Collective Iterative Allocation: Enabling Fast and Optimal Group Decision Making [1]

The Role of Group Knowledge, Optimism, and Decision Policies in Distributed Coordination

Christian Guttmann [a,*], Michael Georgeff [b] and Iyad Rahwan [c]
[a] *Department of General Practice, Faculty of Medicine, Nursing and Health Sciences, Monash University, Melbourne, Australia*
E-mail: christian.guttmann@gmail.com
[b] *Department of General Practice, Faculty of Medicine, Nursing and Health Sciences, Monash University, Melbourne, Australia*
E-mail: michael.georgeff@med.monash.edu.au
[c] *British University of Dubai, UAE and (Fellow) School of Informatics, University of Edinburgh, UK*
E-mail: irahwan@acm.org

Abstract.
A major challenge in the field of Multi-Agent Systems is to enable autonomous agents to allocate tasks efficiently. This paper extends previous work on an approach to the collective iterative allocation problem where a group of agents endeavours to find the best allocations possible through refinements of these allocations over time. For each iteration, each agent proposes an allocation based on its model of the problem domain, then one of the proposed allocations is selected and executed which enables us to assess if subsequent allocations should be refined. We offer an efficient algorithm capturing this process, and then report on theoretical and empirical results that analyse the role of three conditions in the performance of the algorithm: accuracy of agents' estimations of the performance of a task, the degree of optimism, and the type of group decision policy that determines which allocation is selected after each proposal phase.

Keywords: Distributed Problem Solving, Group Decision Making, Iterative Allocation, Sequential Allocation, Multi-User Multi-Agent Modelling

1. Introduction

Assigning agents to tasks is a challenging problem in the coordination of *Multi-Agent Systems (MAS)*, where each agent is autonomous and has its own unique knowledge of the agents and tasks involved [9, 20]. This problem requires efficient solutions as it is experienced in a wide range of applications, such as network routing, crisis management, logistics, computational grids, and collaborative student support environments [14, 31, 4].

This paper addresses the problem of Collective Iterative Allocation (CIA) which involves allocating tasks to teams (this paper uses the terms "tasks" and "teams" to describe allocations, but these terms are specific to individual application domains) [20].[1] We assume that each agent has estimations of the performance of different teams (i.e., they do not know the performance accurately). The accuracy of estimations maintained by each agent is of importance as it influences the

[1]This research was supported in part by Linkage Grant LP0774944 from the Australian Research Council.
[*]Corresponding author. E-mail: christian.guttmann@gmail.com.

[1]This paper extends a previous paper by offering substantially deeper theoretical and empirical results [22].

quality of allocations found. Knowledge of the performance of a team is acquired after it has been selected based on a group decision policy. Following other work, which argued that estimations of other agents' performance can improve individual agent decision-making [18], such estimations are useful in making collective decisions about allocating tasks to teams.

Our first question concerns the conditions under which we can guarantee optimal or near optimal allocations by testing only few teams. This paper considers the case of invariable team performance, i.e., a team's performance is the same every time it performs a particular task (related work often assumes implicitly that the performance of agents is invariable [4]).[2] Using various individual agents' proposals which are based on their estimations of the performance of different teams (a phase we call *proposing*), we select a team to a given task (a phase we call *selection*). The selection is based on group decision policies, where each agent contributes (e.g., through proposing) to the decision as to which team should be selected for a given task (Section 2). After a team is selected and executes the task, we progressively develop a better understanding of the true performance of teams (a phase we call *learning*) which is taken into account in subsequent allocations. Our second question concerns the conditions that determine the number of iterations required to find allocations. This question is addressed by studying different criteria that terminate the algorithm. In this context, we also examine three policies: the *majority policy* which selects the team proposed by the majority of agents; the *maximum policy* which selects the team proposed with the highest estimated performance; and the *minimum policy* which selects the team proposed with the lowest estimated performance. We explore these two questions theoretically as well as empirically.

This paper extends the *state of the art* by offering a comprehensive investigation of theoretical and empirical conditions that enable a group of agents to find optimal solutions in as few iterations as possible. The main contributions are described in three sections. Section 2 offers a formal representation of the approach and problem domain. This formal representation is required for the theoretical and empirical studies that follow in Sections 3 and 4. Section 3 offers results that define theoretical boundaries on two aspects of the efficiency of our approach based on [16]: the *quality* of the allocation calculated by our algorithm; and the number of rounds required to find this solution. We show that these aspects are influenced by the accuracy of agents' estimations of the performance of teams; the group's optimism; and the type of group decision policy. For example, we prove that if agents are completely optimistic, then we are guaranteed to converge to an optimal solution. Further, if agents are optimistic and use particular termination criteria then the algorithm can be significantly faster in finding optimal solutions than an exhaustive approach (that tests each possible team). Section 4 consists of empirical studies that determine the behaviour of the algorithm under empirical conditions. We perform a series of simulation-based experiments that show that the maximum policy converges to better allocations, but when the number of rounds is limited, then the majority policy outperforms the maximum policy. Section 5 discusses related literature and Section 6 summarises the contributions offered by this research.

[2]In previous work [23, 21], we presented empirical results in the case of variable team performance. But this paper offers stronger analytical properties about the performance of the algorithm.

2. Formal Definitions and Assignment Algorithm

This section defines the main components of this approach and an assignment algorithm.

2.1. Definition of Main Components

Our approach to the CIA problem is represented by the following tuple.

$$CIA =< T, AT, A = \{a_1(M_{a_1}, RP_{a_1}), \ldots\}, P >$$

T is a set of Tasks, each task t_i can be assigned to an agent team $at_i \in$ AT. Each agent a_i in A maintains Models M_{a_1} and uses Reasoning Processes RP_{a_1} to make assignments collectively using a group decision Policy P. These elements are defined in the following paragraphs.

Definition 1. *A set of **Tasks** is denoted by $T = \{t_1, \ldots, t_s\}$ with $s = |T|$.*

A task defined in T can be assigned to an agent team.

Definition 2. *A set of **Agent Teams** is denoted by $AT = \{at_1, \ldots, at_p\}$ with $p = |AT|$, where $at_j \in AT$ is a team.*

Definition 3. *A set of **Agents** is denoted by $A = \{a_1, \ldots, a_q\}$ with $q = |A|$, where $a_i \in A$ is an agent.*

The true performance of a team at_j for various tasks is referred to as a team's capability: $C(at_j) = \{V(at_j, t_1), \ldots, V(at_j, t_s)\}$, where $V : AT \times T \rightarrow R^+$ is a value representing the performance of a team for a task (the value 0 corresponds to being unable to perform a task). The capability of a team can only be estimated (in this study, a value of team's true performance is only revealed after a team performed a task, values for the true performance of teams that have not performed a task yet are only specified for illustrative purposes). Each agent $a_i \in A$ maintains models M_{a_i} to estimate the capabilities of teams and each agent is able to execute Reasoning Processes RP_{a_i} using these models.

Definition 4. *M_{a_i} are the **Models** maintained by agent a_i. Models are expressed by $M_{a_i} = \{M_{a_i}(at_1), \ldots, M_{a_i}(at_p)\}$ with p being the number of teams in AT. A specific model of a team at_j is defined by a set of estimations*

$$M_{a_i}(at_j) = \{\hat{V}_{a_i}(at_j, t_1), \ldots, \hat{V}_{a_i}(at_j, t_s)\}, \textit{where}$$

- · *$t_1, \ldots, t_s$ are the tasks defined in T (Definition 1).*
- · *at_j is a team in AT (Definition 2).*
- · *$\hat{V}_{a_i}(at_j, t_k)$ is a value function that maps a team at_j and a task t_k to a value that is an estimation of team at_j's true task performance $V(at_j, t_k)$.*

Definition 5. *RP_{a_i} are a_i's **Reasoning Processes**.*

- *For each agent a_i and task t_k, the* INITIALISE *process returns a set of Models M_{a_i}.*
- *For each agent a_i and task t_k, the* PROPOSE *process returns a proposal defining a team and its estimated performance: proposal$_{a_i}$ = $\langle at_j, \hat{V}_{a_i}(at_j)\rangle$.*
- *For each agent a_i, task t_k and team at_j, the* UPDATE *process returns a set of updated models M'_{a_i} (in this study, we assume that models are updated after a team performs a task t_k).*

For a given task, the estimations in the models maintained by each agent are initialised based on its own unique experience and each agent can propose a particular team with an estimation of that team's performance (Definition 5). A central issue in our approach is to fully utilise the proposals submitted by the agents A for the selection of a team. To this end, we define a policy that governs the process of making a group decision. We consider the following three notations that underly the definition of a group decision policy.

– A set of combinations of proposals that could be submitted by agents A is denoted by *SETPROPOSALS*$_A$ for a particular group decision.
– Let *PROPOSALS*$_A$ $\in$ *SETPROPOSALS*$_A$ be a specific set of proposals, such that *PROPOSALS*$_A$ =$\{$*proposal*$_{a_1}, \ldots,$ *proposal*$_{a_q}\}$, where $q = |A|$ (i.e., each agent makes one proposal). As per Definition 4, *proposal*$_{a_i}$ is an ordered pair of the proposed team and its proposed performance as specified by an agent a_i's PROPOSE process.
– Finally, a set of proposed teams is denoted by $AT_{\textit{PROPOSALS}_A} = \{at_j | \langle at_j, \hat{V}_{a_i}(at_j, t_k)\rangle \in$ *PROPOSALS*$_A\}$.

With these notations, we define the policy as follows.

Definition 6. *A group decision Policy P is an algorithmic process for selecting one proposal from a set of proposals, such that P : SETPROPOSALS$_A$ $\rightarrow AT$. Then,*

$$at_{\textit{selected}} := P(\textit{PROPOSALS}_A), \textit{where}$$

- · *$at_{\textit{selected}}$ is a team in $AT_{\textit{PROPOSALS}_A}$.*
- · *PROPOSALS$_A$ is a proposal set in SETPROPOSALS$_A$.*

Note that a policy selects a team only from those that have been proposed, that is, the selected team is in $AT_{\textit{PROPOSALS}_A}$.

2.2. TAP Assignment Algorithm

We now introduce a formal algorithm that coordinates the main processes of the assignment procedure.

2.2.1. Preliminaries

Each agent follows the assignment algorithm (depicted in Figure 1) to coordinate processes with other agents and to make collective assignments of a team to a task over multiple rounds of proposing, selecting and learning. The algorithm uses the definitions introduced in the previous section.

Two more clarifications are required to define the algorithm. First, the algorithm assumes the execution

TAP ASSIGNMENT ALGORITHM.

INPUT: Task $t_k \in T$, 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) $\mathit{PROPOSALS}_A = \bigcup_{a_i \in A} \mathit{PROPOSE}_{a_i}(M_{a_i}, t_k)$
    (b) $at_{\mathit{selected}} := P(\mathit{PROPOSALS}_A)$
    (c) UPDATE$_{a_i}(M_{a_i}, V(at_{\mathit{selected}}, t_k))$ ($\forall a_i \in A$)
4. Until a termination criterion is satisfied
```

Fig. 1. A task is repeatedly assigned to different teams until a criterion is satisfied (e.g., a team is believed to perform the task best).

of an ANNOUNCE process that broadcasts the task $t_k \in T$ to all agents. Unlike reasoning processes RP executed by a specific agent a_i (Definition 4), the ANNOUNCE process can be executed by any party. We also assume that the policy is applied by an *agent* a_{policy} that does not alter proposals or the outcome of the policy. Note that finding an agent a_{policy} can be a difficult problem in MAS, particularly if the MAS encourages opportunistic behaviour and the agent modifies the outcome of the policy.

2.2.2. How Does the Assignment Algorithm Work?

The assignment algorithm presented in Figure 1 works as follows. In step 1, a task is announced and broadcasted to all agents A. Upon receiving this announcement, each agent initialises its models of the performance of all teams (step 2 in Figure 1).

The loop (steps 3a, 3b, and 3c) is called an *assignment round* r_i with i being the i'th iteration of the loop. A reasoning process or group decision executed in a particular round is denoted with a lower index indicating the round. For example, the team that has been selected in round 3 is denoted by $at_{selected,r_3}$, and the policy outcome in that round is expressed by P_{r_3}. The estimated performance of the selected team $at_{selected,r_3}$ is denoted by $\hat{V}_{a_i,r_3}(at_{\mathit{selected}}, t_k)$.

Each assignment round involves proposing a team by each agent (step 3a), selecting a team (step 3b) and learning from the performance of the assigned team after task execution (step 3c).

– In step 3a, each agent proposes a team for the announced task and the proposals communicated by the agents in A are stored in $\mathit{PROPOSALS}_A$. For example, agent a_1's PROPOSE process returns $\mathit{proposal}_{a_1}$=$\langle\{a_1, a_2\}, 0.4\rangle$ and agent a_2 PROPOSE process returns $\mathit{proposal}_{a_2}$=$\langle\{a_2\}, 0.25\rangle$. The proposals communicated by a_1 and a_2 are: $\mathit{PROPOSALS}_A$=$\{\mathit{proposal}_{a_1} = \langle\{a_1, a_2\}, 0.4\rangle, \mathit{proposal}_{a_2} = \langle\{a_2\}, 0.25\rangle\}$.
– In step 3b, an agent a_{policy} uses the $\mathit{PROPOSALS}_A$ and a group decision policy P to select a team at_{selected} for the announced task (Definition 6). The selected team at_{selected} is assigned to the task and will then perform it. For example, a policy can select the highest proposed team at_3 (consisting of agent a_1 and a_2).
– In step 3c, each agent updates its models based on the true performance of the selected team: $V(at_{\mathit{selected}}, \mathit{bake})$. For example, the estimated value $\hat{V}_{a_1,r_i}(at_3, t_k) = 0.4$ is replaced with the true performance of the selected team $\hat{V}_{a_1,r_{i+1}}(at_3, t_k) = V(at_{\mathit{selected}}, \mathit{bake}) = 0.2$. The models are now changed and this may alter the selection of a team for the task in the next round.

An assignment round is repeated until the agent a_{policy} terminates the algorithm according to a criterion (step 4). A simple example of a criterion is to limit the number of assignment rounds to $k \in N$ (e.g., a similar criterion is used in [34]). Such a criterion may be inefficient as it can terminate the algorithm well before or well after an optimal team has been selected. Other criteria may terminate the algorithm more efficiently. For example, the algorithm may terminate when no proposals specifies a team which estimated performance is higher than any previously selected team. This paper considers criteria that uses the proposed estimations and information of each team's true performance (as it becomes known). As our studies unfold in this paper, it will become clear that developing efficient criteria is a difficult problem.

2.2.3. Solution Computed by Algorithm

Finally, we need to specify the solution returned by the algorithm when it terminates. We define this solution to be a team whose actual performance is no worse than any team that has performed the designated task in previous assignment rounds. More formally,

– Let $AT_{\mathit{KnownSoFar}} = \{at_j \in AT : at_j = at_{r_t}, r_t \leq r_{\mathit{current}}$, where r_{current} is the current round$\}$. That is, if the algorithm is currently in round r_{current}, and the performance of the selected team $at_{r_{\mathit{current}}}$ becomes known, then $AT_{\mathit{KnownSoFar}}$

is a set of teams that were selected in rounds $r_1, \ldots, r_{current}$. The performance of these teams is known accurately $V_{AT_{KnownSoFar}} = \{V(at_j) : at_j \in AT_{KnownSoFar}\}$. At the beginning of the algorithm, before the first selection of a team, $AT_{KnownSoFar} = \emptyset$.

– Let $at_{BestSoFar} = \underset{at_l \in AT_{KnownSoFar}}{\mathrm{argmax}} V(at_l)$ be the team with the highest true performance $V(at_{BestSoFar})$ in $AT_{KnownSoFar}$.

Note that more than one team may satisfy this definition, so that the solution returned by the algorithm may be non-deterministic.

2.3. Formalising the CNET protocol in our Framework

The contract net protocol is a well known coordination mechanism for distributed systems [36]. In CNET, each agent proposes itself for an announced task (using its estimation of its own task performance) and the highest bidding agent is selected for the task. The CNET protocol is a traditional market-based allocation scheme which assumes that each agent knows its own performance best. Given its prevalence in many application domains, this section briefly shows how the CNET protocol can be formalised using our formalism.

The algorithm in Figure 1 works similar to the CNET protocol [36] by specifying the following components of our framework.

– Each agent is also a team and vice versa. That is, the group of agents is self-contained as $A = AT$.
– Each agent proposes itself with a self-estimate of its own performance (and to this end, each agent only needs to maintain a model of its own performance).
– A maximum policy is used to select a team (which is similar to a manager selecting the best bid).
– Our algorithm is terminated after one assignment round.

Under this setting, finding an optimal allocation relies on the accuracy of each agent's estimations of its own performance. For example, if an optimal agent's estimations remain lower than that of other agents, it will never win the bid, and thus the agent's true performance will never be known. This paper shows that our approach substantially extends the CNET protocol to offer better and faster solutions. Note that [20] uses this framework and algorithm to represent other well known assignment processes, such as the process of aggregating preferences in classical voting.

3. Theoretical Considerations

The previous section introduced a formal framework and an algorithm that defines our approach to the Collective Iterative Allocation (CIA) problem. This section presents a theoretical study of our algorithm focusing on conditions under which we are guaranteed to find optimal solutions, and different termination criteria that influence the computational requirement of the algorithm. Section 3.1 first discusses the underlying assumptions of our study and Section 3.2 discusses the efficiency aspects of the algorithm. Sections 3.3 and 3.4 then analyse the role of the agents' estimations of team performance in the efficiency of the algorithm. In particular, we are interested in identifying conditions under which the algorithm is guaranteed to find a team that performs a task optimally, and conditions that can reduce the computational requirement of the algorithm.

3.1. Preliminaries

This theoretical study assumes that the capability of each team will not change (it is invariant and deterministic) when performing the same task under the same conditions. This assumption is made to focus our investigations of the efficiency of the algorithm on three aspects: accuracy of estimations, policies and termination criteria. In related work, the behaviour of a team is often implicitly assumed to be the same whenever it performs the same task, because the context under which the behaviour occurs does not change [14, 37, 40].

It is worthwhile noting that the assumption on invariant and deterministic performance is not that restrictive, as we could in fact model *normalised* performance with respect to the environment or situation. For example, if we applied this to long distance running, and each iteration involves a different length race, we could simply normalise by dividing the length of the race by the race time and using this as the performance measure.

For clarity of exposition, we have chosen to simplify two aspects of the notation used in this chapter. From now on, we assume a fixed task $t_k \in T$ and we use an abbreviated form of the notation for the processes and estimations. For example, instead of notating an estimation with $\hat{V}(at_j, t_k)$, we now simply refer to $\hat{V}(at_j)$. Analytical results that hold for a task t_k also hold for any task selected from T if performing this task does not depend on first performing other tasks. Finally, values, processes and policy outcomes are associated with the same round, if not indicated otherwise. That is, if we have not indicated a value, process or policy with a lower index of the round, they are associated with the same round.

3.2. *Evaluating the Efficiency of the Assignment Algorithm*

[15, 16] propose three aspects to evaluate the efficiency of allocation algorithms: *solution quality* – a value of the solution computed for an allocation problem, *computational requirement* – the number of times a dominant operation is repeated, and *communication requirements* – total number of messages sent over the network [15, 16].[3] This paper uses two of these efficiency aspects to evaluate our algorithm.

– **Solution quality**: performance of the team for a particular task found by our algorithm.
– **Computational requirement**: number of assignment rounds required before a solution is found.

The third aspect, communication requirement, can be derived from the computational requirement by multiplying the computational requirement with the number of agents in a group. For example, for three rounds and two agents, the number of communicated proposals is: $|\text{rounds}| \times |A| = 3 \times 2 = 6$. This calculation underlies the assumption that each agent in A communicates one proposal in each round.

To offer useful theoretical properties of the efficiency of our algorithm, we need to make assumptions about the reasoning processes, estimations maintained by agents and the termination condition of the algorithm. For example, if each agent proposes the same team in each round and does not update its models, and the algorithm terminates at some round in the future, we would not be able to make precise predictions of the efficiency of the algorithm. Hence, a specification of the reasoning processes, estimations and termination conditions is required to offer theoretical properties on the efficiency of the algorithm.

Particularly, we show that under certain conditions, we can guarantee to find an *optimal solution* which is a team denoted by at_* whose performance is not lower than the performance of other teams in AT, i.e., $at_* = \operatorname*{argmax}_{at_j \in AT} V(at_j)$ for all $at_j \in AT$. Note that if $|\operatorname*{argmax}_{at_j \in AT} V(at_j)| > 1$, then there are several optimal solutions.

3.3. *Reaching Optimal Solutions under Complete Group Optimism*

This section examines the efficiency of the algorithm under the assumption that a group of agents is completely optimistic (i.e., each agent in the group is optimistic). The next section examines if complete optimism is a prerequisite for guaranteeing an optimal solution.

3.3.1. *Policy – Independent Optimality under Complete Group Optimism*

Research on reinforcement learning found that optimistic initial estimations have a significant influence on converging to optimal solutions as they encourage the exploration of the problem space and eventually lead to better results [38]. However, this research does not offer any guarantees on finding optimal solutions or the computational requirement of finding solutions. Further, previous research only considers how a single agent makes individual decisions and how this influences its own learning behaviour (and not how groups of agents make decisions together). Research on reinforcement learning has not investigated the issue of optimism in collective decision making and learning. We extend existing work to address this issue.

For this investigation, we need to define the notion of complete optimism in a group of agents. To this end, we first specify two reasoning processes used by each agent (Definition 5), starting with the INITIALISE process.

Definition 7. ***Optimistic Initialiser.*** *An agent a_i is an optimistic initialiser if, and only if, it does not initialise its estimations of the performance of each team $at_j \in AT$ with values lower than the team's true performance. That is, the INITIALISE process (Definition 5) of agent a_i returns models M_{a_i} where $\hat{V}_{a_i}(at_j) \geq V(at_j)$ for all teams $at_j \in AT$.*

[3]Note that the term of computational requirement to describe a performance aspect of a coordination mechanism is not universally shared. For example, Endriss and Maudet use the term "communication complexity" which refers to the number of deals required to find an optimal assignment in the context of negotiation [10]. Economic literature often refers to the notion of "transaction costs" and [20] discusses this notion in the context of CIA problems.

Example 1. *There might be various situations under which agents are optimistic initialisers. For example, assume that we have two agents a_1 and a_2 and two teams at_1 and at_2. Each agent observes each team running a certain relay for a certain distance, and records the time they take. Each agent observes the same teams but over different distances. For example, a_1 observes at_1 running 1 kilometre (km), a_2 observes the same team running 2 km; and similarly a_1 observes at_2 running 3 km and a_2 observes at_2 running 1 km. Now, the task given is to run a marathon of approximately 42 km. Each agent bases their estimations of each team's running performance by linearly extrapolating from their prior observations of the running performance of teams of shorter distances. Assume that teams actually become slower on average as they run longer distances. In this case, each agent overestimates the performance of each team, all by differing amounts as they all observed the teams over different distances.*

Definition 8. ***Optimistic Updater.*** *An agent a_i is an optimistic updater if, and only if, it does not update its estimations of the performance of each team $at_j \in AT$ for a task with values lower than the team's true performance. That is, the UPDATE process (Definition 5) of this agent returns updated models M_{a_i} where $\hat{V}_{a_i}(at_j) \geq V(at_j)$ for all teams $at_j \in AT$.*

Example 2. *Agents may have several reasons not to update their models with underestimations. Following from the previous example (Example 1), consider that agents have observed an improvement of each team's running performance over a shorter distance (but the teams have not run a marathon yet). Agents may therefore believe that a similar improvement is to be expected for longer distances. For example, assume that team at_1 and at_2 improved their performance on shorter distances. Assume that team at_1 requires 4 hours and team at_2 requires $4\frac{1}{2}$ hours to run a first marathon and they would have already reached their peak performance. Now, agents may assume a similar improvement of the marathon performance as was observed for teams running shorter distances. Then, for a next marathon, each agent's updated models will estimate at_1's time shorter than 4 hours and at_2's time shorter than $4\frac{1}{2}$ hours.*

For clarity of exposition, the following definition refers to an optimistic agent as being one that is an optimistic initialiser and updater.

Remark 1. **Optimistic Agent.** We say that an agent is *optimistic* if, and only if, the agent is an *optimistic initialiser* (Definition 7) and an *optimistic updater* (Definition 8).

Definition 9. ***Complete Group Optimism.*** *A group of agents A is completely optimistic if, and only if, each agent in this group is optimistic, so that $\hat{V}_{a_i}(at_j) \geq V(at_j)$ for all $a_i \in A$ and for all $at_j \in AT$ at all times.*

We also define an agent's PROPOSE process representing the agent's contribution to find an optimal solution (which we also refer to as *task rationality*). For this definition, recall the following expressions which were first introduced in Section 2.2.3.

- Let $AT_{\mathit{KnownSoFar}} = \{at_j \in AT : at_j = at_{r_t}, r_t \leq r_{\mathit{current}}$, where r_{current} is the current round$\}$. That is, if the algorithm is currently in round r_{current}, and the performance of the selected team $at_{r_{\mathit{current}}}$ becomes known, then $AT_{\mathit{KnownSoFar}}$ is a set of teams that were selected in rounds $r_1, \ldots, r_{\mathit{current}}$. The performance of these teams is known accurately $V_{AT_{\mathit{KnownSoFar}}} = \{V(at_j) : at_j \in AT_{\mathit{KnownSoFar}}\}$. At the beginning of the algorithm, before the first selection of a team, $AT_{\mathit{KnownSoFar}} = \emptyset$.
- Let $at_{\mathit{BestSoFar}} = \underset{at_l \in AT_{\mathit{KnownSoFar}}}{\operatorname{argmax}} V(at_l)$ be the team with the highest true performance $V(at_{\mathit{BestSoFar}})$ in $AT_{\mathit{KnownSoFar}}$.

Definition 10. ***Task Rational Proposer.*** *An agent a_i is a task rational proposer if, and only if, its PROPOSE process (Definition 5) is as follows.*

1. *if $\exists at_j : \hat{V}_{a_i}(at_j) > V(at_{\mathit{BestSoFar}})$, then a_i proposes a team with an estimation higher than $V(at_{\mathit{BestSoFar}})$, such that $proposal_{a_i} = \langle at_l, \hat{V}_{a_i}(at_l)\rangle$ where $\hat{V}_{a_i}(at_l) > V(at_{\mathit{BestSoFar}})$ for a team $at_l \in AT$ (note that $at_l = at_j$ if $\hat{V}_{a_i}(at_j)$ is the only estimation higher than $at_{\mathit{BestSoFar}}$).*
2. *if $\neg\exists at_j : \hat{V}_{a_i}(at_j) > V(at_{\mathit{BestSoFar}})$, then a_i proposes any team.*

If an agent has estimations higher than $V(at_{\mathit{BestSoFar}})$ for several teams, a “cautious” task rational proposer may, for example, propose the next best team to $at_{\mathit{BestSoFar}}$. Also consistent with this definition is a “maximal” task rational proposer that specifies a team with the overall highest estimated performance in the agent's models.

The first theorem demonstrates that complete group optimism and task rationality are important premises for guaranteeing the finding of an optimal solution with our algorithm. These premises combined with testing the proposals communicated in each round enable us to identify the precise conditions under which an optimal solution is guaranteed. The importance of this theorem is that the algorithm identifies an optimal solution (under conditions of complete group optimism and task rationality), regardless of the type of group decision policy used. Hence, we call this theorem the sufficient condition.

Theorem 1. ***Optimality under Complete Group Optimism and Task Rationality: Sufficient Condition.*** *If*

- ***(A)*** *the group of agents A is completely optimistic (Definition 9),*[4]
- ***(B)*** *each agent in A is a task rational proposer (Definition 10), and*
- ***(C)*** *the algorithm does not terminate if $\forall\langle at_j, \hat{V}(at_j)\rangle \in \mathit{PROPOSALS}_A$ such that $\hat{V}(at_j) > V(at_{\mathit{BestSoFar}})$,*

then if the algorithm terminates it will terminate with an optimal solution.

Proof. The theorem is proven by contradiction, i.e., we assume that if the algorithm terminates, then we have found a suboptimal team $at_{\mathit{BestSoFar}}$.

1. If the team $at_{\mathit{BestSoFar}}$ is not an optimal team, then there must be a team at_*, such that $V(at_*) > V(at_{\mathit{BestSoFar}})$.
2. According to **(C)**, if the algorithm terminates, we know that $\exists\langle at_j, \hat{V}(at_j)\rangle \in \mathit{PROPOSALS}_A$ such that $\hat{V}(at_j) \leq V(at_{\mathit{BestSoFar}})$. Assume that $a' \in A$ is the agent that made this proposal.
3. According to **(B)**, we know that $V(at_{\mathit{BestSoFar}}) \geq \hat{V}_{a'}(at_j)$ for all $at_j \in AT$. In particular, we know that $V(at_{\mathit{BestSoFar}}) \geq \hat{V}_{a'}(at_*)$.

According to 1., $V(at_*) > V(at_{\mathit{BestSoFar}})$, and according to 3. $V(at_{\mathit{BestSoFar}}) \geq \hat{V}_{a'}(at_*)$. So, we know that $V(at_*) > \hat{V}_{a'}(at_*)$. However, this is a contradiction, because we assumed that no agent has initial underestimations of the performance of any team nor does it update its models with underestimations (Condition **(A)**). □

[4]The minimum condition for this theorem is that each agent is optimistic of the performance of only an optimal team. However, this premise is difficult to prove in many realistic problem domains.

To illustrate this theorem recall Examples 1 and 2. The best marathon team is identified as soon as a proposed estimation is not greater than the best marathon team known so far, if we have a completely optimistic and task rational group of agents (Definitions 9 and 10).

Note that if all estimations are higher than the true performance of an optimal team, then we can define a stronger Condition **C** enabling an earlier identification of an optimal solution and thus reducing the computational requirement of the algorithm (which is further investigated in Section 3.3.3). This condition is true if any proposed estimation communicated in any previous round is not greater than $at_{\mathit{BestSoFar}}$. In this case, we know that, if the algorithm terminates, we have found an optimal solution.

3.3.2. Optimal Solution with Three Termination Criteria

We now define three termination criteria and investigate if they are guaranteed to terminate the algorithm with an optimal solution. We first prove that an optimal solution is found with each of these criteria. In Section 3.3.3, we order these criteria with respect to their computational requirement (i.e., the required number of rounds before the algorithm terminates with an optimal solution).

For the following criteria, we assume that the policy is restricted to not select a team more than once (to prevent looping), hence the algorithm will eventually terminate, because the set of teams AT is finite (in the worst case after all teams have been selected at least once).

The following criterion terminates the algorithm if all proposed estimations of the team selected at_{selected} by the policy are not greater than the best one found so far: $at_{\mathit{BestSoFar}}$. In other words, we only terminate the algorithm if no agent believes that we can find a better team for the task.

Termination Criterion 1. *The algorithm terminates if the following condition is true: $\forall a_i \in A : \mathit{proposal}_{a_i} = \langle at_{\mathit{selected}}, \hat{V}_{a_i}(at_{\mathit{selected}})\rangle$ such that $\hat{V}_{a_i}(at_{\mathit{selected}}) \leq V(at_{\mathit{BestSoFar}})$.*

The next criterion is more restrictive as it terminates the algorithm if only one proposed estimation of the performance of the selected team is not greater than $at_{\mathit{BestSoFar}}$.

Termination Criterion 2. *The algorithm terminates if the following condition is true:*

$\exists a_i \in A$: *proposal*$_{a_i} = \langle at_{selected}, \hat{V}_{a_i}(at_{selected})\rangle$ *such that* $\hat{V}_{a_i}(at_{selected}) \leq V(at_{BestSoFar})$.

The third criterion is most restrictive as it will terminate if at least one proposed team has an estimated performance not greater than $V(at_{BestSoFar})$.

Termination Criterion 3. *The algorithm terminates if the following condition is true:* $\exists a_i \in A$: *proposal*$_{a_i} = \langle at_j, \hat{V}_{a_i}(at_j)\rangle$ *such that* $\hat{V}_{a_i}(at_j) \leq V(at_{BestSoFar})$.

We first prove that the most restrictive criterion, TC 3, will terminate with an optimal solution.

Lemma 1. ***Optimality under Complete Group Optimism and Task Rationality: Termination Criterion 3.*** *Under complete group optimism (Definition 9) and under the condition that each agent is task rational (Definition 10), the algorithm terminates with an optimal solution using Termination Criterion 3.*

Proof. We will prove optimality by showing that the algorithm does not terminate before Condition **(C)** in Theorem 1 is satisfied. Recall that *SETPROPOSALS*$_A$ is a set of all possible combinations of proposals that can be communicated by agents (Definition 6).

1. Let *SETPROPOSALS*$_{TC3} \subseteq$ *SETPROPOSALS*$_A$ be a set of proposal sets, where each proposal set *PROPOSALS*$_{TC3} \in$ *SETPROPOSALS*$_{TC3}$ will satisfy TC 3.
2. Let *PROPOSALS'*$_{TC3} \in$ *SETPROPOSALS*$_{TC3}$ be an arbitrary set of proposals that will satisfy TC 3.
3. According to TC 3, when the algorithm terminates, there exists at least one proposal' $\in$ *PROPOSALS'*$_{TC3}$ that does not estimate a team greater than $V(at_{BestSoFar})$, such that proposal'$=$ $\langle at_j, \hat{V}(at_j)\rangle$*where* $\hat{V}(at_j) \leq V(at_{BestSoFar})$.

Since proposal' also satisfies Condition **(C)** specified in Theorem 1, the algorithm that uses TC 3 will terminate with an optimal solution. □

Lemma 2. ***Optimality under Complete Group Optimism and Task Rationality: Termination Criterion 2.*** *Under complete group optimism (Definition 9) and under the condition that each agent is task rational (Definition 10), the algorithm terminates with an optimal solution using Termination Criterion 2.*

Proof. Since *SETPROPOSALS*$_{TC2} \subseteq$ *SETPROPOSALS*$_{TC3}$, *SETPROPOSALS*$_{TC2}$ will also consist of proposal sets which have at least one proposal, proposal', with a proposed performance that is not greater than $at_{BestSoFar}$ (as specified for *PROPOSALS*$_{TC3}$). Since proposal' satisfies condition **(C)** specified in Theorem 1, the algorithm terminates with an optimal solution using Termination Criterion 2. □

Lemma 3. ***Optimality under Complete Group Optimism and Task Rationality: Termination Criterion 1.*** *Under complete group optimism (Definition 9) and under the condition that each agent is task rational (Definition 10), the algorithm terminates with an optimal solution using Termination Criterion 1.*

Proof. Since *SETPROPOSALS*$_{TC1} \subseteq$ *SETPROPOSALS*$_{TC3}$, *SETPROPOSALS*$_{TC1}$ will also consist of proposal sets which have at least one proposal, proposal', with a proposed performance that is not greater than $at_{BestSoFar}$ (as specified for *PROPOSALS*$_{TC3}$). Since proposal' satisfies condition **(C)** specified in Theorem 1, the algorithm that uses Termination Criterion 1 will terminate the algorithm with an optimal solution. □

3.3.3. Ordering Three Termination Criteria with respect to Computational Requirement

For comparative purposes, consider an exhaustive procedure that assigns each team at least once and after each team has performed a task, the true performance of each team is known and an optimal solution can be determined. By testing each team, one-by-one, it is possible that an optimal team could be assigned in the first round, but it will not be known if it is an optimal team before testing all remaining teams. Our aim in this section is to determine a ranking of Termination Criteria 1, 2 and 3 with respect to the number of required rounds to find an optimal solution.

For the purpose of analysing the computational requirement of the algorithm, the UPDATE and PROPOSE processes specified in Definitions 8 and 10 are too general. For example, according to Definition 10 (task rational proposer), agents may propose any team as long as they propose one with an estimation higher than $V(at_{BestSoFar})$ (if there is such a team). Assume that a_1 is a task rational proposer and has three performance estimations $\hat{V}_{a_1}(at_1) = 0.6$, $\hat{V}_{a_1}(at_2) = 0.7$ and $\hat{V}_{a_1}(at_3) = 0.8$, $V(at_{BestSoFar}) = 0.5$, and $V(at_1) = V(at_2) = 0.5$ and $V(at_3) = 0.8$. As a_1 is task rational it could propose at_1 first, then at_2, and finally at_3 or it could propose at_2 first, and then at_3. It could also propose at_3 in the first round. As seen from this example, we cannot predict how many rounds are required before the algorithm terminates if agents act according to Definition 10. Also, if agents use the optimistic UPDATE process specified in Definition 8, each agent may only make slight changes to its models and propose different teams before the algorithm terminates.

An assessment of the computational requirement of the algorithm with a particular criterion requires more restrictive definitions of the PROPOSE and UPDATE processes. These definitions will enable us to prove specific theorems of the computational requirement of the algorithm.

For Theorem 1, we assumed that each agent is an optimistic updater – the following definition is a special case of such an agent (Definition 8).

Definition 11. ***Accurate Updater.*** *An agent a_i is an accurate updater if, and only if, the UPDATE process of this agent replaces an estimated value of a team with the value that represents the observed performance of the selected team (after it performs the task): $\hat{V}_{a_i}(at_{selected}) = V(at_{selected})$.*[5]

The following definition is a special case of a task rational proposer (Definition 10).

Definition 12. ***Maximal Task Rational Proposer.*** *An agent is a maximal task rational proposer if, and only if, it proposes a team with the highest estimated performance according to the estimations in its models. That is, an agent a_i proposes a team at_j with $\max_{at_j \in AT} \hat{V}_{a_i}(at_j)$.*

We can now rank the Termination Criteria 1, 2 and 3 according to the number of rounds the algorithm requires to find an optimal solution. We say that “TC x is at least as efficient as TC y”, if the algorithm terminates with TC x in no greater number of rounds than it would take for the algorithm to terminate with TC y (in other words, TC y will need at least as many rounds as TC x). The next theorem states that Criterion 2 is at least as efficient as Criterion 1.

Theorem 2. ***Termination Criterion 2 is at least as efficient as Termination Criterion 1.*** *If each agent in A is an optimistic initialiser (Definition 7), a maximal task rational proposer (Definition 12), and an accurate updater (Definition 11), then TC 2 will terminate the algorithm with an optimal solution in at least as many rounds as TC 1.*

[5]Since we assume that the performance of teams is invariable, the observed performance equals the true performance of a team.

Proof. We will prove this theorem by showing that there is a set of proposals that terminates the algorithm with TC 2, but not with TC 1.

Let $\mathit{SETPROPOSALS}_{TC1} \subseteq \mathit{SETPROPOSALS}_A$ be a set of all proposal sets that will satisfy TC 1 and let $\mathit{SETPROPOSALS}_{TC2} \subseteq \mathit{SETPROPOSALS}_A$ be a set of proposal sets that will satisfy TC 2 ($\mathit{SETPROPOSALS}_A$ is a set of a combination of proposals communicated by agents, Definition 6).

According to Lemma 2, if the algorithm terminates with TC 2 and finds an optimal solution, then it will also terminate and find an optimal solution with TC 1. We also know that $\mathit{SETPROPOSALS}_{TC2} \subseteq \mathit{SETPROPOSALS}_{TC1}$. Then, we know that $\mathit{SETPROPOSALS}_{TC2} / \mathit{SETPROPOSALS}_{TC1}$ are proposal sets that will terminate with TC 2, but not with TC 1. In other words, any sequence of rounds that terminates with TC 1 will terminate with TC 2 after the same number of rounds, if not earlier. □

Theorem 3. ***Termination Criterion 3 is at least as efficient as Termination Criterion 2.*** *If each agent in A is an optimistic initialiser (Definition 7), a maximal task rational proposer (Definition 12), and an accurate updater (Definition 11), then TC 3 will terminate the algorithm with an optimal solution in at least as many rounds as TC 2.*

Proof. According to Lemma 2, we know that $\mathit{SETPROPOSALS}_{TC1}$ are the proposal sets that terminate the algorithm with an optimal solution. We also know that $\mathit{SETPROPOSALS}_{TC2} \subseteq \mathit{SETPROPOSALS}_{TC3}$. Hence, this proof is similar to the proof for Theorem 2. In brief, we have $\mathit{SETPROPOSALS}_{TC3} / \mathit{SETPROPOSALS}_{TC2}$ which are the proposal sets that will terminate with TC 3, but not with TC 2. That is, any sequence of rounds that terminates with TC 2 will terminate with TC 3 after the same number of rounds, if not earlier. □

Corollary 1. *From Theorems 2 and 3, we can see that TC 3 is at least as efficient as TC 1 and TC 2.*

That is, the algorithm with TC 3 never requires more rounds than with the other two criteria (if agents are completely optimistic, accurate updaters and maximal task rational).

Note that the algorithm may require even fewer rounds if each agent uses the estimations specified in proposals made by other agents, because the lowest estimation maintained by one agent is still an overestimation and can be used by other agents to update their models. For example, if a proposed estimation of a team at_1 by agent a_1 is lower than the estimation maintained by agent a_2, then a_2 could improve its models by using a_1 proposed estimation.

3.3.4. *Optimal Solutions with Three Policies: P_{min}, P_{max} and P_{maj}*

We now introduce three policies, the minimum policy P_{min}, the maximum policy P_{max} and the majority policy P_{maj}. To define the first two policies, P_{min} and P_{max}, we will denote a set $V_{PROPOSALS_A}$ which are the estimations specified in *PROPOSALS*$_A$:

$$\hat{V}_{PROPOSALS_A} = \{\hat{V}_{a_i}(at_j) | \langle at_j, \hat{V}_{a_i}(at_j)\rangle \in PROPOSALS_A\}$$

Under complete group optimism, the minimum policy can be viewed as a conservative group decision policy as it selects the team proposed with the lowest estimation (but not an underestimation of the true performance of a team if each agent is optimistic).

Definition 13. ***Minimum Policy.*** *Formally, the minimum policy is defined by $P_{min}(PROPOSALS_A) = at_j$, where at_j is a random team selected from set $AT_{min} = \{at_{min} | \langle at_{min}, \hat{V}_{min}\rangle \in PROPOSALS_A$, where $\hat{V}_{min} = \operatorname{argmin}_{\hat{V}_i \in \hat{V}_{PROPOSALS_A}} (\hat{V}_i)\}$.*

The maximum policy always selects the team proposed with the highest estimated performance.

Definition 14. ***Maximum Policy.*** *Formally, the maximum policy is defined by $P_{max}(PROPOSALS_A) = at_j$, where at_j is a random team selected from set $AT_{max} = \{at_{max} | \langle at_{max}, \hat{V}_{max}\rangle \in PROPOSALS_A$, where $\hat{V}_{max} = \operatorname{argmax}_{\hat{V}_i \in \hat{V}_{PROPOSALS_A}} (\hat{V}_i)\}$.*

The third policy is the majority policy which selects the team that is preferred by the largest number of agents. As opposed to the minimum or maximum policy, the advantage of the majority policy is that it can prevent the selection of a team that has been proposed based on an unrealistically high estimation which is far removed from the true performance of a team. To define the policy P_{maj}, we use a set $AT_{PROPOSALS_A}$ which are the teams specified in *PROPOSALS*$_A$:

$$AT_{PROPOSALS_A} = \{at_j | \langle at_j, \hat{V}_{a_i}(at_j)\rangle \in PROPOSALS_A\}$$

Definition 15. ***Majority Policy.*** *Formally, the majority policy is defined by $P_{maj}(PROPOSALS_A) = at_j$, where at_j is the team selected from the set $AT_{maj} = \{at_{maj} | \operatorname{argmax}_{at_{maj} \in AT_{PROPOSALS_A}} |\{a_i : \langle at_{maj}, \hat{V}_{a_i}(at_{maj})\rangle \in PROPOSALS_A\}|\}$.*

We adopt a definition of the term majority which related literature often refers as "plurality rule", "first past the post" and "winner takes all". Related research also defines majority such that at least half the agents have selected a particular team.

Using Theorem 1 and any of the policies, we know that the algorithm will find an optimal solution under complete group optimism. Note that this corollary assumes the PROPOSE and UPDATE processes in Definition 7 and Definition 8.

Corollary 2. ***Optimality with P_{min}, P_{max} and P_{maj}.*** *Under the conditions **(A)–(C)** introduced for Theorem 1, we know that using P_{min}, P_{max} and P_{maj} will find an optimal solution, if the algorithm terminates.*

Proof. Condition **(C)** of Theorem 1 states that an optimal solution is found if there exists a proposal with an estimation not greater than $V(at_{BestSoFar})$. The selection of this team is independent of the group decision policy. □

Example 3. *Let us illustrate Corollary 2 by an example run of the algorithm using the policy P_{max} (depicted in Table 1). We consider a self contained group of agents $A = AT$, where three agents assign a task to each other. Formally, the group of agents is $A = \{a_1, a_2, a_3\}$ is also a set of agent teams $AT = \{at_1 = a_1, at_2 = a_2, at_3 = a_3\}$. The true performance (capability C) of each of these agents is represented by the values specified in the last row in each round to illustrate the difference to the updated estimations (Table 1).*

*In this example, Condition **(A)** is met, because estimations of the performance of a team are not initialised or updated with values lower than that of the true performance of the team, i.e., the group of agents is completely optimistic. For example, in round r_1, agent a_1 has an initialised estimation of its own performance (here at_1 since $at_1 = a_1$) that is far greater*

r_1	at_1	at_2	at_3	Proposals	Policy Selection
M_{a_1}	0.85	0.5	0.8	$\langle at_1, 0.85\rangle$	$P_{max} = at_2$
M_{a_2}	0.3	0.95	0.75	$\langle at_2, 0.95\rangle$	
M_{a_3}	0.35	0.55	0.9	$\langle at_3, 0.9\rangle$	
C	0.2	0.4	0.7		
$AT_{KnownSoFar}$		**0.4**		$(at_{BestSoFar} = at_2)$	
r_2	at_1	at_2	at_3		
M_{a_1}	0.85	0.6	0.8	$\langle at_1, 0.85\rangle$	$P_{max} = at_3$
M_{a_2}	0.3	0.5	0.75	$\langle at_3, 0.75\rangle$	
M_{a_3}	0.35	0.4	0.9	$\langle at_3, 0.9\rangle$	
C	0.2	0.4	0.7		
$AT_{KnownSoFar}$		0.4	**0.7**	$(at_{BestSoFar} = at_3)$	
(If an estimation in any proposal$_{r_2+i} \leq 0.7$, then condition **D** in Theorem 1 is true) (In this example, the next round consists of such a proposal)					
r_3	at_1	at_2	at_3		
M_{a_1}	0.85	0.6	0.8	$\langle at_1, 0.85\rangle$	
M_{a_2}	0.3	0.5	0.7	$\langle at_3, 0.7\rangle$	
M_{a_3}	0.35	0.4	0.8	$\langle at_3, 0.8\rangle$	
C	0.2	0.4	0.7		
$AT_{KnownSoFar}$		0.4	0.7	$(at_{BestSoFar} = at_3)$	

Table 1

An example run illustrating Theorem 1 with the policy P$_{max}$ (boldfaced value is the true performance of team $at_{BestSoFar}$ – when the algorithm terminates, the solution will specify this team)

than its true performance (0.85>0.2). Also, an updated estimation of a team remains at least as large as its true performance. For example, in round 2, agent a_1 updates its model of the previously selected team with values higher than the team's true performance at_2 ($\hat{V}_{a_1,r_1}(at_2) = 0.5$ in round r_1 and $\hat{V}_{a_1,r_2}(at_2) = 0.6$ in round r_2).

*According to Condition **(B)**, each agent proposes a team with an estimation higher than the true performance of $at_{BestSoFar}$ (for the proposals in the first round, there are no restrictions about which team can be proposed since no team has performed the task yet). For example, in round 2, agent a_3 proposes at_3 as the estimation of at_3's performance is the only one in a_3's models that is not smaller than the true performance of $at_{BestSoFar}$ (0.9>0.4, but 0.4=0.4 and 0.35<0.4).*

*After task execution in round 2, we know that the true performance of $at_{BestSoFar}$ is 0.7 which is also the best performing team known so far. As soon as an estimation in a proposal is not greater than 0.7, then Condition **(C)** is satisfied. In this example, in round r_3, an estimation is equal to that of the best team selected so far (at_3). As all Conditions **(A)–(C)** are satisfied after round 2, we know that if the algorithm terminates after round 2 (or any following round) we have found an optimal solution (as stated by Corollary 2).*

In summary, teams at_2 and at_3 have been assigned, but not at_1, thus we know that at_3 is an optimal solution before the algorithm assigns at_1. This example demonstrates how the algorithm is guaranteed to find an optimal solution, if it terminates in any round. That is, it is not depending on any termination criterion. Since the following theorems are proven under more specific assumptions regarding criteria and processes, they enable us to make stronger assertions about the computational requirement. Finally, the algorithm may run for 5, 10, or 20 rounds as we have assumed an optimistic update process that replaces an initial estimate with a value no smaller than the original estimate. A special case of this process is when the update process replaces the estimate with exactly the observed value.

3.4. Reaching Optimal Solutions under Partial Group Optimism

The theorems in the previous section relied on the assumption of complete group optimism (Definition 9). We now consider a *partially optimistic* group of agents, where agents also have underestimations of the true performance of teams. Under partial group optimism, the proof in Theorem 1 does not guarantee an optimal solution as a proposal that is not greater than $at_{BestSoFar}$ may be an underestimation of the performance of a team (and this does not satisfy the Conditions **(A)–(C)**, Theorem 1). To guarantee an optimal solution, we now have to more closely examine the role of the policies defined in Section 3.3.4.

3.4.1. Optimality under Minimal Group Optimism

An optimal solution can be found with the maximum policy if we know that at least one agent is optimistic about at least one optimal team.

Definition 16. ***Minimal Group Optimism.*** *We say that a group of agents is minimal optimistic if, and only if, at least one agent in that group is optimistic, so that $\exists a_i \in A, \forall at_j \in AT : \hat{V}_{a_i}(at_j) \geq V(at_j)$ at all times.*

Theorem 4. ***Maximum Policy: Sufficient Condition.*** *If*

- ***(A)*** *a group of agents A is minimal optimistic (Definition 16),*
- ***(B)*** *optimistic agents are also task rational proposers (Definition 12),*

(C) *we use the maximum group decision policy: P_{max} (Definition 14), and*

(D) *the algorithm does not terminate if $\exists\langle at_{selected}, \hat{V}(at_{selected})\rangle \in PROPOSALS_A$ such that $\hat{V}(at_{selected}) > V(at_{BestSoFar})$,*

then if the algorithm terminates it will terminate with an optimal solution.

Proof. The theorem is proven by contradiction, i.e., we assume that if the algorithm terminates, then we have found a suboptimal team $at_{BestSoFar}$.

1. If the team $at_{BestSoFar}$ is not an optimal team, then there must be a team at_*, such that $V(at_*) > V(at_{BestSoFar})$.
2. According to **(D)**, if the algorithm terminates, we know that $\forall\langle at_{selected}, \hat{V}(at_{selected})\rangle \in PROPOSALS_A$ such that $\hat{V}(at_{selected}) \leq V(at_{BestSoFar})$.
3. Let a' be the optimistic and task rational agent in A. According to **(C)**, if the algorithm terminates, we know that $\hat{V}(at_{selected}) > \hat{V}_{a'}(at_j), \hat{V}_{a'}(at_j) \in \hat{V}_{PROPOSALS_A}$, because at_j has not been selected by P_{max}.
4. According to **(B)**, if the algorithm terminates, we know that $V(at_{BestSoFar}) \geq \hat{V}_{a'}(at_j)$ for all $at_j \in AT$ and for a'.

According to 1., $V(at_*) > V(at_{BestSoFar})$, and according to 3. and 4. $V(at_{BestSoFar}) \geq \hat{V}_{a'}(at_*)$. So, we know that $V(at_*) > \hat{V}_{a'}(at_*)$. However, this is a contradiction, because we assumed that agent a' is an optimistic initialiser and initial updater with respect to all teams in AT (Condition **(A)**). □

3.4.2. Optimality under Majority Group Optimism

Definition 17. ***Majority Group Optimism.*** *We say that a group of agents is majority optimistic if, and only if, the majority of agents in that group are optimistic, so that $\hat{V}_{a_i}(at_j) \geq V(at_j)$ for the majority of $a_i \in A$ and for all $at_j \in AT$ at all times.*

A proof of a theorem stating optimality with a group of majority optimistic agents is a variation of the proof for Theorem 4. In brief, the majority policy will select the team that is overestimated by the majority of agents. By assumption, an optimal team is overestimated by the majority of agents. We know that teams that are overestimated will be selected until we find a team whose performance is greater than the estimations of other teams. The algorithm will not terminate until the majority of agents has estimations about an optimal team which are higher than estimations about other teams.

3.4.3. Policy: Independent Optimality under Minimal Group Optimism

The following theorem is an important variation of Theorem 1 as it does not assume a completely optimistic group of agents. Under minimal group optimism, we can still guarantee optimality with a less restrictive Condition **(C)** involving an assessment of the proposals made by the agents.

Theorem 5. ***Minimal Group Optimism: Sufficient Condition.*** *If*

(A) *a group of agents A is minimal optimistic (Definition 16),*[6]

(B) *optimistic agents are also task rational proposers (Definition 12), and*

(C) *the algorithm does not terminate if $\exists\langle at_j, \hat{V}(at_j)\rangle \in PROPOSALS_A$ such that $\hat{V}(at_j) > V(at_{BestSoFar})$,*

then if the algorithm terminates it will terminate with an optimal solution.

Proof. The theorem is proven by contradiction, i.e., we assume that if the algorithm terminates, then we have found a suboptimal team $at_{BestSoFar}$. Let $a' \in A$ be an agent that remains optimistic about the performance of an optimal team (according to **(A)**).

1. If the team $at_{BestSoFar}$ is not an optimal team, then there must be a team at_*, such that $V(at_*) > V(at_{BestSoFar})$.
2. According to **(C)**, if the algorithm terminates, we know that $\forall\langle at_j, \hat{V}(at_j)\rangle \in PROPOSALS_A$ such that $\hat{V}_{a_i}(at_j) \leq V(at_{BestSoFar})$ for all agents $a_i \in A$. In particular, we know that a' has made a proposal, where $\hat{V}_{a'}(at_j) \leq V(at_{BestSoFar})$.
3. According to **(B)**, if the algorithm terminates, we know that $V(at_{BestSoFar}) \geq \hat{V}_{a_i}(at_j)$ for all $at_j \in AT$ and for all $a_i \in A$. In particular, we know that $V(at_{BestSoFar}) \geq \hat{V}_{a'}(at_j)$.

According to 1., $V(at_*) > V(at_{BestSoFar})$, and according to 3. $V(at_{BestSoFar}) \geq \hat{V}_{a'}(at_*)$. So, we know that $V(at_*) > \hat{V}_{a'}(at_*)$. However, this is a contradiction, because we assumed that agent a' is an optimistic initialiser and initial updater with respect to all teams in AT (Condition **(A)**). □

[6]The minimum condition for this theorem is that there exists one agent that is optimistic regarding the performance of one optimal team. However, this premise is difficult to prove in many realistic problem domains.

Another important variation of Theorem 1 is one where we keep the original Condition **C** (i.e., the algorithm does not terminate if all proposals are greater than $V(at_{\mathit{BestSoFar}})$) and then assume that each agent is a maximal task rational proposer (Definition 12). Note that in this case, the insights obtained with respect to the computational requirement (Theorems 2 and 3) also apply. We do not prove this statement here as it is similar to the proof provided for Theorems 1 and 5.

3.4.4. Determining Precise Computational Requirement under Partial Optimism

Under partial optimism, Termination Criterion 3 is not guaranteed to terminate the algorithm with an optimal solution. A proposed estimation of an optimal team could be lower than its true performance, and also lower than the team $at_{\mathit{BestSoFar}}$ and this would terminate the algorithm with a suboptimal solution. This reasoning applies to Termination Criterion 2 which could also terminate with a suboptimal solution. Only Termination Criterion 1 ensures that the algorithm does not terminate with an underestimation. Hence, it is assumed for the theorems in this section. Note also that we assume that agents are accurate updaters (Definition 11) and maximal task rational proposers (Definition 12).

The following theorem defines conditions which determine the precise number of rounds until the algorithm converges to an optimal solution with P_{max}. Informally, if there exists an agent that estimates the performance of m teams (of n teams) higher than the true performance of an optimal team (but whose performance is actually lower than that of an optimal team), and if the performance of an optimal team is higher than the estimated performance of the other $n - m$ teams, then an optimal solution is found in exactly m rounds. Let at' be an optimal team. Let $AT_* \subseteq AT$ be a set of teams such that $\forall at_j \in AT_*, \exists a_i \in A$ with $\hat{V}_{a_i}(at_j) \geq V(at')$, but where $V(at_j) < V(at')$.

Theorem 6. ***Maximum Policy: Assignment Rounds.*** *If* $\max_{a_k \in A} \hat{V}_{a_k}(at') \geq V(at')$ *and* $V(at') \geq \max_{a_l \in A, at'' \in |AT - AT_*|} \hat{V}_{a_l}(at'')$, *then* P_{max} *is guaranteed to find an optimal team* at' *in* $|AT_*|$ *rounds.*

Proof. P_{max} selects the team with the highest proposed performance, and we know that $|AT_*|$ teams have higher estimations than the true and estimated performance of an optimal team, but do actually perform a task worse than an optimal team. Each team in AT_* is selected before an optimal team, since $\hat{V}_{a_i}(at_j) > V(at_j), \forall at_j \in AT_*$. Since these teams have a true performance that is lower than the estimated performance of an optimal team, each of them will be selected once and then not be selected again. The other teams $AT_* - AT$ will never be selected as their estimations are lower than that of an optimal team and the teams in AT_*. Thus, exactly $|AT_*|$ teams perform the task, after which the algorithm terminates with an optimal team. □

The majority policy will also find an optimal solution.

Theorem 7. ***Majority Policy: Assignment Rounds.*** *If the majority of agents estimate the performance of m teams (of all n teams) higher than the performance of an optimal team, and if the majority of agents estimates the performance of an optimal team higher than its true performance, then P_{maj} finds an optimal solution in m rounds.*

Theorems 6 and 7 imply that in the best case an optimal solution can be found in only one round. The theorems also imply that an optimal solution can be found before all teams have been assigned once (i.e., the algorithm does not require more rounds than the exhaustive procedure). For the maximum policy, we can derive the following corollaries (similar corollaries can easily be derived for the majority policy).

Corollary 3. ***Minimum Number of Assignment Rounds.*** *If the performance of an optimal team is estimated to be higher than the performance of the other teams, and if the other estimations are lower than the true performance of an optimal team, then we find an optimal solution in one round with the maximum policy P_{max}.*

We can also show that the maximum number of rounds is not greater than the number of teams, which means that the maximum number of rounds for the algorithm is not greater than those required when trying each team at least once.

Corollary 4. ***Maximum Number of Assignment Rounds.*** *For n teams, if we have at least one agent that is optimistic of the performance of an optimal team, and if the estimations about the other $n - 1$ teams are higher than the true and estimated performance of an optimal team, then we will find an optimal solution in n rounds with the maximum policy P_{max}.*

COMPLETE GROUP OPTIMISM	Optimality	Computational requirement
Complete Group Optimism (Definition 9) $\forall a_i \in A$ Task Rational Proposer (Definition 10)		
Theorem 1 the algorithm does not terminate if $\neg\exists\langle at_j, \hat{V}(at_j)\rangle \in \mathit{PROPOSALS}_A, \hat{V}(at_j) \leq V(at_{\mathit{BestSoFar}})$	**YES**	Not Determined
Corollary 1 Minimum Policy P_{min} (Definition 13) Maximum Policy P_{max} (Definition 14) Majority Policy P_{maj} (Definition 15)	**YES**	Not Determined
Lemma 1 TC 1: $\forall a_i \in A : \mathit{proposal}_{a_i} = \langle at_{\mathit{selected}}, \hat{V}_{a_i}(at_{\mathit{selected}})\rangle$ such that $\hat{V}_{a_i}(at_{\mathit{selected}}) \leq V(at_{\mathit{BestSoFar}})$	**YES**	$\boldsymbol{TC1 \leq \lvert AT\rvert}$
Lemma 2 TC 2: $\exists a_i \in A : \mathit{proposal}_{a_i} = \langle at_{\mathit{selected}}, \hat{V}_{a_i}(at_{\mathit{selected}})\rangle$ such that $\hat{V}_{a_i}(at_{\mathit{selected}}) \leq V(at_{\mathit{BestSoFar}})$	**YES**	$\boldsymbol{TC2 \leq \lvert AT\rvert}$
Lemma 3 TC 3: $\exists a_i \in A : \mathit{proposal}_{a_i} = \langle at_j, \hat{V}_{a_i}(at_j)\rangle$ such that $\hat{V}_{a_i}(at_j) \leq V(at_{\mathit{BestSoFar}})$	**YES**	$\boldsymbol{TC3 \leq \lvert AT\rvert}$
$\forall a_i \in A, \forall at_j \in AT$ Accurate Updater (Definition 11) $\forall a_i \in A$ Maximal Task Rational Proposer (Definition 12)		
Theorem 2	**YES**	$\boldsymbol{TC2 \leq TC1 \leq \lvert AT\rvert}$
Theorem 3	**YES**	$\boldsymbol{TC3 \leq TC2 \leq \lvert AT\rvert}$
Corollary 2	**YES**	$\boldsymbol{TC3 \leq TC2 \leq TC1 \leq \lvert AT\rvert}$

Table 2

Optimality and computational requirement of the algorithm under *complete group optimism*.

If we know that agents are optimistic in a given domain, we can use the assignment algorithm to exploit the above property in reducing the number of rounds required to find an optimal solution. That is, Theorems 6 and 7 enable us to make statements about the computational requirements of our algorithm under partial optimism. For example, the theorems show that if we have prior knowledge of the agents' estimations, then we can determine an acceptable upper bound on the number of reassignments required. Another useful insight is that if we know that the majority of agents are not optimistic about an optimal team (but we have at least one agent that is optimistic about an optimal team), then we are guaranteed to find an optimal team with P_{max}.

Note that the computational requirement of the algorithm can be reduced if we know that one agent is optimistic. The algorithm can be terminated more efficiently with Termination Criterion 3. The proof provided in Section 3.3.3 can be adopted to prove this statement.

This paper does not offer proofs for variations of previous theorems, but instead we outline a few ideas. For example, we can order policies according to their computational requirement under partial op-

PARTIAL GROUP OPTIMISM	Optimality	Computational requirement
Minimum Group Optimism (Definition 16)		
Theorem 4 Maximum Policy P_{max} the algorithm does not terminate if $\neg\exists\langle at_j, \hat{V}(at_j)\rangle \in \mathit{PROPOSALS}_A, \hat{V}(at_j) \leq V(at_{\mathit{BestSoFar}})$	**YES**	Not Determined
Theorem 5 the algorithm does not terminate if $\exists\langle at_j, \hat{V}(at_j)\rangle \in \mathit{PROPOSALS}_A, \hat{V}(at_j) > V(at_{\mathit{BestSoFar}})$	**YES**	Not Determined
Theorem 6 $\forall at_j \in AT_*, \exists a_i \in A$: $\hat{V}_{a_i}(at_j) \geq V(at')$, where at' is an optimal team	**YES**	$\boldsymbol{\|AT_*\|}$
Theorem 7 $\exists$ majority $\|AT_*\|$ overestimations	**YES**	$\boldsymbol{\|AT_*\|}$
Corollary 3	**YES**	$\boldsymbol{\|AT_*\| = 1}$
Corollary 4	**YES**	$\boldsymbol{\|AT_*\| = \|AT\|}$
Theorem 8 Group pessimism (Definition 19)	**NO**	Not Determined

Table 3

Optimality and computational requirement of the algorithm under *partial group optimism*.

timism which requires a stricter definition of Definition 16.

Definition 18. ***Strict Minimal Group Optimism.*** *We say that a group of agents is strictly minimal optimistic if, and only if, one agent a' in that group is optimistic and the other agents are pessimistic, so that $a' \in A, \forall at_j \in AT : \hat{V}_{a'}(at_j) \geq V(at_j)$ and $a' \neq a_i \in A, \forall at_j \in AT : \hat{V}_{a_i}(at_j) < V(at_j)$ at all times.*

Under strict minimal group optimism, the minimum policy is no more efficient than the maximum policy. A proof to this assertion is a variant of proofs in the previous section.

3.4.5. Suboptimal Solutions under Group Pessimism

In the previous section, we focused on guaranteeing to find an optimal team with optimistic estimations. However, it is also important to identify conditions under which an optimal team can not be found.

If all agents have estimations of optimal teams that are lower than the true performance of other teams, then it is called a pessimistic group.

Definition 19. ***Group Pessimism.*** *A group of agents A is pessimistic if, and only if, each agent in this group initialises its estimation of all optimal teams lower than that of other teams, so that $\hat{V}_{a_i}(at_j) < V(at_j)$ for all $a_i \in A$ and for all $at_j \in AT$ at all times.*

The following theorem assumes group pessimism and task rationality, thus preventing the algorithm from finding an optimal solution.

Theorem 8. *If a group of agents is pessimistic, and if each agent is maximal task rational (Definition 12), then an optimal solution will never be found (regardless of the group decision policy used).*

Proof. Since an optimal team at_* is always estimated to be lower than other teams by all agents, then an optimal team will never be proposed in any round. Recall that we defined a group decision policy as one that selects a team from those that have been proposed, and therefore an optimal team will never be selected, because it has not been proposed. Thus, an optimal solution will never be found. □

If we know that a group of agents is pessimistic and each agent is task rational, then a different policy may be more appropriate, for example, a policy that selects a random team from teams that have not been proposed.

Tables 2 and 3 provide an overview of the theoretical results.

4. Empirical Study

This section presents an empirical simulation-based study investigating the influence of model accuracy and group decision policies on the efficiency of the TAP algorithm. This study is illustrated by a model of a domain involving surf rescues, where a group of rescue teams endeavours to allocate the best rescue team to a rescue task (Section 4.1). Section 4.4 defines an experiment that examines the role of agents' initial estimations of the rescue performance of rescue teams and the role of the maximum, minimum and majority group decision policy in the assignment procedure.

4.1. Illustration of Study: A Model of a Surf Rescue Domain

RoboCup Rescue is a research initiative that offers many practical domains in which autonomous systems perform and coordinate tasks in disaster situations, such as bushfires and earthquakes [28, 26, 29]. As part of this initiative, [26] have been the first to propose a surf rescue domain involving rescues coordinated by autonomous robots on remote beaches. The main research interest of [26] is on building autonomous rescue teams endowed with specific hardware components. Such components cope with the harsh environmental conditions in the surf, e.g., coping with noisy sensory input and meeting high power demands. Our interest is in using this domain as a platform to illustrate the main components of a CIA problem.

4.2. Formalising the Main Components in a Surf Rescue Domain using our Framework

A Surf Rescue (SR) domain involves a panel of senior lifesavers that endeavours to allocate an optimal rescue team to the task of rescuing a distressed swimmer. Each rescue requires a team of individual lifesavers that perform different roles (and this paper assumes that each team is estimated as a unique object). Each panel member has an initial notion of how a team would perform the rescue, and the panel can refine rescue allocations based on their observations of how well different teams perform a rescue.

A SR domain is represented using the formal definitions of our framework presented in Section 2.

- *Task* domain. We consider a specific rescue task denoted by t=*rescue* (Definition 1). The execution of a rescue task does not depend on first executing other tasks (because we assume task independency). The panel will re-allocate a team to the rescue on a regular basis.
- *Teams*. Each rescue team consists of lifesavers performing different roles during a rescue. We have a finite number of rescue teams that are considered for performing a rescue task.
- *Agents*. Agents are represented as a panel of senior lifesavers and each agent (or panel member) actively participates in making an allocation of a team to the rescue task.
- *Policy*. The policy considers the proposals submitted by individual panel members to select one of the proposed rescue teams.

Our focus is on finding an optimal team after assigning and testing as few teams for the rescue task as possible.

Each agent maintains models of the rescue performance of the teams and each agent has processes to use these models (Definitions 4 and 5). In particular, each agent uses a value function that represents how well a team rescues the swimmer (e.g., the faster the distressed swimmer is retrieved, the higher the value returned by this function). The values of this function range from 0 to 1, where 0 represents that a team is not able to perform a rescue and 1 represents an optimal rescue performance. For example, agent a_1 has a model of the rescue performance of team at_3, i.e., a_1's model of at_3's rescue performance: $M_{a_1}(at_3) = \{\hat{V}_{a_1}(at_3, \textit{rescue}) = 0.6\}$ (Definition 4).[7] To illustrate the difference between the

[7]Each agent may have preconceived notions of the performance of the rescue teams and we assume that agents derive these initial estimations from prior experience. Since these estimations are idiosyncratic, building a model about where they come from is an open research issue [6] and is not addressed in this paper.

models maintained by agents and the capabilities of rescue teams, we specify each team's true rescue performance by its capability C, e.g., for at_3: $C(at_3) = \{V(at_3, \textit{rescue}) = 0.3\}$. According to these two values, a_1 overestimates at_3's rescue performance $(0.6 > 0.3)$.

Each agent a_i has three reasoning processes that use the estimations stored in its models (Definition 5): INITIALISE, PROPOSE and UPDATE. Each agent proposes a team that has the highest estimated rescue performance according to the agent's models (i.e., each agent is a maximal task rational proposer, Definition 12). Each agent updates its models accurately when it observes the rescue performance of a team (i.e., each agent is an accurate updater, Definition 11).

The process of *announcing the task* and *applying the policy* are steps in our algorithm, but they are not central to the analysis of our experiments. We explain them briefly for clarity of exposition. The ANNOUNCE process is executed once to all agents as the first step in the assignment procedure (e.g., at the beginning of a season, the senior lifesavers identify the rescue as being a central part of patrolling surf beaches). For the second process, a designated "chairman" of the panel $a_{\textit{policy}}$ will apply a group decision policy P and will collect proposals by senior lifesavers throughout the assignment rounds. As discussed in Section 2.2.1, this paper does not assume specific features of the entities that execute these processes.

The four components, Tasks – T, Agent Teams – AT (i.e., rescue teams), Agents – A (i.e., panel of senior lifesavers), Policy – P, are used as input by the assignment algorithm as described in the next section.

4.3. *Sample Run of the Assignment Algorithm in the Surf Rescue Domain*

This section describes a sample run of the assignment algorithm to find a rescue team for a rescue (based on the algorithm depicted in Figure 1). This sample run has the following properties.

– Policy $P_{\textit{max}}$ (Definition 14) is used to select the rescue team with the highest proposed rescue performance.
– Termination Criterion 1 (page 8) is used to terminate the algorithm (i.e., if all estimations of the selected team contained in proposals are not greater than the true performance of the best rescue team found so far, and the policy is restricted to not select the same rescue team more than once).
– Interactions are confined between agents a_1 and a_2 (for clarity of exposition). That is, only agents a_1 and a_2 maintain models of rescue teams at_3 and at_4 ($M_{a_1}(at_3), M_{a_1}(at_4), M_{a_2}(at_3)$ and $M_{a_2}(at_4)$), propose a rescue team and update models of selected rescue teams. Only rescue teams at_3 and at_4 can be assigned to the rescue.

The sample run is depicted in Table 4.

– Column 1 indicates the step of the algorithm.
– Column 2 indicates the status of the group.
– Columns 3–4 show the processes executed by agents a_1 and a_2 at a particular step of the algorithm.
– Columns 5–6 contain values of the true performance of the rescue teams. The true performance of the rescue teams selected for a rescue is boldfaced.
– Columns 7–10 contain the models maintained by agents (the estimated performance of rescue teams proposed by agent a_1 is highlighted in lightgrey and rescue teams proposed by agent a_2 is highlighted in darkgrey).

At the beginning of the run, the rescue task is announced to all agents, step 1 in Table 4. Upon receiving the task announcement, the agents a_1 and a_2 initialise their models of the rescue performance of the rescue teams at_3 and at_4. As a result of initialising their models, agent a_1 has an estimation of each rescue team's rescue performance (0.6 for at_3 and 0.1 for at_4) as does agent a_2 (0.4 for at_3 and 0.5 for at_4) . To illustrate the interaction of each agent's models with the true rescue performance of rescue teams, we specify the true rescue performance of the rescue teams at_3 (0.3) and at_4 (0.6). As seen from step 2, the estimated performance in each agent's models is not consistent with the true performance of the rescue teams in question. For example, agent a_1 overestimates the performance of rescue team at_3 (0.6>0.3), but underestimates the performance of rescue team at_4 (0.1<0.6). Agent a_2 also overestimates the performance of rescue team at_3 (0.4>0.3), and underestimates the performance of rescue team at_4 (0.5<0.6).

In the first allocation round, the rescue task is assigned as follows. Agents a_1 and a_2 propose the best rescue team according to the estimations stored in their models (step 3). In particular, a_1's proposal is $\textit{proposal}_{a_1} = \langle at_3, 0.6\rangle$, and a_2's proposal is $\textit{proposal}_{a_2} = \langle at_4, 0.5\rangle$. In this example, agent a_2 is

				True Performance		*Estimated Performance*			
Step of Algorithm	**Group Status**	**Process executed by a_1**	**Process executed by a_2**	$C(at_3)$	$C(at_4)$	$M_{a_1}(at_3)$	$M_{a_1}(at_4)$	$M_{a_2}(at_3)$	$M_{a_2}(at_4)$
1	announcing	ANNOUNCE(*rescue*)		0.3	0.6				
2	initialising	INITIALISE(M_{a_1}, *rescue*)	INITIALISE(M_{a_2}, *rescue*)	0.3	0.6	0.6	0.1	0.4	0.5
	Round 1								
3	proposing	PROPOSE=$\langle at_3, 0.6\rangle$	PROPOSE=$\langle at_4, 0.5\rangle$	0.3	0.6	0.6	0.1	0.4	0.5
4	selecting		$P_{max} = at_3$	**0.3**	0.6	0.6	0.1	0.4	0.5
5	updating	UPDATE (M_{a_1},0.3)	UPDATE (M_{a_2},0.3)	0.3	0.6	0.3	0.1	0.3	0.5
	Round 2								
6	proposing	PROPOSE=$\langle at_3, 0.3\rangle$	PROPOSE=$\langle at_4, 0.5\rangle$	0.3	0.6	0.3	0.1	0.3	0.5
7	selecting		$P_{max} = at_4$	0.3	**0.6**	0.3	0.1	0.3	0.5
8	updating	UPDATE (M_{a_1},0.6)	UPDATE (M_{a_2},0.6)	0.3	0.6	0.3	0.6	0.3	0.6
9	terminate	(Criterion is satisfied)		0.3	**0.6**	0.3	0.6	0.3	0.6

Table 4

A sample run of the assignment algorithm in the Surf Rescue domain with two assignment rounds

agent a_{policy} applying the maximum policy P_{max} to select a rescue team for the rescue task. In this round, rescue team at_3 is selected and specified in proposal$_{a_1}$ (step 4 in Table 4). Rescue team at_3 does not rescue the swimmer as well as a_1 had estimated. After observing the true performance of at_3, which is 0.3, both agents update their models of at_3 (step 5).

In the second allocation round, a_1 still proposes at_3, and a_2 still proposes at_4 (step 6). However, this time, the maximum policy selects at_4 (step 7), since its proposed performance (0.5) is now higher than the proposed performance of at_3 (0.3). After observing the true performance of at_4 (0.6), both agents update their models $M_{a_1}(at_4)$ and $M_{a_2}(at_4)$ (step 8). Upon completion of this step, the models maintained by a_1 and a_2 are identical and both agents now believe that at_4 is the optimal rescue team.

In step 9, the termination criterion of the algorithm is satisfied and the algorithm terminates (we use Termination Criterion 1, page 8). The reason it terminates is that the policy selects a rescue team whose estimated performance specified in proposals from both agents is not greater than the true performance of the best rescue team that is known so far (which, in this example, is rescue team at_4). Note that models are not always identical to the capabilities of teams when the algorithm terminates – they are only identical if all rescue teams have been selected at least once (as occurred in this sample run).

4.4. *Setting up the First Experimental Series: Interaction of Initial Models and the Minimum, Maximum and Majority Policy*

This section addresses the question: How does the assignment algorithm perform under empirical domain conditions? This study examines the role of model accuracy (i.e., how accurate each agent knows the performance of each rescue team) and three group decision policies: the minimum, maximum and majority policy. We are interested in examining the following aspects.

4.4.1. *Effect of Model Accuracy on Solution Quality and Computational Requirement*

In Section 3, we have proven that an important condition to guarantee an optimal solution is that agents are completely or partially optimistic. To further extend on this insight, we test under which combinations of initial estimations and group decision policies our algorithm is likely to find optimal or near optimal solutions. We hypothesise that more accurate models increase the likelihood of finding optimal or near optimal solutions in fewer assignment rounds. Further, we also hypothesise that agents with optimistic estimations are more likely to find optimal or near optimal solutions than agents that with pessimistic estimations.

4.4.2. *Comparison to Optimal and Random Assignments*

How do the results obtained from the assignment algorithm compare to those obtained from benchmark settings? To evaluate the efficiency of the algorithm, we consider two benchmark allocation procedures (neither use the TAP algorithm and are further defined in Section 4.6).

- An exhaustive procedure is guaranteed to find an optimal solution, but requires knowledge of the true performance of each rescue team which is achieved by assigning each rescue team to the task once.
- A random procedure selects a random rescue team without any assignments and without knowledge of the performance of any rescue team. It only finds an optimal solution sometimes depending on the number of optimal rescue teams and the total number of rescue team (no assignments are required as no knowledge is required of a rescue team's performance).

4.4.3. Convergence Characteristics of Group Decision Policies

How fast does the algorithm converge to a solution? Our interest is in determining when a group decision policy enables the algorithm to converge to a solution faster. This knowledge enables us to select one policy over the others if the number or rounds is limited or not accurately known (that is, the algorithm would terminate regardless of when the termination criterion is satisfied).

The assumptions introduced for our theoretical study (Section 3.1) offer a starting point for a first empirical experiment. That is, our experimental study makes three explicit assumptions about the PROPOSE and UPDATE processes executed by the agents and the rescue performance of each rescue team.

- Agents are maximal task rational (each agent proposes the rescue team with the highest estimated performance according to its models), Definition 12.
- Each agent updates its model with the observed performance of the rescue team that has performed a task, Definition 11.
- The rescue performance of each rescue team is invariant (the rescue performance of a given team will not vary when it performs various rescue), Section 3.1.

4.5. Experimental parameters

To investigate the above issues empirically, we implemented the assignment algorithm (specified in Figure 1) and the components defined in the formal framework (Section 2). We then evaluated the algorithm by varying four experimental parameters: Capability Initialisation CI of rescue teams (a distribution representing the true rescue performance of rescue teams), Model Initialisations MI of agents (a distribution representing estimated rescue performance of rescue teams), group decision Policy P, and the Group Size GS. These experimental parameters are assigned a range of values in the simulations.

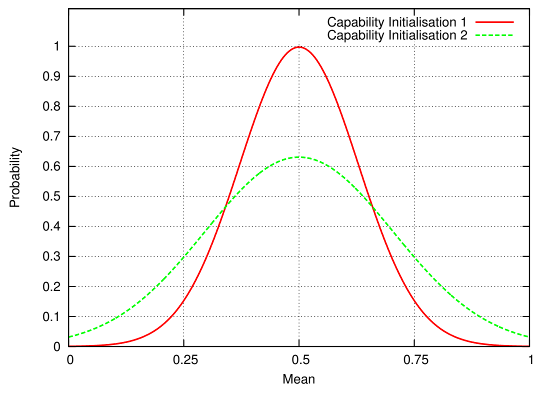


Fig. 2. A sketch of two truncated normal distributions used to set the experimental parameter Capability Initialisation CI.

- **Capability Initialisation (CI)** defines two types of distributions that initialise the values V for the true performance of rescue teams. These distributions are normal and truncated (Figure 2).
  - ∗ *Capability Initialisation 1*: $N(CI_\mu = 0.5, CI_\sigma = 0.1)$
  - ∗ *Capability Initialisation 2*: $N(CI_\mu = 0.5, CI_\sigma = 0.25)$

Since the interval of performance values ranges from 0 (worst performance) to 1 (optimal performance), the mean for both distributions is 0.5. Other values can be used to simulate the performance of other types of rescue teams. For example, 0.25 may be an appropriate mean of a distribution representing low performing rescue teams and 0.75 for high performing rescue teams (or 0.1 for extremely low performing rescue teams, or 0.9 for extremely high performing rescue teams). A standard deviation of 0.1 represents a rescue team population where performance values are distributed closer around the mean while a standard deviation of 0.25 represents a wider scattering (Figure 2). The specification of the mean and standard deviation will depend on which type of rescue team is considered in the simulation.

Truncation is required so that the performance does not exceed the [0, 1] thresholds. We have chosen to truncate the normal distribution by setting an outlier to its closest threshold (an outlier is a value that lies outside the [0, 1] threshold after is has been drawn from the normal distribution). For example, assume that we have a value 1.2 that is drawn from a capability distribution, e.g., when CI=2. For CI=2, this value will be set to 1.0 – the upper threshold. This truncation method maintains a normal distribution that represents a population with extremely low or high performance rescue teams. Section 4.10 discusses methods that may be more adequate. s.

Next, we discuss the parameter that initialises the models maintained by agents. As for the experimental parameter CI, models are initialised based on a normal truncated distribution with a mean $MI_{\hat{\mu}}$ and standard deviation $MI_{\hat{\sigma}}$.

There are various methods to simulate model accuracy and define values for $MI_{\hat{\mu}}$ and $MI_{\hat{\sigma}}$. A naive method is simply to draw random estimates from the same distribution as used for the parameter CI, and then add or subtract values to simulate optimism and pessimism. The problem with this initialisation is that the estimates are not related to the true performance of rescue teams. Hence, there is no reason to expect that our algorithm is more likely to choose an optimal rescue team than a random selection. Such a naive method of initialising estimations can be considered a worst case setting for our investigations as it assumes that agents are *ignorant* of a team's true performance.[8] In other cases, agents have estimations that are related to the capabilities of rescue teams, that is, agents are knowledgeable about the performance of teams. For this study, we aim to simulate model initialisation along a dimension where agents exhibit a certain degree of ignorance.

We have chosen to specify the distribution of models using four values: group knowledge and group optimism specify the mean $MI_{\hat{\mu}}$ of the distribution, and spread and homogeneity specify the standard deviation $MI_{\hat{\sigma}}$. This study focuses on the effect of group knowledge and group optimism which are specified in the next two paragraphs.

[8]In this paper, the notion of ignorance is qualified as "not knowing accurately" as opposed to "complete absence of knowledge". This notion follows from research by [3, 35] who suggested three different types of “ignorance”: incompleteness (e.g., absence of facts), imprecision (e.g., ambiguous information) and uncertainty (e.g., uncertain information).

The degree of group knowledge of agents is specified by a value K indicating the closeness of $MI_{\hat{\mu}}$ to the true performance of an agent. When K is 1, agents are knowledgeable and the distribution mean $MI_{\hat{\mu}}$ is based on the true performance $V(at_i)$ of a rescue team at_i (recall that the true performance $V(at_i)$ is drawn from the distribution specified by the experimental parameter Capability Initialisation CI). For example, if K=1, then $MI_{\hat{\mu}}(at_i) = V(at_i)$, if we apply no optimistic or pessimistic modifications (as explained in the next section). When K is 0, all agents are (equally) ignorant of the individual performance of each rescue team. In particular, the mean of each estimation is based on the mean of the capability distribution of rescue teams (e.g., $MI_{\hat{\mu}}(at_i) = CI_{\mu} = 0.5$). This ignorance may be present when agents assemble in an ad-hoc manner and initial estimates are not related to the true performance of each rescue team. Any value of K between 0 and 1 represents a distribution with a mean that is close to the true performance of the rescue team in question. That is, agents have some estimates of the performance of rescue teams strongly related to the true value and other estimates which are far removed from the true value.

To empirically evaluate the implications of optimistic and pessimistic agents (based on our theoretical study in Section 3), we define a value Δ that modifies the mean of the model distribution. Since each agent is unlikely to have exactly the same optimistic and pessimistic estimations of rescue teams, Δ is also drawn from a normal distribution. That is, each rescue team will under- or overestimate the performance of each rescue team to a different extent.

We also define a standard deviation $MI_{\hat{\sigma}}$ that determines the divergence of the estimations from the mean. The value $MI_{\hat{\sigma}}$ is also drawn from a normal distribution, where the mean represents the spread of estimations, while the standard deviation represents the homogeneity of estimations among different agents. As the spread increases the divergence of estimations increases. A large degree of homogeneity means that the spread among different agents is similar. This study confines $MI_{\hat{\sigma}}$ to two settings where agents have a low and a high spread with moderate homogeneity.

– **Model Initialisation (MI)** defines a distribution to simulate the estimations maintained by agents. The values $\hat{V}$ for the estimated performance of each rescue team are drawn from a truncated and normal distribution with a mean $MI_{\hat{\mu}}$ and standard deviation $MI_{\hat{\sigma}}$, such that $\hat{V}(at_i) \sim$

$N_{MI}(MI_{\hat{\mu}}(at_i), MI_{\hat{\sigma}})$ for the estimate of a rescue team at_i (we apply the same method for truncation as for the experimental parameter CI).
The mean of models $\mathbf{MI}_{\hat{\mu}}$ of a rescue team at_i is defined as follows.

$$MI_{\hat{\mu}}(at_i) = (K * V(at_i) + (1 - K) * CI_{\mu}) + \Delta$$

- K ranges from 0 to 1 and defines the degree of closeness of the mean of the estimations to the true mean of individual rescue teams. If K is 1 then the estimated mean $MI_{\hat{\mu}}$ is based on the mean of the true performance of an individual rescue teams (agents are informed). If K is 0 then the estimated mean is based on the population mean (and agents are equally ignorant of the performance of individual rescue teams). In this study, we will set K to 0, 0.5 and 1.
- $\mathbf{\Delta}$ is a value drawn from one of the following five normal distributions. Each distribution specifies a different degree of over- and underestimations: *Very Pessimistic, Pessimistic, Neutral, Optimistic* and *Very Optimistic*.
  - ∗ *Very Pessimistic*: $\Delta \sim N_{\Delta}(-0.25, 0.1)$
  - ∗ *Pessimistic*: $\Delta \sim N_{\Delta}(-0.1, 0.1)$
  - ∗ *Neutral*: $\Delta \sim N_{\Delta}(0.0, 0.0)$
  - ∗ *Optimistic*: $\Delta \sim N_{\Delta}(0.1, 0.1)$
  - ∗ *Very Optimistic*: $\Delta \sim N_{\Delta}(0.25, 0.1)$

The value for the standard deviation of model initialisation $\mathbf{MI}_{\hat{\sigma}}$ is drawn from two normal distributions representing estimations with *Low* and *High-Spread* (both distribution have a moderate degree of homogeneity with 0.1).

∗ *Low-Spread*: $MI_{\hat{\sigma}} \sim N_{MI_{\hat{\sigma}}}(0.1, 0.1)$
∗ *High-Spread*: $MI_{\hat{\sigma}} \sim N_{MI_{\hat{\sigma}}}(0.25, 0.1)$

To illustrate the parameter MI, consider a situation in which agents are informed and optimistic. Further, agents have estimations that are closely spread around the mean performance of rescue teams and have a moderate level of homogeneity among agents. We can simulate this situation by setting K to 1, draw the value for Δ from $N_{\Delta}(0.1, 0.1)$ (optimistic) and draw $MI_{\hat{\sigma}}(at_1)$ from $N_{MI_{\hat{\sigma}}}(0.1, 0.1)$ (low-spread). Assume Δ is 0.14 and $MI_{\hat{\sigma}}$ is 0.08. Assume we have a rescue team at_1 with $V(at_1) = 0.4$. Under these settings, $MI_{\hat{\mu}}(at_1) = (1 * 0.4) + (0 * 0.5) + 0.14 = 0.54$. Hence, an estimation drawn from the distribution N_{MI} has a mean $MI_{\hat{\mu}}(at_1)$ of 0.54 and $MI_{\hat{\sigma}}(at_1)$ of 0.08. In this example, the value $\hat{V}(at_1)$ is 0.58 and overestimates the true performance of at_1 by 0.18 (0.58-0.4).

Note that we do not investigate an experimental setting where the mean $MI_{\hat{\mu}}(at_i)$ is equal to $V(at_i)$ and the standard deviation $MI_{\hat{\sigma}}$ is 0 as this represents a trivial case where all agents have accurate models (i.e., the algorithm will always find an optimal solution after one assignment round).

We have chosen the following group decision policies for the experimental study, because under given theoretical conditions, they are guaranteed to find optimal solutions (Sections 3.3 and 3.4).

– **Policy (P)** defines three types of group decision policies: *minimum, maximum* and *majority*.
  - ∗ The *minimum* policy P_{min} selects the rescue team with the lowest proposed performance of all rescue teams proposed in a given round (Definition 13, page 11).[9]
  - ∗ The *maximum* policy P_{max} selects the rescue team with the highest proposed performance of all rescue teams proposed in a given round (Definition 14, page 11).
  - ∗ The *majority* policy P_{maj} selects the rescue team proposed by the greatest number of agents (Definition 15, page 11).

Different group sizes are expected to show a trend of our results with different settings of the experimental parameters CI, MI and P. For example, as group size becomes larger, we expect that solution quality reaches a higher, more stable level due to an increased chance that more agents have more accurate models overall.

– **Group Size (GS)** defines the number of agents in a surf rescue domain: *5, 10, 20, 40* and *50*. GS also specifies the number of rescue teams in this simulation. So, we have the same number of agents and rescue teams in every simulation.

The study in this paper simulates a self-contained group where the number of agents and teams is the same ($AT = A$). We plan to alter the number of agents A and AT and will investigate if this has a significant influence on the efficiency of the algorithm.

[9]This policy selects a conservative estimate and is expected to be particularly efficient in settings where agents are optimistic as the selected estimate is still likely to find an optimal or near optimal solution, but with fewer rounds than the other two policies.

4.6. Benchmark Settings

To evaluate the outcome of TAP settings, we constructed two benchmark settings called EXHAUSTIVE and RANDOM. As opposed to TAP settings, these benchmark settings do not use the TAP assignment algorithm (i.e., agents do not make proposals, update models, or make a group decision to assign a rescue team to a task).

– An EXHAUSTIVE setting uses a procedure that finds an optimal rescue team for the rescue after obtaining accurate knowledge of each rescue team's rescue performance. This requires n assignment rounds for n rescue teams (as the performance of these rescue teams will only be known accurately after each rescue team has performed a rescue). The average result obtained from an EXHAUSTIVE setting defines a benchmark that presents an optimal average solution that can be reached when assigning teams, we also refer to this benchmark as the "upper benchmark".
– A RANDOM setting uses a procedure that selects a random rescue team to a rescue. Assignments are not required to find a random solution because this procedure does not use knowledge of the rescue performance of assigned rescue teams. The procedure is expected to assign an optimal rescue team sometimes (i.e., for m rescue teams performing at an optimal value and a total number of n rescue teams, the chance of assigning an optimal rescue team is $\frac{m}{n}$).[10] The average result obtained from a RANDOM setting defines a lower benchmark, and approximates the mean of the capability distribution.

We have chosen these benchmark settings, because each setting represents an opposing extreme on a scale that measures the number of assignments required before a solution is calculated. At one extreme is the procedure used in the EXHAUSTIVE setting which requires as many assignments as there are rescue teams. At the other extreme is the procedure used in the RANDOM setting that requires no assignments to provide a solution.

Other benchmark settings are possible, but not used in this paper. For example, [14] use an "oracle setting" that always predicts the performance of teams accurately and finds optimal solutions without any explicit specification of how the performance of teams would be tested. An oracle setting is not useful for our analysis as it requires an explicit measure of the number of assignments. An alternative worst-case procedure is one where a worst-performing rescue teams is selected. This procedure requires as many rounds as the one used in an EXHAUSTIVE setting, and finds worse solutions than those found by the procedure used in a RANDOM setting. This is useful to determine the worst possible average solution quality, but it would be difficult to find realistic situations that would justify a comparison with this setting. We decided to use the random setting as it is more suitable than the worst-case setting.

4.7. Simulation Run of TAP, Exhaustive and Random Settings

A simulation run of a TAP setting executes the assignment algorithm (Section 1) under a particular combination of the experimental parameters. At the beginning of the simulation run, the number of agents (and rescue teams) is instantiated as specified by the parameter group size GS. The simulated true performance of rescue teams is initialised as specified by the CI parameter (e.g., for CI=1, values are drawn from a truncated distribution with a mean of 0.5 and a standard deviation of 0.1). The agents' models (that consist of estimations of the performance of rescue teams) are initialised as specified by the MI parameter (e.g., values are drawn from a distribution with a mean equal to the true performance of each rescue team, K=1 and $\Delta = 0$, and a standard deviation $MI_{\hat{\sigma}}$ of 0.1).

Initially, each agent's estimation of a rescue performance is likely to be different to the true performance of each team. Hence, different agents may propose different rescue teams for a rescue task due to the discrepancy between each agent's models. The models will change as agents get to know the performance of teams (by applying the update process after task execution). The capability of each team remains constant over different rounds.

The parameter P determines the type of group decision policy used in each assignment round. The simulation run is completed if the algorithm terminates according to criterion 1 (i.e., the proposed estimations of

[10]A RANDOM allocation procedure is guaranteed to find an optimal solution if all rescue teams have the same rescue performance as there is no benefit in selecting one rescue team over the other.

rescue teams are not greater than the true performance of $at_{\mathit{BestSoFar}}$).[11]

A simulation run of the procedure in an EXHAUSTIVE setting works as follows. The procedure selects each rescue team for the rescue task once, stores the rescue performance of each rescue team, and then selects an optimal one. The procedure in a RANDOM setting simply selects a random rescue team using the random function provided by a DGJPP compiler distribution underlying the implementations for our experiments. The DGJPP compiler distribution includes a complete 32-bit C/C++ development system for Intel 80386, and higher, processors. The random function is initialised with a different seed in each run based on the current time and date of the system.

4.8. *Efficiency Metrics*

Upon termination of a simulation run, we store the solution quality and computational requirement of the algorithm.

4.8.1. *Solution Quality (SQ): Rescue Performance of Selected Rescue Team*

The quality of a solution is measured by the rescue performance of the rescue team $at_{\mathit{BestSoFar}}$ found by the algorithm after termination (Section 2.2), and the rescue team found by the two benchmark procedures. This performance is represented by the value obtained from the rescue team's simulated rescue performance (as specified by Capability Initialisation CI).[12]

- TAP setting: we measure the performance of the rescue team $at_{\mathit{BestSoFar}}$ computed by the assignment algorithm (Section 2.2).
- EXHAUSTIVE setting: we measure the rescue performance of an optimal rescue team (a rescue team with a rescue performance not lower than that of any other rescue team).
- RANDOM setting: we measure the rescue performance of a rescue team that has been selected randomly.

4.8.2. *Computational Requirement (CR): Number of Assignments*

The computational requirement of the algorithm is measured as follows.

- TAP setting: we store the number of assignment rounds required until the algorithm terminates.
- EXHAUSTIVE setting: the CR is set to the number of rescue teams GS (e.g., for a group of 50 rescue teams, the EXHAUSTIVE procedure requires 50 assignments).
- RANDOM setting: the CR is set to 0.

4.8.3. *Average Solution Quality and Average Computational Requirement*

We have simulated each TAP setting (Capability Initialisation (CI) $\times$ Model Initialisation (MI) $\times$ Policy (P) $\times$ Group Size (GS)= $2\times30\times3\times5$=900) and each benchmark setting, RANDOM and EXHAUSTIVE. Note that $MI = K \times N_{\Delta} \times MI_{\hat{\sigma}} = 3 \times 5 \times 2 = 30$ (but owing to space limitations, settings with $\mu = 0.0$ of N_{Δ} and $MI_{\hat{\sigma}} = 0.25$ have not been plotted).

To obtain representative results of each of the 902 simulation settings, the solution quality and computational requirement are averaged over 10000 simulation runs. We have chosen this number, because the results showed stable patterns and are statistically significant. The significance test of the results is based on a 95% confidence interval. This was calculated by multiplying 1.96 with the standard error (the degrees of freedom are 10000).[13]

Each simulation setting is initialised with a different random seed based on the time of the system. Further, to mitigate the effect of repeating cycles of random numbers (as is the case with many standard implementations of random generators), the random function is initialised with a different seed value every 1000

[11]Termination criteria 2 and 3 defined in Section 3.3.2 will be investigated in future research. As they guarantee optimal solutions when agents are completely optimistic, we expect that they are more efficient in experimental settings where estimations are optimistic with $\Delta > 0$.

[12]Other measures are possible, but not used in this paper. For example, TAP settings can be measured by the sum of the performance of each rescue team in each round (and not only the performance of $at_{\mathit{BestSoFar}}$). This measure can be useful in understanding the efficiency of the algorithm over the entire simulation run. Another measure is to rank rescue teams according to their capabilities. For example, an optimal rescue team found in an EXHAUSTIVE setting is ranked 1. Rescue teams selected by a TAP or RANDOM setting are ranked between 1 (optimal rescue team) to the number of all rescue teams (e.g., for 50 rescue teams, the worst rank is 50). However, this ranking assumes that we have knowledge of each rescue team's true performance, but we do not make this assumption.

[13][5] offers a comprehensive introduction to analyse data obtained from empirical experiments in artificial intelligence. [8] provide convincing arguments why confidence intervals are adequate means to determine the significance of different data sets. Note that other statistical means might be more adequate for the analysis of simulation data as significance can be obtained by running a large number of experiments even if the difference between results is very small.

simulation runs. That is, we have ten trials, each divided into 1000 simulation runs. We used a random number function provided by the standard C++ library "stdlib.h" in the DGJPP compiler distribution.

4.8.4. *Normalisation*

We normalise the solution quality and computational requirement of the TAP algorithm as this enables a better comparison with the optimal and worst results of a particular setting. For this normalisation, solution quality is transformed into a range of 0 and 1, where 0 represents the average mean performance of the worst rescue teams $SQ_{\min}$ and 1 represents the average performance of an optimal rescue team $SQ_{\max}$ (result of the exhaustive setting). In particular, we subtract a result with $SQ_{\min}$ and then divide it by $(SQ_{\max} - SQ_{\min})$. For example, assume that the average optimal mean performance of rescue teams selected in the exhaustive procedure is $SQ_{\max} = 0.83$, and the average mean performance of rescue teams selected in the worst case procedure is $SQ_{\min} = 0.17$ (in this study, the average of the worst case procedure $SQ_{\min} = 1 - SQ_{\max}$). If the average solution quality of the TAP algorithm has the value 0.67, then the normalised value is $\frac{0.67-0.17}{0.83-0.17} = 0.\overline{75}$. This result means that the algorithm provides an average solution that is 25% worse than an optimal solution.

We apply a similar normalisation for the computational requirement. The value assignments are reversed with 0 representing an optimal computational requirement of the algorithm, and 1 representing the worst. That is, when the normalised value of CR is 0 then we measure one assignment round, while CR being 1 means that GS assignment rounds are required.

4.9. *Results and Analysis*

The purpose of this experiment is to examine the efficiency of the TAP algorithm empirically under different simulation settings (Section 4.5).

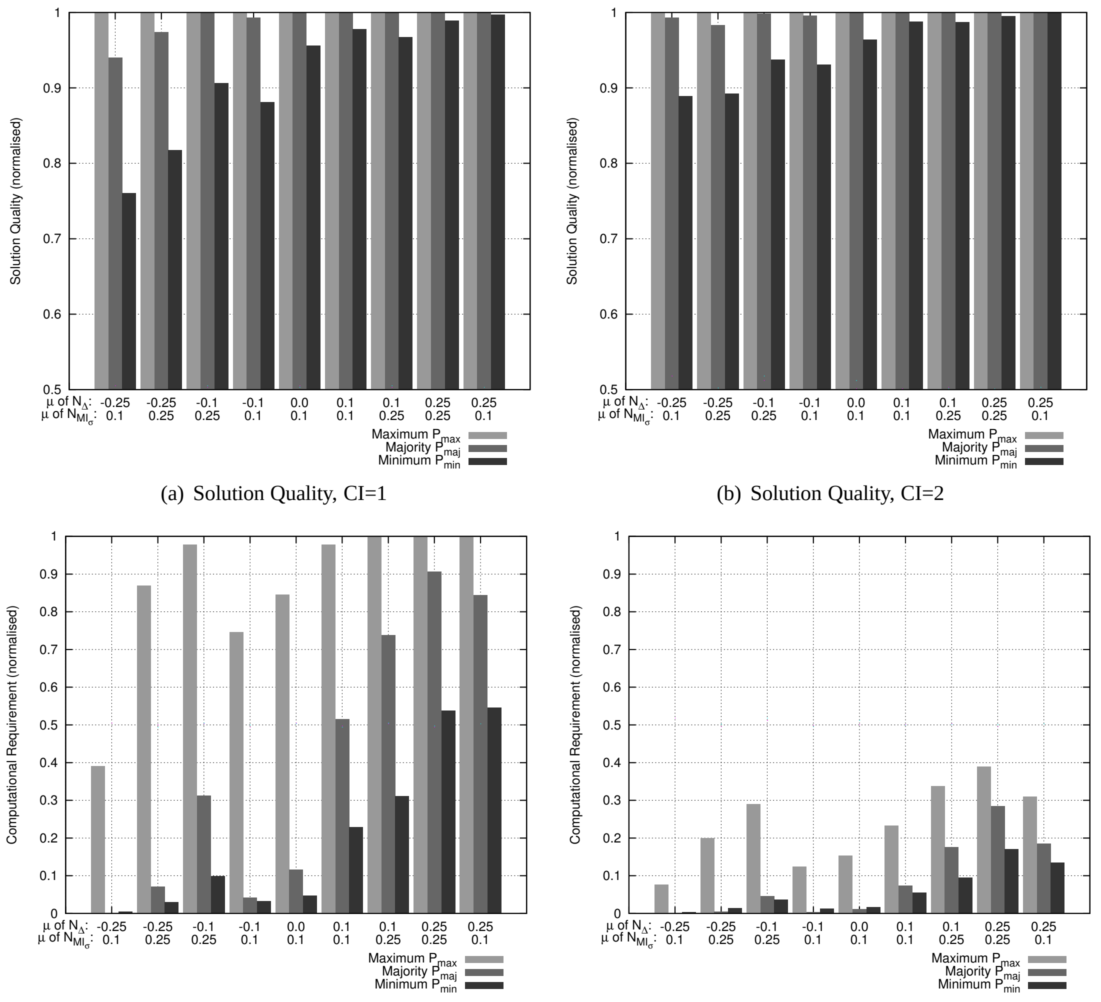
4.9.1. *Results of Benchmark Settings*

We first comment on the benchmark results obtained from EXHAUSTIVE and RANDOM settings. The average solution quality of the rescue teams assigned in EXHAUSTIVE settings is never worse than that obtained with TAP and RANDOM settings, because assignments are optimal and based on accurate knowledge of the performance of all rescue teams. The number of assignment rounds for an EXHAUSTIVE setting is always equal to GS (the number of rescue teams) because each rescue team has to perform the rescue once. The normalised value for solution quality and computational requirement of the EXHAUSTIVE setting is 1.

The average simulation results obtained from RANDOM settings averages the rescue performance of randomly assigned rescue teams and requires no selections. The average performance of rescue teams assigned in RANDOM settings is always lower than the average performance of rescue teams assigned in TAP and EXHAUSTIVE settings. This is because a RANDOM setting does not retain any information of the performance of rescue teams as opposed to a TAP setting where allocations are refined as the number of rounds increases. The normalised average performance of rescue teams in RANDOM settings is 0.5 as this is the actual mean of the distribution specified by the experimental parameter Capability Initialisation (CI). Note that the measures for RANDOM settings are not plotted in the figures in this section.

4.9.2. *Knowledgable Agents (K>0)*

The essence of this approach is how well agents estimate actual performance of rescue teams. This section focuses on results where estimates are based on the mean of the true performance of individual rescue teams (K=1). Figures 3(a) and 3(b) shows the SQ and Figures 3(c) and 3(d) shows the CR of the algorithm with the three policies and under different degrees of optimism and pessimism when K=1. We observe that SQ is less variable and higher when CI=2 (Figure 3(a)), than CI=1 (Figure 3(b)). The reason for this effect is that the true performance of each rescue team becomes more divergent from that of other rescue teams when the standard deviation becomes larger (i.e., when $CI_\sigma = 0.25$). That is, the distance of the performance value of an optimal rescue team is generally larger to that of other rescue teams (compared to values based on a smaller standard deviation $CI_\sigma = 0.1$). Hence, estimations are more divergent (and better performing agents more easily identified compared to a setting with a small standard deviation) and the algorithm is more likely to converge to an optimal rescue team. This effect is reinforced by the method of truncating the capability distribution (Section 4.5). When CI=2, there are more outliers set to 0 and 1 than with CI=1. In fact, if the capabilities are initialised based on a distribution with a very large standard deviation, SQ will reach 1 and CR will reach 0 (we ran the algorithm using $CI_\sigma = 0.5$ or $CI_\sigma = 1.0$ with K = 1, which are not shown in this paper, but can be found in [20]). Section 4.10 discusses other truncation methods and their effect on the SQ and CR.



(a) Solution Quality, CI=1

(b) Solution Quality, CI=2

(c) Computational Requirement, CI=1

(d) Computational Requirement, CI=2

Fig. 3. Model Accuracy: Knowledge (K) = 1, Group Size (GS) =50.

We also observe a significant decrease of CR when CI=2 (Figure 3(d)) compared to CI=1 (Figure 3(c)). Again, the reason is that as variability increases, the modelled performance values are more differentiated and the algorithm is less likely to select suboptimal rescue teams (as their values are much lower than in CI=1 on average).

As seen from Figures 3(a) and 3(b), settings with optimistic agents have a higher SQ than settings with pessimistic agents (particularly with the minimum policy). The reason is that optimistic agents will continue to propose rescue teams as they are not "satisfied" with the true performance of a rescue team as fast as pessimistic agents. That is, as the true performance is likely to be smaller than the estimations of optimistic agents, the algorithm will continue to select agents with a higher estimate.

Policies Let us analyse these results focusing on the role of group decision policies. Consider the setting under CI=1, where agents are very optimistic (μ of N_Δ is 0.25) and estimates are low variable (μ of $N_{MI_{\hat{\sigma}}}$ is 0.1), Figures 3(a) and 3(c). Under this setting, SQ is near optimal for the minimum, maximum and majority policy as almost all estimates are optimistic for each rescue team. Further, it is likely that an optimal rescue team is selected first, as on average the estimates will be higher for an optimal solution than for others. Because the true performance will turn out to be less than the estimates for almost all other rescue teams, the algorithm will not terminate until we have (almost) exhaustively gone through every rescue team for the majority and maximum policy. In short, for the maximum and majority policy, the following observations are made.

– An optimal solution is found first or early on.
– The CR is close to exhaustive.
– There is no difference between solution quality.

The minimum policy has a significantly lower average CR than that obtained for the maximum and majority policy as overestimates of the lowest true performance are lower than an optimal performance and therefore more likely to terminate the algorithm faster.

Now consider optimistic agents (μ of N_Δ is 0.1) with low variability (μ of $N_{MI_{\hat{\sigma}}}$ is 0.1). Again, it is most likely that an optimal solution is found first or early on. Recall the termination criterion which terminates the algorithm when no proposed estimation is higher than $at_{BestSoFar}$. For the maximum policy, the proposed estimates will continue to be about 0.2 (= 2 $\times\mu$ of $N_{MI_{\hat{\sigma}}}$) higher than the mean of the estimates: $MI_{\hat{\mu}}$ (including Δ). This means that the rescue team with the lowest true performance is still likely to have an optimistic estimation. For example, on average we can expect the lowest true value to be at 0.3 since CI_μ is 0.5 and $C_\sigma = 0.1$ ($0.3 = 0.5 - 2 \times 0.1$), and the best average solution to be at 0.7. Assume that $\Delta = 0.1$ and $MI_{\hat{\sigma}} = 0.1$, then a worst rescue team at_{worst} is likely to be overestimated with $0.3 + \Delta + 2 \times MI_{\hat{\sigma}} = 0.3 + 0.1 + 2 \times 0.1 = 0.6$ on average (note that as the number of agents increases, it is more likely that each rescue team is overestimated higher than 0.6). In short, it is likely that the algorithm, while selecting an optimal solution early on, will not terminate until most rescue teams are tested. For the majority policy, the estimates of the selected rescue team will be approximately $MI_{\hat{\mu}}$ (including Δ). Again, an optimal rescue team will be selected first, but the algorithm will likely terminate earlier than the maximum policy, as the estimates of selected rescue teams will be the estimate of the majority (i.e., $MI_{\hat{\mu}}$ including Δ) rather than the estimate of the maximum (i.e., $\Delta + 2 \times MI_{\hat{\sigma}}$ above $MI_{\hat{\mu}}$ including Δ). That is, the overestimates of the worst performing rescue team are lower than the true performance of an optimal rescue team: $V(at_{worst}) + MI_{\hat{\sigma}} = 0.3 + 0.1 = 0.4$. Therefore, the majority policy reaches close to the same solution as the maximum policy, but terminates earlier. In short, we make the following observations.

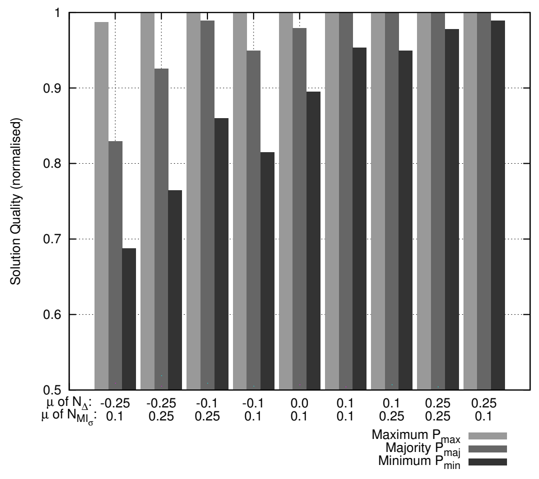
– An optimal solution is found first or early on for the maximum and majority policy.
– The CR is close to exhaustive for maximum, but lower for majority policy.
– There is no difference between solution quality for the maximum and majority policy.

Consider a pessimistic setting with $\mu = -0.25$ (N_Δ) and $\mu = 0.1$ ($N_{MI_{\hat{\mu}}}$), Figures 3(a) and 3(c). Again, the estimates are the highest for an optimal rescue team, so it is selected at the beginning of the run (as in the optimistic case). As opposed to the optimistic settings, the algorithm will terminate after only few rounds as many estimates are below the true performance of an optimal rescue team. The minimum and majority policy will terminate even earlier than the maximum policy, because the average estimates of the selected rescue team are likely to be below the true performance of the first selected rescue team. We obtained the following results.

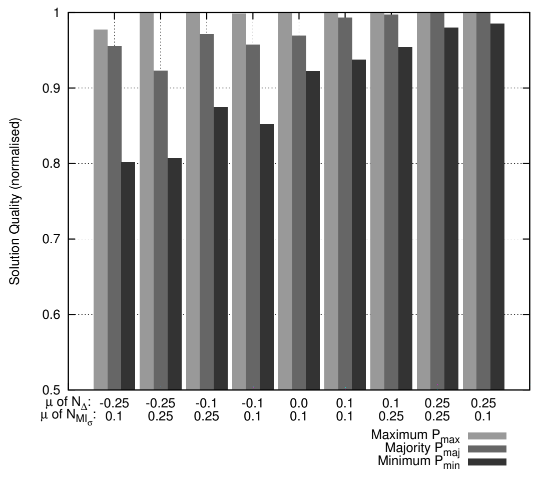
– An optimal solution is found first or early on with either the maximum or majority policy.
– The maximum policy requires fewer rounds than in optimistic settings, and majority and minimum policy require almost no rounds.
– Solution quality for the maximum policy is highest followed by that of majority and minimum policy.

4.9.3. *Ignorant Agents (K=0)*

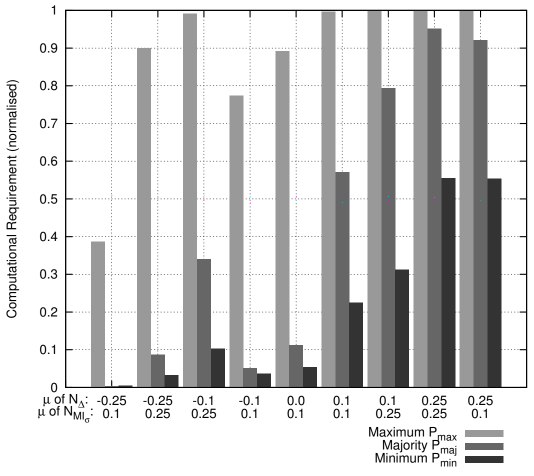
This section briefly discusses the results of agents that have no knowledge of the individual performance of each rescue teams (K=0). As seen from Figure 4, when agents are partially ignorant and knowledgeable with K=0.5, the results are in between those obtained for K=0 and K=1 in general. As seen from Figures 3, 4, and 5, as agents become more knowledgeable, the better the solution quality and computational requirement. The reason is that there is "more guesswork" involved when agents are ignorant as their estimates are not related to the true performance of rescue teams. As seen from these figures, when agents are ignorant, the order of selection of rescue teams is *independent* of actual capabilities. This means that selections are essentially random and "estimations" are independent of how well they match capabilities. Settings that terminate earlier have on average a lower SQ than those that terminate later. That is, if we consider a setting with respect to the termination criterion that terminates earlier than some other setting, then on average the SQ of the first setting will be worse but found in fewer assignment rounds than those for the other settings. In summary, the longer the algorithm runs the better solution quality.



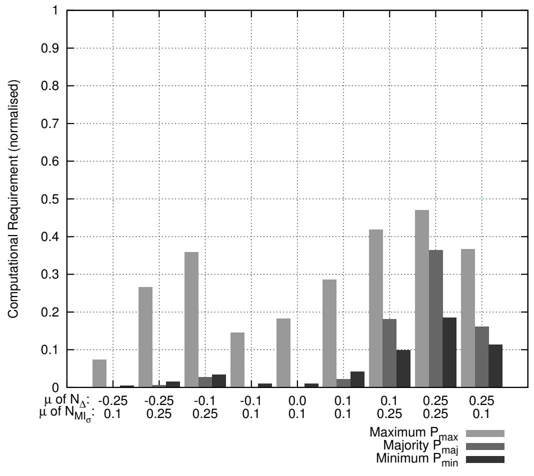
(a) Solution Quality, CI=1



(b) Solution Quality, CI=2



(c) Computational Requirement, CI=1



(d) Computational Requirement, CI=2

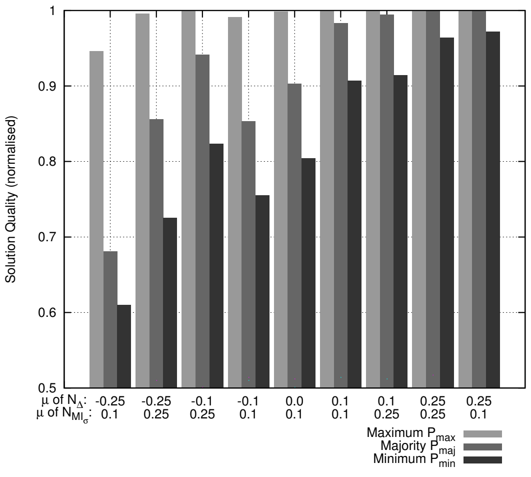
Fig. 4. Model Accuracy: Knowledge (K) = 0.5, Group Size (GS) =50.

4.9.4. Convergence: Performance Measure plotted against Assignment Rounds

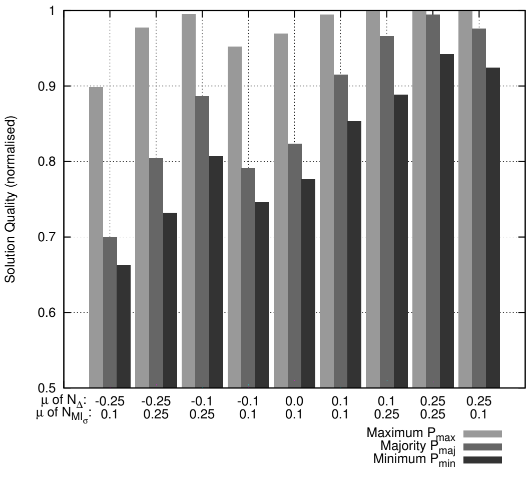
The previous results show that settings with the maximum policy find better performing rescue teams than settings with the majority policy. Also, the majority and minimum policy require fewer rounds, but reach worse solutions than those found with the maximum policy. This behaviour is observed under the condition that the algorithm will run until it meets termination criterion 1 (which states that the algorithm terminates if all agents' proposed estimations are not greater than $V(at_{BestSoFar})$). However, does the maximum policy always find better solutions in any round during the execution of the algorithm?

Figure 6 shows the normalised SQ as a function of the assignment round. A closer look at the convergence behaviour of the algorithm reveals that the majority policy often selects better rescue teams than those selected by the maximum or minimum policy in the first one to ten rounds.

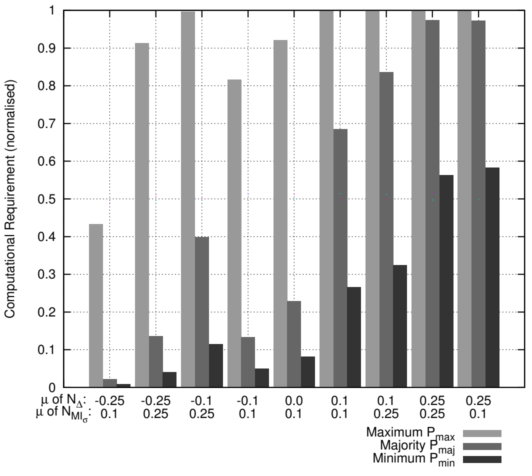
Figure 6(a) shows that when agents are ignorant (K=0), selecting each rescue team one by one as done with the exhaustive procedure is most efficient in any round. Hence, the SQ obtained when agents are only aware of the average performance of a rescue team population is not better than selecting a new rescue team in each round (as for the exhaustive procedure). That is, for ignorant agents, applying the assignment algorithm may not offer a benefit and instead



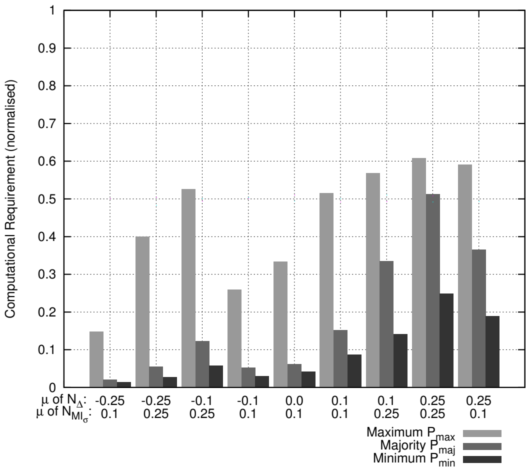
(a) Solution Quality, CI=1



(b) Solution Quality, CI=2



(c) Computational Requirement, CI=1



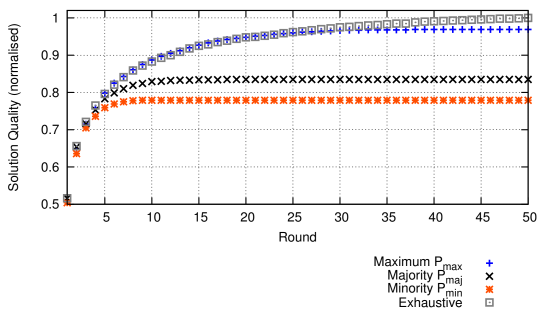
(d) Computational Requirement, CI=2

Fig. 5. Model Accuracy: Knowledge (K) = 0, Group Size (GS) =50.

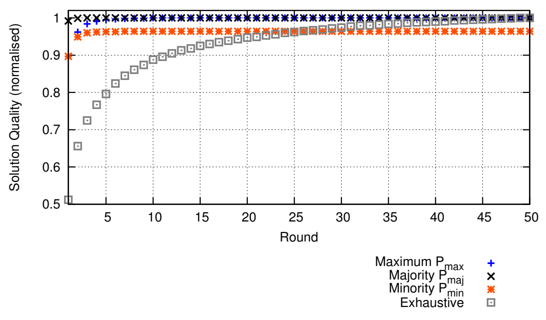
rescue teams should be allocated one by one (as for the exhaustive procedure). Figure 6(b) shows that under the informed setting an optimal solution is found quickly for all policies.

Consider a setting where agents are ignorant and knowledgable (K=0.5), Figure 6(c). We can observe that the SQ of rescue teams with the majority policy reaches a stable level after only 1 round. The minimum and maximum policy require more rounds before they reach their local maximum. Since P_{max} selects a rescue team preferred by only a minority of agents (based on one or a few agents' proposals), a slightly better rescue team is selected in each round. In TAP settings where the majority policy is applied the algorithm converges faster to a local maximum than in settings where the maximum or minimum policy is applied. This is explained by the fact that the outcome of the majority policy reflects the preference of many agents (and not only the preference of one or a few agents as reflected by the outcome of the maximum policy). Since the majority policy satisfies the preferences of the majority of agents it also converges faster and will provide a better solution quality in settings where the number of reassignment rounds is limited.

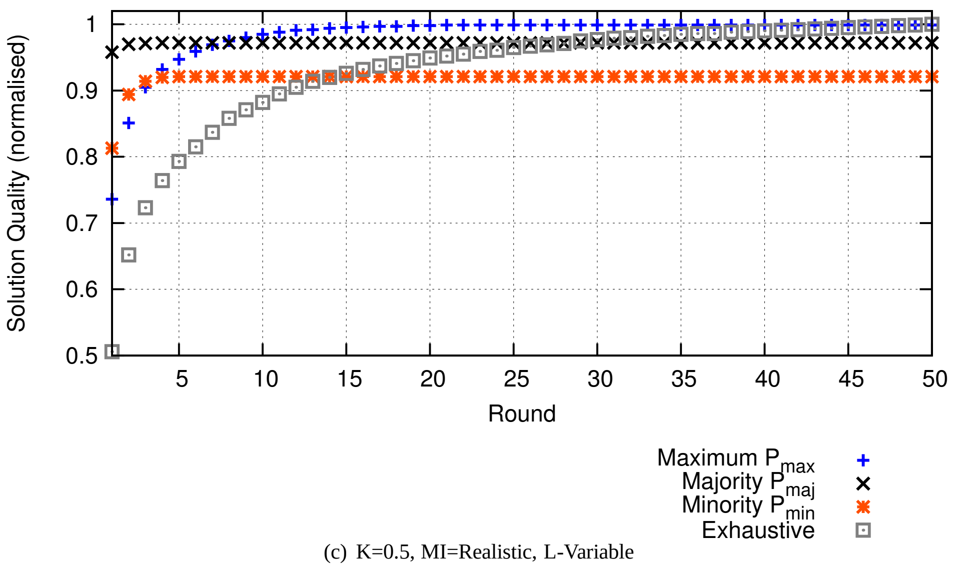
In summary, if the number of assignment rounds is small, then the majority policy will generally find better solutions than the maximum policy, and the maximum policy will find better solutions than the minimum policy on average. The reason is that the group of agents has generally more accurate knowledge than



(a) K=0, MI=Realistic, L-Variable



(b) K=1, MI=Realistic, L-Variable



(c) K=0.5, MI=Realistic, L-Variable

Fig. 6. Model Accuracy – Solution Quality (average task performance): Knowledge (K) = 0.5, Group Size (GS) = 50, Capability Initialisation (CI) = 2.

an individual agent. Hence, we observe that the maximum policy will take longer to converge to a solution as it only follows the best estimate by one agent (or sometimes a few agents), while the majority policy will converge faster as it follows the best estimates of many agents.

4.9.5. *Convergence: Performance Measure plotted against Assignment Rounds*

The previous results show that settings with the maximum policy find better performing rescue teams than settings with the majority policy. Also, the majority and minimum policy require fewer rounds, but reach worse solutions than those found with the maximum policy. This behaviour is observed under the condition that the algorithm will run until it meets termination criterion 1 (which states that the algorithm terminates if all agents' proposed estimations are not greater than $V(at_{BestSoFar})$). However, does the maximum policy always find better solutions in any round during the execution of the algorithm?

Figure 6 shows the normalised SQ as a function of the assignment round. A closer look at the convergence behaviour of the algorithm reveals that the majority policy often selects better rescue teams than those selected by the maximum or minimum policy in the first one to ten rounds.

Figure 6(a) shows that when agents are ignorant (K=0), selecting each rescue team one by one as done with the exhaustive procedure is most efficient in any round. Hence, the SQ obtained when agents are only aware of the average performance of a rescue team population is not better than selecting a new rescue team in each round (as for the exhaustive procedure). That is, for ignorant agents, applying the assignment algorithm may not offer a benefit and instead rescue teams should be allocated one by one (as for the exhaustive procedure). Figure 6(b) shows that under the informed setting an optimal solution is found quickly for all policies.

Consider a setting where agents are ignorant and knowledgable (K=0.5), Figure 6(c). We can observe that the SQ of rescue teams with the majority policy reaches a stable level after only 1 round. The minimum and maximum policy require more rounds before they reach their local maximum. Since P_{max} selects a rescue team preferred by only a minority of agents (based on one or a few agents' proposal), a slightly better rescue team is selected in each round. In TAP settings where the majority policy is applied the algorithm converges faster to a local maximum than in settings where the maximum or minimum policy is applied. This is explained by the fact that the outcome of the majority policy reflects the preference of many agents (and not only the preference of one or a few agents as reflected by the outcome of the maximum policy). Since the majority policy satisfies the preferences of the majority of agents it also converges faster and will provide a better solution quality in settings where the number of reassignment rounds is limited.

In summary, if the number of assignment rounds is small, then the majority policy will generally find better solutions than the maximum policy, and the maximum policy will find better solutions than the minimum policy on average. The reason is that the group of agents has generally more accurate knowledge than an individual agent. Hence, we observe that the maximum policy will take longer to converge to a solution as it only follows the best estimate by one agent (or sometimes a few agents), while the majority policy will converge faster as it follows the best estimates of many agents.

4.10. Discussion

This section discusses the issues of truncation method, alternative methods of empirically investigating the accuracies of models, and the role of learning. All these issues turned out to play an important role in the efficiency of the algorithm.

4.10.1. Truncation of Distributions

The truncation method used in this study sets outliers to the nearest threshold value. We have implemented and tested three other methods to obtain a truncated distribution.

1. Outliers are redrawn from the same distribution until they are within the threshold [0,1]. This method maintains the original shape of the normal distribution. However, this redrawing process may never terminate as redrawn values may also lie outside this threshold. This becomes more likely with a larger standard deviation.
2. Outliers are redrawn from a uniform distribution from the interval [0,1]. As opposed to (1.), the redrawing process will specify a new random value after one iteration. The problem here is that in pessimistic settings, randomly redrawn values are often much larger than the mean of the model distribution $MI_{\hat{\mu}}$. This has a significant influence on the solution quality and the computational requirement as both are much higher for the maximum policy than they would be if we use (1.).
3. Outliers are set to the original mean of the distribution (for example, 0.5 for CI=1 and 2). If the standard deviation of the distribution is very large, then many values will be set to the same mean, and there is little variation in the true and estimated performance of rescue teams. This is opposed to the truncation method used in this study which sets the performance of some rescue teams to a very high value and some to a very low performance.

In future research, we plan to use a beta distribution or rejection sampling as generated values fit the data better than the truncation method used in this paper. The *beta distribution* is a continuous probability distribution defined on the interval [0, 1] and uses two positive shape parameters denoted by α and β [7, 24, 19].

Rejection sampling is a mix of (1.) and (2.) as values are drawn from a distribution function g that is easier to sample from than from the actual distribution f (in our case, the truncated normal distribution). We can sample from g and accept the sampled value with probability M g/f and sample again until we accept a value. This method works best if g is close to f and M is small. [17] have suggested a particularly effective rejection sampling method.

This issue is referred to as distribution fitting – depending on the distribution that we wish to simulate, we need to find the values of parameters that maximise the resemblance between a presumed theoretical distribution and a given data set [7, 24, 19].

4.10.2. Designing an Experiment with Model Accuracy

This study has identified and varied four parameters that represent the accuracy of models maintained by different agents: group knowledge, group optimism, spread, and homogeneity. Model accuracy can be investigated in many different ways.

For example, the notion of (partial) ignorance investigated in this study is that estimations are more likely to move closer to the mean of the distribution of the rescue teams' performance (e.g., $CI_\mu = 0.5$). Another way of defining this experimental parameter is to replace the value CI_μ with a value that has been drawn from a normal distribution. This would then represent a situation where agents are also ignorant of the distribution mean of rescue teams.

The investigation of homogeneity among agents can also be approached in different ways. For example, we can set the estimation of a different rescue team to an accurate level for each agent such that each agent has one accurate estimation of one rescue team. Extensions to our theoretical study would show that in this case, we are also guaranteed to find optimal solutions. Empirically, the overall accuracy of each agent can be initialised based on a normal distribution, where some agents have highly accurate models of all rescue teams and some inaccurate models, but most will have moderately accurate models.

Under certain settings of homogeneity and model accuracy, our algorithm may likely obtain a similar solution quality than if only one designated agent (or a small group of designated agents) would make a decision. An example of such a setting is when an agent that is more knowledgeable than other agents, may be able to find optimal solutions faster on its own, than if it participates in the group decision process. Under which conditions will our collective algorithm find better solution quality and reduce computational requirement? One main issue is the identification of agents that have the most accurate models which is addressed in [20].

4.10.3. Role of Learning

Although learning is part of our formal framework, it is not as important in this study as in related studies [20]. In fact, if we remove the learning processes from the algorithm and modify the framework slightly, then the algorithm will terminate as it does currently. This slight modification can be done as follows. We restrict agents to propose a rescue team that has not been selected before and redefine the termination criterion such that it will terminate the algorithm if no proposals are made. This has not been done in this paper as agents would progressively have "no input" to the decision making process and restricts future work on more complex forms of learning.

Another aspect of learning disregarded in this study is that each agent could replace its estimates of a rescue team's performance with the highest proposed estimate made by other agents. In this case, the solution quality will stay the same (as in our current setting when using the majority policy), but the computational requirement is likely to decrease as agents will have the same estimate of previously proposed rescue teams. However, the issue of revising and updating beliefs if new input is presented is a research field in its own right [12, 32, 13, 11].

Learning becomes a more important issue in future research. For example, it is important when studying models of the variable performance of rescue teams [20]. Also, our study implements a model-updating process that only modifies the model of a selected team after task execution. A more sophisticated model-update process may also modify the models of other teams at the same time. For example, after an agent observes the performance of a particular team is better than expected, it will increase the estimation of the performance of this team, but also could lower the estimations of all other teams. We discuss different aspects of model updating in more detail in [20].

5. Related Research

Distributed coordination procedures are often investigated using the Multi-Agent Systems (MAS) paradigm, because it makes realistic assumptions of the autonomous and distributed nature of the components in system networks [36, 25, 2, 27, 41, 33, 4]. Many Multi-Agent System approaches do not adequately address the CIA problem as they use each agent's models separately to improve coordination as opposed to all agents using their models together. That is, each agent uses its own models to decide on allocating a team to a task even if other, more knowledgeable agents would suggest better allocations.

This section offers a brief overview of well-known MAS approaches to allocation problems (note that this paper considers task or group-rational agents that *collaborate* when allocating tasks). These approaches can be divided into two classes.

- Market-driven schemes enable the coordination of MAS by using *each agent's knowledge of its own task performance* [4]. A well-known market-driven scheme is the Contract Net (CNET) protocol [36]. The CNET protocol is the first protocol that enables agents to assign tasks autonomously [42]. This protocol is based on a contract metaphor involving a manager and contractors. In the context of a task allocation problem, a manager announces a task, each contractor provides a bid specifying how well it can perform the announced task, and the manager then selects the contractor that specified a performance no worse than those specified by other contractors in his bid (we refer to this as the *highest* or *best* bid). Note that agents may have knowledge of the performance other agents, but the protocol does not require an agent to use this knowledge for making bids.
- Agent-modelling schemes rely on *each agent's knowledge of the behaviour of other agents* [39, 37, 1, 30, 18]. A well-known approach is the Recursive Modelling Method (RMM) where each agent uses a utility function to make decisions that estimates utility functions maintained by other agents [18]. For example, to make a decision, agent a_i estimates the utility function of agent a_j ($a_i \neq a_j$), and agent a_j in turn models the utility function of agent a_i. Since a_i knows that its utility function is estimated by a_j, a_i's utility function will change accordingly, and so will a_j's. This type of "recursive nesting" will eventually exceed an agent's memory, which is prevented by limiting the nesting depth (thus keeping an agent ignorant). An agent will use its estimated utility function to assess whether or not it should perform the task in question, but RMM does not involve the consultation of other agents.

The isolated use of each agent's contribution renders a market-driven or agent-modelling approach most useful when each agent has accurate or near accurate estimations. For example, in the CNET protocol, the bid of each contractor must reflect its true performance to find an optimal solution. That is, the bid of an optimal contractor should not be lower than that of suboptimal contractors. If an optimal contractor underestimates its performance (and underbids) or if a suboptimal contractor overestimates its performance (and overbids optimal contractors), an optimal contractor will not be selected by the manager and an optimal solution is not found. Our approach enables the use of each contractor's contribution in a collective manner as other contractors may have more accurate estimations of optimal contractors than optimal contractors themselves. Hence, an optimal contractor is no longer required to be the only one to have accurate or near accurate estimations to guarantee an optimal allocation.

A comprehensive review of related research is offered in [20].

6. Conclusion

This paper offers an efficient algorithm to the Collective Iterative Allocation (CIA) problem and studies theoretical and empirical behaviour of this algorithm. Our theoretical study showed that complete optimism and task rationality of agents are two theoretical premises that guarantee optimal solutions (under the condition that the team's performance is invariable and deterministic). We verified optimality of three policies in particular under such conditions: the minimum policy P_{min} (which selects a team with the lowest proposed performance), the maximum policy P_{max} (which selects a team with the highest proposed performance) and the majority policy P_{maj} (which selects a team that has been proposed by most agents). Complete optimism is a condition that converges the algorithm to an optimal solution in no greater number of assignment rounds than testing each team once (as for an exhaustive procedure).

We have performed a series of empirical simulation-based experiments that investigates the influence of model accuracy and group decision policies on the efficiency of the assignment algorithm (measured by solution quality and computational requirement). The experiment is illustrated by using a model of a surf rescue domain involving a group of rescue teams that endeavours to find an optimal rescue team for rescues.

This study offers several lessons about the empirical efficiency of the TAP assignment algorithm (under the assumption that performance is invariable and deterministic).

- We have investigated two factors that characterise the accuracy of models: group knowledge and group optimism (we have identified two additional factors: spread and homogeneity of estimations among agents, but not investigated them rigourously in this study).

  - ∗ **Group Knowledge** defines how closely the agent's estimates match the true performance of rescue teams. As the knowledge of agents increases, the efficiency of the algorithm becomes better. If agents are knowledgable, solution quality will be near optimal with the maximum and majority policy. As the true performance of rescue teams becomes more distinct (i.e., if the spread of the true performance of rescue teams is larger) or the number of rescue teams becomes larger, solution quality and computational requirement reaches optimal levels regardless of the policy used.
  - ∗ **Group Optimism** defines how the performance of individual rescue teams is under- or overestimated. While solution quality for both the maximum and majority policy is near optimal when agents are optimistic, the algorithm terminates earlier with the majority policy than with the maximum policy, and earlier with the minimum policy than with the majority policy. Our study shows that informed, but pessimistic agents still find optimal solutions with the maximum and majority policy, but require fewer assignment rounds than in the case of optimistic agents.

- **Convergence** If the computational requirement is restricted to only a few rounds (i.e., the algorithm terminates after a constant and small number of rounds), the majority policy should be selected over the maximum and minimum policy as the majority policy offers a better solution quality.

These theoretical and empirical insights form a basis to explore other dimensions of our approach to the CIA problem. [20] discusses many dimensions in detail. For example, currently, our policies do not require agents to compromise on their preference, but as it might be necessary in many domains that all agents agree on the direction, we need to develop algorithms that allow an individual agent to assess conditions that will allow it to relinquish its preferences. Further, a policy receives as input several proposals specifying one team and its estimated performance, but the efficiency of the algorithm may improve if an agent proposes a list of preferred teams. This requires more sophisticated policies as well as an evaluation of the communication requirement of the algorithm. Also, the agent a_{policy} is currently assumed to be task rational and capable when applying the policy. However, many domains would encourage some agents to be more opportunistic than others, thus we need trust mechanisms that will allow the identifications of the most trusted agent that is least likely to jeopardise the application of the policy. These are just some, but important directions for future research that will further elaborate on whether our approach is successful in more complex domains.

References

[1] R. Alterman and A. Garland. Convention in joint activity. *Cognitive Science*, 25(4):611–657, July–August 2001.

[2] A. H. Bond and L. Gasser. *Distributed Artificial Intelligence*. Morgan Kaufmann publishers Inc., 1988.

[3] P. Bonissone and R. M. Tong. Reasoning with uncertainty in expert systems. *International journal of man-machine studies*, 22(3):241–250, 1985.

[4] Y. Chevaleyre, P. E. Dunne, U. Endriss, J. Lang, M. Lemaître, N. Maudet, J. Padget, S. Phelps, J. A. Rodríguez-Aguilar, and P. Sousa. Issues in multiagent resource allocation. *Informatica*, 30:3–31, 2006.

[5] P. R. Cohen. Empirical methods for artificial intelligence. *IEEE expert: Intelligent systems and their applications*, 11(6):88, 1996.

[6] V. Conitzer and T. Sandholm. Common voting rules as maximum likelihood estimators. *Proceedings of the twenty-first Annual Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 145–152, 2005.

[7] H. Cramér. *Mathematical Methods of Statistics*. Princeton University Press, 1946.

[8] G. Cumming and S. Finch. Inference by eye: Confidence intervals and how to read pictures of data. *American Psychologist*, 60(2):170–180, 2005.

[9] E. H. Durfee, V. R. Lesser, and D. D. Corkill. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1(1):63–83, 1989.

[10] U. Endriss and N. Maudet. On the communication complexity of multilateral trading: Extended report. *Journal of Autonomous Systems and Multi-Agent Systems*, 11(1):91–107, 2005.

[11] J. Galliers. Autonomous belief revision and communication. In P. Gärdenfors, editor, *Belief revision*, pages 220–246, Cambridge, Massachusetts, United States of America (USA), 1992. Cambridge University Press.

[12] P. Gärdenfors. *Knowledge in flux: Modeling the dynamics of epistemic states*. Cambridge University Press, Cambridge, Massachusetts, United States of America (USA), 1988.

[13] P. Gärdenfors. Belief revision. In P. Gärdenfors, editor, *Belief revision*, Cambridge, Massachusetts, United States of America (USA), 1992. Cambridge University Press.

[14] L. Garrido, K. Sycara, and R. Brena. Quantifying the utility of building agents models: An experimental study. In *Agents-00/ECML-00 Workshop on Learning Agents*, Barcelona, Spain, 2000.

[15] B. Gerkey and M. Mataric. A framework for studying multi-robot task allocation. *Multi-robot systems: From swarms to intelligent automata*, 2:15–26, 2003.

[16] B. Gerkey and M. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *Robotics Research*, 23(9):939–954, September 2004.

[17] W. R. Gilks and P. Wild. Adaptive rejection sampling for gibbs sampling. *Applied Statistics*, 41(2):337–348, 1992.

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

[19] A. Gupta and S. Nadarajah. *Handbook of Beta Distribution and Its Applications*. Marcel Dekker Inc, 2004.

[20] C. Guttmann. *Collective Iterative Allocation*. PhD thesis, Monash University, 2008.

[21] C. Guttmann. Making allocations collectively: Iterative group decision making under uncertainty. In R. Bergmann, G. Lindemann, S. Kirn, and M. Pechoucek, editors, *Proceedings of the sixth German Conference on Multi-Agent system TEchnologieS (MATES)*, volume 5244 of *Lecture Notes in Computer Science*, pages 73–85. Springer, 2008.

[22] C. Guttmann, I. Rahwan, and M. Georgeff. An approach to the collective iterative task allocation problem. In *Proceedings of the International Conference of Intelligent Agent Technology (IAT)*, pages 363–369, United States of America (USA), 2007. IEEE Press.

[23] C. Guttmann and I. Zukerman. Agents with limited modeling abilities: Implications on collaborative problem solving. *International Journal of Computer Science and Software Engineering (CSSE)*, 21(3):183–196, 2006.

[24] G. Hahn and S. Shapiro. *Statistical models in engineering*. John Wiley & Sons, 1967.

[25] C. Hewitt. The challenge of open systems. *Byte*, 4(10), 1985.

[26] A. Jennings and D. Bradby. Lifebots for surf rescue. Unpublished Manuscript, Royal Melbourne Institute of Technology, Melbourne, Australia, 2000.

[27] N. Jennings, K. Sycara, and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*, 1(1):7–38, 1998.

[28] H. Kitano. Robocup rescue: A grand challenge for multi-agent systems. In *Proceedings of the International Conference on Multi-Agent Systems*, page 5, Washington, DC, United States of America (USA), 2000. IEEE Press.

[29] H. Kitano and S. Tadokoro. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI Magazine*, 22(1):39–52, 2001.

[30] J. R. Kok and N. Vlassis. Mutual modeling of teammate behavior. Technical Report UVA-02-04, Computer Science Institute, University of Amsterdam, Netherland, August 2001.

[31] X. Li. Hybrid negotiation for resource coordination in multiagent systems. volume 3, pages 231–259. IOS Press, 2005.

[32] A. S. Rao. Dynamics of belief systems: A philosophical, logical, and ai perspective. Technical Report 02, Australian Artificial Intelligence Institute, Melbourne, Australia, July 1989.

[33] W. R. Scott. *Organizations: Rational, Natural, and Open Systems*. Prentice-Hall, Upper Saddle River, New Jersey, United States of America (USA), 2002.

[34] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998.

[35] P. Smets. Varieties of ignorance and the need for well-founded theories. *Information Sciences*, 57(58):135–144, 1991.

[36] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.

[37] P. Stone, P. Riley, and M. M. Veloso. Defining and using ideal teammate and opponent agent models. In *Proceedings of the Innovative Applications of Artificial Intelligence Conference (IAAI)*, pages 1040–1045, 2000.

[38] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. The MIT Press, Cambridge, Massachusetts, United States of America (USA), 1998.

[39] M. Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[40] J. Vassileva, G. I. McCalla, and J. E. Greer. Multi-agent multi-user modeling in I-Help. *User Modeling and User-Adapted Interaction*, 13(1–2):179–210, 2003.

[41] G. Weiß. *Multiagent systems: A modern approach to distributed artificial intelligence*. The MIT Press, Cambridge, Massachusetts, United States of America (USA), 1999.

[42] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, June 1995.