Management Techniques for Multiagent Virtual Organization

Xiaoqin Shelley Zhang[1], Ping Xuan[2] and Bhumit Patel [1]

[1]University of Massachusetts Dartmouth, USA

shelley.zhang@umassd.edu, bhumitmpatel@gmail.com

[2]Clark Univeristy, USA

pxuan@clarku.edu

Abstract: A virtual organization (VO) is formed when agents decide to temporarily build collaborative teams in order to accomplish tasks that individual agents cannot do. While the benefit may be apparent, there are several challenges to this business model. First, agents in a virtual organization perform not only tasks for the team (VO task), but also other tasks that benefit themselves (known as direct tasks). Also, since tasks may be achieved in difference ways (via different plans), and more than one agents may be capable of performing the same tasks, there is a need for the virtual organization to be able to determine which plan to adopt and which agents to assign which set of subtasks to. This is known as the initial commitment decision problem. Furthermore, when the plan is underway, an agent may decide to abandon a previously committed VO task in order to pursue a newly arrived better direct task. This is known as the crisis management problem. This paper presents several techniques to deal with these challenges. A multiagent virtual organization testbed is built for evaluation of the strategies and a building construction domain problem is used to show the effectiveness of our approach.

Keywords: Virtual organization, initial commitment decision problem, task reallocation, multiagent syetems.

1. Introduction

A virtual organization (VO) [1,10] can be defined as "a cooperation of legally independent enterprises, institutions or individuals, which provide a service on the basis of a common understanding of business. The cooperating units mainly contribute their core competences and they act to externals as a single corporation. The corporation refuses an institutionalization – for example, by central offices; instead, the cooperation is managed by using feasible information and communication technologies" [11]. One example of a virtual organization in the building construction domain would be a group of independent contractors participate in construction projects. There, external buyers offer the projects, developers plan, manage, and coordinate the construction related activities, and contractors bid on the projects and perform the construction tasks as a virtual organization.

To translate this in multiagent system context, this means that there are three general types of agents in the system that simulates the operations of a virtual organization (following the names used in the building construction domain):

- Buyer: the buyer agent purchases product or service from the virtual organization.
- Developer: the developer agent forms, manages and coordinates the activities in the virtual organization to ensure the completion of tasks as assigned by the buyer agent.
- Contractor: the contractor agents actually perform the tasks.

Once a virtual organization is formed, the developer agent receives requests from the buyer agent, and decides how to accomplish this request. There are multiple alternatives to accomplish a request, each called a plan that includes a set of tasks and temporal constraints. To decide which plan to select for the current request is known as initial commitment decision problem (ICDP) [2]. We adopt a combinatorial auction approach to solve ICDP in VO. A combinatorial auction is a special type of auction that deals with many types of parameters that collectively associate to a bid. Each agent generates a bid for each plan candidate based on its local reasoning process. The bid includes a heuristic called the *preference number*, which shows the agent's likeliness of implementing that plan, together with the VO subtasks it can perform without affecting any previously committed tasks in the agent's local schedule. The developer finds the highly preferred plan using a winner determination mechanism, thus obtains a solution to the ICDP as the result of the combinatorial auction. The scheduling and the management of the tasks are done using the *Simple Temporal Network (STN)* [3], which is a representation of task's temporal constraints.

After solving the ICDP, agents are committed to perform tasks as assigned by the developer. However, these agents also receive direct tasks in real-time that do not belong to the virtual organization. Since it is committed to some tasks in virtual organization, it may lose opportunities of having more profits. On the other hand, abandoning the commitment to VO plan

subtasks puts the overall plan in jeopardy. To improve the performance of the virtual organization and the individual agents, we introduce *Crisis Management* mechanism to ensure that the one agent may perform other tasks instead of its own VO subtasks without making entire VO task being in danger of being incomplete. Using crisis management mechanism, other agents are encouraged to perform the abandoned subtask for receiving some incentives and, at the same time, the agent that abandons the VO subtask is penalized according to the current penalty policy in the VO. Several penalty policies are implemented and tested.

In this work we present our experimental work via simulation on an imaginary building construction domain. The simulation of this research is conducted with the *multiagent virtual organization testbed* that is developed for this research work; however it is extensible enough to other multi-agent virtual organization application in general. It provides communication mechanism within the agents for cooperation, negotiation or coordination.

Our experiment results show the effectiveness of various mechanisms and techniques applied to the virtual organization. The experiments also expose some details of issues for VOs in the real world. For example an individual participant in a virtual organization may have less efficiency as compared to the other individuals in the virtual organization.

In the following sections we will discus related work, agent model and testbed setup, the plan and their temporal constraints, followed by the mechanism for solving ICDP and crisis management techniques. Finally, we show experimental results for the building construction problem and the analysis of the results.

2. Related Work

Hunsberger and Grosz [2] have proposed a mechanism that agents may use to solve the ICDP. This mechanism is based on a combinatorial auction in which agents bid on sets of roles in the group activity, each role comprising constituent subtasks that must be done by the same agent. Each bid may specify constraints on the execution times of the subtasks it covers. Results indicate a significant improvement in performance when constituent subtasks are grouped into roles. In our work, the bids structure is different and it includes three types of parameters (plan, subtasks in the plan, and preference number) instead of one type of parameter (roles). This way our testbed facilitates more options but with the price of higher complications in the system. The winner determination is directly based on the one type of the parameter of the bid (preference number).

Our work adopts the *Simple temporal networks* (STNs) [3,7] model for distribued control of a temporl network among multiple agents. Agents collaborating on a set of tasks subject to temporal constraints must coordinate their activities to ensure that all of the temporal constraints are ultimately satisfied. STNs can be used to concisely represent temporal constraints. In our work, we use the same method in [3] to convert a set of time point variables into a STN, which allows us to check plan consistency.

Our multiagent virtual organization testbed is similar to other agent architectures and testbeds [4,5,6] such as DECAF and JADE at the high level of abstraction such as the components structure and the message mechanism. However, at details level, our testbed is designed for ease of use and understanding. For example, we use a simpler protocol similar to electornic mail rather than FIPA protocols.

3. Agent Model And Testbed Setup

The multiagent virtual organization testbed uses a Java-based component architecture. It provides a set of core libraries to develop and simulate a multiagent virtual organization. It is designed to provide extensibility and facilitate code reuse for any other virtual organization application. The major components include:

VO-Comunicator: handles communication messages for agent. It uses VO-Mail message format that is similar to E-Mail.

VO-Simulator: handles setup, registeration and pulsing the agents/components.

Inspector: displays and monitors the execution process.

Config: uses a xml file to specify the properties for simulation runs: number of each experiment, deadline of the task assigned by the buyer, number of agents, name of the agents, roles of the agents, subtask associated with each role, communication lag time, and task decomposition at different levels.

Plan Parser: creats possible plans for tasks requested by the buyer agents, based on pre-defined task structures.

Agents: as mentioned in Section 1, there are three types of agents: buyer, developer, and contractor. There are one developer and multiple contractors in the system. Each contractor agent has two additional properties: *Role* and *Nature*. The role of an agent bounds it to perform the tasks of its expertise only. For example, a plumber agent can perform jobs related to plumbing only. However, an agent may be versatile and can have multiple roles.

We introduce a feature called *nature* for contractor agent to represent the variety of individuals in real world: some are aggressive while others are conservative. An aggressive agent tends to be more responsive (i.e. efficient) to changes in the system (such as new tasks). In our work, we use a numerical value to represent the nature: value 1 means a conservative nature and a higher value (up to 2) indicates a more aggressive nature. To be more specific, a conservative agent may skip bidding a subtask even if it is available during the same time. The skipping of a task is based on the value skip task seed,

which is calculate as the following: *Skip task seed = 20 – (Nature * 10).* A skip task seed value 10 (conservation with *Nature* = 1) means that the agent skips 10 tasks out of 20 direct tasks; while 0 (aggressive with = 2) means skipping none.

4. Plans, Tasks, And Constraints

After the virtual organization is formed, the buyer agent sends a contract to the developer agent. There is numerous ways of execution in order to achieve what the buyer agent requires. For each strategy of execution, a plan is formed and added to the developer's plan library. This plan is decomposed into subtasks. Figure 1 shows one of the task structure used for the building construction domain problem. Note that each subtask may in turn be represented a task structure at a more detailed level, thus allowing hierarchical decomposition.

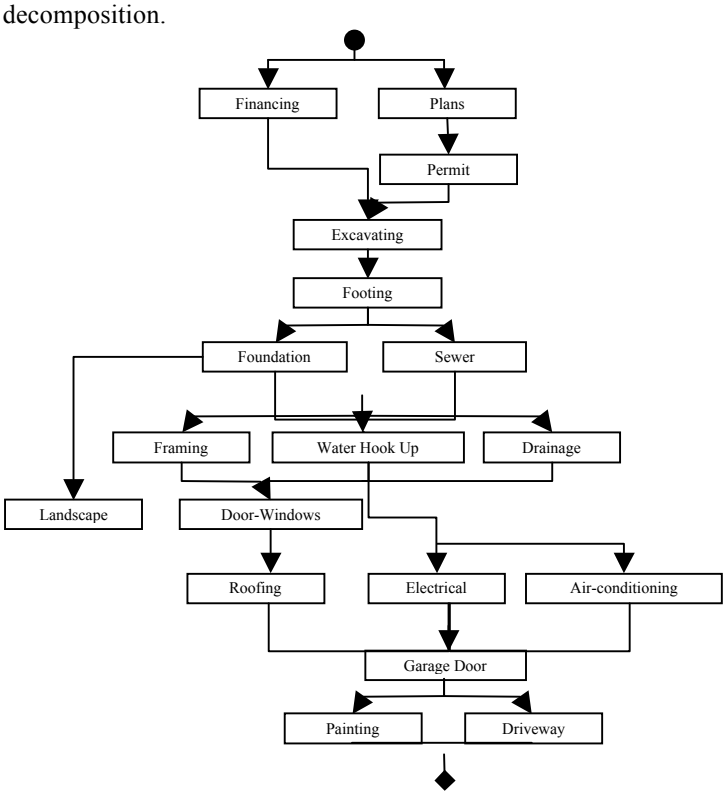

Figure 1: Task structure for constructing a building (Level 1)

A task in virtual organization is also decomposed into different time regions. Each task has a communication gap appended to it for communication to take place before actually initiating the task. The task contains a fixed start time known as earliest start time. It is set by the developer and the task can be started only after this earliest start time. The developer also assigns a deadline to the task. The task needs to be completed before this deadline. The contractor is responsible for finding the actual start time to fit the current task in its local schedule. The actual end time is calculated by adding the actual start time to the duration of the task.

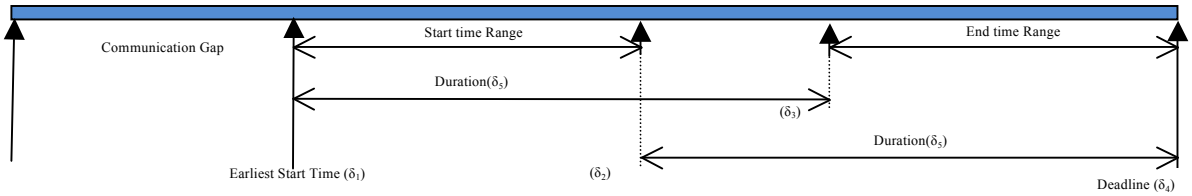

Figure 2: Task with STN parameters.

To represent temporal constraints, we use the *Simple Temporal Networks* (STN) [3] representation: the network is a pair, (T, C), where T is a set $\{t_0, t_1 \ldots t_n\}$ of time-point variables and C is a set of binary constraints on those variables, each constraint having the form $t_j - t_i \leq \delta$ for some real number δ. The "variable" t_0 (also known as z) represents an arbitrary, fixed

reference point on the time line (typically 0). Thus, given the task's start and end time points, as shown in Figure 2, we have the following:

$T = \{z, S_{i\text{-start}}, S_{i\text{-end}}\}$

$C = \{$

$z - S_{i\text{-start}}$	$<= -\delta_1$	(Start after δ_1)
$S_{i\text{-start}} - z$	$<= \delta_2$	(Start before δ_2)
$z - S_{i\text{-end}}$	$<= -\delta_3$	(End after δ_3)
$S_{i\text{-end}} - z$	$<= \delta_4$	(End before δ_4)
$S_{i\text{-start}} - S_{i\text{-end}}$	$<= -\delta_5$	(Minimum duration δ_5)
$S_{i\text{-end}} - S_{i\text{-start}}$	$<= \delta_5$	(Maximum duration δ_5)

$\}$

The first two constraints represent the range of actual start time. The third and forth constraints represent the range of the finish time of the task. The last two constraints represent the range of the duration. Since the duration is fixed, minimum and maximum duration are equal.

A simple temporal network is *consistent* if there exists some values assigned for those time variables so that all the constraints in C are satisfied. Each variable has a value range defined by the STN. For example, The value of the start time for task i belongs to the range of $[-D(S_{i\text{-start}}, z), D(z, S_{i\text{-start}})]$ where D(x,y) represents the shortest distance between x and y. $-D(S_{i\text{-start}}, z)$ and $D(z, S_{i\text{-start}})$ can be found out by checking the STN array, which are δ_1 and δ_2 respectively. By trying all the combinations of the values of the variables in T in its range defined in constraints C, we can find out if the subtask can be executed or it would have conflict with other committed tasks. To check if a contractor agent can execute a subtask, the subtask's STN is compared with each task in it's local schedule. If all the tasks are consistent with the new subtask, and then the agent can execute the subtask.

5. Initial Commitment Decision

When a new VO tasks arrives, the VO decides to which plan to choose for this task through a combinatorial auction process.

5.1 Bid Generation

When a contractor agent is informed that a new task has arrived the VO, it first checks if there is some subtasks of its role for each plan associated with this task. If so, it performs a consistency check with its local schedule, and then generates a bid for each plan. A bid contains the plan ID, the list of consistent VO subtasks included in that plan, and a preference number. A preference number is likeliness of implementing that plan, and is calculated using the following formula:

*Preference $P_{i\text{-}Pref}$ = (Nature of the Agent * Deadline of the Plan P_i) ^ (Number of Subtasks Fitting in Local Schedule / Total Number Subtasks of its Role in the Plan P_i)*

For example, if deadline is 550, the agent is conservative (nature = 1) and there are 5 subtasks of its role fitting in its local schedule, and there are 10 subtasks of the same role as those in the agent's bid, then the preference number is (1*550)^(5/10) = 23.45. The contractor agent bids for each plan existing in the plan library.

There are many different ways to measure an agent's preference of a plan, which way is the most appropriate one depends on the agent's utility function and it's current problem solving context. The formula presented here shows that an agent prefers more a plan with a longer deadline and more subtasks that can be fitted in its current local plan. This formula shows the agent's desire for more scheduling flexibility and to perform more tasks.

5.2 Winner Determination

The general winner-determination (WD) problem for combinatorial auctions is NP-complete [8,9]. However, Sandholm [8] and Fujishima et al. [9] have independently presented WD algorithms that scale to problems involving scores of items and thousands of bids. The main insight is that bids in practice necessarily only sparsely populate the space of possible bids; thus the search for the winning bid set should be restricted to the space of bid-sets composed of actual and not just possible bids.

In our work, the developer agent determines the winner by adding up corresponding preference number associated with each plan and finding out the plan with the highest preference sum. However, we also need to check that the subtasks of the plan to be selected are contained in the bids. Therefore we modify the winner determination mechanism by selecting the highest preference plan whose subtasks are covered by the bids. If none of the plans have all subtasks covered by the bids, we select - as a hopeful attempt - the highest preference plan, and randomly assign the leftover subtasks (i.e. subtasks not in any bid) to agents with suitable roles. Figure 3 shows the interactions involved when assigning the initial commitment.

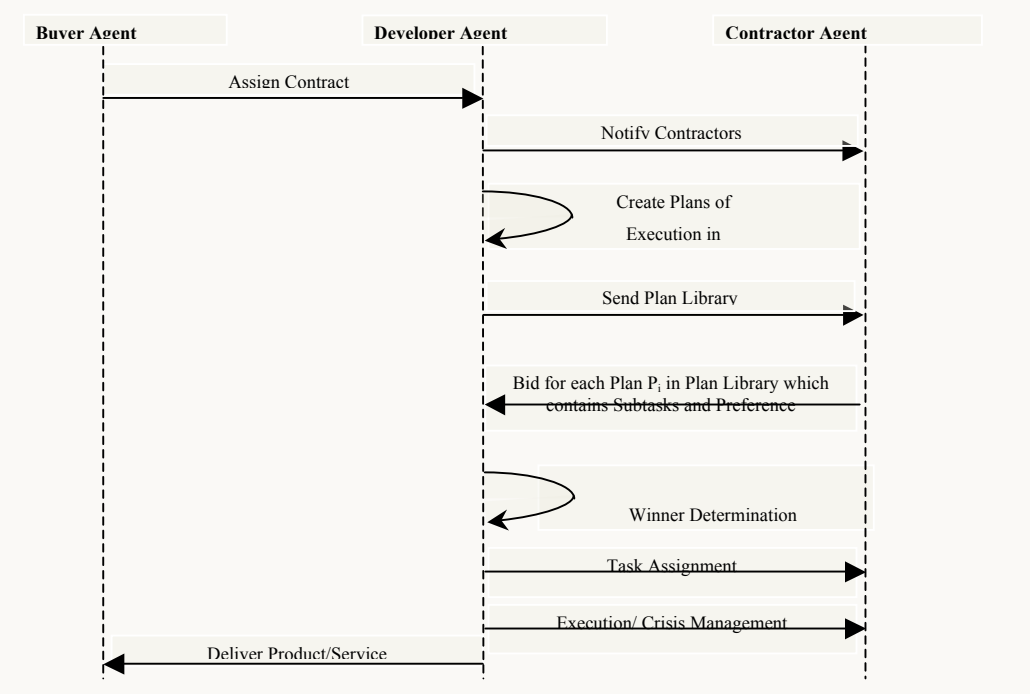

Figure 3: Agents interactions for preference features.

6. Crisis Management

We implemented a crisis management mechanism in multi-agent virtual organizations to handle task de-commitment when agents have more profitable direct tasks arriving. Agents bid for the tasks in virtual organizations based on their availabilities. At the same time, agents are also committed to direct tasks - tasks that do not belong to the virtual organization. At run time, when the plan for one VO task is underway, new direct tasks are offered to the contractors, who may break its initial commitment to VO subtasks. Crisis management mechanism ensures that the VO task assigned by the buyer is completed even when a contractor agent backs off from its own commitment, hence reduces other agents' risk of being a part of the virtual organization.

6.1 Penalty Policies

Each task (whether a VO subtask or a direct task) is associated with a price. On completion of a subtask the price is awarded. During operation, the agent may, in self-interested fashion, favor other opportunities and leave the commitment to the virtual organization unfulfilled. A penalty for fewer commitments than what it has promised is the most straightforward way to increase participating agents' bonding to VO and can be easily implemented. When a VO subtask is replaced by direct task, the penalty is applied to the agent who drops this VO subtask. On completion of a rescheduled VO subtask, price of that subtask along with the bonus (penalty paid by the de-committing agent) is awarded to the agent who completes it. Depending on how the penalty is calculated, there are different penalty policies. All the contractor agents are aware of the penalty policy that virtual organization imposes, so they can evaluate offers from other buyers (direct tasks) compared to the tasks in the local schedule.

A *linear penalty policy* has a fixed penalty rate for each unfulfilled commitment:

*Linear Penalty = Penalty_constant * VO_price*

For example, the linear penalty of a task with a price of 10 units and *penalty_constant* = 2, is 2 * 10 = 20.

A *progress-based penalty policy* has a decreasing penalty rate as more commitments have been fulfilled by the agent. It charges a heavy penalty if the agent cannot fulfill a minimum percentage of its promises, and it charges less penalty if the agent has fulfilled a certain percentage of obligations. For instance, if the agent cannot fulfill 30% of its promised commitments, there are 100 units penalty for each unfulfilled commitment; if the agent has fulfilled 90% of its promise, there is only 10 units penalty for each unfulfilled commitment. Progress-based penalty policy is dynamic and depends upon how many VO tasks have been accomplished at given time:

*Progress-based penalty = Penalty_constant * VO_price * [(VO_task rescheduled)/(VO_task accomplished)]*

This penalty policy, as well as the linear penalty one, ensures that the agent prefers VO tasks to direct tasks, thus encourages the completion of the entire VO task. However it is lenient when an agent has completed a good amount of VO

tasks. For example, the progress-based penalty for requesting reschedule when 2 tasks are already rescheduled and 1 is accomplished, with penalty constant of 2, is 2 * 10 * (2/1) = 40 (assuming VO task price as 10). But, penalty for requesting rescheduling when 1 task is already rescheduled and 2 are accomplished, with penalty constant of 2, is 2 * 10 * (1/2) = 10.

6.2 Feasibility Check

To decide whether to accept a direct task, an agent performs a feasibility check, in three different cases:

Case I: The direct task fits in local schedule's free time slot. In this case, there is no task in the local schedule conflicting with the direct task, so it passes the consistency check test. Once the consistency check test has been passed, the agent checks its skip task seed and performs a random test accordingly to see if the task should be skipped. If not, the direct task is accepted and added to local schedule. The developer is not notified about the change in the local schedule, as it does not affect the virtual organization.

Case II: The direct task conflicts with another direct task in the agent's local schedule. In this case, the agent checks if the direct task in local schedule is committed and not allowed to de-commit. If the task cannot be replaced, the new direct task is simply rejected. However, if it can be replaced, the price of the new direct task is compared with the old one. If the price of new direct task is higher, then the direct task is replaced by the new direct task. The developer agent is not informed, as the virtual organization task is not affected.

Case III: The direct task conflicts with a VO subtask. In this case, the agent checks if there is still a profit when accepting the direct task and paying the penalty for the pre-committed virtual organization subtask. In this work we assume that the agent does not de-commit from a VO subtask if the task has been started. Once the contractor agent decides to reject the pre-committed VO subtask, it needs to notify the developer agent and wait for its approval. The developer sends a request to complete the VO subtask left by this contractor agent to other contractor agents with the same role to see if anyone can pick it up. The price of this VO subtask now increases to the sum of the original price of the VO subtask plus the penalty paid by the original contractor.

With higher price, other contractor agents are more likely to accept this VO subtask. When other contractor agents receive the request by the developer agent, they perform feasibility check with as described earlier for direct task. However, the only difference is that an agent cannot replace a committed VO subtask with another VO subtask (leftover subtask of another contractor). This rule is constructed in order to prevent deadlock situation.

Once the feasibility check of the leftover VO subtask is completed by other contractors, they sends accept/reject message to the developer agent. If the developer agent receives more than one acceptance message, then it chooses the agent who preferred the ongoing plan the most. If no other contractor agents are willing to perform the leftover task, the original contractor is not approved for accepting the direct task and it needs to complete the VO subtask. This mechanism makes the agent more productive and increases its local profits, without actually harming other agents in the virtual organization.

Crisis management ensures some freedom of the contractor agents to accept more profitable direct tasks during the execution of the virtual organization. Experiment results show that the implementation of crisis management boosts the agents' performance while maintaining the completion of the virtual organization task as assigned by the buyer.

7. Experimental Results

To study the operations of multiagent virtual organizations, we have implemented a building construction organization scenario, which including one developer agent, multiple buyers agent and five contractor agents. One buyer agent generates VO tasks and several other buyer agents create direct tasks. Each contractor has three roles associated to it. Each role defines a set of subtasks. The agent who carries this role is able to perform the subtasks related to the role. The tasks used in the experiments are similar to the one showed in Section 4 (Figure 1). Each subtask has duration 5. There are six roles in this experiment. There are three subtasks associated with each role in this experiment. Table 1 shows the roles and the names of the subtasks for each type of role, and Table 2 shows the assumed roles of the 5 contractor agents.

Table 1: Roles and subtasks.

Roles	**Subtasks**
Manager	Financing, Plans, Permit.
Builder	Excavating, Footing, Foundation.
Plumber	Drainage, Sewer, WaterHookUp.
Framing	Framing, Roofing, DoorsWindows.
Interior	AirConditioning, Electrical, Painting.
Exterior	GarageDoors, Driveway, Landscaping.

Table 2: Roles of agents.

Agent	**Roles**		
Agent 1	Manager	Builder	Plumber
Agent 2	Builder	Plumber	Framing
Agent 3	Plumber	Framing	Interior
Agent 4	Framing	Interior	Exterior
Agent 5	Interior	Exterior	Manager

Direct tasks arrive at random time. At any time pulse, the probability of the arrival of a direct task is 0.25. Thus, the number of direct tasks depends upon the deadline of the plan. If a plan has a deadline of 100 then there will be approximately 25 direct tasks. The VO tasks have fixed price of 5 units. The direct tasks have the price range of 1 to 10.

Several sets of experiments are carried out in order to compare results; each containing different features like preference feature, crisis management and different penalty policies. Each experiment contains 100 Buyer's VO Tasks and with the deadline of 550 pulses assigned by the buyer agent. Other direct tasks are sent to each contractor.

For each experiment, we calculate the total profit of each agent. In the figures below, we show two rows of bars, with each row representing the result of one experiment. We compare the results of two experiments and show the agents' profit growth between the front and rear row, i.e. *growth = rear row profit / front row profit*.

Table 3: Utility achieved with plan preference compared to utility achieved without plan preference.

Agent	1	2	3	4	5
Utility growth from VO tasks	80.00%	86.00%	97.00%	109.29%	110.56%
Utility growth from Direct tasks	115.07%	115.30%	115.89%	116.20%	116.64%

7.1 Performance With Plan Preference

First, we show the result of choosing a plan based on agents' preference votes versus randomly choosing a plan. Note that the 5 contractor agents in the experiments. Each agent has a different nature of aggressiveness: the nature values are 1.0 (conservative), 1.25, 1.5, 1.75, and 2.0 (most aggressive) for agents 1 through 5. Figure 4 shows the total profit for each agent. The front row is the baseline, with randomly chosen plan. The rear row uses preference votes to choose a plan. With the help of the preference feature, all agents have increased profits. Table 3 shows the VO subtask profit growth due to the preference feature. Clearly, due to the difference in agent nature, more aggressive agents are more efficient in terms of being able to perform more tasks and therefore earning more profit.

Figure 5 shows the portion of the profit made from VO subtasks. In this case, aggressive agents get to perform more VO tasks because they have higher preference numbers than conservative agents,. Since the number of VO subtasks are fixed, conservative agents' performance decreases (for VO tasks) and the aggressive agents absorb the difference and improved their performance. Table 3 shows utility achieved with plan preference feature as a percentage of utility achieved without this feature.

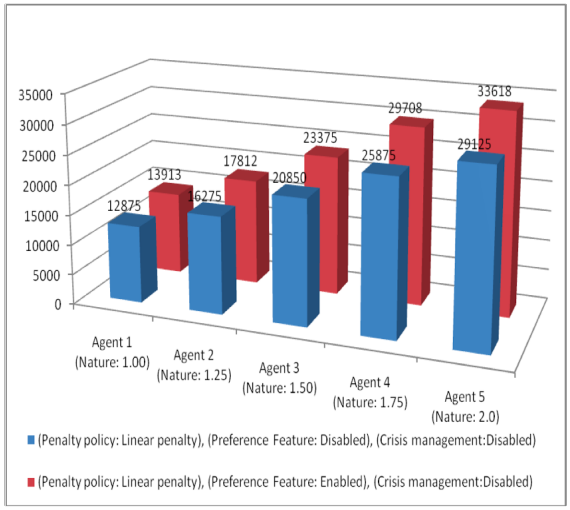

Figure 4: Total profit.

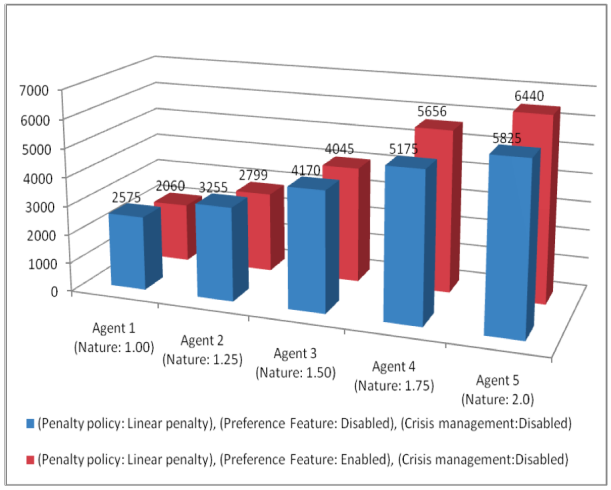

Figure 5: Profit from VO subtasks.

7.2 Performance With Plan Preference And Crisis Management

When crisis management feature is enabled, a contrator agent may decline an originally assigned VO subtask by paying a certain amount of penlaty (determined by the penalty policy used); another contractor agent that takes this declined task receives not only the VO price for this task but also the penalty paid by the orignial assigned agent. When this feature is disabled, the declined VO tasks fail to be executed. Figure 6 shows the result of using linear penalty for crisis managent (rear row) and one without crisis management enabled (front row). The total profit increases drastically when crisis management is enabled. This is expected as crisis management allow agents to replace current tasks with higher profit ones.

7.3 Linear Penalty Policy Versus Progress-Based Penalty Policy

Finally we compare the two penalty polices: Figure 7 shows the total profit using linear (front) and progress-based penalty policy (rear). The latter slightly dominates the former, indicating that the linear penalty is probably harsh on the agents for producing unnecessarily high penalty even if the agents are on the verge of completing the VO tasks. The progress-based penalty seems more reasonable.

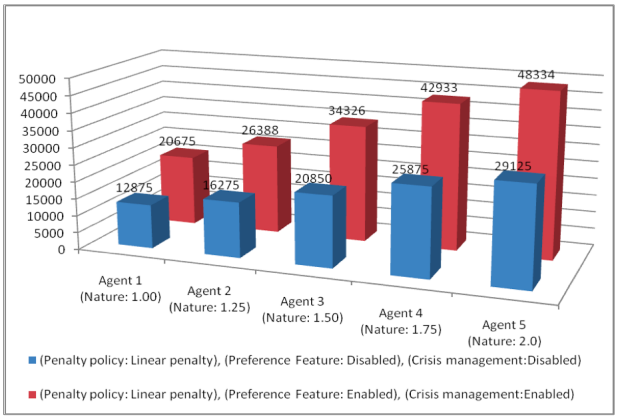

Figure 6: Total profits: preference and crisis management.

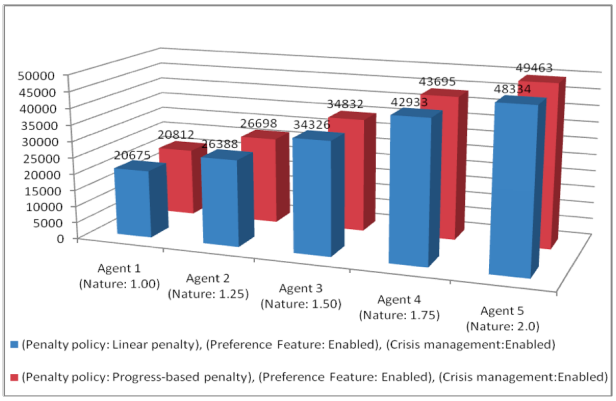

Figure 7: Total profit: linear penalty (front) vs progress based penalty (rear).

8. Conclusion

We have created a test bed for multiagent virtual organization applications. This test bed allows us to rapidly develop VO applications and evaluate mechanisms for virtual organization operations. Based on the experiments performed, we showed that, preference-based plan selection produces overall utility growth of all the agents in the virtual organization and achieves better efficiency. Similarly, crisis management produces significant benefits, as it not only offers some freedom toward maximizing local utility, but also safeguards virtual organization as it reschedules a task that can be performed by other agent in virtual organization which otherwise would be idle in that particular time slot. With careful choice of penalty policies, such as the progress-based penalty, the VO can offer a more fine-tuned strategy toward de-commitment, therefore providing more flexibility and fairness to agents in virtual organizations.

References

[1] Q. Zheng, X. Zhang, "Automatic Formation and analysis of multi-agent Virtual organization". *Journal of the Brazilian Computer Society (JCBS) Special Issue on agents Organizations*, Vol. 11(1): 74-89, July, 2005.

[2] L. Hunsberger and B.J. Grosz, "A combinatorial auction for Collaborative Planning". In *Proceedings of the Fourth International Conference on multi-agent Systems (ICMAS-2000)*, pages: 151-158.

[3] L. Hunsberger, "Distributing the Control of a Temporal Network among multiple agents". In *Proceedings of the Second International Joint Conference on Autonomous agents and multiagent Systems (AAMAS-2003)*, pages 899-906.

[4] J.R. Graham, K.S. Decker, M. Mersic, "DECAF - A Flexible multi agent System Architecture", in *Autonomous agents and multi-agent Systems*, 7(1-2): 7-27, 2003.

[5] F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, "JADE: A White Paper." EXP, 3:6-19, 2003.

[6] J. Collins, W. Ketter, and M. Gini, "A multi-agent negotiation testbed for contracting tasks with temporal and precedence constraints". *Int'l Journal of Electronic Commerce*, 7(1):35-- 57, 2002.

[7] L. Castillo, J. Fdez-Olivares, and A. Gonzalez, "A temporal constraint network based temporal planner". In *Workshop of the UK Planning and Scheduling Special Interest Group*, PLANSIG 2002, pages 99–109, 2002.

[8] T. W. Sandholm, "An algorithm for optimal winner determination in combinatorial auctions". In *Proceeding of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI'99, pages 542–547, August1999.

[9] Y. Fujishima, K. Leyton-Brown, and Y. Shoham, "Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches". In *Sixteenth International Joint Conference on Artificial Intelligence* (IJCAI-99), pages, 548-553, 1999.

[10] M. Ader, "Technologies for the Virtual Enterprise", *Workflow & Groupware Strategies*. France, 2001.

[11] M.T. Martinez, K.H. Fouletier, K.H. Park and J. Favrel, "Virtual enterprise-organization, evolution and control". *International Journal of Production Economics*. v74. 225-238. 2001.