast to all other agents in the society. All agents can then update their organizational knowledge. These steps are described below.

**Decomposition protocol.** The decomposition protocol becomes:

1. Create a new agent: Agent A creates a new agent, agent B which starts task execution cycles.
2. Select domain knowledge to be transferred to the new agent: This is the knowledge required for achieving the goals in the set GD if C3 is satisfied. Otherwise half of the domain knowledge is transferred to the new agent in such a way that the more frequently requested goals (determined from the counters Cgi) are equally distributed between the two agents.
3. Request synchronization: Agent A sends a synchronization request message to agent B. Agent A also sends synchronization request message to its neighbours. Agent A then waits for acknowledgements from all its neighbours. While waiting for acknowledgement agent A can process messages as in step 1 of the goal execution protocol.
4. Transfer domain and organizational knowledge: Agent A transfers domain and organizational knowledge to agent B, updates its own organizational knowledge, and also propagates changes to all other agents.
5. Release Synchronization: Agent A sends synchronizat-
ion release message to agentB and all its neighbours. This ends reorganization.

**Composition Protocol.** An agent can compose with another agent by a similar process. First agentA sends composition request messages to its neighbours. If some agent, say agentB, acknowledges, agentA transfers all domain and organizational knowledge to agentB and leaves the organization. The transfer method is the same as that for decomposition.

Two new messages, namely the composition request and acknowledgement messages therefore need to be handled in step 1 of the task execution protocol as follows:

**Composition request:** If the agent is not already in the process of composition then send composition acknowledgement to the requesting agent, obtain its domain and organizational knowledge and integrate it with its own domain and organizational knowledge.

**Composition acknowledgement:** Send all the domain and organizational knowledge and leave the MAS.

### IV. Simulation Results

To evaluate the effectiveness of the proposed approach under varying problem solving demands a simulation environment is implemented. Experiment begins with one agent, agentA, with all the domain knowledge required to achieve 10 different goals g1 to g10. The set GD = \{g1,g3,g7\}. Table 1 lists the time and funds required for processing each of these goals. In order to simplify the discussion we assume that all agents have equal processing speed and all agents charge equally for the same service. To start with, agentA does not have any organizational knowledge as it is the only agent in the society and all goal processing requests arrive at this agent. However this knowledge keeps changing as agents enter and leave the society. The simulation results are summarised in tables 2 and 3.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Proc. Time (Tp)</th>
<th>Funds</th>
</tr>
</thead>
<tbody>
<tr>
<td>g1</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>g2</td>
<td>12</td>
<td>150</td>
</tr>
<tr>
<td>g3</td>
<td>11</td>
<td>120</td>
</tr>
<tr>
<td>g4</td>
<td>14</td>
<td>150</td>
</tr>
<tr>
<td>g5</td>
<td>15</td>
<td>50</td>
</tr>
<tr>
<td>g6</td>
<td>12</td>
<td>50</td>
</tr>
<tr>
<td>g7</td>
<td>10</td>
<td>80</td>
</tr>
<tr>
<td>g8</td>
<td>14</td>
<td>90</td>
</tr>
<tr>
<td>g9</td>
<td>12</td>
<td>140</td>
</tr>
<tr>
<td>g10</td>
<td>14</td>
<td>50</td>
</tr>
</tbody>
</table>

*Table 1*

We now describe the entries in tables 2 and 3. Column 1 indicates the goal for which a request has been made. There are two sources from which an agent can receive a request; its own sensors (S), and from other agents. Column 2 gives the source of request. Columns 3 and 4 show the arrival time and deadline associated with the request. Column 5 indicates the agent processing the request. Column 6 gives the time till which the agent is busy. Column 7 lists the expected completion time of a goal. The actual completion time may extend beyond this if the agent receives higher priority synchronization requests that lead to goal deactivation. Column 8 indicates the overhead incurred in achieving a goal. There are three possible ways in which a goal can be processed. The first possibility is for agentA itself to process the goal. In this case the total processing time is equal to the time indicated in table 1 (Tp) for that goal. No overhead is involved here. The second possibility arises when some existing agent is given the contract for the goal as a result of coalition formation. In this case the total processing time is equal to the time required for coalition formation (Tcoal) + the time required to process the goal as given in table 1 (Tp) + the time required to obtain the results from the contractor (Tc). The overhead incurred is equal to Tcoal + Tc. The third possibility arises when an attempt for coalition formation with existing agents fails and a new agent is created via decomposition which then processes the request. In this case the total processing time is equal to the time spent in trying for coalition formation (Tcoal) + time required for decomposition (Tdecomp) + time required to process the goal (Tp) + time required to obtain the results from the newly created contractor (Tc).

The overhead incurred is equal to Tcoal + Tdecomp + Tc. In case the contractor decommit is a goal, the overhead also includes the time that it actually spent in processing the goal before decommitting and the time spent in trying coalition formation the second time. In each of these cases the overhead also includes the time required to achieve synchronization (Ts), if necessary, which is taken to be 1.0 (the minimum possible value). This overhead may be more than 1.0 when some other agent in the system is in the process of achieving a higher priority goal that is an element of GD. Let Tcomp be the time required for composition. We assume Tcoal, Tdecomp, Tcomp and Tr values to be 1.5, 2.0, 2.0 and 0.5 respectively.

The first request at agentA arrives at time 0 for g1. AgentA starts processing it immediately and completes it at time 10 (Tp). The next two requests are also handled individually by agentA. There is no overhead for these three requests. However the deadline of the fourth request cannot be met. If therefore does a decomposition, creates a new agent, agentB and transfers half of its domain knowledge (in such a way that the more frequently requested goals are evenly distributed between the two agents i.e. for g1, g3, g5, g6 and g8) to it. AgentB comes into existence at time 14+ Tdecomp = 16, and completes the fourth request of
Table 3: Requests arriving at agentB

<table>
<thead>
<tr>
<th>Req. No</th>
<th>Goal</th>
<th>Req. Agent</th>
<th>Time of Arrival</th>
<th>Deadline</th>
<th>Agent</th>
<th>Busy Time</th>
<th>Expected Completion Time</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>g1</td>
<td>S</td>
<td>0</td>
<td>13</td>
<td>A</td>
<td>10</td>
<td>10</td>
<td>nil</td>
</tr>
<tr>
<td>2</td>
<td>g2</td>
<td>S</td>
<td>4</td>
<td>25</td>
<td>A</td>
<td>22</td>
<td>22</td>
<td>nil</td>
</tr>
<tr>
<td>3</td>
<td>g4</td>
<td>S</td>
<td>8</td>
<td>38</td>
<td>A</td>
<td>36</td>
<td>36</td>
<td>nil</td>
</tr>
<tr>
<td>4</td>
<td>g8</td>
<td>S</td>
<td>14</td>
<td>40</td>
<td>A</td>
<td>36</td>
<td>36.5</td>
<td>2.5</td>
</tr>
<tr>
<td>5</td>
<td>g9</td>
<td>S</td>
<td>16</td>
<td>52</td>
<td>A</td>
<td>48</td>
<td>48</td>
<td>nil</td>
</tr>
<tr>
<td>6</td>
<td>g6</td>
<td>S</td>
<td>18</td>
<td>54</td>
<td>B</td>
<td>48</td>
<td>42.5</td>
<td>2.0</td>
</tr>
<tr>
<td>7</td>
<td>g7</td>
<td>S</td>
<td>22</td>
<td>60</td>
<td>A</td>
<td>59</td>
<td>59</td>
<td>1.0</td>
</tr>
<tr>
<td>8</td>
<td>g7</td>
<td>S</td>
<td>24</td>
<td>73</td>
<td>A</td>
<td>70</td>
<td>70</td>
<td>1.0</td>
</tr>
<tr>
<td>9</td>
<td>g4</td>
<td>S</td>
<td>25</td>
<td>85</td>
<td>A</td>
<td>84</td>
<td>84</td>
<td>nil</td>
</tr>
<tr>
<td>10</td>
<td>g7</td>
<td>S</td>
<td>29</td>
<td>98</td>
<td>A</td>
<td>95</td>
<td>95</td>
<td>1.0</td>
</tr>
<tr>
<td>11</td>
<td>g7</td>
<td>S</td>
<td>32</td>
<td>110</td>
<td>A</td>
<td>106</td>
<td>106</td>
<td>1.0</td>
</tr>
<tr>
<td>12</td>
<td>g7</td>
<td>S</td>
<td>35</td>
<td>120</td>
<td>A</td>
<td>117</td>
<td>117</td>
<td>1.0</td>
</tr>
<tr>
<td>13</td>
<td>g10</td>
<td>S</td>
<td>39</td>
<td>132</td>
<td>A</td>
<td>131</td>
<td>131</td>
<td>nil</td>
</tr>
<tr>
<td>14</td>
<td>g9</td>
<td>S</td>
<td>42</td>
<td>144</td>
<td>A</td>
<td>143</td>
<td>143</td>
<td>nil</td>
</tr>
<tr>
<td>15</td>
<td>g10</td>
<td>S</td>
<td>45</td>
<td>146</td>
<td>C</td>
<td>143</td>
<td>61.5</td>
<td>2.5</td>
</tr>
<tr>
<td>16</td>
<td>g10</td>
<td>S</td>
<td>50</td>
<td>150</td>
<td>C</td>
<td>143</td>
<td>75.5</td>
<td>2.0</td>
</tr>
<tr>
<td>17</td>
<td>g9</td>
<td>S</td>
<td>55</td>
<td>154</td>
<td>C</td>
<td>143</td>
<td>87.5</td>
<td>2.0</td>
</tr>
<tr>
<td>18</td>
<td>g4</td>
<td>S</td>
<td>60</td>
<td>160</td>
<td>A</td>
<td>157</td>
<td>157</td>
<td>nil</td>
</tr>
<tr>
<td>19</td>
<td>g4</td>
<td>S</td>
<td>65</td>
<td>172</td>
<td>A</td>
<td>171</td>
<td>171</td>
<td>nil</td>
</tr>
<tr>
<td>20</td>
<td>g4</td>
<td>S</td>
<td>70</td>
<td>185</td>
<td>A</td>
<td>185</td>
<td>185</td>
<td>nil</td>
</tr>
</tbody>
</table>

agentA at time 30. This result becomes available to agentA at time $30 + Tr = 30.5$. Its overhead is equal to $T_{decomp} + Tr = 2.5$. The 5th request is again handled individually by agentA. In order to complete the 6th request before its deadline, agentA forms a coalition with agentB and awards the contract for g6 to it. In this case the overhead is equal to $T_{coal} + Tr = 2.0$. Requests 7 to 14 are also processed individually by agentA. However the processing of requests for g7 requires synchronization as g7 belongs to the set GD. Thus the overhead for processing g7 is equal to the synchronization overhead ($Ts$) which is initially assumed to be 1.0.

While agentA processes g7, agentB has requests for goals g1 and g3 that belong to GD. These goals have a higher priority than g7. Goals in the decreasing order of priority are g1 to g10. Thus before starting the fourth request for g3 at time 54, agentB achieves synchronization. At this point agentA suspends g7. AgentB repeats this synchronization process for its 5th, 6th, 7th, and 8th requests. The overhead for each of these is equal to $Ts$. During this period, the goal g7 remains suspended at agentA. AgentB sends the last synchronization release message after completing its 8th request, i.e. at time 111. This is received by agentA at time $111 + Tr = 111.5$. At this time agentA can resume g7 but as the deadlines of these requests cannot be met now, it decommits these requests. As the condition C3 is satisfied, a new agent, agentX, is created and allocated all domain knowledge for goals in the set GD. This knowledge is eliminated from agentA and agentB.
The 15th request at agentA cannot be completed either individually or as a coalition. AgentA therefore initiates another decomposition, creates a new agent, agentC, and transfers domain knowledge for g9 and g10 to it. The 15th, 16th and 17th requests of agentA are therefore processed by agentC. AgentB does not receive any requests after the 12th one. It therefore broadcasts a composition request message. AgentA responds with an acknowledgement. AgentB now transfers all its domain and organizational knowledge to agentA and leaves the society. In this way agents keep entering and leaving the society. From these figures the following conclusions can be drawn.

Adaptiveness of the organization: We have shown that when the response time exceeds the time limit the organization starts decomposition. Similarly when the response time falls below the time limit the organization starts composition. The organization of the system thus varies with varying demands from the environment.

Time constrained problem solving: In the given example the number of agents in the system varies between 1 and 4. It can be said that in order to meet the deadlines of all the requests, an average of around two agents is required. If the same experiment is done using the conventional approach with 2 permanent agents, the deadlines of all the requests cannot be met. Thus the reorganization approach is more effective for time constrained problem solving.

Effective resource utilisation: In order to meet the deadlines of all the requests, the conventional approach would require 4 permanent agents. Thus the organization centred approach which requires around 2 agents on an average is more economical.

Communication and reorganization overhead: The overhead indicated in column 8 of tables 2 and 3 is the sum of the communication and reorganization overheads. Reorganization overhead does not pose a problem as it is a temporary overhead. Communication overhead will not be significant in message passing machines. However communication overhead can become a problem when using wide area networks to perform distributed problem solving.

V. Conclusion

To overcome the drawbacks of the existing organizational policies when used in isolation, we presented a protocol that integrates these approaches by exploiting the negotiation mechanism of the CNP, social laws of the SRM, market economy feature of the DCF, and organization self design to meet the needs of time constrained domains where computational load on the organization cannot be predicted ahead of time. Our simulation study shows that the proposed model has the characteristics of adaptiveness, meeting deadlines, and efficient resource utilisation. This study is preliminary and we propose to put this model in operation in more realistic problem domains and study in detail issues like scalability and stability of the model.

References

12. Toru Ishida, “Parallel, Distributed and Multi-Agent Production Systems”, Proc. ICMAS-95