

# Making Allocations Collectively: Iterative Group Decision Making under Uncertainty

Christian Guttman\*

Department of General Practice  
Faculty of Medicine, Nursing and Health Sciences  
Monash University, Melbourne, Australia  
christian.guttman@gmail.com

**Abstract.** A major challenge in the field of Multi-Agent Systems (MAS) is to enable autonomous agents to allocate tasks and resources efficiently. This paper studies an extended approach to a problem we refer to as the Collective Iterative Allocation (CIA) problem. This problem involves a group of agents that progressively refine allocations of teams to tasks. This paper considers the case where the performance of a team is variable and non-deterministic. This requires that each agent is able to maintain and update its probabilistic models using observations of each team's performance. A key result is that each agent needs the capacity to store only two or three observations of a team's performance to find near optimal allocations, and a further increase of this capacity will reduce the number of reallocations significantly.

## 1 Introduction

Efficient approaches to distributed allocation problems are required in a wide range of applications, such as network routing, crisis management, logistics, computational grids, and collaborative student support environments [1]. We consider a problem we refer to as the Collective Iterative Allocation (CIA) problem [2–4]. This problem involves a group of agents that endeavours to find an optimal allocation of a team to a task, and subsequent allocations are then refined as the true performance of a team becomes known.<sup>1</sup> Note that this paper describes allocations using the terms *tasks* and *teams*, but this terminology is specific to the domain of application. The Multi-Agent System (MAS) paradigm

---

\* The author would like to thank Michael Georgeff, Iyad Rahwan and Ingrid Zukerman for their assistance related to this research. The author is also grateful for the insightful comments of the reviewers. Part of this work was done at the School of Computer Science and Software Engineering at the Faculty of Information Technology and at the Monash Institute of Health Services Research at the Faculty of Medicine, Nursing and Health Sciences, Monash University, Melbourne, Australia.

<sup>1</sup> In this paper, we assume that agents are collaborative and task-rational. That is, each agent proposes a team with the highest performance according to the agent's models. In [3], we investigated issues related to competitive agents that follow a strategy when making group decisions.

is useful to study the efficiency of algorithms to distributed coordination problems [5], and particularly to the CIA problem [2–4].

Many research frameworks on allocation problems make a simplistic assumption: the performance of a team is deterministic and invariant [1]. This paper extends our previous framework [3] that copes with agents that exhibit variable and non-deterministic performance in the CIA problem. In particular, in [3], we consider how each agent maintains a probabilistic model which represents a team’s performance by a mean (average performance of team) and standard deviation (stability of performance). Each agent updates its model when new information of a team is available. The focus of this paper is on (a) identifying conditions that influence finding near-optimal solutions and the time it takes to find such solutions, and (b) developing methods that refine the process of selection under conditions of uncertainty. This paper extends our work in [2–4] as follows.

- *Framework Extensions:* We offer extensions of the modelling and reasoning abilities of agents. For example, each agent uses a shared observation memory that determines how many observations of all teams can be remembered by an agent for the refinement of models (in previous research [3], each agent stores an observation of each agent in a separate memory).
- *Empirical Study:* We offer a deeper empirical understanding of the efficiency of our algorithm, as we explore parameters additional to those used in [3]. For example, we vary the termination criteria (which indicates the stability of converging to a solution), and also the level of ignorance that each agent exhibits prior to running the algorithm.
- *Evaluation:* We analyse the efficiency of the algorithm using measures of solution quality (average task performance of a team) and in addition to previous research [3], we also evaluate the algorithm’s computational requirement (the average number of iterations before a solution is found).

Section 2 offers a formal representation of our extended approach. Section 3 investigates the efficiency of this algorithm under the condition that the performance of teams is variable. Related research is discussed in Section 4. Section 5 summarises the insights and contributions offered by this research.

## 2 Framework Coping with Uncertainty

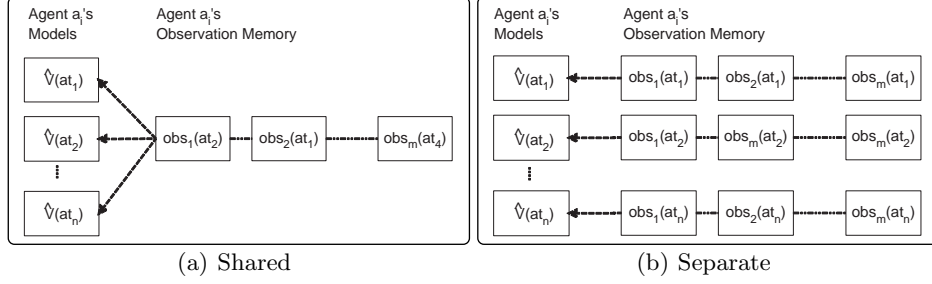
Section 2.1 defines the main components of the CIA problem and Section 2.2 defines and illustrates our algorithm.

### 2.1 Defining the Main Components

The CIA problem is represented by the following tuple.

$$CIA = \langle T, AT, A = \{a_1(M_{a_1}, RP_{a_1}), \dots, a_q(M_{a_q}, RP_{a_q})\}, P \rangle$$

T is a set of Tasks, each can be assigned to an Agent Team ( $\in AT$ ). Each agent  $a_i$  in A maintains Models M and uses Reasoning Processes RP to make assignments collectively using a group decision Policy P. These elements are defined in the following paragraphs. Note that the models  $M_{a_i}$  and reasoning processes  $RP_{a_i}$  maintained by each agent  $a_i$  are substantially refined and extended compared to those offered in [3].



**Fig. 1.** Observation Memory.

**Definition 1.**  $T = \{t_1, \dots, t_s\}$  is a set of Tasks with  $s = |T|$ .

**Definition 2.**  $AT = \{at_1, \dots, at_p\}$  is a set of Agent Teams with  $p = |AT|$ .

This paper assumes that a team’s performance varies each time it performs a task. In particular, we assume that this varying performance is based on a normal distribution  $N_{at_j, t_k}$  with a mean  $\mu_{at_j, t_k}$  (representing the average performance of a team), and a standard deviation  $\sigma_{at_j, t_k}$  (representing the stability of the performance), so that  $N_{at_j, t_k}(\mu_{at_j, t_k}, \sigma_{at_j, t_k})$  for a given team  $at_j$  and task  $t_k$ . To be consistent with the notation used in previous research, we also refer to this distribution as  $V(at_j, t_k)$  [4]. A set of distributions (that describe the true performance for several tasks) is called the *capability* of a team:  $C(at_i)$ .

We now define a group of agents that actively participates in assigning a team  $at_i$  to a task  $t_k$ .

**Definition 3.**  $A = \{a_1, \dots, a_q\}$  is a set of Agents with  $q = |A|$ .

The capability of a team  $C(at_i)$  is *not* directly observable (as is the case for a team’s deterministic performance [4]). Hence, each agent has a model consisting of a set of estimated distribution parameters (Definition 4). Each agent also executes reasoning processes that use these models (Definition 5).

**Definition 4.**  $M_{a_i}$  are the **Models** maintained by agent  $a_i$ :  $M_{a_i} = \{M_{a_i}(at_1), \dots, M_{a_i}(at_p)\}$  with  $p = |AT|$ . A specific model of a team  $at_j$  is defined by a set of estimations:  $M_{a_i}(at_j) = \{\hat{V}_{a_i}(at_j, t_1), \dots, \hat{V}_{a_i}(at_j, t_s)\}$ , where  $\hat{V}_{a_i}(at_j) = \hat{N}_{a_i, at_j, t_k}(\hat{\mu}_{at_j, t_k}, \hat{\sigma}_{at_j, t_k})$  for  $j = 1, \dots, n$ .

For example, for a given task, assume that lifesaver  $at_2$ ’s performance is represented by a normal distribution  $N_{at_2}$  with a mean  $\mu_{at_2} = 0.4$  (representing the average rescue performance of  $at_2$ ), and a standard deviation  $\sigma_{at_2} = 0.1$  (representing its stability), such that  $N_{at_2}(\mu_{at_2} = 0.4, \sigma_{at_2} = 0.1)$ . Also, assume that  $a_1$ ’s model estimates the mean and stability of  $at_2$ ’s performance:  $\hat{N}_{a_1, at_2}(\hat{\mu}_{at_2} = 0.5, \hat{\sigma}_{at_2} = 0.2)$ . In this example,  $a_1$  overestimates the mean of  $at_2$ ’s true performance as well as its variability when it performs the task.

Each agent in  $A$  executes reasoning processes that use the information stored in its models.

**Definition 5.**  $RP_{a_i}$  denote agent  $a_i$ ’s **Reasoning Processes**.

- For agent  $a_i$ ,  $INITIALISE_{a_i}(M_{a_i}, t_k)$  returns a set of initial models.

- For agent  $a_i$ ,  $PROPOSE_{a_i}(M_{a_i}, t_k)$  returns a proposal specifying  $a_i$ 's preference of a team that should perform a task:  $proposal_{a_i} = \langle at_j, \hat{\mu}_{at_j} \rangle$ . An agent is assumed to be task-rational and therefore proposes a team with the highest estimated value in its models (related studies investigate agents that make strategic proposals, e.g., [3]).
- For agent  $a_i$ ,  $UPDATE_{a_i}(M_{a_i}, observation(at_j))$  returns a set of updated models based on information of team  $at_j$ 's performance. Let  $observation(at_j)$  be a function that returns a measure of the performance of the team  $at_j$  for a given task. This value is drawn from a team's capability distribution ( $observation(at_j) \sim V(at_j)$ ).

The update process requires a memory to store these observations. We refer to this as *shared observation memory* (Figure 1(a)) and it works as follows. Each agent retains a window of the last  $k$  observations of the performance of all teams (each observation is tagged with the team that performed the task). The update process recalculates the estimated mean and standard deviation of the estimated distribution of a team each time an observation is made. When an agent updates its models of a team, the observations of this team are extracted from the shared observation memory using the corresponding tags. The results obtained by this process are moderated by the number of observations stored by each agent. That is, if an agent  $a_i$  has stored  $k$  observations of team  $at_k$ 's performance, then the mean and standard deviation are calculated from  $k$  observations, such that  $\hat{N}_{a_i, at_j}(\hat{\mu}_{at_j, k}, \hat{\sigma}_{at_j, k})$  for  $j = 1, \dots, n$ .<sup>2</sup>

**Definition 6.** A group decision Policy  $P$  is a function that selects a team based on the proposals submitted by agents  $A$ , such that

$$at_{selected} := P(PROPOSALS_A), \text{ where}$$

- agent  $at_{selected}$  is one of the teams proposed by agents  $A$ , and
- $PROPOSALS_A$  is a set of proposals submitted by these agents.

## 2.2 Algorithm and Example

All agents in  $A$  follow the algorithm shown in Figure 2. This section illustrates how this algorithm interacts with the variable performance of lifesavers (this example is based on [2, 3]). This example is illustrated using a surf rescue domain which consists of a group of lifesavers  $AT = A = \{a_1, a_2, a_3\}$  which assign the task of rescuing a distressed person to each other. The values describing the capability distribution  $C$  of  $a_1$ ,  $a_2$  and  $a_3$  of this task are

- $C(a_1) = \{V(a_1, rescue) = N_{a_1}(\mu_{a_1, rescue} = 0.5, \sigma_{a_1, rescue} = 0.4)\}$ .
- $C(a_2) = \{V(a_2, rescue) = N_{a_2}(\mu_{a_2, rescue} = 0.8, \sigma_{a_2, rescue} = 0.3)\}$ .
- $C(a_3) = \{V(a_3, rescue) = N_{a_3}(\mu_{a_3, rescue} = 0.3, \sigma_{a_3, rescue} = 0.2)\}$ .

That is, lifesaver  $a_1$  has a medium performance and is unstable, lifesaver  $a_2$  has a high performance and is more stable, and lifesaver  $a_3$  has a low performance

<sup>2</sup> Note that sharing memory to store observations (as opposed to separately storing the observations of each team as done in [3] and explained in Figure 1(b)) can significantly reduce the number of unused storage entries. As such, shared observation memory will be particularly efficient if we have a large number of teams and only few of them will perform a task.

---

**TAP ASSIGNMENT ALGORITHM.**

*INPUT:* Task  $t_k \in T$ , Agent Teams  $AT$ , Agents  $A$ , Policy  $P$

*OUTPUT:* Assignment of  $at_j \in AT$  to  $t_k \in T$

---

1. ANNOUNCE task  $t_k \in T$
  2. INITIALISE $_{a_i}(M_{a_i}, t_k)$  ( $\forall a_i \in A$ )
  3. Repeat
    - (a)  $PROPOSALS_A = \bigcup_{a_i \in A} PROPOSE_{a_i}(M_{a_i}, t_k)$
    - (b)  $at_{selected} := P(PROPOSALS_A)$
    - (c) UPDATE $_{a_i}(M_{a_i}, V(at_{selected}, t_k))$  ( $\forall a_i \in A$ )
  4. Until a termination criterion is satisfied
- 

**Fig. 2.** A task is repeatedly assigned to different teams until a criterion is satisfied (e.g., a team is believed to perform the task best).

and is the most stable. These distributions are learnt by the agents to find an optimal lifesaver.

For clarity of exposition, we assume the following settings in this example.

- Only agents  $a_1$  and  $a_2$  can propose lifesavers for a rescue and each agent learns from their performance. These two agents (which are both observers and lifesavers) maintain models of lifesavers  $a_1$ ,  $a_2$  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 these lifesavers.
- In this example,  $a_1$  stores up to eight observations of the performance of the lifesavers in its shared observation memory, as does  $a_2$ .
- The majority policy is applied for selecting a lifesaver for a rescue. This policy selects the lifesaver backed by most agents (in the event of a tie, the top agent in an ordered list of lifesavers is selected – the order is based on the index of the lifesaver).

Table 1 illustrates the assignment of lifesavers to rescues under the majority policy (values obtained after each rescue are boldfaced).

- Column 1 indicates the allocation round.
- Column 2 shows the observer agents.
- Column 3 shows the lifesaver proposed by each observer agent.
- Column 4 shows the lifesaver selected by the majority selection policy.
- Columns 5–12 contain values of the observed performance of the lifesavers.
- Columns 13–15 show the mean of the updated models.

The first two rows in Table 1 (before the first round  $r_1$ ) contain values representing initial conditions: columns 5-12 show that no observations have been stored yet, and columns 13-15 contain the initial values of the models maintained by  $a_1$  and  $a_2$  for the rescue performance of  $a_1$ ,  $a_2$  and  $a_3$ . These initial values, which are *not* consistent with the true performance of the agents in question, are also recorded as the first “observed” performance of  $a_1$ ,  $a_2$  and  $a_3$ . This is to model a behaviour whereby an agent’s initial “opinion” of lifesavers can be influenced, but not immediately replaced, by observations of the lifesavers’ performance.

COLUMNS														
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Observer agent	Proposed lifesaver	Selected lifesaver	Shared Observation Capacity of Performance $a_1, a_2, a_3$								Models			
			1	2	3	4	5	6	7	8	$M(a_1)$	$M(a_2)$	$M(a_3)$	
$a_1$												0.3	0.4	0.5
$a_2$												0.6	0.5	0.7
$r_1$	$a_1$ $a_2$	$a_3$ $a_3$	$P_{maj} = a_3$	0.5 0.7	0.4 0.4	0.4 0.4						0.3 0.6	0.4 0.5	0.45 0.55
$r_2$	$a_1$ $a_2$	$a_3$ $a_1$	$P_{maj} = a_1$	0.5 0.7	0.4 0.4	0.3 0.6	0.3 0.3	0.3 0.3	0.2 0.2			0.3 0.45	0.4 0.5	0.45 0.55
$r_3$	$a_1$ $a_2$	$a_3$ $a_3$	$P_{maj} = a_3$	0.5 0.7	0.4 0.4	0.3 0.6	0.3 0.3	0.2 0.2	0.4 0.8	0.8 0.8		0.3 0.45	0.4 0.5	0.37 0.43
$r_4$	$a_1$ $a_2$	$a_2$ $a_2$	$P_{maj} = a_2$	0.5 0.7	0.4 0.4	0.3 0.6	0.3 0.3	0.2 0.2	0.4 0.5	0.8 0.8	0.7 0.7	0.3 0.45	0.6 0.65	0.37 0.43
$r_5$	$a_1$ $a_2$	$a_2$ $a_2$	$P_{maj} = a_2$	0.5 0.7	0.4 0.4	0.3 0.6	0.3 0.3	0.2 0.2	0.4 0.5	0.8 0.8	0.7 0.7	0.3 0.45	0.63 0.67	0.37 0.43
$r_6$	$a_1$ $a_2$	$a_2$ $a_2$	$P_{maj} = a_2$	0.9 0.9	0.4 0.4	0.3 0.6	0.3 0.3	0.2 0.2	0.4 0.5	0.8 0.8	0.7 0.7	0.3 0.45	0.7 0.68	0.37 0.43

**Table 1.** Sample run of the assignment algorithm (explained in Section 2.2)

According to the models maintained by  $a_1$  and  $a_2$ , the lifesaver  $a_3$  has the best rescue performance. Hence,  $a_3$  is selected by both  $a_1$  and  $a_2$  when a rescue is announced in round  $r_1$ . However, the true performance of  $a_3$  (0.4 at round  $r_1$ , column 6) is lower than that anticipated by the observer agents. Both agents observe this performance, and update their models accordingly (column 15).

Now, when a new rescue must be performed (in round  $r_2$ ), agent  $a_1$  proposes  $a_3$ , as  $a_3$  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 selected. 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 in round  $r_2$ , column 5), which significantly lowers  $M_{a_2}(a_1)$  to 0.45 (column 8). As a result, lifesaver  $a_3$  is once more the top choice of both observer agents for the next rescue (in rescue  $r_3$ ). But  $a_3$  performs quite low (0.2 at round  $r_3$ , column 7), thereby further lowering its expected performance according to the models maintained by the observers (column 10).

At this stage, the low performance of both  $a_1$  and  $a_3$  yields models with low mean values for these lifesavers. Hence, for the next rescue (in round  $r_4$ ),  $a_2$  is proposed by both observer agents. This is a high-performing lifesaver that has initially been underestimated by both observers. As it performs the rescue well (0.8 in round  $r_4$ , column 6), it raises the estimated value in the models maintained by both observers (column 9). Agent  $a_2$  is now clearly preferred by both observers and selected for the rescue in round  $r_5$ . Again, lifesaver  $a_2$  offers a good performance (0.7 in round  $r_5$ , column 6).

At this point, the models maintained by the observer agents are closer to the capabilities of the lifesavers than the initial models. Since both observer agents have a shared observation memory of eight observations, the next time a rescue is performed, the initial value representing the performance of lifesaver  $a_3$  will be dropped, which will further increase the accuracy of the models. In round  $r_6$ ,  $a_2$  is selected again and performs the rescue with 0.9. Both agents use this value to replace the initial observation of  $a_3$  (0.5 and 0.7, respectively). The algorithm is

terminated if we have an indication that one lifesaver is repeatedly selected over other lifesavers. In this example, the algorithm provides as solution  $a_2$ , because it has been selected three times in this run (more often than other lifesavers, which is an indication of a lifesaver’s selection being reliable).

### 3 Experiment: Impact of Variable Performance

This experiment evaluates the framework extensions under the condition that the performance of lifesavers is variable and non-deterministic. Our simulation is illustrated by the Surf Rescue (SR) domain introduced in Section 2.2.

#### 3.1 Experimental Parameters

We evaluated five experimental parameters which are varied as follows.

- **Capability Deviation (CD)** – We define lifesaver groups with different degrees of stability: *Invariant*, *Stable*, *Medium*, *Unstable* and *Mixed*. That is, the deviation  $\sigma$  of each lifesaver’s performance distribution is drawn from one of the following distributions.
  - *Invariant* lifesavers exhibit the same performance in all rescues.
  - *Stable* lifesavers: low performance variability,  $N(\mu = 0.25, \sigma = 0.1)$ .
  - *Medium* lifesavers: medium performance variability,  $N(\mu = 0.5, \sigma = 0.1)$ .
  - *Unstable* lifesavers: highly unstable performance,  $N(\mu = 0.75, \sigma = 0.1)$ .
  - *Mixed* lifesavers represent a mixture of stable, medium and unstable agents,  $N(\mu = 0.5, \sigma = 0.25)$ .
- **Observation Capacity (OC)** – The OC of each agent is varied between 1 and 30 concurrently.<sup>3</sup> When  $OC=i$ , each agent retains the last  $i$  observations made, and when  $OC=1$ , their observation capacity is as for previous studies that assume deterministic performance [3, 4]. This parameter specifies how many observations of the performance of lifesavers 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).
- **Stability Indicator (SI)** – The algorithm is terminated after a lifesaver is selected  $SI=25, 50, 75$  and  $100$  for a rescue. A greater SI value indicates that the solution is more reliable, because one lifesaver is selected more often than others. Note that the algorithm terminates if a lifesaver is selected SI times for a rescue during the entire simulation run (but not necessarily in SI consecutive rounds).
- **Policy (P)** defines two types of group decision policies: *maximum* and *majority*. We have chosen these two group decision policies for the experimental studies because under given theoretical conditions, they are guaranteed to find optimal solutions [4].
  - The *maximum* policy  $P_{max}$  selects a lifesaver with the highest proposed performance of all lifesavers proposed.

---

<sup>3</sup> The results did not change significantly when agents had a capacity of storing more than 30 observations. Hence, values greater than 30 are not assigned to the OC parameter. Note also, that another way of setting the value for observation capacity for each agent is to draw this value from a normal distribution to simulate a more heterogenous distribution of observation capacity.

- The *majority* policy  $P_{maj}$  selects a lifesaver proposed by a majority of agents.
- **Group Size (GS)** defines the number of agents (lifesavers) in a surf rescue domain: 5, 10, 20, 40 and 50. These values are expected to show a representative trend of our results for a range of populations of agents.

We run one simulation for each combination of the experimental parameters ( $CD \times OC \times SI \times GS \times P = 5 \times 30 \times 4 \times 5 \times 2 = 6000$ ), plus one simulation for each of two benchmark settings, EXHAUSTIVE and RANDOM. In short, both benchmark settings do not execute the TAP algorithm. Instead, an exhaustive setting assigns an optimal lifesaver, but requires that each lifesaver is assigned several times to better know the performance of each lifesaver (in this experiment, we assign each lifesaver GS multiplied by SI, as this assigns an optimal lifesaver with high certainty). A random setting assigns a random lifesaver and does not require any testing or remembering of the performance of lifesavers.

### 3.2 Efficiency Metrics

**Solution Quality (SQ).** The measure of SQ for a run is the mean of the capability distribution for the lifesaver on which the algorithm eventually converges. For instance, in the example in Table 1, the solution consists of lifesaver  $a_2$ , whose  $C(a_2) = \{V(a_2, rescue)\}$  has a mean of  $\hat{\mu}_{jk} = 0.8$  (in this example, SI=3, which means that lifesaver  $a_2$  has been selected in three rounds). This measure reflects the final outcome of the combination of the parameters of the simulation run in question.<sup>4</sup>

**Computational Requirement (CR).** The measure of the CR of the EXHAUSTIVE setting is GS multiplied by SI (as this enables us to assign an optimal lifesaver with high certainty). For example, if we have a group of GS=10 lifesavers and a stability indicator SI=50, then the CR is 500 assignment rounds (each lifesaver is assigned 50 times to a rescue). The CR of the procedure in the RANDOM setting is 0.

**Averaging and Normalising SQ and CR.** Results for each setting are averaged over 10000 trials (we selected this number of runs because it yields statistically significant results according to a 95% confidence interval). We normalise the SQ and CR of these 10000 trials as this enables a better comparison with the optimal and worst results of a particular setting. SQ and CR (which have been averaged over 10000 runs) is transformed into a range of 0 and 1. In particular, 0 represents the average mean performance of the worst lifesavers  $SQ_{min}$  and 1 represents the average performance of an optimal lifesaver  $SQ_{max}$ . These value assignments are reversed for CR (0 is optimal and 1 is as poor as for the EXHAUSTIVE setting).

### 3.3 Initialising a Simulation Run of a Setting

At the beginning of each run, each lifesaver’s capability mean  $\mu_{at_j}$  is initialised using a truncated normal distribution with a mean of  $\mu_{all} = 0.5$  and standard

<sup>4</sup> Another way to measure solution quality is to average the performance of a lifesaver that has been selected (one or more times) during a simulation run. However, this does not enable us to compare the results of the true capabilities of teams.

deviation of  $\sigma_{all} = 0.25$  (truncation is required to keep performance values within a 0 to 1 interval).

Agents are “partially knowledgeable” of the true performance of each lifesaver’s performance. We have chosen the following method to initialise the estimated mean of each agent’s model. For a given lifesaver  $at_j$ , the estimated mean is drawn from a normal distribution with a mean  $\hat{\mu}_{at_j}$ . The value  $\hat{\mu}_{at_j}$  is an average of the mean  $\mu_{at_j}$  (i.e., the mean of the distribution of a specific team’s capability) and the mean of the distribution that initialises the mean of all team estimates (i.e., 0.5).

The parameter Capability Deviations CD initialises the performance stability of each lifesaver (e.g., Stable or Unstable, Section 3.1). As defined by the parameter Observation Capacity (OC), each agent is endowed with the capacity to store 1 to 30 observations. For instance, if OC=8, then each agent is able to store 8 observations of the performance of different lifesavers (Section 2.2). The values for the capability distribution of each lifesaver remain constant throughout a run, while the mean of each agent’s models is updated after a rescue has been performed (Section 2.2). Each run consists of the assignment of a rescue task that is repeated until a lifesaver has been selected SI times.

### 3.4 Results and Analysis

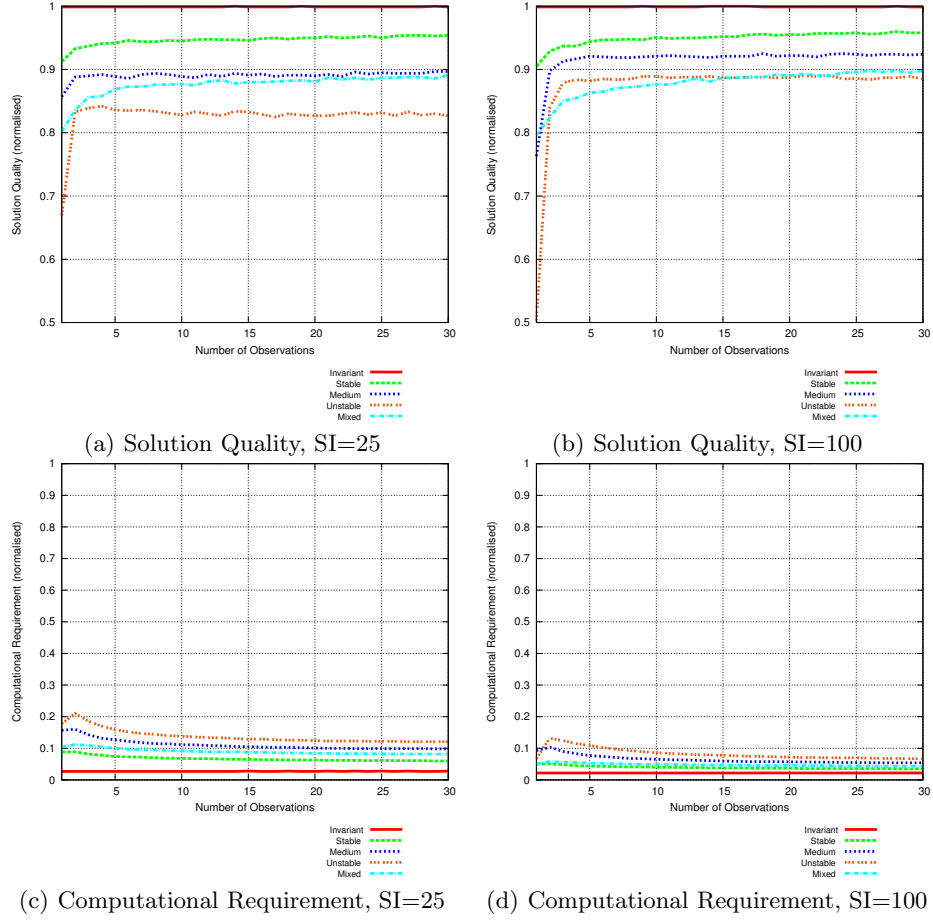
The results of our experiment are shown in Figure 3, which depicts (averaged and normalised) SQ and CR as a function of OC for our five types of lifesaver’s capability deviations – Invariant, Stable, Medium, Unstable and Mixed, plus the two benchmark settings RANDOM and EXHAUSTIVE.

**Results of Benchmark Settings are as expected.** The results for the RANDOM and EXHAUSTIVE settings correspond to the worst and best performance respectively, and are used as a benchmark for comparison with the other settings. The solution quality for the Invariant setting is slightly lower than that for the EXHAUSTIVE setting. These results are consistent with the results obtained for the RANDOM, EXHAUSTIVE and INVARIANT settings in previous experiments [3, 4]. This is due to the fact that the algorithm with Invariant lifesavers sometimes converge to a suboptimal solution. For example, this result is reached when each agent underestimates the performance of optimal lifesavers [4].

**The higher CD the worse SQ.** As seen in Figures 3(a) and 3(b), the average performance obtained for the Stable, Medium, Unstable and Mixed settings is on average worse than that obtained for the Invariant setting. This is due to the higher variability in rescue performance. The more unstable the performance of lifesavers is (i.e., the higher the CD), the worse the solution quality becomes. We posit that the main reason for this outcome is that the observer agents are unable to build reliable models when lifesavers exhibit unstable performance.

**As SI increases, SQ increases (for Unstable lifesavers).** Figures 3(a) and 3(b) also show how solution quality increases for unstable lifesavers as the SI increases. This benefit comes with a significantly higher computational requirement (as seen in Figures 3(c) and 3(d)). The reason is that agents have the opportunity to assess more observations and are able to build more accurate models.

**SQ improves dramatically with increasing OC across all GS, CD and P.** Average solution quality improves for all settings of CD, population sizes



(c) Computational Requirement, SI=25      (d) Computational Requirement, SI=100

**Fig. 3.** Policy =  $P_{max}$ , Group Size=50.

GS and policies P when agents are able to store more than one observation of the performance of lifesavers. The greatest improvement of solution quality is achieved when each agent stores 1-3 observations (when  $OC > 3$ , solution quality still improves, but at a lesser rate). The reason for this dramatic improvement of solution quality is that the models updated by each agent capture much of the previous performance of the lifesaver, despite keeping only a small window of 1-3 observations. It is unnecessary to store a large number of observations to build accurate models. The main difference of the two policies is that the maximum policy offers better results on average than the majority policy (this has been explored in [4]).

**CR becomes worse up to  $OC=2$  and then gradually improves.** Figures 3(c) and 3(d) show that the computational requirement is worst when  $OC=2$ . However, CR improves significantly when more than 2 observations are stored – the largest improvement is observed for  $OC = 2, \dots, 15$ . This is due to the additional storage of observations that compensates for testing lifesavers in additional rounds required to build accurate models.

## 4 Related Research

Previous theoretical and empirical studies assume that the performance of a team is deterministic and invariant [4, 1]. If the performance of a team is invariant, it can be adequately represented by one value. However, as a domain becomes more complex, predicting the performance of a team becomes more difficult due to the influence of factors that are not known. Such factors cannot be modelled explicitly and an accurate prediction of a team’s future performance is not possible using a deterministic ”one-value” representation. For example, human performance varies from task to task due to the complex nature of human behaviour. Hence, a prediction of human performance will not be accurate using a deterministic representation of human behaviour [6, 7]. To deal with uncertainty, many research projects have successfully applied the theory of probability [8]. For example, an agent interacts more efficiently with human users if it predicts their behaviour using a probabilistic model [6, 7]. Following this research, our extended framework enables agents to maintain probabilistic models to predict variable and non-deterministic performance of teams in the CIA problem.

Our work differs from existing research on Multi-Agent Reinforcement Learning (MARL) [9] and agent modelling [10]. Most work on MARL focuses on multiple agents that execute tasks individually and use Reinforcement Learning (RL) to coordinate their actions, taking into account various configurations (e.g., if agents can observe each others’ actions). Perhaps the best-known agent modelling technique to coordinate agents uses a decision-theoretic approach whereby each agent makes decisions to maximise its own payoff by recursively estimating the payoff of collaborators [10]. A key difference is that agents described in [10] and [9] is that our agents make group decisions which that guide how the entire multi-agent system learns.

## 5 Conclusion

This paper offers extensions to our approach to the Collective Iterative Allocation (CIA) problem and evaluates an algorithm that is designed to cope with uncertain conditions regarding the performance of teams (i.e., the performance of a team varies each time it performs a task). In particular, agents are endowed with the ability to model the performance of a team by its average performance and its stability. Given different levels of stability (invariant, stable, medium, unstable and mixed), our experiment evaluated an algorithm by varying each agent’s capacity to store observations (which are required for the refinement of its models). This paper offers the following insights.

1. Our results show that variable performance has a large impact on the efficiency of the algorithm. As the performance variability of teams increases, agents find it difficult to build accurate models of teams, which in turn results in lower solution quality and higher computational requirements.
2. The solution quality of the algorithm will improve dramatically if at least two or three past observations of a team’s performance are used by each agent to refine its models (for any type of performance variability, group size and policy). The use of probabilistic models is particularly effective when the performance of lifesaver is highly variable. However, the benefit on solution quality by using more than two or three observations is small.

3. This initial improvement of solution quality extends to the computational requirement of the algorithm if more than two observations are made. That is, the computational requirement of the algorithm is greatest when agents store only two observations, but it decreases steadily as agents store more than 2 observations.

We now have a starting point to address many future challenges based on the uncertain nature of allocation domains. For example, the results stated in 2. and 3. are obtained with a self-contained group of agents (i.e., as many agents as we have teams,  $A = AT$ ). We plan to analyse how many observations are required when the number of agents is different to the number of teams. Further, we plan to extend this framework such that each agent will also submit its confidence value of a lifesaver's performance (based on the estimated standard deviation). These confidence values can be used to make risk-averse group decisions where greater weight is assigned to a proposal of an agent that is confident about a medium performing lifesaver (and vice versa for risk-tolerant policy). Further extensions are required to deal with uncertainties present in communication and observation, such as information specified in a proposal may change due to interference introduced during a transmission.

## References

1. Chevalyere, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaître, M., Maudet, N., Padget, J., Phelps, S., Rodríguez-Aguilar, J.A., Sousa, P.: Issues in multiagent resource allocation. *Informatica* **30** (2006) 3–31
2. Guttman, C., Zukerman, I.: Towards models of incomplete and uncertain knowledge of collaborators' internal resources. In Denzinger, J., Lindemann, G., Timm, I.J., Unland, R., eds.: Proc. of the second German Conference on Multi-Agent system TEchnologieS (MATES). Volume 3187 of Lecture Notes in Artificial Intelligence., Erfurt, Germany, Springer-Verlag, Berlin, Germany (2004) 58–72
3. Guttman, C., Zukerman, I.: Agents with limited modeling abilities: Implications on collaborative problem solving. *Journal of Computer Science and Software Engineering (CSSE)* **21**(3) (2006) 183–196
4. Guttman, C., Rahwan, I., Georgeff, M.: An approach to the collective iterative task allocation problem. In: Proc. of the International Conference of Intelligent Agent Technology (IAT), USA, IEEE Press (2007) 363–369
5. Bond, A.H., Gasser, L.: An analysis of problems and research in DAI. In Bond, A.H., Gasser, L., eds.: Readings in Distributed Artificial Intelligence. (1988)
6. Billsus, D., Pazzani, M.: Learning probabilistic user models. In: Proc. of the Workshop on Machine Learning for User Modeling, Chia Laguna, Italy (1997)
7. Horvitz, E., Breese, J., Heckerman, D., Hovel, D., Rommelse, K.: The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In: Proc. of the fourteenth Conference on Uncertainty in Artificial Intelligence, Madison, Wisconsin, United States of America USA (July 1998) 256–265
8. Pearl, J.: Probabilistic reasoning in intelligent systems: Networks of plausible inference. Morgan Kaufmann publishers Inc. (1988)
9. Sandholm, T.: Perspectives on Multiagent Learning. *Artificial Intelligence (Special Issue on Multiagent Learning)* **171** (2007) 382–391
10. Gmytrasiewicz, P.J., Durfee, E.H.: Rational communication in multi-agent environments. *Autonomous Agents and Multi-Agent Systems* **4**(3) (2001) 233–272