

Agents with Limited Modeling Abilities: Implications on Collaborative Problem Solving

Christian Guttman and Ingrid Zukerman

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

Abstract. Collaboration plays a critical role when a group is striving for goals which are difficult or impossible to achieve by an individual. Knowledge about collaborators' contributions to a task is important when solving problems as a team. However, a problem in many collaboration scenarios is the uncertainty and incompleteness of such knowledge. To investigate this problem, we present a collaboration framework where team members use models of collaborators' performance to estimate contributions to a task, and propose agents for tasks based on these estimations. We conducted a simulation-based study to assess the impact of modeling limitations on task performance. The main results of our simulation are that maintaining models of agents improves task performance, but exhaustive model maintenance is not essential. Additionally, we found that the ability of agents to update their models has a large impact on task performance.

We then extended our framework to support more refined agent models, and performed additional simulated studies. Our results indicated that task performance is improved by the availability of additional reasoning resources and the use of probabilistic models that represent variable agent performance.

1 Introduction

Knowledge of collaborators (or team members) is important when activities are planned and performed together in order to reach a goal. However, estimating contributions of team members to a task is a difficult problem, particularly when agents do not have accurate knowledge of collaborators' internal resources (i.e., capabilities and expertise). This can be due to a lack of experience with new team members or new tasks, or due to agents' limited information-gathering capabilities.

In this paper, we investigate the problem of estimating contributions of team members based on incomplete and uncertain knowledge of collaborators in the context of three assumptions commonly made with respect to open systems [1, 2, 3, 4, 5]:

Assumption 1. Individuals have local knowledge rather than global knowledge (referred to as decentralized knowledge acquisition and administration). This

means that different agents have distinct models of collaborators' internal resources with different levels of uncertainty and incompleteness.

Assumption 2. Individuals exert control on the environment autonomously, rather than merely following commands given by an overarching administration or a team member (referred to as decentralized control). As a result, the behaviour of individuals should be coordinated as part of the coordination of the team.

Assumption 3. Performance is influenced by the resources in the environment and the agents' interactions with the environment and with other agents. Thus, while in principle agents may have the capability to perform a task, success is not guaranteed if resources become restricted or unavailable.

These open system assumptions are more realistic than those made for closed systems. Closed systems are self-contained systems which assume that knowledge of collaborators' internal resources is global. This means that agents have the same knowledge of collaborators' capabilities, as opposed to a situation where each agent has local knowledge (Assumption 1). In a closed system, a central agent controls a collaboration of agents, in contrast to a collaboration that requires the consent of each agent (Assumption 2). Finally, task performance in a closed system depends on factors that are explicitly represented, while task performance in open systems is influenced by external factors that are not explicit (Assumption 3).

Open system assumptions hold for various application scenarios, such as routing in peer-to-peer networks, decentralized task allocation in autonomous robot groups, coordination of mixed robot and human teams, fault detection in a computational grid, planning in multi-modal transport logistics, the design and simulation of natural immune systems, and intrusion detection in a network of systems. As an example of an open system, consider a situation where several distinct groups of mobile devices intend to establish a decentralized telecommunication infrastructure (the devices in each group are made by the same company, and the devices in different groups are made by different companies). The mobile devices in each group have diverse capabilities, which are not known to the mobile devices in other groups prior to the collaboration. Thus, an agent that collaborates well with mobile devices in its own group may encounter problems when collaborating with agents in other groups.

In this paper, we present *ETAPP* (*Environment Task Agents Policy Protocol*) – a computational framework that was developed to support the investigation of different aspects of collaboration in a team of agents, with a particular emphasis on the study of the impact of maintaining models of teammates when agents have limited modeling abilities. The ETAPP framework describes a collaboration scenario, where team members maintain their own models of collaborators' capabilities to estimate their contribution to a task, and use these models to propose agents for tasks. Coordination is important in this framework, particularly when proposals made by several team members are considered to make a group decision about allocating agent(s) to a task.

We report on the results of two experimental studies conducted using this framework. In our first experiment, we examined the influence of modeling limitations on task performance. In particular, we considered variations in memory capacity and learning ability of agents, and variations in accuracy of the models maintained by the agents. The insights obtained from this experiment led to refinements which allowed our framework to take into account variable agent behaviour. The impact of these refinements were investigated in our second experiment.

The remainder of this paper is structured as follows. In Section 2, we describe the ETAPP framework for basic collaboration scenarios. In Section 3, we present our initial simulation-based experiment and analyze its results. In Section 4, we discuss the extensions of the ETAPP framework, followed by our second experiment in Section 5. Related research is considered in Section 6, followed by our conclusions and plans for future work.

2 The ETAPP Framework

The ETAPP framework is based on the assumptions of open systems mentioned in the previous section. In addition, we assume that

- agents exhibit deterministic performance. This assumption implies that an agent’s performance is the same every time a task is performed under the same conditions. Hence, the true performance of an agent can be learned from a single observation.
- agents use a common language and interaction protocol.
- agents are committed to strive for an optimal task outcome. This means that they optimize the team performance according to the criteria of a task rather than their own criteria.

In this section, we describe the ETAPP framework for collaborative agents. For our description we adopt the following notation. Groups of agents are denoted in uppercase letters (e.g., A), individual agents are denoted in uppercase letters with an index (e.g., A_i), and sub-groups of agents in uppercase letters with tilde and an index (e.g., \widetilde{A}_i). Activities are denoted in lowercase letters with an index (e.g., a_i). Evaluation parameters of activities are denoted in Greek letters with an index (e.g., α_i), and estimations of evaluation parameters are denoted in Greek letters with hat and an index (e.g., $\widehat{\alpha}_i$). Cardinalities of sets are denoted in lowercase letters (e.g., $m = |\widetilde{A}_i| = |\{A_1, \dots, A_m\}|$).

Definition 1. *A collaborative scenario S is represented by a tuple with five elements E, T_E, A, P_A , and P , i.e.,*

$$S = (E, T_E, A, P_A, P)$$

E corresponds to an environmental state space under which a task T_E is specified, A denotes a group of agents that considers task T_E , P_A is a policy which enables the agents in A to make joint decisions, and P is a protocol that controls the interaction between agents.

In the following sub-sections, we define the elements of a collaborative scenario.

2.1 Environmental State Space E

Definition 2. *An environment E is a state space described by predicates which represent properties of (and relations between) locations and objects. A state $e \in E$ describes the status of these predicates at a particular step in a collaboration.*

For example, an environment E may consist of a set of locations $L = \{loc_1, \dots, loc_k\}$ and a set of objects $O = \{obj_1, \dots, obj_l\}$. Each location is described by its coordinates, e.g., $loc_i = (x_i, y_i, z_i)$ for $i = 1, \dots, k$. Each object is described by features such as location ($loc(obj_j) = loc_i$), weight ($weight(obj_j)$) and size ($size(obj_j)$) for $j = 1, \dots, l$. A specific instance of this environment would be a “table environment”, which has two locations, $house_a$ and $house_b$, and two objects, a car and a table. An example of a state e is $at(car, house_a) \wedge in(table, car)$, viz the car is located at $house_a$ and the table is located in the car. Currently, we do not represent uncertainties regarding observations of locations and objects in the environment E .

2.2 Task T_E

To assess how well a task has been accomplished, task performance is measured based on criteria that evaluate the outcome of activities.

Definition 3. *A task T_E is represented by a tuple with three elements EC_T , EF_T and ACT_T , i.e.,*

$$T_E = (EC_T, EF_T, ACT_T)$$

The elements of the tuple T_E are described as follows.

- EC_T specifies the Evaluation Criteria relevant to task T , e.g., speed, quality or profit. EC_T provides an array of place holders $EC_T = \{ec_1, \dots, ec_n\}$ which receive values for these criteria when agents are proposed for an activity. 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 following evaluation function maximizes a linear combination of n evaluation criteria:

$$EF_T = \max \sum_{i=1}^n ec_i w_i$$

where $w_i \in [0, 1]$ for $i = 1, \dots, n$ are the weights assigned to the evaluation criteria. A weight close to 0 indicates a minimal impact of a criterion on task performance, and a weight close to 1 indicates a high impact.

- ACT_T is a set of activities $ACT_T = \{a_1, \dots, a_k\}$. Agents perform activities a_j for $j = 1, \dots, k$ to accomplish the goal of a task (a predefined environmental state). Individual agents use the evaluation function EF_T to evaluate the agent performance of activities (Section 2.3.2).

For example, consider the task of moving the *table* in the table environment from $house_a$ to $house_b$ by using a *car*. The goal of this task is that the *table* is located in $house_b$, i.e., $in(table, house_b)$, and the activity of this task is *lift*. The evaluation criteria are ec_{time} and $ec_{quality}$, with ec_{time} being more important than $ec_{quality}$. These conditions are represented in an evaluation function such as the following.

$$EF_T = \max\{ec_{time} \times 1.0 + ec_{quality} \times 0.8\} \quad (1)$$

The task specification T_E (EC_T , EF_T and ACT_T) is known and accessible to all the agents in a group during a collaboration. That is, they know the evaluation criteria, the evaluation function (and weights), and the activities. When an agent assesses team members for a task, the values for the evaluation criteria of an activity are filled in according to an agent's estimations of team members' capabilities (Section 2.3).

2.3 Agents and Teams of Agents A

Our framework considers a team of autonomous agents that perform tasks.

Definition 4. A team of agents A is a set $\{A_1, \dots, A_q\}$, such that A_i for $i = 1, \dots, q$ is an individual agent and \widehat{A}_j for $j = 1, \dots, r$ is a subset (or sub-team) of A , where $r = |\wp(A)|$ is the cardinality of the powerset of A .

The behaviour of an autonomous agent is based on its own skills, reasoning procedures, and knowledge of members in the team. Hence, we propose the following definition of an autonomous agent.

Definition 5. An agent $A_i \in A$ is represented by a tuple with three elements IR_{A_i} , M_{A_i} and RA_{A_i} , i.e.,

$$A_i = (IR_{A_i}, M_{A_i}, RA_{A_i})$$

The elements of the tuple A_i are described as follows.

- IR_{A_i} specifies agent A_i 's Internal Resources, which represent an agent's capabilities, such as skills or knowledge. It is encoded as a value that describes how well the agent can perform an action in terms of the Evaluation Criteria of the task (Section 2.3.1). The internal values of performance 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, Sections 3 and 5).

- M_{A_i} denotes Models maintained by agent A_i , which are represented as estimations of the performance of agents and sub-groups of agents in A (Section 2.3.2).
- RA_{A_i} , the Reasoning Apparatus of A_i , consists of processes for reasoning about agents in the team (Section 2.3.3). The RA also allows for interacting with team members according to a group protocol (Section 2.5).

When a group of agents A is given a task T_E , the agents in the group act in order to optimize the evaluation function EF_T . Each agent in the group participates in the process of selecting agent(s) to perform activities of a task. The Reasoning Apparatus (RA) of the agents is used in this selection process, and the performance of the candidate agent(s) is estimated by the models M . Once agents are selected for the task, their real performance is determined by their Internal Resources (IR), and the RA updates the models maintained by the agents in the team for the selected agent(s). We now describe each element of the tuple A_i in more detail.

2.3.1 Internal Resources IR_{A_i}

Formally, we say that $IR_{A_i} = \{act_1, \dots, act_s\}$ is a set of activity tuples, where

- s denotes the number of activities defined in the task specification T_E .
- $act_k = (a_k, \alpha_1(a_k), \dots, \alpha_n(a_k))$ for $k = 1, \dots, s$ is a tuple with $n + 1$ elements for activity a_k (n is the number of Evaluation Criteria). The activities a_k for $k = 1, \dots, s$ correspond to the activities defined in the task specification T_E .
- $\alpha_j(a_k) \in [0, 1]$ for $j = 1, \dots, n$ is an internal factor that represents the level of performance of activity a_k with respect to Evaluation Criterion ec_j when performed by A_i . 0 represents the worst performance and 1 represents the optimal performance.

2.3.2 Models M_{A_i}

Models represent estimations of agents' abilities as opposed to the real abilities of agents. Formally, we say that M_{A_i} is a family of models $\{M_{A_i}(\widetilde{A}_1), \dots, M_{A_i}(\widetilde{A}_r)\}$, where

- r is the cardinality of the powerset of A .
- Model $M_{A_i}(\widetilde{A}_l) = \{\widehat{act}_1, \dots, \widehat{act}_s\}$ for $l = 1, \dots, r$ is a set of activity tuples.
- s denotes the number of activities defined in the task specification T_E .
- $\widehat{act}_k = (a_k, \widehat{\alpha}_1(a_k), \dots, \widehat{\alpha}_n(a_k))$ for $k = 1, \dots, s$ is a tuple with $n + 1$ elements for activity a_k (n is the number of Evaluation Criteria). The activities a_k for $k = 1, \dots, s$ correspond to the activities defined in the task specification T_E .
- $\widehat{\alpha}_j(a_k) \in [0, 1]$ for $j = 1, \dots, n$ is an estimated evaluation parameter of the level of performance of a_k with respect to Evaluation Criterion $ec_j \in EC_T$ when performed by \widetilde{A}_l . That is, $\widehat{\alpha}_j(a_k)$ is an estimation of $\alpha_j(a_k) \in IR_{A_i}$.
- If a model for \widetilde{A}_l is not known to A_i then $M_{A_i}(\widetilde{A}_l) = \emptyset$.

If an agent has sufficient memory, it could maintain $2^q - 1$ models (one model for each subteam excluding a model of the empty set), where q is the number of agents. However, in realistic settings, an agent has limited memory and cannot

maintain an exhaustive number of models. This constraint is represented as a parameter pertaining to an agent’s memory boundedness (Section 4).

Additionally, the accuracy of models at the beginning of a collaboration is important, because they influence how well agents estimate team members before any observations have been made. This condition is represented as a parameter that describes the relationship between the real capabilities of agents and the models of agents (in our first experiment, we consider models that underestimate the real capabilities of an agent).

The influence of memory boundedness and the accuracy of models on task performance is investigated in our first experiment (Section 3).

2.3.3 Reasoning Apparatus RA_{A_i}

The RA receives as input an environment E , a task T , models M_{A_i} , a policy P_A , and protocol P . It consists of processes for

- making a proposal, i.e., selecting agent(s) for a task according to the estimated value of their evaluation function for the activity,
- communicating this proposal to other agents (Section 2.5),
- applying a policy P_A in order to select a proposal from all communicated proposals (Section 2.4), and
- updating its models M_{A_i} based on the observed performance of the selected agent(s) (Section 2.5). The learning ability of an agent is important in improving the accuracy of agent models. We assume invariant and deterministic agent performance, and hence agents need only one observation to accurately predict future performance of an agent. If an agent is not able to learn from observations it will make predictions based on obsolete models. Therefore, we experimented with teams that consist of agents that update their models and agents that do not change their models (Section 3).

The estimations used by an agent A_i , the procedures for making and communicating a proposal, and the process for updating M_{A_i} may differ from those employed by other agents.

2.3.4 Example

To illustrate the difference between agents’ internal resources IR_{A_i} and the models of internal resources M_{A_i} , let us return to the table scenario, where the evaluation criteria are time and quality. Two agents A_1 and A_2 are considered for the table task and both agents are able to perform the *lift* activity, which moves a “liftable” object from one location to another. The internal resources for A_1 and A_2 are

- $IR_{A_1} = \{act_1\}$, where $act_1 = (lift, \alpha_{time}(lift) = 0.25, \alpha_{qual}(lift) = 0.25)$,
- $IR_{A_2} = \{act_1\}$, where $act_1 = (lift, \alpha_{time}(lift) = 0.15, \alpha_{qual}(lift) = 0.2)$.

The activity tuple act_1 of each agent has two internal factors which reflect the agent’s performance of the activity *lift*. For example, 0.15 is the internal factor

for agent A_2 's performance in relation to time, and 0.2 is the internal factor for its performance in relation to quality. According to IR_{A_1} and IR_{A_2} , A_1 lifts an object faster than A_2 (i.e., $0.25 > 0.15$), and the outcome of A_1 's *lift* is of better quality than that of A_2 's *lift* (i.e., $0.25 > 0.2$).¹

IR_{A_1} and IR_{A_2} are not known or accessible to A_1 and A_2 . Therefore, the internal resources of A_1 and A_2 are estimated through models. An agent can access only its own models (it can not access the models maintained by other agents), e.g., A_1 can access only M_{A_1} . These models allow each agent to estimate the contribution of individuals or sub-groups of the team A . For instance, in the above example let us say that the set M_{A_1} for A_1 is as follows:

$$\begin{aligned}
M_{A_1} &= \{M_{A_1}(\widetilde{A}_1 = A_1), M_{A_1}(\widetilde{A}_2 = A_2), M_{A_1}(\widetilde{A}_3 = (A_1, A_2))\}, \text{ where} \\
&\cdot M_{A_1}(A_1) = \{\widehat{act}_1\} \\
&\quad \cdot \widehat{act}_1 = (\widehat{lift}, \widehat{\alpha_{time}}(\widehat{lift}) = 0.15, \widehat{\alpha_{qual}}(\widehat{lift}) = 0.1), \\
&\cdot M_{A_1}(A_2) = \{\widehat{act}_1\} \\
&\quad \cdot \widehat{act}_1 = (\widehat{lift}, \widehat{\alpha_{time}}(\widehat{lift}) = 0.25, \widehat{\alpha_{qual}}(\widehat{lift}) = 0.35), \\
&\cdot M_{A_1}(\{A_1, A_2\}) = \{\widehat{act}_1\} \\
&\quad \cdot \widehat{act}_1 = (\widehat{lift}, \widehat{\alpha_{time}}(\widehat{lift}) = 0.4, \widehat{\alpha_{qual}}(\widehat{lift}) = 0.45),
\end{aligned}$$

and the set M_{A_2} for A_2 is as follows:

$$\begin{aligned}
M_{A_2} &= \{M_{A_2}(\widetilde{A}_1 = A_1), M_{A_2}(\widetilde{A}_2 = A_2), M_{A_2}(\widetilde{A}_3 = (A_1, A_2))\}, \text{ where} \\
&\cdot M_{A_2}(A_1) = \{\widehat{act}_1\} \\
&\quad \cdot \widehat{act}_1 = (\widehat{lift}, \widehat{\alpha_{time}}(\widehat{lift}) = 0.1, \widehat{\alpha_{qual}}(\widehat{lift}) = 0.2), \\
&\cdot M_{A_2}(A_2) = \{\widehat{act}_1\} \\
&\quad \cdot \widehat{act}_1 = (\widehat{lift}, \widehat{\alpha_{time}}(\widehat{lift}) = 0.25, \widehat{\alpha_{qual}}(\widehat{lift}) = 0.1), \\
&\cdot M_{A_2}(\{A_1, A_2\}) = \{\widehat{act}_1\} \\
&\quad \cdot \widehat{act}_1 = (\widehat{lift}, \widehat{\alpha_{time}}(\widehat{lift}) = 0.15, \widehat{\alpha_{qual}}(\widehat{lift}) = 0.2).
\end{aligned}$$

The values for activity parameters for models with $|\widetilde{A}_j| > 1$ are determined by combining models of individual agents using specific aggregation functions for different types of activities. These functions reflect the observation that the performance of an activity improves up to a point as the number of agents increases, and beyond that point, the performance reaches a plateau or deteriorates. For example, lifting a table typically requires 2 or 3 agents; fewer agents may be unable to lift the table, and more agents may jeopardize a successful completion of the task.

2.4 Policy P_A

In order to exhibit coordinated behaviour, the agents in a group should adopt only one *proposal*. One way to ensure this outcome is for the agents to apply a joint decision policy P_A .

¹ Although a shorter time is better than a longer time, the values for the evaluation criteria are mapped to a $[0,1]$ range where higher values always indicate better performance than lower values.

Definition 6. Policy P_A consists of an election mechanism which selects a proposal in a democratic fashion from those made by several agents (by incorporating the opinion of each individual in a fair manner). The outcome of this election is a collective choice of one proposal.

Definition 7. A proposal made by an agent A_i is a tuple that specifies an activity a_k to be carried out by agent(s) \widetilde{A}_j , and provides estimations of the proposed sub-group’s performance (*prop_perform*) with respect to the evaluation criteria $\{ec_1, \dots, ec_n\}$. These estimations are values $\widehat{\alpha}_m$ of A_i ’s model of \widetilde{A}_j . The activity a_k corresponds to one of the activities defined in the Task Specification (Section 2.2). Formally, a proposal has the form

$$\text{proposal}(A_i) = (\text{activity } a_k, \text{agent } \widetilde{A}_j, \text{prop_perform } (\widetilde{A}_j)(\widehat{\alpha}_1, \dots, \widehat{\alpha}_n))$$

Every agent subscribes to policy P_A upon entering a team and applies this policy locally. The local application of a policy by agents is a decentralized version of a central application of the policy followed by a broadcast of the outcome. That is, proposals made by team members are processed by each agent according to the policy as opposed to one central agent that selects a team member and then distributes the selection outcome to team members. A reason for a local application of the policy is that a central agent could manipulate the policy for selfish purposes (e.g., the central agent could always select itself before other agents and distribute this selection outcome to team members). Only the central agent collects and processes the proposals, and other agents receive outcome of the policy from the central agent. If agents apply the policy locally, each agent knows which agent is selected. Agents are then able to identify an invalid proposal made by an agent that applies its own criteria (and not the policy), and thus the team might decide to exclude this agent from the team. In future work, we will compare the local and central application of the policy.

In our initial experiments, we considered an *optimistic policy* for selecting a proposal. An optimistic policy selects the proposed agent with the most promising performance compared to all the other proposed agents. For example, according to the evaluation function in Equation 1 and the values for M_{A_1} and M_{A_2} in the previous section, A_1 proposes that A_1 and A_2 lift the table together (with $EF = 0.4 \times 1 + 0.45 \times 0.8 = 0.76$), while A_2 proposes that A_2 lifts the table alone (with $EF = 0.25 \times 1 + 0.1 \times 0.8 = 0.33$). Each of these proposals is the best proposal according to the models maintained by each agent. However, A_1 ’s proposal is selected, because it is more optimistic than the proposal of A_2 (i.e., A_1 ’s proposal has a higher value compared to A_2 ’s proposal).

A significant shortcoming of the optimistic policy is that its outcome may be unrealistic when agents make a proposal based on models that are far removed from reality. This shortcoming could lead to undesirable results, e.g., where a task can not be completed in time. Hence, in our subsequent experiments we also considered the *majority policy* – a more stable policy which chooses the agent that is preferred by most agents [6].

Another way to reach consensus between agents is by negotiation. This process may reveal a better agent for an activity than the voting process. However,

-
1. $\forall A_i \in A : \text{PROPOSAL}_{A_i}, \text{PROPOSAL_LIST} \leftarrow \emptyset$
 2. **for** each activity **do**
 3. $\forall A_i \in A : A_i$ generates PROPOSAL_{A_i}
 4. $\forall A_i \in A : A_i$ communicates PROPOSAL_{A_i}
 5. $\text{PROPOSAL_LIST} \leftarrow \{\text{PROPOSAL}_{A_1}, \dots, \text{PROPOSAL}_{A_q}\}$
 6. $\text{SELECTED_PROPOSAL} \leftarrow \text{select_proposal}(\text{PROPOSAL_LIST}, P_A)$
 7. observe $\text{REAL_PERFORM}(\tilde{A}_{\text{selected}})$, where $\tilde{A}_{\text{selected}} \in \text{SELECTED_PROPOSAL}$
 8. $M'_{A_i}(\tilde{A}_{\text{selected}}) \leftarrow \text{REAL_PERFORM}(\tilde{A}_{\text{selected}})$
 9. **end for**
-

Figure 1. Pseudocode of the group interaction protocol.

agents must possess the capability to negotiate, which is a research field in its own right [7, 8, 9]. In the future, we propose to investigate a combination of voting with limited negotiation.

2.5 Interaction Protocol

Each agent follows the interaction protocol depicted in Figure 1 in order to coordinate activities with other agents. This protocol uses the following variables:

- $\text{PROPOSAL}(A_i)$ is a PROPOSAL made by agent A_i for $i \in 1, \dots, q$,
- PROPOSAL_LIST is a list of PROPOSALS communicated by agents,
- $\text{REAL_PERFORM}(\tilde{A}_l)$ is an array of values that represent the real performance of agent(s) \tilde{A}_l with respect to the evaluation criteria $\{ec_1, \dots, ec_n\}$.
- M'_{A_i} is an updated version of M_{A_i} after the performance of the agents selected in the proposal has been observed.

In line 3 of the protocol in Figure 1, each agent generates a proposal that matches one or more agents with an activity. Each agent A_i uses its evaluation parameters $\hat{\alpha} \in M_{A_i}$ to estimate the performance of each agent and subgroup of agents for the activity in question. These estimates are sorted according to the evaluation function EF_T , and the agent then selects the best proposal (in [10] we considered other ways of selecting a proposal). For instance, as shown in the above example, this process results in A_1 generating the proposal $[\text{lift}, (A_1, A_2), \text{PROP_PERFORM}(A_1, A_2)(0.4, 0.45)]$ for the table scenario.

Once an agent has generated a proposal, it is communicated to the other agents (line 4 in Figure 1). This is done by broadcasting the proposal to every collaborator. Each agent then stores the proposals received from the other agents in PROPOSAL_LIST (line 5 in Figure 1). In order to reach a joint decision regarding a proposal, each agent applies policy P_A locally (line 6), selects a proposal, and commits to the outcome of this selection.

Upon completion of the selection process, the agents specified in the selected proposal ($\tilde{A}_{\text{selected}}$) perform the task or activity in question (line 7). The real performance of these agents, $\text{REAL_PERFORM}(\tilde{A}_{\text{selected}})$, is based on their IR (in contrast to the proposals, which use the models M). This real performance



Figure 2. A typical Surf Rescue scenario.

is then used to update the existing models. At present, we employ a simple updating function where the values that estimate a proposed performance (i.e., $\hat{\alpha}$) are replaced by values obtained from the real performance. In Section 4, we consider algorithms which learn probabilistic agent models and take into account the recency of observations during this process.

2.6 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 empirical studies (Section 3 and 5). In this scenario, which is depicted in Figure 2, the environment E consists of the *beach* and the *ocean*, and the task is to rescue the *Distressed Person* (DP) in the shortest time possible. The lifesaver team assigns an agent to the *rescue* activity based on models of an agent’s capabilities. This means that the set of evaluation criteria is $EC_T = \{ec_{time}\}$. The evaluation function is $EF_T = \max\{ec_{time}\}$, where a high value of ec_{time} signifies high performance, and thus less time. EC_T and EF_T are accessible to the agents that perform the activities of the task.

In this example, we have five lifesavers $A = \{A_1, A_2, A_3, A_4, A_5\}$ (at the beach), and the task activity is to *rescue* the DP . This activity is represented in the IR and the models M , and its value (according to ec_{time}) depends on the distance between the beach and the DP . The models in M are of individual lifesavers (i.e., $|\tilde{A}_j| = 1$), rather than of sub-groups of lifesavers. We do not assume that agents have prior knowledge of the performance of team members, and hence the models of each agent are initialized with random values between 0 and 1 (in our second experiment, we investigate other methods of initialization, where all initial models have a value of 0.5, or all initial models have a value of 1.0).

Although each agent (lifesaver) in A possesses the capability to perform the *rescue* activity, the agents endeavour to assign the best lifesaver to the rescue based on models that each lifesaver has of the lifesavers in the team. To this effect, the agents employ the optimistic policy P_A , which selects the proposal

Group Status	Action A_1	Action A_2	IR_{A_3}	IR_{A_4}	$M_{A_1}(A_3)$	$M_{A_1}(A_4)$	$M_{A_2}(A_3)$	$M_{A_2}(A_4)$
1 initialize	initialize IR, M	initialize IR, M	0.3	0.6	0.6	0.1	0.4	0.5
Rescue 1								
2 select agent	$\max\{M_{A_1}\}$	$\max\{M_{A_2}\}$	0.3	0.6	0.6	0.1	0.4	0.5
3 broadcast	$\text{PROP}_{A_1}=(A_3, 0.6)$	$\text{PROP}_{A_2}=(A_4, 0.5)$	0.3	0.6	0.6	0.1	0.4	0.5
4 apply P_A	select PROP_{A_1}	select PROP_{A_1}	0.3	0.6	0.6	0.1	0.4	0.5
5 observe	$\text{REAL_PERFORM}(A_3)$	$\text{REAL_PERFORM}(A_3)$	0.3	0.6	0.6	0.1	0.4	0.5
6 update M	$\text{REAL_PERFORM}(A_3)$	$\text{REAL_PERFORM}(A_3)$	0.3	0.6	0.3	0.1	0.3	0.5
Rescue 2								
7 select agent	$\max\{M_{A_1}\}$	$\max\{M_{A_2}\}$	0.3	0.6	0.3	0.1	0.3	0.5
8 broadcast	$\text{PROP}_{A_1}=(A_3, 0.3)$	$\text{PROP}_{A_2}=(A_4, 0.5)$	0.3	0.6	0.3	0.1	0.3	0.5
9 apply P_A	select PROP_{A_2}	select PROP_{A_2}	0.3	0.6	0.3	0.1	0.3	0.5
10 observe	$\text{REAL_PERFORM}(A_4)$	$\text{REAL_PERFORM}(A_4)$	0.3	0.6	0.3	0.1	0.3	0.5
11 update M	$\text{REAL_PERFORM}(A_4)$	$\text{REAL_PERFORM}(A_4)$	0.3	0.6	0.3	0.6	0.3	0.6
Rescue 3								
12 select agent	$\max\{M_{A_1}\}$	$\max\{M_{A_2}\}$	0.3	0.6	0.3	0.6	0.3	0.6
13 broadcast	$\text{PROP}_{A_1}=(A_4, 0.6)$	$\text{PROP}_{A_2}=(A_4, 0.6)$	0.3	0.6	0.3	0.6	0.3	0.6
14 apply P_A	select PROP_{A_2}	select PROP_{A_2}	0.3	0.6	0.3	0.6	0.3	0.6
15 observe	$\text{REAL_PERFORM}(A_4)$	$\text{REAL_PERFORM}(A_4)$	0.3	0.6	0.3	0.6	0.3	0.6
16 update M	$\text{REAL_PERFORM}(A_4)$	$\text{REAL_PERFORM}(A_4)$	0.3	0.6	0.3	0.6	0.3	0.6

Table 1. Sample trace of the interaction protocol.

that promises the best performance. The group interaction between the agents is based on the protocol in Figure 1.

To demonstrate the effect of the reasoning processes and the interaction protocol on the models of individual agents, we confine our discussion to the interaction between agents A_1 and A_2 . In this example, A_1 and A_2 maintain only models of agents A_3 and A_4 (i.e., $M_{A_1}(A_3), M_{A_1}(A_4), M_{A_2}(A_3)$ and $M_{A_2}(A_4)$), and generate proposals involving these agents. The values for the internal resources of A_3 and A_4 , and the initial models for these agents are shown in line 1 of Table 1. As seen from this line, the models are not consistent with the real performance of the agents in question.

The first rescue task is performed as follows. A_1 and A_2 first select the best agent according to the estimated time performances in their models (line 2 in Table 1), and broadcast a proposal based on the selected agent (line 3). For A_1 this proposal is $\text{PROP}_{A_1}=(A_3, 0.6)$, while for A_2 it is $\text{PROP}_{A_2}=(A_4, 0.5)$.² Both agents then apply the optimistic policy P_A to select the most optimistic proposal, which in this case is PROP_{A_1} (line 4 in Table 1). Hence, A_3 will perform the rescue. However, A_3 does not swim as well as A_1 had thought. After observing the real performance of A_3 , which is $\text{REAL_PERFORM}(A_3)=0.3$ (line 5), both agents update their models of A_3 (i.e., $M_{A_1}(A_3)$ and $M_{A_2}(A_3)$) using the value for $\text{REAL_PERFORM}(A_3)$ (line 6).

² We use an abbreviated form of PROPOSAL, which contains only the proposed agent and the proposed time performance.

In the second rescue, A_1 still proposes A_3 , and A_2 still proposes A_4 (line 8). However, this time, the optimistic policy selects A_4 (line 9), since its proposed performance (0.5) is now better than the proposed performance of A_3 (0.3), which is obtained from its updated model. After observing the real performance of A_4 , $M_{A_1}(A_4)$ and $M_{A_2}(A_4)$ are updated using the value for $\text{REAL_PERFORM}(A_4)$, which is 0.6 (line 11). Upon completion of this step, the models maintained by A_1 and A_2 are identical. Hence, in the third rescue, both agents propose A_4 .

3 Experiment 1: Impact of Resource Boundedness

Our experiment was run under the surf-rescue scenario described in Section 2.6, which had the following conditions.

- the rescue team consists of five lifesaver agents.
- one distressed person is rescued by the team.
- in each rescue, one lifesaver agent is allocated to rescue one distressed person.
- the evaluation criterion in the surf rescue domain is time.

We used this relatively simplistic collaboration scenario in order to obtain preliminary insights into the influence of the modeling limitations of agents on task performance. In the future, we will design more advanced experiments with more complex collaboration processes. Such experiments may include larger teams (of more than five lifesaver agents), agents that perform several activities (as opposed to one agent that performs one rescue), and several evaluation criteria (e.g., time and quality).

Next, we present our experimental parameters, followed by a description of the collaboration settings used in our simulations. We then provide a specification of a simulation run, and present our results.

3.1 Experimental Parameters

To assess the impact of resource limitations on task performance, we considered the following parameters: Memory Boundedness (MB), M-IR Relationship (MIRR), and Model Update (MU). These parameters pertain to an agent’s modeling ability, and are assigned a range of values in our simulations.

- *Memory Boundedness (MB)* represents the average memory capacity of the agents in a team that can be used to accommodate the models for the team members. For example, $MB=0\%$ corresponds to no memory; $MB=100\%$ corresponds to sufficient memory to store all models; and $MB=50\%$ means that on average 50% of the models are randomly discarded by the agents in the team. Since this parameter models the average conditions for the team (as opposed to the conditions for each agent), some agents may have sufficient memory to store all the models, while others may have no memory (in our second experiment, we model the memory capacity of individual agents outlined in Section 5). This parameter is used to model the effect of memory

capacity on task performance. It is set when the models maintained by the agents in the team are initialized.

The *MB* parameter is similar to the attentional resource limits considered by Walker [11], and the memory boundedness investigated by Rubinstein [12]. However, both Walker and Rubinstein also considered inferential limitations, while we consider agent-modeling limitations and group-related parameters.

- *M-IR Relationship (MIRR)* provides an upper bound for the degree to which agents’ real capabilities are underestimated by M_{A_i} for each agent A_i in a team (various relationships of models and internal resources are investigated in future research, such as overestimation). For example, $MIRR=0\%$ means that according to each agent A_i , all the agents in the team are estimated to have no capabilities; and $MIRR=50\%$ means that for each agent A_i , the agents modelled by M_{A_i} are randomly assigned capabilities between 0% and 50% of their real capabilities. This parameter constitutes a measure of the accuracy of the models being maintained by the agents in the team. It is set when the agents’ models are initialized.
- *Model Update (MU)* specifies the capacity of the agents in a team to update (or learn) their models of the team members. For example, $MU=0\%$ corresponds to no agents updating their models; $MU=100\%$ corresponds to all agents updating their models; and $MU=50\%$ means that half the agents (randomly selected) update their models. This parameter models the influence of agents’ learning capabilities on task performance. It is used in the updating procedure carried out in Step 8 of the interaction protocol in Figure 1.

These parameters pertain to the average modeling capabilities of a team. In our first experiment (Section 3), we used team-based parameters (as opposed to parameters pertaining to individual agent-based modeling capabilities). The team-based parameters can be used to represent estimations of individuals with average modeling capabilities. Team-based modeling capabilities are easier to estimate than modeling capabilities of each individual agent. However, these team-based parameters provide a basis for more refined agent-based parameters, as done in Study 2 (Section 5).

3.2 Collaboration Settings

We constructed three settings where we varied the values of the collaboration parameters *MB*, *MIRR*, and *MU*. These settings are denoted $Set(SimParam)$, where $SimParam \in \{MB, MIRR, MU\}$ varies from 0% to 100% by 1% increments, while the other parameters stay at a constant 100%.

We also constructed three benchmark settings, *RAND*, *OMNI* and *DEF*, for comparative purposes.

- 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 group. 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 optimal agent of the group is assigned to a rescue. This setting is consistent with the traditional assumption of teams in Multi-Agent Systems (MAS), where agents have accurate knowledge about the internal resources of team members prior to the collaboration (i.e., all models are accurate with respect to the real internal resources of agents), which results in an optimal assignment of agents to an activity. In the OMNI setting, agents do not update their models or communicate proposals, because all agents have the same accurate models.
- The DEF (or default) setting defines agents where all the parameters are set to 100%, i.e., all agents update their models ($MU=100\%$), have enough memory to maintain all models ($MB=100\%$), and have a maximum upper bound for model accuracy ($MIRR=100\%$). The DEF setting offers the best conditions for agents to reach high levels of performance. However, this does not mean that agents in the DEF setting will eventually have accurate and complete knowledge of collaborators, because the group can converge to a local maximum (Section 3.4).

3.3 The Simulation

The simulation consists of 100 *steps*, where each step increments the simulation parameter of a setting by 1% (e.g., MU is the parameter being incremented in $Set(MU)$). Each simulation step in turn consists of 1000 *runs* (we selected this number of runs because it yields stable and continuous patterns of behaviour).

A simulation run consists of a rescue task that is repeated until consensus is reached. IR and M are initialized at the beginning of each run. In this paper, we assume that team members have no prior knowledge of collaborators’ capabilities. That is, lifesavers have not observed any performance of team members at the beginning of a simulation run. Hence, IR is initialized with random values, and M is initialized as specified in the collaboration and benchmark settings (Section 3.2). IR remains constant throughout a run, while M is updated for each rescue in the run. Models are different to the internal resources of each agent at the beginning of each run. The models converge to the real internal resources of agents and become more accurate with each rescue. The process of convergence is influenced by an agent’s ability to learn and the availability of memory to store models.

A simulation run is based on the interaction protocol outlined in Section 2.5 and 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). Both experiments use a process for reaching convergence that defines the *end of a run*. A run is terminated when the same lifesaver is chosen in N consecutive turns (we have experimented with

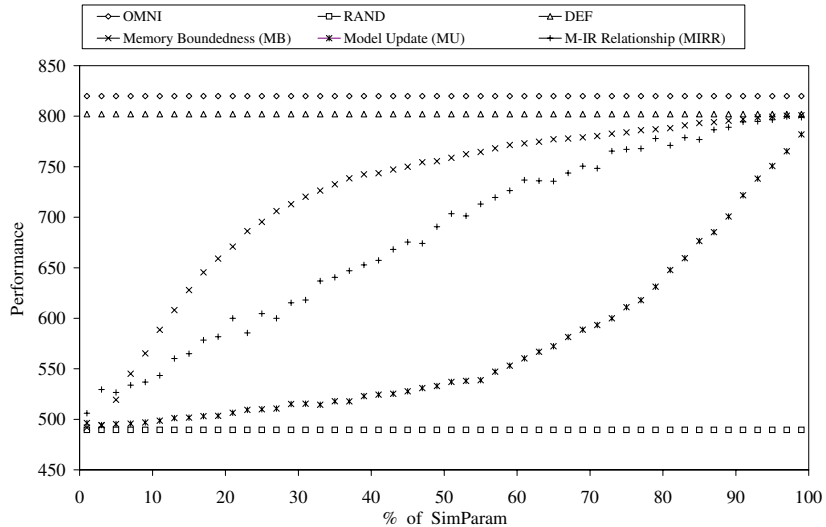


Figure 3. Average task performance for 1000 simulations plotted against the simulation parameters for several collaboration settings.

$N = 2$ and $N = 5$; the results presented in Section 3.4 are for $N = 2$, and the results presented in Section 5.4 are for $N = 5$).

Upon completion of a simulation run, we calculate task performance as follows. The *task performance* for a run is the performance of the agent on which the observers converged. This measure reflects the final outcome of the combination of the parameters of the simulation for the run in question.

The task performance of an agent is based on the values obtained from its *IR* (Section 2.3.1) for the evaluation criteria relevant to the task (Section 2.2). In the SR scenario, the only criterion is speed, i.e., the quicker the person in distress is rescued, the better the task is performed.

3.4 Results

Figures 3 to 5 plot task performance as a function of the different parameters in the collaboration settings. The x-axis in both figures shows the percentage of the simulation parameter, e.g., when % of SimParam=50, then 50% of the agents update their models ($MU=50\%$). The y-axis in Figures 3 to 5 shows the total level of performance for 1000 runs.

Overall, our experiments demonstrate that task performance is better in settings where agents maintain models of the agents in the group than in settings where agents do not have such models. Further, our results show that limitations on the ability to model agents have a large influence on task performance.

As expected, the results for the RAND and OMNI settings correspond to the worst and best performance respectively, which are used as a benchmark for comparison with other settings. The performance for the DEF setting is slightly

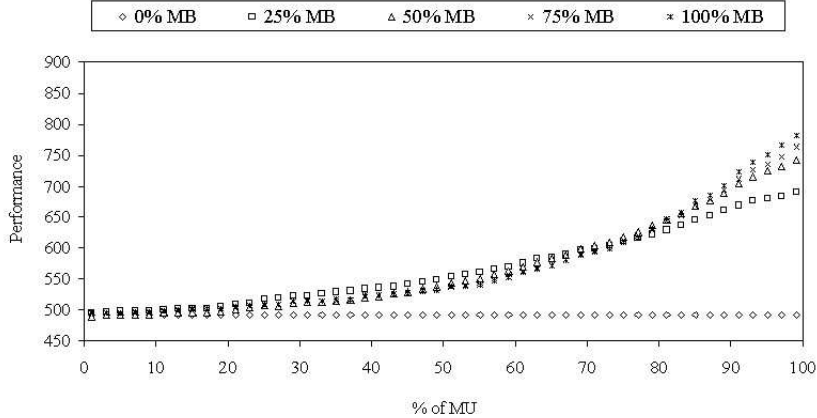


Figure 4. Impact on task performance of the interaction between the increasing MU parameter and five values of the MB parameter.

lower than that for the OMNI setting. This is due to the fact that agents in the DEF setting sometimes converge to a local maximum, which is reached when all the agents in the team estimate the performance of a particular agent below the performance of other agents. Hence, this agent, which is in fact better for the task at hand than a currently chosen agent, will never be proposed by any agent in the group, and will never perform the task.

As seen in Figure 3, the local maximum obtained in other settings is generally lower than the local maximum for the DEF setting. This is due to the lower values of the collaboration parameters for these settings. We found that task performance increases logarithmically as the available memory increases for the agents in the team (*MB* setting). Further, if these agents store only 50% of all the models on average, the performance is above 90% of the performance obtained for the OMNI setting. Thus, all agents *do not* need to store all the models in order to reach a high level of performance. This means, a designer of such a system needs to allocate less memory to each agent and can still expect a high performance outcome. In the *MU* setting, performance remains low until about 60% of the agents are able to update their models, at which point it increases polynomially. Finally, the influence of the *MIRR* setting on task performance is quasi-linear, with performance increasing when agents’ estimations of team members become more accurate (*MIRR* setting).

The results in Figure 3 show that *MU* has the most dramatic effect on performance, in the sense that a large percentage of the agents need to be able to update their models in order to reach high levels of performance. Figure 3 also shows that the influence of *MB* is important when proposing an agent for a task, i.e., agents perform well even without storing a large number of models. These results prompted us to investigate the influence of *MU* in combination with the *MB* parameter. This combination yields the following two evaluation sets: $Set(MU, MB)$, which varies the *MU* parameter from 0% to 100% (by 1%),

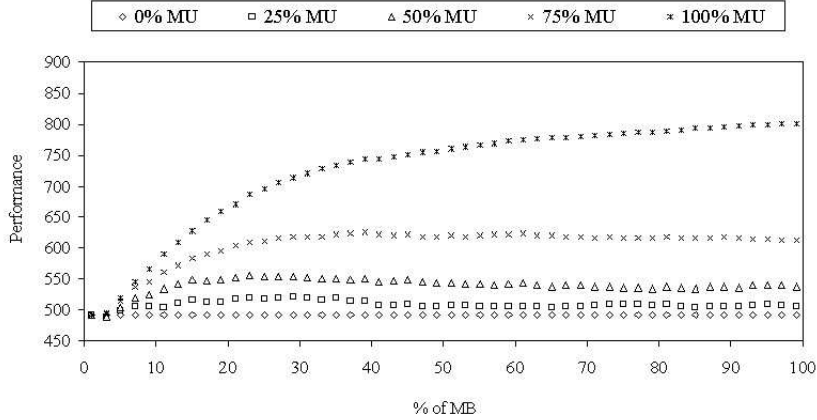


Figure 5. Impact on task performance of the interaction between the increasing MB parameter and five values of the MU parameter.

while the MB parameter receives five values, and $Set(MB, MU)$, which reverses these value assignments.

Figure 4 shows the impact of the varying MU parameter and the five values of the MB parameter on task performance, and Figure 5 shows the impact of the varying MB parameter and the five values of the MU parameter. According to Figure 4, varying the memory of agents between 25% and 100% has little impact on task performance until the percentage of agents that are able to update their models reaches 80%. The results in Figure 5 indicate that task performance for each value of MU increases proportionally to the value of MB until MB reaches about 30%; from then on MB seems to have a small effect on task performance.

4 Extensions to the ETAPP Framework: Modeling and Reasoning Resources

In Section 3.4, we showed that memory boundedness and model updating capability have a significant impact on task performance. These insights were obtained under simplifying assumptions whereby an agent’s performance is deterministic and invariant, and hence observers can learn an accurate model of an agent after a single observation (Section 2). In this section, we consider extensions of the ETAPP framework regarding modeling and reasoning resources of agents, in order to determine if these insights hold under more realistic conditions.

4.1 Extensions to Internal Resources IR_{A_i}

In the original framework, IR_{A_i} consisted of a set of numbers between 0 and 1 that described the level of performance of an activity. However, in realistic settings, agents exhibit variable performance (e.g., they could be having a bad

day). We represent such 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). Thus, the values of the evaluation parameters $\alpha_j(\text{activity})$ for $j = 1, \dots, n$ (where n is the number of Evaluation Criteria, Section 2.3.1) are drawn from a truncated normal distribution with mean μ_{jk} and standard deviation σ_{jk} , $N_T(\mu_{jk}, \sigma_{jk})$, for $j = 1, \dots, n$. As for the deterministic agents, the distribution is not directly observable, but the drawn values yield the observed performance of an agent during simulations.

4.2 Extensions to Models M_{A_i}

Originally, due to the deterministic performance of agents, an agent's performance could be modeled by means of a single number (Section 2.3.2). When the performance is variable, the observer agents need to learn the parameters of the distribution from which the performance is drawn (in our case, this is a normal distribution). Thus, $\widehat{\alpha}_j(\text{activity}) = N_T(\widehat{\mu}_{jk}, \widehat{\sigma}_{jk})$ for $j = 1, \dots, n$.

These modeling requirements give rise to an experimental parameter *Observation Capacity (OC)*, which is a refinement of the *MB* parameter introduced in Section 3.1. This parameter specifies how many observations of the performance of each agent 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). In the future, we investigate different weighing functions that can be applied to these observations (thus we keep a list of K most recent events for our current implementation as opposed to a list with only two numbers: current average of observed performance, and the number of observations made). For example, greater weight could be given to more recent events than to earlier events which in turn would influence the assessment of an agent's ability (which is similar to work by Hirsh and Davison [13]).

4.3 Extensions to Reasoning Apparatus RA_{A_i}

The variable performance of agents also demands the implementation of a new model-updating procedure. We propose a simple procedure whereby an agent re-calculates the mean and standard deviation of the observed performance of an agent every time it performs an activity. 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.

5 Experiment 2: Impact of Modeling Refinements

The second experiment is designed to evaluate the extensions pertaining to variable agent performance. Our simulation experiments assess the impact of

Internal Resources and Observation Capacity on task performance. The model-updating procedure described in Section 4.3 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 2.6, where the task is to rescue a person in distress.

5.1 Experimental Parameters

The requirements of the probabilistic models discussed in Section 4.2 give rise to the experimental parameter OC , which is a refinement of the MB parameter introduced in Section 3.1. This parameter specifies how many observations of the performance of an agent performing a task can be stored by an observer agent in its memory. When this limit is exceeded, the observer agent retains a window of the last K observations (forgetting the initial ones).

5.2 Collaboration Settings

The collaboration parameters corresponding to our extensions were varied as follows.

- **Internal Resources** – We defined 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. In the future, we propose to conduct experiments with high-performing, medium-performing and low-performing teams.
- **Observation capacity** – We varied the OC of the agents between 1 and 8 (in our experiments, the performance did not change when agents had a capacity of more than 8 observations, because more than 8 observations were not made by an agent). When $OC=i$, agents retain the last i observations made, and when $OC=1$, their observation capacity is as for the original ETAPP framework.

In addition, we used the three benchmark collaboration settings that we have defined for Experiment 1: RAND, OMNI and DEF.

5.3 The Simulation

We ran one simulation for each combination of the collaboration parameters ($IR \times OC \times \mathcal{P}_A = 5 \times 8 \times 2 = 80$), plus one simulation for each of the benchmark settings, RAND and OMNI. Each simulation consisted of ten trials, each divided into 1000

runs (we selected this number of trials and runs because it yields stable and continuous behaviour patterns). Each run consisted of a rescue task that was repeated until convergence was 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 $Models$ (M) are initialized with random values.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.

5.4 Results

The results of our experiments are shown in Figure 6, which depicts the average task performance as a function of OC for our seven types of teams – RAND, OMNI, Invariant, Stable, Medium, Unstable and Mixed.

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.

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 lower than that for the OMNI setting. This is due to the fact that the Invariant team sometimes converges to a local maximum, which is reached when the agents in the team repeatedly 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 experiment.

As seen in Figure 6, 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 build reliable models when team members exhibit unstable performance.

Task performance improves for Medium and Mixed teams when agents are able to remember observations of the performance of team members. This im-

³ 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 5.4).

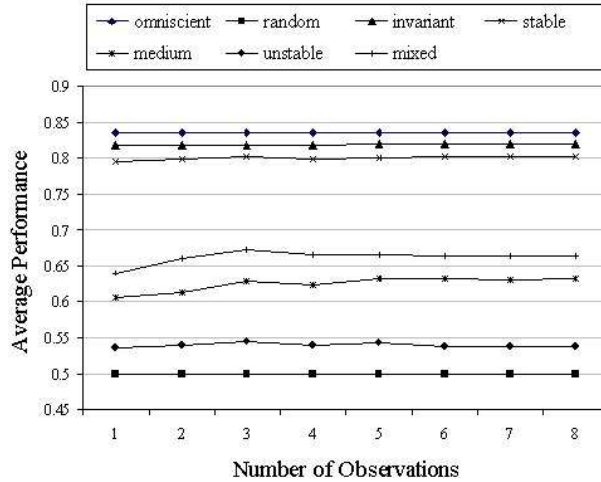


Figure 6. Average task performance plotted against observation capacity for several types of teams.

provement is achieved with only 3 observations. Other types of teams or additional observations do not seem to have an impact on performance.

Finally, the performance of Unstable teams is not affected by the agents’ observation capacity, as the agents in these teams exhibit too much performance variation for the observer agents to reach reliable conclusions.

6 Related Research

Collaboration between complex computational systems is a challenging problem which has been addressed by many researchers, e.g., [14, 15, 16, 17, 18, 19]. Several research projects have demonstrated that maintaining models of features of collaborators can benefit different aspects of task performance. For example, such features have been modelled in order to determine behaviour based on collaborators’ utility-functions [20], achieve flexible team behaviour in military domains [21], establish domain-dependent conventions among agents [22], or predict behaviour of “soccer-agents” under commonly-known domain dynamics [23] or commonly-known action policies [24]. These systems are discussed in more detail in the remainder of this section.

Suryadi and Gmytrasiewicz [25] and Gmytrasiewicz and Durfee [20] (groundwork in [26]) investigated a decision-theoretic approach where each agent makes decisions to maximize its own individual payoff by estimating the payoff of collaborators. According to this approach, an agent is given payoff matrices of collaborators. When agent A_i estimates the payoff matrix of agent A_j ($i \neq j$), agent A_j may in turn model the payoff of agent A_i , which would influence agent A_i ’s payoff matrix, and so on. This recursive nesting of models may lead to an infinite recursion of models of collaborators, thereby outstripping the limited reasoning capabilities of an agent. Gmytrasiewicz and his collaborators prevent this

problem by restricting the information provided about other agents. It is worth noting that a payoff matrix enables an agent to reason about the behaviour of another agent in order to increase its own utility. However, agents described in [26, 25, 20] make individual decisions and do not communicate with each other, our agents communicate proposals in order to make a joint decision.

Stone *et al.* [23] assume an ideal world model and suggest a model called Ideal Model Based Behavior Outcome Prediction (IMBBOP) used by robot soccer agents to make decisions based on values that represent the highest possible performance of the agents in a team (e.g., highest possible speed of an agent). Although information about upper-bound agent performance may be readily available in synthetic domains, such as the robot soccer scenarios, it may not be obtainable in realistic domains, and may not be generally assumed in open system scenarios.

Kok and Vlassis [24] developed an approach called Mutual Modeling of Teammate Behavior which assumes that the action policy of each agent is known to every agent. Hence, one agent A_i can apply the policy of another agent A_j ($i \neq j$) in order to predict which action agent A_j will perform. The state s_{A_i} of agent A_i together with the policy of this agent $\phi_i(s_i)$ determines the action that will be chosen next. The success of this technique relies on the assumption that agent A_i (that predicts the action of agent A_j) has knowledge of the state s_{A_j} of agent A_j which ought to resemble the actual state of A_j . However, in open systems such as that implemented in the Surf Rescue scenario, the proposal-selection procedure of an agent is usually not known to other agents, and hence can not be used by collaborators in a team.

Garland and Alterman [27] designed agents that store successful collaboration events in the form of execution traces. An agent can then re-use these execution traces in situations that are similar to the situations where the traces were made. An advantage of these traces is that they are dynamically generated, and do not require a significant design effort. However, they do not contain generic information about agents, and they do not explicitly represent resources of collaborators. Hence, they are not readily re-usable in different domains or different situations.

The research area of User Modeling (UM) deals with the problem of modeling collaborators or agents, and cannot assume direct access to or an accurate model of collaborators' resources. In UM, an agent can form a model of a user in order to assist her/him [28]. For example, Vassileva *et al.* developed I-Help [29], which is a large scale multi-agent system that provides students with distributed help resources. Personal agents represent students' personal preferences. Matchmaker agents collect this information from personal agents, and match students that require help in a certain topic with students that are able to provide help. The incorporation of our model-update mechanism into the models maintained by the matchmaker agents would increase the accuracy of these models, and hence improve their usefulness for help-seeking students.

7 Conclusion and Future Work

We have offered the ETAPP framework for agent collaboration which explicitly models the following aspects of a collaboration: the environment, the requirements of the task, the capabilities of collaborators, the decision-making process, and the interaction protocol.

Our framework provided the basis for two empirical studies where we investigated the effect of different limitations of modeling abilities on task performance. Specifically, our first experiment considered the following modeling based parameters: memory boundedness, model accuracy, and the ability to learn from observations. Our second experiment investigated refinements pertaining to the modelling and reasoning resources of agents.

The main conclusion of our first experiment is that limiting the modeling abilities of agents has a large negative influence on task performance. Specifically, limitations on the ability to update models have the most dramatic effect on performance. Our results show that a well performing team requires at least 50% to 60% of agents able to update their models. Regarding memory resources, agents perform well only when they are able to store models of at least 20% to 30% of all team members. Finally, our results indicated that the relationship between the underestimation of agents' abilities and agent performance is quasi-linear, with performance increasing when agents' estimations of team members become more accurate.

In our second experiment, we have extended our ETAPP collaboration framework to model team members that exhibit variable performance. This requires a probabilistic representation of agent performance, the specification of the number of observations retained by observer agents, and a procedure for building agent models from these observations. To evaluate our extensions, we varied the performance stability of teams of agents, and the number of observations retained by observer agents. Our results show that performance variability has a large impact on task performance, and that a small number of observations of agent behaviour is sufficient to improve task performance.

We propose the following extensions to our research.

- Investigate models of team performance as an extension of our current implementation, where we consider models of only one agent.
- Decentralize the evaluation of performance. This means that each agent uses a different function to evaluate observed performance, as opposed to our current approach where one evaluation function is used by all agents.
- Compare central with distributed decision making procedures, specifically decisions made by a leader, or decisions derived from voting and auctioning.
- Build additional communication protocols for transmitting proposals between agents (Section 2.5).

8 Acknowledgments

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

authors would like to thank Fredrik Hacklin, Lara Kornienko, Mike Tyson, and Andrea Bunt for their advice on writing this paper. We would also like to thank the anonymous reviewers for their comments.

References

- [1] Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers* **29** (1980) 1104–1113
- [2] Hewitt, C.: The challenge of open systems. *Byte* **4** (1985)
- [3] Bond, A.H., Gasser, L.: *Distributed Artificial Intelligence*. Morgan Kaufmann Publishers Inc. (1988)
- [4] Davidsson, P.: Categories of artificial societies. In: *ESAW '01: Proceedings of the Second International Workshop on Engineering Societies in the Agents World II*, London, UK, Springer-Verlag (2001) 1–9
- [5] Scott, W.R.: *Organizations: rational, natural, and open systems*. Prentice-Hall, Upper Saddle River, NJ, USA (2002)
- [6] Guttman, C., Zukerman, I.: Voting policies that cope with unreliable agents. In Dignum, F., Dignum, V., Koenig, S., Kraus, S., Singh, M.P., Wooldridge, M., eds.: *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*. LNAI, New York, NY, ACM Press (2005)
- [7] Chu-Carroll, J., Carberry, S.: Conflict Resolution in Collaborative Planning Dialogues. *International Journal of Human Computer Studies* **53(6)** (2000) 969–1015
- [8] Li, C., Giampapa, J.A., Sycara, K.: A Review of Research Literature on Bilateral Negotiations. Technical Report CMU-RI-TR-03-41, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania (2003)
- [9] Rahwan, I., Ramchurn, S.D., Jennings, N.R., McBurney, P., Parsons, S., Sonenberg, L.: Argumentation-based negotiation. *The Knowledge Engineering Review* **18** (2003) 343–375
- [10] Zukerman, I., Guttman, C.: Modeling Agents that Exhibit Variable Performance in a Collaborative Setting. In Ardissono, L., Brna, P., Mitrovic, A., eds.: *User Modeling 2005*. LNAI 3538, Edinburgh, United Kingdom, Springer (2005)
- [11] Walker, M.A.: The Effect of Resource Limits and Task Complexity on Collaborative Planning in Dialogue. *Artificial Intelligence* **1-2** (1996) 181–243
- [12] Rubinstein, A.: *Modeling Bounded Rationality*. Zeuthen lecture book series. MIT Press, Cambridge, Massachusetts (1998) .
- [13] Davison, B., Hirsh, H.: 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)
- [14] Cohen, P.R., Levesque, H.J.: *Teamwork*. Technical Report 504, Stanford University, Menlo Park, California (1991)

- [15] Kinny, D., Ljungberg, M., Rao, A.S., Sonenberg, E., Tidhar, G., Werner, E.: Planned Team Activity. In C. Castelfranchi and E. Werner, ed.: *Artificial Social Systems — Selected Papers from the Fourth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-92* (LNAI Volume 830), Springer-Verlag: Heidelberg, Germany (1992) 226–256
- [16] Jennings, N.R.: Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. *Artificial Intelligence* **75** (1995) 195–240
- [17] Grosz, B.J., Kraus, S.: Collaborative Plans for Complex Group Action. *Artificial Intelligence* **86** (1996) 269–357
- [18] Kaminka, G.A., Tambe, M.: Robust agent teams via socially-attentive monitoring. *Journal of Artificial Intelligence Research* **12** (2000) 105–147
- [19] Panzarasa, P., Jennings, N., Norman, T.: Formalizing Collaborative Decision-Making and Practical Reasoning in Multi-Agent Systems. *Journal of Logic and Computation* **12** (2002) 55–117
- [20] Gmytrasiewicz, P.J., Durfee, E.H.: Rational Communication in Multi-Agent Environments. *Autonomous Agents and Multi-Agent Systems* **4** (2001) 233–272
- [21] Tambe, M.: Towards Flexible Teamwork. *Journal of Artificial Intelligence Research* **7** (1997) 83–124
- [22] Alterman, R., Garland, A.: Convention in Joint Activity. *Cognitive Science* **25** (2001) 611–657
- [23] Stone, P., Riley, P., Veloso, M.M.: Defining and Using Ideal Teammate and Opponent Agent Models. In: *Proceedings of the Twelfth Annual Conference on Innovative Applications of Artificial Intelligence*. (2000) 1040–1045
- [24] Kok, J.R., Vlassis, N.: Mutual Modeling of Teammate Behavior. Technical Report UVA-02-04, Computer Science Institute, University of Amsterdam, The Netherlands (2001)
- [25] Suryadi, D., Gmytrasiewicz, P.J.: Learning Models of Other Agents Using Influence Diagrams. In: *Proceedings of the Seventh International Conference on User Modeling*. 223–232, Banff, Canada (1999)
- [26] Gmytrasiewicz, P.J.: A Decision-Theoretic Model of Coordination and Communication in Autonomous Systems (Reasoning Systems). Ph.D. dissertation, University of Michigan (1992)
- [27] Garland, A., Alterman, R.: Autonomous Agents that Learn to Better Coordinate. *Autonomous Agents and Multi-Agent Systems* **8** (2004) 267–301
- [28] Kobsa, A., Wahlster, W.: *User models in dialog systems*. Springer-Verlag New York, Inc. (1989)
- [29] Vassileva, J., McCalla, G., Greer, J.: Multi-agent Multi-user Modeling in I-Help. *User Modeling and User-Adapted Interaction* **13** (2003) 179–210