Voting Policies that Cope with Unreliable Agents *

Christian Guttmann†
School of Computer Science and Software Engineering
Monash University
Clayton, VICTORIA 3800, Australia
xtg@csse.monash.edu.au

Ingrid Zukerman
School of Computer Science and Software Engineering
Monash University
Clayton, VICTORIA 3800, Australia
ingrid@csse.monash.edu.au

ABSTRACT

Collaboration plays a critical role when a team is striving for goals that are difficult to achieve by an individual. In previous work, we defined the ETAPP (Environment-Task-Agents-Policy-Protocol) framework, which describes the collaboration of a team of agents. According to this framework, team members propose agents to perform a task, and the team applies a voting policy to choose an agent for the task. In this paper, we expand on three parameters of this framework. We model team members that have variable proposal making attitudes, and team members whose performance exhibits different levels of stability. We then consider two new voting policies for group decision-making, and use a simulation-based evaluation to investigate the interaction between the different types of team members and the voting policies. Our results show that our previous *optimistic* voting policy, which chooses the agent that seems to have the best performance, yields an unstable task performance for teams where even a few agents do not make the best possible proposal. In contrast, our new voting policies yield a stable task performance.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Artificial Intelligence—*Multiagent systems*

General Terms

Performance, Design, Reliability

Keywords

Voting Policies, Agent Collaboration, Unreliable Team Members, Variable Performance

1. INTRODUCTION

In many multi-agent collaboration scenarios, the agents involved have different opinions about how to perform a task and who should perform it. Voting policies are methods that elicit the opinions of individuals in order to make group decisions. However, agents may not always be reliable when they propose agents for a task, thus compromising the performance of the group. Making reliable proposals means that agents make proposals to optimize utility according to the criteria of a task (rather than an agent's own criteria). Two examples of task criteria are to improve the quality and to reduce the time spent to achieve a task.

The goal of our research is to provide insights into how unreliable agents influence team performance, and propose group decision procedures that cope with unreliable agents in the context of a collaborative activity – the allocation of agents to tasks. We investigate these aspects under the *ETAPP* framework. This framework expresses the collaboration of a team of agents in terms of five operating parameters: Environment, Task, Agents, Policy and Protocol [5]. In our framework, agents do not accurately know the capabilities of team members, and combine their knowledge and coordinate their activities by means of a group decision procedure. Specifically, agents decide which agent should perform a task by applying an optimistic voting policy, which chooses the proposed agent with the most promising performance compared to all the other proposed agents. This policy was applied under the simplistic assumption that agents always make proposals in a reliable manner with the best intention for the group. However, decisions made under this assumption could lead to bad team performance when some agents exhibit unreliable proposal-making behaviour.

To investigate the impact of such behaviour, we extend the ETAPP framework by considering the following types of unreliable agents, which can make proposals that are not beneficial for a team.

- **Selfish**. If the agent considers a task to be onerous, it proposes other, perhaps less qualified, agents for the task; or alternatively, if a task is of benefit to the agent, it may propose itself, while disregarding more qualified candidates.
- **Lazy**. The agent knowingly uses less information than is available when it proposes agents for a task.
- **Corrupt**. The agent attempts to undermine the collaboration when it proposes agents for a task.
- **Conservative**. The agent does not use the most recent information to update its opinion of team members, thus proposing agents on the basis of old information.

*This research was supported in part by Linkage Grant LP0347470 from the Australian Research Council, and by an endowment from Hewlett Packard.

†Student Author.

AAMAS'05, July 25-29, 2005, Utrecht, Netherlands.

We propose to cope with unreliable team members by applying different voting policies that enable a team to perform in a stable manner. In this paper, we consider *majority voting*, where the agent preferred by most team members is chosen, and *weighted-observation voting*, where a proposal for an agent who has been previously seen performing a task is weighed higher than a proposal for an agent that was not observed in action before.

Some researchers assume a trusted third party that is determined by reputation mechanisms [6]. This would be an agent that is deemed to make proposals in a more reliable manner than other agents in a team. However, in our framework, there is no reason to assume that any agent will make more reliable decisions than other agents. This means that agents do not know if team members are unreliable or reliable, and therefore the team is not able to exclude unreliable team members from the decision making process.

An additional extension considered in this research consists of building models of agents that exhibit a nondeterministic performance. That is, each time an agent performs a task, its level of performance may change due to the influence of factors that are not explicit. This extension requires a probabilistic representation of an agent's *task-related capabilities*, such as mean level of performance and stability; a procedure for building agent models from a sequence of observations; and a representation of an observer's *observation capacity* (which is similar to attention span), i.e., how many observations can the observer remember.

We assess the influence of these factors on task performance by means of a simulation where we vary the stability of agents and their observation capacity, and the type and quality of the proposals made by agents (by considering teams composed of reliable agents and selfish, lazy, corrupt or conservative agents), and apply different policies for selecting agents for a task.

Our approach can be applied to a number of examples, such as the routing problem in peer-to-peer networks, where a group of peers (also called nodes) is selected based on their bandwidth to establish a transmission routing between two remote peers. Each node that is part of a routing provide bandwidth and consume battery power in order to establish a transmission. Single peers have incomplete knowledge of the bandwidth of peers in the network, and may ask other peers for their opinion about which routing is optimal. This process of collecting and assessing other peers' opinion resembles a voting process. Each peer in such a network is concerned to preserve its own bandwidth and battery power, and thus a peer may only propose other peers for the routing (rather than itself). The voting policies proposed in this paper are designed to cope with the problem of unreliable peers.

Section 2 outlines the ETAPP framework and discusses the extensions where we model agents with a nondeterministic performance and unreliable agents, and consider new voting policies. Our evaluation is described in Section 3. In Section 4, we consider related research, followed by our conclusions.

2. THE ETAPP FRAMEWORK

The *ETAPP* [5] framework is designed for a decentralized setting, where agents in a group act autonomously to collaborate with their teammates. Our framework provides an explicit representation of five operating parameters of a collaboration: *Environment, Task, Agents, Policy* and *Protocol*. The *Task* given to the group is to be performed in the *Environment*, and the *Policy* and *Protocol* are procedures agreed upon by all the agents in the group, but performed autonomously (this is similar to abiding by certain rules in order to belong to a society). Central to the *ETAPP* framework is the idea that the real capabilities of the agents in a team are not known to the team members. Hence, individual agents employ models of collaborators' capabilities in order to estimate the value of contributions of team members to a task. The *Agents* component stores these models and the mechanisms to reason about them.

The elements of the ETAPP framework are outlined below (for more details see [5], but note that the *Agents* component has been substantially modified since that publication). Our extensions are described later in this section.

An ***Environment*** $\mathcal{E}$ is a state space described by predicates which represent properties of objects and relations between objects. A state in the environment describes the values of these predicates at a particular step in a collaboration.

A ***Task*** $\mathcal{T}$ is represented by a tuple with three elements $< EC_T, EF_T, MS_T >$.

- EC_T specifies the *Evaluation Criteria* relevant to task $\mathcal{T}$, e.g., speed, quality or profit. The value for each criterion ranges between 0 and 1, where 0 corresponds to the worst possible performance and 1 corresponds to the optimal performance.

- EF_T denotes the *Evaluation Function* for the task, which specifies the weights assigned to the Evaluation Criteria (i.e., their relative importance to the task), and the way in which the values for these criteria are combined. For instance, the Evaluation Function $EF_T = \max \sum_{i=1}^{n} ec_i w_i$ specifies that the task should maximize a linear combination of n Evaluation Criteria, where w_i for $i = 1, \ldots, n$ are the weights assigned to these criteria. These weights range between 0 and 1, where 0 indicates no impact of a criterion on task performance, and 1 indicates a maximum impact.

- MS_T denotes a set of *MileStones* for the task: $MS_T = \{ms_0, \ldots, ms_m\}$, where ms_0 represents the initial state of the task (and is satisfied by default) and ms_m represents the goal state. Each milestone is reached by performing an action.

A team of ***Agents*** $\mathcal{A}$ comprises agents $\{A_1, \ldots, A_m\}$, where m is the number of agents in $\mathcal{A}$. Individual agents and groups of agents have *Internal Resources (IR)*, which represent the task-related capabilities of an agent or group. Individual agents also have *Modeling Resources (MR)*, which represent the ability of an agent to model agents and reason about them.

The *IR* of an agent or group of agents represents how well they can perform an action in terms of the Evaluation

Criteria of the task. The values for IR range between 0 and 1, with 0 indicating the worst performance and 1 the best. For instance, if the Evaluation Criteria of a task are time and quality, and one of the actions in the environment is *drive*, then $IR_{A_i}(drive)$ represents the driving performance of agent A_i in terms of time and quality, i.e., $IR_{A_i}(drive) = \{Perf_{A_i}^{\text{time}}(drive), Perf_{A_i}^{\text{qual}}(drive)\}$. These capabilities are *not* directly observable (only the resultant behaviour can be observed). Hence, they cannot be used to propose agents for tasks (but they are necessary to simulate agent performance, Section 3).

The MR of an agent comprise its *Models (M)* of the Internal Resources of agents and groups of agents, the *Resource Limits (RL)* of the agent in question, and its *Reasoning Apparatus (RA)*.

- M_{A_i} are the models maintained by agent A_i to estimate IR_{A_j} for $j = 1, \ldots, m$, and $IR_{\tilde{A}_k}$, where $\tilde{A}_k$ is a subset of k agents in $\mathcal{A}$ for $k \in \{2, ..., |\mathcal{A}|\}$ (agent A_i can model the performance of different subsets of agents). A_i's estimation of the capabilities of agents in the team (including its own capabilities) may differ from their actual performance, in particular if agent A_i has never observed the team in action. This estimation may be updated as agent A_i observes the real performance of the agents in the team.

- The RL of an agent pertain to the amount of memory available to store models of agents and groups, the agent's ability to update these models and generate proposals, and its ability to send and receive proposals (an agent that has become disconnected cannot send proposals, even if it can generate them).

- The RA consists of the processes required by protocol $\mathcal{P}$, which enable an agent to act in an environment and interact with collaborators. These processes are: (1) proposing agents for an action (selecting agents from a list of candidates); (2) communicating this proposal to other agents; (3) applying a policy P_A to select a proposal from the communicated proposals; and (4) updating M based on the observed performance of the selected agent(s).

A ***Policy*** $\mathcal{P}_A$ is a joint policy (adopted by all the agents in the team) for making group decisions about assigning agents to activities. Each agent proposes one or more agents for an action (according to its models M and its RA). Upon receiving all the proposals, each agent uses $\mathcal{P}_A$ for selecting one proposal.

A ***Protocol*** $\mathcal{P}$ is a process that is followed by all the agents in the group to coordinate their interaction. According to this protocol, all agents generate a proposal and communicate it to the other agents. Next, each agent applies $\mathcal{P}_A$ to select a proposal, observes the performance of the selected agent(s), and updates its models accordingly. It is worth noting that even though all agents follow the same protocol, the manner in which individual steps are performed is determined by the agents' RA.

2.1 Extension of ETAPP

In this paper, we extend the ETAPP framework along three agent-modeling dimensions – Internal Resources, Resource Limits and Reasoning Apparatus, with particular emphasis on the proposal-generation procedure of unreliable agents, and the voting policies that deal with the proposals made by these agents.

Internal Resources. In the original framework we assumed that agents' performance is deterministic and invariant. Thus, $IR_{A_i}(action)$ is a single number between 0 and 1. However, in realistic settings, agents exhibit variable performance (e.g., they could be having a bad day). We represent such a performance by means of a truncated normal distribution, where the mean represents the ability of an agent, and the standard deviation represents its stability (truncation is required so that we don't exceed the [0,1] thresholds). As stated above, these values are not observable, but they are the basis from which the observed performance of an agent is obtained during simulations.

Resource Limits. Originally, due to the deterministic performance of agents, a single observation of an agent's performance yielded an accurate model of its ability. However, this is clearly not the case if the performance is non-deterministic. In order to cope with this situation, we include *Observation Capacity (OC)* in our model of the Resource Limits of agents. This parameter, which is similar to attention span [11], specifies how many observations of the performance of each agent or group can be stored by an agent in its memory. When this limit is exceeded, the observer agent retains a window of the last K observations (forgetting the initial ones).

Reasoning Apparatus. The variable performance of agents demands the implementation of a new model-updating procedure. As for Resource Limits, our previous single-update method is unlikely to yield accurate results. We therefore propose a simple procedure whereby an agent re-calculates the mean and standard deviation of the observed performance of an agent or group every time they perform an action. Notice, however, that the results obtained by this procedure are moderated by the observation capacity of the observing agent. That is, if the observing agent can remember only the last K observations of an agent's performance, then the mean and standard deviation are calculated from these observations.

The attitude of an agent influences its proposal-making procedure and its model-updating procedure. An agent's attitude is a feature that is not known to other agents. We define four types of unreliable agents in the context of these RA processes as follows.

- A **selfish** agent proposes itself with a high performance when it expects a positive reward for a task.[1]

- A **lazy** agent does not maintain models of team members, thus it randomly selects an agent when making a proposal, and attributes to it a random performance.

- A **corrupt** agent proposes a random agent, attributing to it a high level of performance, in order to undermine the collaboration.

[1]In future research, we will consider agents that propose other agents when they expect a negative reward for a task.

- A **conservative** agent does not observe the behaviour of the team members, and hence does not update its models, using only the initial models of collaborators when it makes proposals.

Policy. In previous work, we implemented an *optimistic* voting policy, where the agent with the most promising performance was chosen for a task. This policy is adequate when the agents that make proposals are reliable. However, since unreliable agents can now make proposals, we consider two additional policies: *majority* and *weighted-observation*. According to the majority policy, the agent that receives the most votes is chosen. In contrast, the weighted-observation policy gives more weight to proposals for agents that have some "credibility". That is, a proposal for an agent who has been previously seen performing a task is weighed higher than a proposal for an agent that was not observed in action before. This weight is proportional to the number of times the observed agent has been seen in action. For example, if agent A_1 has just performed a task, while A_2 has never performed a task, the votes of the agents that prefer A_1 are weighed higher ($\times 2$) than the votes of the agents that prefer A_2.

2.2 Example – Surf Rescue Scenario

In this section, we present an example that illustrates the ETAPP framework in the context of the Surf Rescue (SR) scenario used in our simulation-based evaluation (Section 3). In this scenario, the environment $\mathcal{E}$ consists of the *beach* and the *ocean*, and the task is to rescue a distressed person (DP) in the shortest time possible. This means that the set of evaluation criteria is $EC_T = \{ec_{time}\}$, and the evaluation function is $EF_T = \max\{ec_{time}\}$ (recall that a short time has a high score). The milestones in this scenario are

$MS_T = \{ms_0, ms_1\}$, where

- $ms_0 = at(loc(A), beach) \wedge at(loc(DP), ocean)$,
- $ms_1 = at(loc(DP), beach) \wedge at(loc(A), beach)$.

In other words, ms_0 represents the initial state, where the group of agents is located at the beach and the distressed person is in the ocean, and ms_1 represents the goal state, where the distressed person is brought back to the beach.

In this example, we have three lifesavers $\mathcal{A} = \{A_1, A_2, A_3\}$ at the beach. The task consists of performing one action – to rescue the distressed person. The values for the *IR* of A_1, A_2 and A_3 for this action are $IR_{A_1}(rescue) = 0.5$ (STDV=0.4), $IR_{A_2}(rescue) = 0.8$ (STDV=0.3), and $IR_{A_3}(rescue) = 0.3$ (STDV=0.2). That is, agent A_1 has a medium performance and is unstable, A_2 has a high performance and is a bit more stable, and A_3 has a low performance and high stability.

For clarity of exposition, we assume that only agents A_1 and A_2 can select agents for a rescue. These two agents (which are both observers and lifesavers) maintain models of lifesaver agents A_1, A_2 and A_3 ($M_{A_1}(A_1), M_{A_1}(A_2)$ and $M_{A_1}(A_3)$, and $M_{A_2}(A_1), M_{A_2}(A_2)$ and $M_{A_2}(A_3)$), and generate proposals involving the lifesaver agents. The models are initialized randomly (i.e., each agent has an *a priori*, random opinion of the other agents). Both A_1 and A_2 store the last three observations made of the performance of the lifesavers (OC=3), and apply the majority policy for selecting a lifesaver for a rescue. This policy chooses the lifesaver that most agents voted for (in the event of a tie, the top agent in an ordered list of agents is selected).

Table 1 illustrates the assignment of agents to a sequence of rescues under the majority voting policy (the values obtained after each rescue are boldfaced). The first column shows the time of the rescue; the second column lists the observer agents; the third and fourth columns show the agent proposed by each observer agent and the agent selected by the majority voting policy, respectively. Columns 5-7 contain the observed performance of the lifesaver agents; and columns 8-10 contain the models resulting from these observations (we have listed only the mean of the observed performance).

The first two rows in Table 1 (corresponding to time T_0) contain the initial conditions of the collaboration. Columns 8-10 contain the initial values of the models maintained by A_1 and A_2 for the Internal Resources (rescue performance) of A_1, A_2 and A_3. These initial values, which are *not* consistent with the real performance of the agents in question, are also recorded as the first "observed" performance of A_1, A_2 and A_3. This is done to model a behaviour whereby an agent's initial "opinion" of the members of its team can be influenced, but not instantly replaced, by observations of their performance.

According to the models maintained by A_1 and A_2, A_3 has the best performance. Hence, A_3 is selected by both A_1 and A_2 when a rescue is announced at time T_1. However, as expected from the *IR* of A_3, the agent's actual performance (0.4 at time T_1, Column 7) is poorer than that anticipated by the observer agents. Both agents observe this performance, and update their models accordingly (Column 10).

Now, when a new rescue must be performed (at time T_2), agent A_1 proposes A_3, as it is still the best according to its models, but agent A_2 proposes A_1. As indicated above, according to our tie-breaking rule, the first agent in the ordered list of agents is chosen. This is A_1, as it appears in the list before A_3. However, A_1 does not perform well in the rescue (0.3 at time T_2, Column 5), which lowers $M_{A_2}(A_1)$ to 0.45 (Column 8). As a result, A_3 is once more the top choice of both observer agents for the next rescue (at time T_3). But A_3 performs quite badly (0.2 at time T_3, Column 7), thereby further lowering its expected performance according to the models maintained by the observers (Column 10).

At this stage, the bad performance of both A_1 and A_3 has yielded models with low mean values for these agents. Hence, for the next rescue, A_2 is chosen by both observer agents (at time T_4). This is a high-performing agent that has been under-estimated by both observers. Its good performance (0.8 at time T_4, Column 6) raises the expected value in the models maintained by both observers (Column 9). As a result, A_2, who is now clearly preferred by both observers, is chosen for the rescue at time T_5, rendering once more a good performance (0.7 at time T_5, Column 6).

At this point, the models maintained by the observer agents are closer to the *IR* of the lifesavers than the initial (random) models. Since both observer agents have an observation capacity of three observations, the next time a rescue is performed, the initial value will be dropped, which will further increase the accuracy of the models.

Table 1: Sample agent assignment to a sequence of rescues.

Time	Observer agent	Proposed agent	Selected agent	Observed performance of			Models		
				$\mathbf{A_1}$	$\mathbf{A_2}$	$\mathbf{A_3}$	$\mathbf{M(A_1)}$	$\mathbf{M(A_2)}$	$\mathbf{M(A_3)}$
T_0	A_1			0.3	0.4	0.5	0.3	0.4	0.5
	A_2			0.6	0.5	0.7	0.6	0.5	0.7
T_1	A_1	A_3	A_3	0.3	0.4	0.5 **0.4**	0.3	0.4	**0.45**
	A_2	A_3		0.6	0.5	0.7 **0.4**	0.6	0.5	**0.55**
T_2	A_1	A_3	A_1	0.3 **0.3**	0.4	0.5 0.4	**0.3**	0.4	0.45
	A_2	A_1		0.6 **0.3**	0.5	0.7 0.4	**0.45**	0.5	0.55
T_3	A_1	A_3	A_3	0.3 0.3	0.4	0.5 0.4 **0.2**	0.3	0.4	**0.37**
	A_2	A_3		0.6 0.3	0.5	0.7 0.4 **0.2**	0.45	0.5	**0.43**
T_4	A_1	A_2	A_2	0.3 0.3	0.4 **0.8**	0.5 0.4 0.2	0.3	**0.6**	0.37
	A_2	A_2		0.6 0.3	0.5 **0.8**	0.7 0.4 0.2	0.45	**0.65**	0.43
T_5	A_1	A_2	A_2	0.3 0.3	0.4 0.8 **0.7**	0.5 0.4 0.2	0.3	**0.63**	0.37
	A_2	A_2		0.6 0.3	0.5 0.8 **0.7**	0.7 0.4 0.2	0.45	**0.67**	0.43

3. SIMULATION-BASED EVALUATION

We evaluated our extensions of the ETAPP framework by means of simulation experiments which assess the impact of the following parameters on task performance: (1) Internal Resources, (2) Observation Capacity, (3) Agent Proposal Making Behaviour, and (4) Voting Policy. The same model-updating procedure was used in all our experiments (when OC=1, this procedure reverts to that used in our original framework). Our simulation is based on the Surf Rescue (SR) scenario introduced in Section 2.2, where the task is to rescue a person in distress. However, in our simulation the team of lifesavers is composed of five agents.

3.1 Simulation parameters

The parameters corresponding to our extensions are varied as follows

- **Internal Resources** – We defined five teams of agents with different degrees of stability: *Invariant, Stable, Medium, Unstable* and *Mixed*. The agents in Invariant teams exhibit the same performance in all the rescues. Agents in Stable teams exhibit low performance variability (the standard deviation of their performance distribution ranges between 0 and 0.2). The standard deviation for the performance of agents in Medium teams ranges between 0.2 and 0.8, and for agents in Unstable teams between 0.8 and 1. The Mixed team includes a mixture of stable, medium and unstable agents. The mean of the performance distribution is randomly initialized for the agents in all types of teams.[2]

- **Observation capacity** – We varied the OC of the agents between 1 and 8. When OC=i, agents retain the last i observations made, and when OC=1, their observation capacity is as for the original ETAPP framework.

- **Group proposal-making behaviour** – We defined two parameters that affect the proposal-making behaviour of a group of agents: *Agent Proposal Making Behaviour (APMB)* and *Number of Unreliable Team members (NUT)*. The *APMB* parameter determines the type of unreliable agents in a team, i.e., selfish, lazy, corrupt or conservative (Section 2.1).[3] The *NUT* parameter determines how many team members exhibit the unreliable behaviour specified by *APMB*. The value of *NUT* varies between 0 and m (the number of agents in the team). NUT=0 means that no agent is unreliable (as for the original ETAPP framework), and NUT=m means that all agents are unreliable.

- **Voting policy** – We experimented with the three policies mentioned in Section 2.1: *optimistic*, *majority* and *weighted observation*.

In addition, we constructed two benchmark collaboration settings: RAND and OMNI.

- The RAND (or random) setting defines a lower bound benchmark, where a rescue is conducted by an agent that has been chosen randomly from the team. In this setting, agents do not maintain models of their collaborators' resources, do not communicate proposals, and do not update models.

- The OMNI (or omniscient) setting defines an upper bound benchmark, where the best-performing agent in the team is always assigned to a rescue. This setting is consistent with the traditional assumption of multi-agent systems whereby agents have accurate knowledge about the performance of team members prior to the collaboration (i.e., $M_{A_i}(A_j) = IR_{A_j}$ for $i, j = 1, \ldots, m$). In this setting, agents do not update their models or communicate proposals, because all agents have the same accurate models.

3.2 Methodology

We conducted two experiments as follows. In the first experiment, we investigated the interaction between team stability, agent observation capacity and voting policy, and the impact of these parameters on task performance. In this experiment, agents were assumed to be reliable.

[2]In the future, we propose to conduct experiments with high-performing, medium-performing and low-performing teams.

[3]In the future, we will model teams composed of different types of unreliable agents.

This assumption was relaxed in our second experiment, where we investigated the impact of number and type of unreliable agents (corrupt, lazy, conservative and selfish) and voting policy on task performance. In this experiment, we assumed that all agents exhibit a stable performance, and have an observation capacity of $OC = 3$ (this value yields the best performance for teams of stable agents, Section 3.3).

We ran one simulation for each combination of the simulation parameters in each experiment. For Experiment 1, we performed 120 simulations (5 types of teams $\times$ 8 values for $OC \times 3$ voting policies), and for Experiment 2 we ran 72 simulations (4 types of unreliable agents $\times$ 6 values for NUT $\times$ 3 voting policies). In addition, we ran one simulation for each of the benchmark settings, RAND and OMNI. Each simulation consists of ten trials, each divided into 1000 runs (we selected this number of trials and runs because it yields stable and continuous patterns of behaviour). Each run consists of a rescue task that is repeated until convergence is reached.

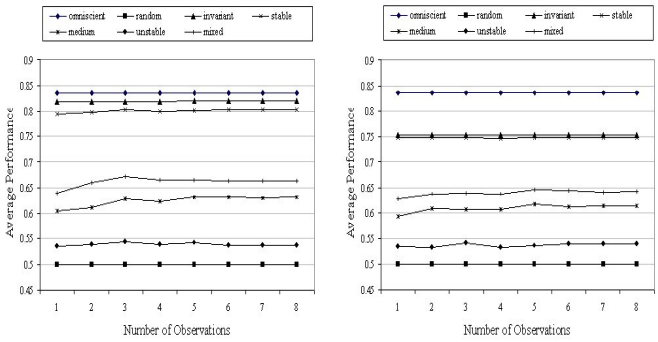
The IR and M for each agent are initialized at the beginning of each run. IR are initialized as specified by the type of the team (e.g., Stable or Unstable), and M are initialized with random values. We also conducted experiments where all the models are initialized with a value of 0.5 (medium expected performance), and with a value of 1.0 (high expected performance). The overall results are similar to those obtained with the randomly initialized models, except for the Invariant and Stable group of agents and the 0.5 initialization, which yield a worse average performance. This is because a run terminates when the chosen agent's performance is repeatedly better than 0.5, and so other agents who may be better are not given a chance, thereby converging to a local maximum (Section 3.3). The IR of each agent remain constant throughout a run (the agent's performance is drawn from the distribution specified in the IR), while M are updated from the observations made for each rescue in the run.

The process for reaching convergence works as follows. At the beginning of a run, different lifesavers may be proposed for a rescue task due to the discrepancy between the models maintained by the different agents. After each rescue, the agents update their models based on the performance of the chosen agent. Hence, when a rescue task is announced in the next turn, more agents are likely to propose the same lifesaver (but not necessarily the lifesaver chosen for the previous task). A run is terminated when the same lifesaver is chosen in N consecutive turns (we have experimented with $N = 2, 3, 4, 5$; the results presented in Section 3.3 are for $N = 5$).

Our measure of task performance for a run is the mean of the IR distribution for the agent on which the observers eventually converged. For instance, in the example in Table 1, this agent is A_2, whose $IR_{A_2}(rescue)$ has mean 0.8 (STDV=0.3). This measure reflects the final outcome of the combination of the parameters of the simulation for the run in question.

3.3 Results

Experiment 1. Figure 1 depicts the average task performance obtained with two voting policies as a function of OC for our seven types of teams – RAND, OMNI, Invariant, Stable, Medium, Unstable and Mixed. Figure 1(a) shows the results obtained with the optimistic policy, and Figure 1(b) shows the results for the majority policy. The results obtained for the weighted-observation policy are similar to those obtained for the majority policy.



(a) Optimistic policy (b) Majority policy

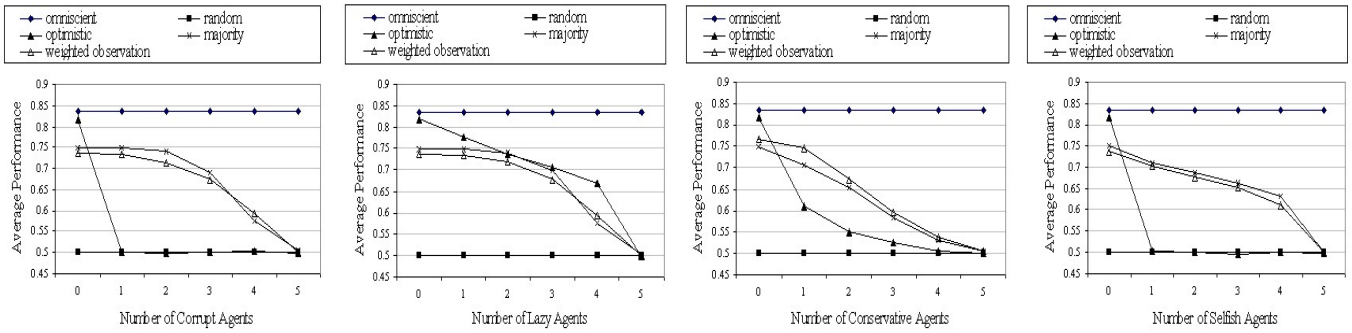
Figure 1: Average task performance obtained with the optimistic and the majority voting policies plotted against observation capacity for several types of teams

As expected, the results for the RAND and OMNI settings correspond to the worst and best performance respectively, and are used as a benchmark for comparison with the other settings. The performance for the Invariant team is slightly worse than that for the OMNI setting. This is due to the fact that agents in the Invariant team sometimes converge to a local maximum, which is reached when the agents in the team select an agent that is not the best. This happens when the agents under-estimate the performance of the best agent to the extent that it will never be proposed by any agent in the group, and hence will never perform the task. These results are consistent with the results obtained for the RAND, OMNI and default scenarios in our previous work [5].

As seen in Figure 1, the average performance obtained for the other types of teams is generally worse than that obtained for the Invariant team. This is due to the higher variability in agent performance. In fact, the more unstable the agents in the team are, the worse the performance becomes. We posit that the main reason for this outcome is that the observing agents are unable to update their models reliably when team members exhibit unstable performance.

The optimistic policy yields a substantially better performance for the Invariant and Stable teams than the majority policy, and it yields a slightly better performance for the Medium and Mixed teams than the majority policy (these results are significant with p=0.01). This is because if we assume that agents are honest and helpful (i.e., they always make the best proposal according to their models, and select the best of the proposals communicated by team members), the optimistic policy is similar to a global optimization, where the agent that appears to be best overall is selected. In contrast, the majority policy selects the most popular agent, which may not be the best overall.

Task performance improves for Medium and Mixed teams when agents are able to remember observations of the per-



(a) Corrupt (b) Lazy (c) Conservative (d) Selfish

Figure 2: Average task performance obtained with the optimistic, majority and weighted-observation voting policies plotted against the number of unreliable agents which are (a) Corrupt, (b) Lazy, (c) Conservative, and (d) Selfish.

formance of team members. This improvement is larger for the optimistic voting policy than for the majority policy (these results are significant with p=0.01). Further, this improvement is achieved with only 2-3 observations for the optimistic policy, and with 4-5 observations for the majority policy. This discrepancy may be caused by the need for additional "evidence" in order to get several agents to prefer the same agent, as required by the majority policy. The performance of Unstable teams is not affected by the voting policy or the agents' observation capacity, as the agents in these teams exhibit too much performance variation for the observer agents to reach reliable conclusions.

Experiment 2. Figure 2 depicts the average task performance obtained with three voting policies as a function of *NUT* and our four types of proposal-making behaviours (selfish, lazy, corrupt and conservative). Additionally, as above, we considered the RAND and OMNI settings for comparison purposes.

We found that the proposal-making behaviour of a group of agents has a significant influence on task performance. As expected, the optimistic policy compares favourably with the other policies when all agents are reliable. However, as soon as agents make selfish or corrupt proposals, the optimistic policy yields a performance that is as bad as that obtained with a random assignment of agents, i.e., the RAND scenario (Figure 2(a) and Figure 2(d)). A similar but less dramatic result is obtained for conservative agents (Figure 2(c)). The reason for these results is that once an agent proposes an inferior team member with a high attributed performance, this deficient proposal has a large influence on the proposal selection process. Only for lazy agents, which propose random agents with a random performance, the optimistic policy yields better results than the majority or weighted-observation policy.

For the corrupt, conservative and selfish agents, as the value of the *NUT* parameter increases, the majority and weighted-observation policies yield a more stable task performance than that obtained with the optimistic policy. In addition, one or two corrupt or lazy agents have almost no effect on task performance under the majority and weighted-observation policies, but as expected, performance deteriorates rapidly when there are four or five of these agents.

For groups with corrupt, lazy and selfish agents, the majority voting policy yields slightly better results than those obtained with the weighted-observation policy. In contrast, for groups with conservative agents, the weighted-observation policy yields better results. This may be explained as follows. The added weight of proposals made for previously observed agents also favours the proposals made by agents who make observations. These agents in turn are able to update their models, and thus make more accurate proposals than those made by conservative agents.

4. RELATED RESEARCH

Several research projects have demonstrated that maintaining models of features of collaborators can benefit different aspects of task performance. Such features have been modeled in order to achieve flexible team behaviour in military domains [10] and determine collaborators' behaviour based on utility functions [4].

Tambe [10] introduced a general teamwork model for communication and behaviour (called STEAM), which enables members of a team to coordinate their activities in order to reach a joint goal. However, the dynamic modeling of agents by team members is limited to a single status variable for each agent (e.g., "busy" or "shot down"), as opposed to the capabilities and resource limits modeled in our work.

Suryadi and Gmytrasiewicz [9] and Gmytrasiewicz and Durfee [4] investigated agents that apply a decision-theoretic procedure to make decisions that maximize their own individual payoffs. This procedure takes into account the "payoff matrix" of collaborators, which in turn is learned from observations of their behaviour. Our system also learns the behaviour of other agents from observations (although we learn only the mean and standard deviation of their performance). However, whereas Suryadi and Gmytrasiewicz's agents make individual decisions and do not communicate with each other, our agents communicate proposals in order to make a joint decision.

Our *OC* parameter is similar to the attentional limitations considered in [11], and is related to the memory boundedness investigated in [8]. However, both Walker [11] and Rubinstein [8] also considered inferential limitations, while we

consider agent-modeling limitations.

Our agents' ability to build models of agents from observations resembles the work of Davison and Hirsh [3]. Their model gave greater weight to more recent events than to earlier events, while we achieve a similar behaviour through our OC parameter, which specifies that only the last K observations should be considered.

Voting has been investigated in the context of collaborative filtering, where the preference of one agent is predicted by taking into account the preferences of other agents [7]. Conitzer and Sandholm [2] have recently studied the problem of voting manipulation. They define "tweaks" that are designed to make manipulation of voting protocols computationally hard. In our research, we consider different types of unreliable proposal-making behaviours, while Conitzer and Sandholm assume a generic behaviour that can influence a voting process.

Busetta *et al.* [1] proposed a form of group communication, called *channeled multicast*, which dynamically selects the optimal agent for a service in a given context. While several assumptions of Busetta *et al.*'s research are similar to the assumptions of our research, their agents apply a different selection mechanism to find the optimal agent for a task. Their agents use selection policies specifically designed to cope with unreliable communication channels, rather than unreliable agents. Agents considered in their project overhear communication and if an agent believes it is suitable for a task, it negotiates a handover with the agent in charge. An unreliable agent could greatly diminish the overall performance of a team, because an unreliable agent may have successfully negotiated a handover. In our work, agents maintain models of team members in order to assess how suitable the team members are to perform a task (Busetta *et al.*'s agents estimate only their own performance).

Finally, if the team would know which team members are unreliable, then the team could simply exclude them from the voting process. Reputation mechanisms aim to identify reliable agents, which can be nominated to make decisions for the group. However, the process of identifying a trusted third party, and how the trusted third party would assess the opinions of reliable and unreliable members of the team are problems in their own right [6].

5. CONCLUSION

We have extended our ETAPP collaboration framework to model team members that exhibit variable task performance and different proposal-making behaviours. We have modeled four types of unreliable proposal-making behaviours, employed a probabilistic representation to model variable task performance, and endowed observer agents with the capability to observe team members and a procedure for building agent models from observations. We then investigated the interaction between these parameters and three voting policies.

We evaluated our extensions by means of a simulated rescue scenario, where we varied the agents' proposal-making behaviour, the performance stability of teams of agents, the number of observations retained by observer agents, and the policy used to allocate agents to tasks. Our results show that performance variability has a significant impact on task performance, and that when agents are reliable, it is enough to make a few observations to improve task performance for stable, mixed and medium teams. Further, the task performance obtained by applying the optimistic selection policy is at least as good as that obtained with the majority policy. In contrast, when agents are unreliable, the majority and weighted-observation policies yield the most stable performance for all proposal-making behaviours, while the optimistic policy produces an unstable task performance, even when only a few agents are unreliable.

6. REFERENCES

[1] P. Busetta, M. Merzi, S. Rossi, and F. Legras. Intra-role coordination using group communication: A preliminary report. In F. Dignum, editor, *Advances in Agent Communication*, LNAI 2922. Springer, 2004.

[2] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *IJCAI-2003 – Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 781–788, Acapulco, Mexico, 2003.

[3] B. Davison and H. Hirsh. Predicting sequences of user actions. In *Notes of the AAAI/ICML 1998 Workshop on Predicting the Future: AI Approaches to Time-Series Analysis*, Madison, Wisconsin, 1998.

[4] P. J. Gmytrasiewicz and E. H. Durfee. Rational Communication in Multi-Agent Environments. *Autonomous Agents and Multi-Agent Systems*, 4(3):233–272, 2001.

[5] C. Guttmann and I. Zukerman. Towards Models of Incomplete and Uncertain Knowledge of Collaborators' Internal Resources. In J. Denzinger, G. Lindemann, I. J. Timm, and R. Unland, editors, *Second German Conference on MultiAgent system TEchnologieS (MATES)*, LNAI 3187, Erfurt, Germany, 2004. Springer.

[6] L. Mui, M. Mohtashemi, and A. Halberstadt. Notions of reputation in multi-agents systems: a review. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, 280–287, New York, USA, 2002. ACM Press.

[7] D. M. Pennock, E. Horvitz, and C. L. Giles. Social choice theory and recommender systems: Analysis of the axiomatic foundations of collaborative filtering. In *AAAI-2000 – Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 729–734, Austin, Texas, 2000.

[8] A. Rubinstein. *Modeling bounded rationality.* Zeuthen lecture book series. MIT Press, Cambridge, Massachusetts, 1998.

[9] D. Suryadi and P. J.Gmytrasiewicz. Learning models of other agents using influence diagrams. In *Proceedings of the Seventh International Conference on User Modeling*, 223–232, Banff, Canada, 1999.

[10] M. Tambe. Towards Flexible Teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[11] M. A. Walker. The effect of resource limits and task complexity on collaborative planning in dialogue. *Artificial Intelligence*, 1-2(85):181–243, 1996.