Available online at www.sciencedirect.com
ScienceDirect
Artificial Intelligence 171 (2007) 491–513
Artificial Intelligence
www.elsevier.com/locate/artint

On the design of coordination diagnosis algorithms for teams of situated agents

Meir Kalech *, Gal A. Kaminka

The MAVERICK group, Department of Computer Science, Bar Ilan University, Israel

Received 5 September 2005; received in revised form 12 March 2007; accepted 16 March 2007

Available online 24 March 2007

Abstract

Teamwork demands agreement among team-members in order to collaborate and coordinate effectively. When a disagreement between teammates occurs (due to failures), team-members should ideally diagnose its causes, to resolve the disagreement. Such diagnosis of social failures can be expensive in communication and computation, challenges which previous work has not addressed. We present a novel design space of diagnosis algorithms, distinguishing several phases in the diagnosis process, and providing alternative algorithms for each phase. We then combine these algorithms in different ways to empirically explore specific design choices in a complex domain, on thousands of failure cases. The results show that different phases of diagnosis affect communication and computation overhead. In particular, centralizing the diagnosis disambiguation process is a key factor in reducing communications, while runtime is affected mainly by the amount of reasoning about other agents. These results contrast with previous work in disagreement detection (without diagnosis), in which distributed algorithms reduce communications.

Keywords: Diagnosis; Multi-agent systems; Situated agents

1. Introduction

With increasing deployment of robotic and agent teams in complex, dynamic settings, there is an increasing need to also be able to respond to failures that occur in teamwork [1,7,23,33]. One type of failure in teamwork is *disagreement*, where agents come to disagree on salient aspects of their joint task. There is thus a particular need to be able to detect and diagnose the causes for disagreements that may occur, in order to facilitate recovery and reestablishment of collaboration, e.g., by negotiations [19]. This type of diagnosis is called *social diagnosis*, since it focuses on finding causes for failures to maintain social relationships [17], i.e., coordination failures.

For instance, suppose a team of four robotic porters carry a table, when suddenly one of the robots puts the table down on the floor, while its teammates are still holding the table up. Team-members can easily identify a disagreement, but they also need to determine its causes, e.g., that the robot believed the table reached the goal location, while its teammates did not. Given this diagnosis, the robots can negotiate in order to resolve the disagreement.

* Corresponding author.
E-mail addresses: kalechm@cs.biu.ac.il (M. Kalech), galk@cs.biu.ac.il (G.A. Kaminka).
URLs: http://www.cs.biu.ac.il/~kalechm/ (M. Kalech), http://www.cs.biu.ac.il/~galk/ (G.A. Kaminka).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2007.03.005

Unfortunately, while the problem of detection has been addressed in the literature e.g., [16,18,24], social diagnosis remains an open challenge. Naive implementations of social diagnosis processes can require significant computation and communications, which prohibits them from being effective as the number of agents is scaled up, or the number of failures to diagnose increases. Previous work did not rigorously address this concern: Kaminka and Tambe [17] guarantee disagreement detection without communications, but their heuristic-based diagnosis often fails. Dellarocas and Klein as well as Horling et al. [4,9] do not explicitly address communication complexity. Fröhlich et al. and later Roos et al. [6,27,29,30] assume fixed communication links, an assumption which does not hold in dynamic teams in which agents may choose their communication partners dynamically (see Section 2 for details).

We seek to examine in depth the communication and computation requirements of social diagnosis. We distinguish two phases of social diagnosis: (i) selection of the diagnosing agents; and (ii) diagnosis of the team state (by the selected agents). We provide alternative algorithms for these phases, and combine them in different ways, to present six diagnosis methods, corresponding to different design decisions. We then examine the runtime and communication complexity and empirically evaluate these parameters in diagnosing thousands of systematically-generated failure cases, occurring in a team of behavior-based agents in two different complex domains.

We draw general lessons about the design of social diagnosis algorithms from the empirical results. Specifically, the results show that centralizing the disambiguation process is a key factor in dramatically improving communications efficiency, but is not a determining factor in runtime efficiency. On the other hand, explicit reasoning about other agents is a key factor in determining runtime: Agents that reason explicitly about others incur significant computational costs, though they are sometimes able to reduce the amount of communications. These results contrast with previous work in disagreement detection, in which distributed algorithms reduce communications (and to some extent, runtime) by reasoning about other agents.

The paper is organized as follows: Section 2 motivates the research and discusses related work. Section 3 presents the architecture of behavior-based agents. Section 4 presents the disambiguating diagnosis hypotheses phase and Section 5 presents the diagnosing agent selection phase. Section 6 specifies diagnosis methods which combine the algorithms in the previous two sections in different ways, and Section 7 evaluates them empirically. Section 8 concludes.

2. Motivation and related work

Agreement (e.g., on a joint plan or goal) is the key to the establishment and maintenance of teamwork [1,7,10,33]. The *Joint Intentions* framework [1] focuses on agreement (mutual belief) in a team's joint goal. The *SharedPlans* framework [7] relies on an intentional attitude, in which an individual agent's intention is directed towards a group's joint action. This includes mutual belief and agreement among the teammates in a complete recipe including many actions. Similarly, the *Joint Responsibility* model [10] establishes the team-members mutual belief in a specific recipe as a corner-stone for their collaboration.

There exist several architectures for building agents, using ideas from teamwork theories; agreement on specific features of the agents' internal state plays a critical role in all. GRATE* implements the joint responsibility model [10] in industrial agent systems. STEAM [33] and TEAMCORE [25] use ideas from both Joint Intentions and SharedPlans, and add reactive team plans which are selected or deselected by a team or sub-team. BITE [14,15] follows in this tradition, and additionally allows for a variety of agreement-synchronization protocols to be used interchangeably, in controlling physical robots.

However, teamwork sometimes fails, causing disagreements—*agreement failures*—among team-members [4,16, 17]. This may be due to sensing failures, or different interpretations of sensor readings. The function of a diagnosis process is to go from disagreement detection (where an alarm is raised when a fault—disagreement—occurs), to fault *identification*, where the causes for the disagreement are discovered, in terms of the differences in beliefs between the agents that lead to the disagreement. Such differences in beliefs may be a result of differences in sensor readings or interpretation, in sensor malfunctions, or communication difficulties.

While diagnosis of a single-agent system is relatively well understood, and known to be computationally difficult [8], *social diagnosis*—diagnosis of coordination failures such as disagreements—remains an open area of research. In particular, to our best knowledge, there has not been an in-depth exploration of disagreement diagnosis algorithms. Our work takes first steps to understand disagreement diagnosis algorithms in terms of their design choices, and the effects of these on computation and communications.

The most closely-related work to ours is reported in [16,17]. This previous investigation provides guarantees on detection of disagreements, but only presented a heuristic approach to diagnosis, which indeed does not always succeed. The algorithms we present here succeed in the same examples where the previous heuristic approaches have failed.

Dellarocas and Klein [4,18] report on a system of domain-independent exceptions handling services. The first component contains a knowledge base of generic exceptions. The second contains a decision tree of diagnoses; the diagnosing process is done by traversing down the tree by asking queries about the relevant problem. The third component is responsible to seek for a solution for the exception, based on a resolution knowledge base. This approach transfers the failure-handling responsibility from the agent to an external system, to alleviate the load on each agent designer (which would now be freed of the responsibility of implementing an exception-handling system in each agent). However, in contrast to our work, communication and runtime concerns are not addressed. In their system *sentinel agents* monitor the agents in the multi-agent system and pro-actively query agents about their status. They do not mention the monitoring method and when a querying is necessary, but both of those actions have a large influence on communication and computation complexity.

Similarly, Horling et al. [9] uses a fault-model of failures and diagnoses to detect and respond to multi-agent failures. In this model, a set of pre-defined diagnoses are stored in acyclic graph's nodes. When a fault is detected a suitable node is triggered and according to the fault characters the node activates other nodes along the graph. The advantage of Horling's fault-model system over Dellarocas and Klein's system is the use of learning algorithm that can be employed to maintain structure as time passes. As with Dellarocas and Klein, Horling's work does not explicitly address communication complexity.

Fröhlich et al. [6] have suggested dividing a spatially distributed system into regions, each under the responsibility of a diagnosing agent. If the fault depends on two regions, the agents that are responsible for those regions cooperate in making the diagnosis. This method is inappropriate for dynamic team settings, where agents cannot pre-select their communication partners. Similarly, Roos et al. [28–30] have analyzed a model-based diagnosis method for spatially distributed knowledge. But, its method assumes that there are no conflicts between the knowledge of the different agents, i.e., no disagreements.

Roos et al. [27] have expanded their own work to address semantically distributed systems, where the knowledge is distributed among the agents. Each agent is an expert in a certain problem domain. In this system, if each agent makes a diagnosis separately, the diagnosis will be incomplete since the dependencies between the agents are not diagnosed. Roos et al. have suggested to keep the dependencies between the agents. Each agent will diagnose its own domain and the related dependencies of its domain. The communication links are fixed, such that each failure is diagnosed strictly by the agents that are associated with their communication link. However, the interactions among system entities, in multi-agent teams, are not known in advance since they depend on the specific conditions of the environment in runtime and the appropriate actions assigned by the agents [21]. We can solve this problem by keeping all the possible interactions between the agents. However, as Roos et al. point out, this may cause a large communication complexity, especially in a large system, since the number of candidate diagnoses is exponential (in the number of dependencies).

Diagnosis is an essential step beyond the detection of the failures. Mere detection of a failure does not necessarily lead to its resolution. First, because the agents that caused a disagreement are not necessarily those that detected it, and may thus be unaware of it. Therefore, they may not be able to replan around it. Second, even if somehow an undiagnosed (though detected) disagreement manages to temporarily overcome the disagreement—it may still continue to occur in various forms, if its causes are not resolved, e.g., via negotiations [19]. In dynamic domains, such as a RoboCup soccer game, it may indeed be more effective (for a short while) to simply replan rather than engage in diagnosis (if the cost of replanning is cheaper than diagnosis, and if it is possible without knowing the causes for the disagreement). However, even in such settings, it often makes sense to use diagnosis post-hoc to discover the reasons for any failures; for instance, in conducting a post-game analysis.

For instance, Parker [23] reports on ALLIANCE, a behavior-based architecture which is robust to many kinds of failures, and is able to recover from them by having robots take over tasks from failing teammates, without on explicit diagnosis process such as those are described in this paper. Similarly to Parker's work, the algorithms here are intended for use by situated, behavior-based agents. However, the focus is on explicit identification of the failures, rather than on reactive responses to them. Thus, our work complements the work on ALLIANCE: Without diagnosis, robots using ALLIANCE may end up failing repeatedly and needlessly due to recurring disagreement failures, or due to lack of agreement on task assignments.

Unfortunately, diagnosis of disagreements can be expensive both in terms of computation as well as in communications (see Table 2, and Section 7). This is because while the constraints for agreement are known in advance, the interactions leading to disagreements are not. They depend on the specific conditions of the environment in runtime. Moreover, in diagnosing disagreements, the agents are part of the system. And so their inputs and outputs are subjective to the point of view of each diagnosing agent. In particular, these inputs and outputs may not be known to all diagnosing agents.

3. Building blocks for diagnosis

We start by describing the basic building blocks for social diagnosis of disagreements. We base our approach on model-based diagnosis. In model-based diagnosis of a single agent, the diagnoser uses a model of the agent to generate expectations which are compared to the observations, in order to form diagnoses [2,3,26].

In model-based *social* diagnosis, the diagnoser models the expected relationships between the agents [12,13]. The goal is to diagnose the failures in these relationships by analyzing the detected deviation of the observations from the model's predictions [16–18,24].

We distinguish two phases of social diagnosis: (i) selecting who will carry out the diagnosis; (ii) having the selected agent(s) generate and disambiguate the diagnosis hypotheses, where a diagnosis hypothesis is a set of conflicting beliefs, and the agents that disagree about them (i.e., that hold these beliefs). These phases can be distinguished for any social diagnosis process.

To explore these phases concretely, we focus on teams of situated (behavior-based) agents [5,20,22,32]. The control process of such agents is relatively simple to model, and we can therefore focus on the core communications and computational requirements of the diagnosis.

Behavior-based agents dynamically switch between alternative behaviors (control modules, see Definition 1 below). Their selection of a controller is done as a result of examining their own internal beliefs, which are influenced by the external world. In such teams we expect to have faults due to the differences between the beliefs of the agents, e.g. because of their different sensing of the external world [16,17].

Definition 1. A *behavior* is a tuple $BHV = \langle VAL, PRE, TER, ACT \rangle$, where *VAL* is the identifier of the behavior, *PRE* and *TER* are sets of logic propositions respectively representing the pre-conditions (which, when satisfied, allow the behavior to be selected), and termination conditions (which terminate its selection if the conditions are satisfied), correspondingly. *ACT* stands for the actions associated with the behavior, which are executed (possibly in sequence, or repeatedly) once the behavior selected.

We model an agent as having a decomposition hierarchy of behavior nodes organized in an acyclic graph:

Definition 2. A *behavior hierarchy* is a directed acyclic graph of behaviors $BH = (V, E)$, where V represents the behavior nodes and E represents TASK-subtask decomposition relations between the behaviors. An edge $\langle b_1, b_2 \rangle \in E$ denotes that b_2 is a subtask of b_1. We then refer to b_2 as a child of b_1.

At any given time, the agent is controlled by a top-to-bottom path through the hierarchy, root-to-leaf:

Definition 3. A *behavior path* is a path of behaviors through the hierarchy, root-to-leaf, organized in a set $BP = \{b_1 \ldots . b_h\}$, where b_i represents behavior b in depth (level) i in the hierarchy. Only one behavior in each level of the hierarchy can be part of a behavior path.

Example 1. This example is taken from ModSAF, an application involving a virtual battlefield environment with synthetic helicopter pilots. In the example, a team of synthetic pilot agents is divided into two: scouts and attackers. In the beginning all teammates fly in formation looking for a specific way-point (a given position), where the scouts move forward towards the enemy, while the attackers land and wait for a signal. Once a signal is sent to the attackers, they take off and fly in formation toward the scouts who await the attackers.

Figs. 1 and 2 describe portions of the behavior hierarchies of the attacker and scout, respectively. A selection of the *Wait Point* behavior is possible only if its pre-condition: *battle-point scouted* = *false* is satisfied, and the behavior will be deselected when the termination-condition *battle-point scouted* = *true* will be satisfied.

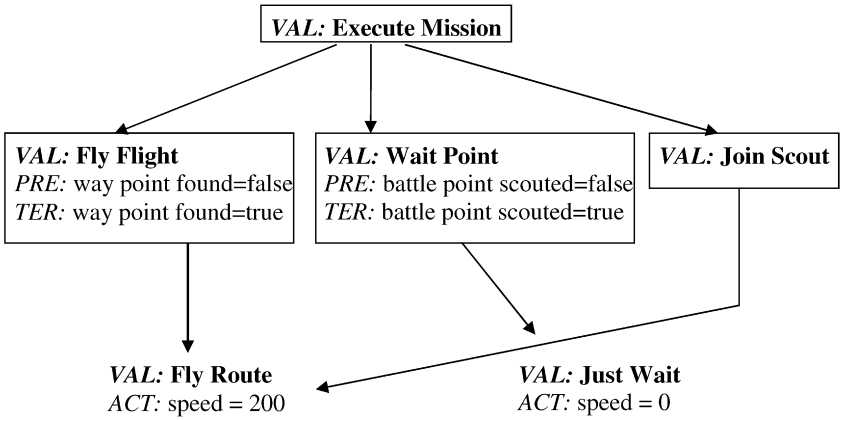


Fig. 1. Attacker's behaviors hierarchy (portion).

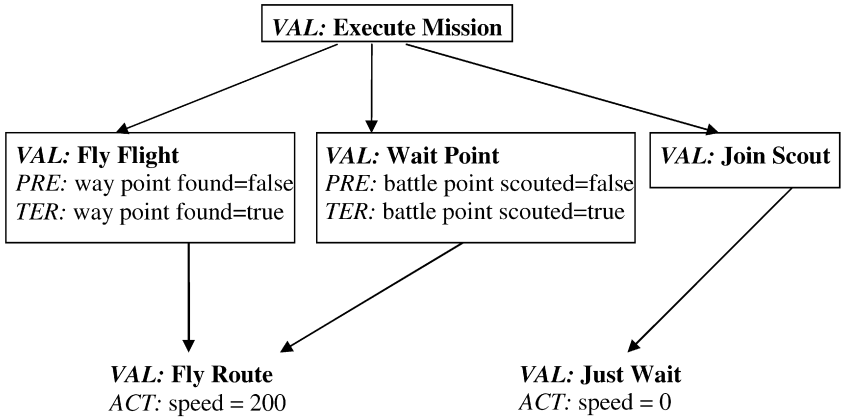


Fig. 2. Scout's behavior hierarchy (portion).

An agent uses a copy of the behavior hierarchy to track its current selections. Using its sensors it determines its beliefs and selects the behavior path which its pre-conditions are satisfied by its beliefs.

Definition 4. The *current state* of an agent is a pair $\langle BP, BL\rangle$ where BP represents its selected behavior path and BL the set of its beliefs. A belief is a pair $\langle p, v\rangle$, where p is a proposition and $v \in \{true, false\}$ is its truth value.

For instance, in Fig. 1 the behavior path of an attacker that is waiting after detecting the way point is: $BP = \{Execute\ Mission, Wait\ Point, Just\ Wait\}$ and it executes a landing action (speed $= 0$). The beliefs set that lead the attacker to select this behavior path is $BL = \{battle\ point\ scouted = false\}$.

Definition 5. A *team* $T = \{a_1 \ldots a_n\}$ is a set of n agents, and B is a set of agents' beliefs $B = \{b_1, \ldots, b_n\}$, where b_i is a set of q beliefs of agent a_i: $b_i = \{b_{i_1}, \ldots, b_{i_q}\}$.

We follow the convention of agent teamwork architectures, where agents coordinate through the joint selection and deselection of team behaviors, by using communications or other means of synchronization [10,15,33]. In other words, while each agent executes its own behavior hierarchy, selection of team behaviors within the hierarchy is synchronized. Team behaviors, typically at higher-levels of the hierarchy, serve to synchronize high-level tasks, while at lower-levels of the hierarchy agents select individual (and often different) behaviors which control their execution of their own individual role. Team behaviors are represented by boxes in Figs. 1 and 2.

Definition 6. A *team behavior* is a behavior which is to be selected and de-selected jointly for all the team: $\forall i, j \in T$, $Id_{i_x} = Id_{j_x}$, where T is a team, and Id_{i_x} is the identifier of team behavior node x of behavior hierarchy of agent a_i.

Disagreement between team-members is manifested by selection of different team behaviors, by different agents, at the same time, i.e. by synchronization failures [17]:

Definition 7. A *disagreement* exists when the following condition holds: $\exists i, j \in T$, such that $tb \in BP_i \wedge tb \notin BP_j$, where T is a team, tb is a team behavior, and BP_i represents the behavior path of agent a_i.

For instance, suppose we have a team of one scout and three attackers $T = \{S, A_1, A_2, A_3\}$ (see Figs. 1 and 2 for their behavior hierarchies). A disagreement (coordination fault) occurs if attacker A_1 selects to wait $BP = \{$*Execute Mission*, *Wait Point*, *Just Wait*$\}$, while the scout S selects to fly in formation $BP = \{$*Execute Mission*, *Fly Flight*, *Fly Route*$\}$. The fault occurs due to the selection of *Fly Flight* by the scout, in contrast with the selection of the behavior *Wait Point* by attacker A_1.

Disagreements can be detected by socially-attentive monitoring [17]. In this process all the agents monitor certain key agents using a behavior recognition algorithm. Once a monitor agent cannot find a matching between its own behavior and the behavior of the monitored key agent, it concludes that there is a fault. Since team behaviors are to be jointly selected (as discussed above), such a disagreement can be traced to a difference in the satisfaction of the relevant pre-conditions and termination conditions, e.g., agent A believes P, while agent B believes $\neg P$, causing them to select different behaviors. In the diagnosis process we investigate these conflicting beliefs:

Definition 8. *Conflicting beliefs* are a pair of two equal belief propositions, of different agents, which have contradictory values. $\langle b_{i_x}, b_{j_y} \rangle$ where $(i \neq j) \wedge (p \in b_{i_x} = p \in b_{j_y}) \wedge (v \in b_{i_x} \neq v \in b_{j_y})$.

It is these conflicting beliefs which the diagnosis process seeks to discover:

Definition 9. A *diagnosis* for a disagreement is a set of conflicting beliefs $D = \{d_1 \ldots d_m\}$ that accounts for the disagreement.

In the example of disagreement presented above, the belief of scout S is the pre-conditions of its behavior ($BP = \{$*Execute Mission*, *Fly Flight*, *Fly Route*$\}$), e.g. $\langle$*way point found*, *false*$\rangle$. The beliefs of attacker A_1 are the termination conditions of its previous behavior ($BP = \{$*Execute Mission*, *Fly Flight*, *Fly Route*$\}$) and the pre-conditions of its current behavior ($BP = \{$*Execute Mission*, *Wait Point*, *Just Wait*$\}$), e.g. $\langle$*way point found*, *true*$\rangle$ and $\langle$*battle point scouted*, *false*$\rangle$. A diagnosis may be that S believes that the way point was not yet found, while A_1 believes that it was. In other words, $D = \{$*way point found*$_S$ = *false*$\}$, *way point found*$_{A_1}$ = *true*$\}$.

4. Disambiguating diagnosis hypotheses

The design space of diagnosis algorithm is composed of two dimensions: First, the selection of one or more team-members to carry out the diagnosis (in the centralized case, only one, and in the distributed case, all or many); and second, the process by which the selected diagnosing agents disambiguate the diagnosis hypotheses to arrive at the correct diagnosis. The algorithms used for selecting the diagnosing agents may depend on the diagnosis process selected in the second phase (disambiguation). Thus for clarity of presentation, this section will first discuss the second phase; The next section (Section 5) discusses alternatives for agent selection.

Let us assume for now that one or more agents have been selected to carry out the diagnosis process. The agents must now identify the beliefs of their peers and then find the disagreements. We present two options: (i) the agents report their status to the diagnosing agents (Section 4.1); (ii) the diagnosing agents actively query agents as to the state of their beliefs 4.2. Obviously, these methods do not exhaust the range of options for the diagnoser selection and the diagnosis methods. For instance, there may be alternative methods which selectively utilize queries for the diagnosis process. However, the methods we chose highlight the extremes of the design space.

4.1. Reporting

Perhaps the simplest algorithm for detecting the beliefs of team-members is to have all team-members send their relevant beliefs to the diagnosing agent (the diagnosing agent can inform the team-members of the detection about a disagreement that triggers this communication). In order to prevent flooding the diagnosing agent with irrelevant information, each team-member sends only beliefs that are potentially relevant to the diagnosis, i.e., only the beliefs that are associated with its currently selected behavior path.

Upon receiving the relevant beliefs from all agents, the generation of the diagnosis proceeds simply by comparing all beliefs of team-members to find conflicting beliefs (e.g., agent A believes P, while agent B believes $\neg P$). Since the beliefs of the other agents are known with certainty (based on the communications), the resulting diagnosis must be the correct one. However, having all agents send their beliefs may severely impact network load.

The procedure `FIND_CONTRADICTION` (Algorithm 1), gets a set of agents' beliefs B (Definition 5) and returns a diagnosis D (Definition 9).

Algorithm 1. FIND_CONTRADICTION
(**input**: set of agents' beliefs B
output: diagnoses set Δ)
1: $D \leftarrow \emptyset$
2: **for all** $b_i \in B$ **do**
3: **for all** $b_j \in B$ where $i < j$ **do**
4: **for all** $b_{i_x} \in b_i$ **do**
5: **for all** $b_{j_y} \in b_j$ **do**
6: compare between the beliefs b_{i_x} and b_{j_y}
7: **if** $\langle b_{i_x}, b_{j_y} \rangle$ are conflicting beliefs (Definition 8) **then**
8: $D = D \cup \langle b_{i_x}, b_{j_y} \rangle$
9: return D

In the first line we initialize Δ—a set of diagnosis. In lines 2–3 the diagnosing agent goes over the belief sets of every pair of agents, in order to compare between their beliefs (b_i is the belief set of agent a_i and b_j is the belief set of agent a_j). Then in lines 4–5 the diagnosing agent goes over the beliefs in the set of agent a_i, comparing it to the belief set of agent a_j. In lines 6–8, the diagnosing agent compares between every pair of beliefs. If they have the same proposition but different truth values, it add these conflicting beliefs to the diagnosis set D, associated with their agents.

Let us analyze the runtime and communications complexity of Algorithm 1. Since both runtime and communications are affected by the number of beliefs in the system, we begin by first examining the number of beliefs of a single agent.

Each behavior has a number of associated beliefs (both the pre-conditions and termination conditions). We denote the worst case for the number of beliefs associated with a single behavior by b. At any given moment, a single agent executes a top-to-bottom path through the hierarchy, root-to-leaf. In the worst case (a degenerate behavior hierarchy) this top-to-bottom path is of length $O(m)$, where m is the size of the behavior-hierarchy (i.e., the number of behaviors in the hierarchy). Thus the total number of beliefs for a single agent is therefore $O(mb)$ in the worst case. In a best case scenario, the length of the top-to-bottom path is the height of a perfectly balanced behavior hierarchy, i.e., $O(\log m)$.

There are n agents in the team. The agents send their beliefs to the diagnosing agent. Under the assumption that each belief message is identical in size; the total number of messages in the worst case is equal to the total number of beliefs communicated by the n agents, $O(nmb)$ (in the best case, the number of messages would be $O(nb \log m)$). The worst-case runtime in this case is where the diagnosing agent compares each agent's beliefs with all others'. Therefore the runtime complexity in the worst case is $O(n^2 m^2 b^2)$.

The complexity of this process can be improved by arranging the beliefs in a sorted order. Instead of comparing between the beliefs in double loop (lines 3–4 in Algorithm 1), we could first lexicographically sort the beliefs (before the loop process) according to the propositions, and then compare between the beliefs, linearly. The sorting process' runtime for each agent would be $O(mb \log(mb))$ and for n agents it would be $O(nmb \log(mb))$. Once the beliefs are sorted the complexity of the comparison process is $O(n^2 mb)$. So, the total complexity in the worst case is

$O(nmb\log(mb) + n^2mb)$. In teams where the number of agents is scaled-up, we expect $n \gg mb$ so the complexity would be $O(n^2bm)$.

Example 2. Example 1, (Figs. 1 and 2) assume the scout was chosen to make the diagnosis then the attackers send their beliefs after the transference from {*Execute Mission, Fly Flight, Fly Route*} to {*Execute Mission, Wait Point, Fly Route*}, to the scout. See in Fig. 1 that the beliefs of A_1 and A_2 are: {*way point found* = *true* ∧ *battle point scouted* = *false*}, a total of four beliefs were sent by communication. The belief of the scout is: {*way point found* = *false*}. Once the scout has the beliefs of all the agents, it compares between them and finds the contradiction. In our example, the diagnosis is that the attackers' belief is: {*way point found* = *true*}, in contrast to the scout's belief: {*way point found* = *false*}.

4.2. Querying

In the previous algorithm the agents send all the beliefs that are associated with their behaviors. However, some of these beliefs may not be necessary for the diagnosis. We thus propose a novel selective monitoring algorithm, in which the diagnosing agent controls the communications, by initiating targeted queries which are intended to minimize the amount of communications. To do this, the diagnosing agent builds hypotheses as to the possible beliefs held by each agent, and then queries the agents as necessary to disambiguate these hypotheses.

Querying proceeds in three stages (Fig. 3):

(1) *Behavior recognition*: the diagnoser observes its peers and uses a behavior recognition process (see below) to identify their possibly-selected behavior paths, based on their observed actions.
(2) *Belief recognition*: based on the hypothesized behavior paths it further hypothesizes the beliefs held by the teammates (which led them to select these behavior paths, by enabling sets of preconditions and termination conditions).
(3) *Querying*: the diagnoser queries the diagnosed agents as needed to disambiguate between these belief hypotheses.

Once it knows about the relevant beliefs of each agent, it compares these beliefs to detect contradictory beliefs which explain the disagreement in behavior selection.

4.2.1. Behavior recognition

This process begins with *RESL*, a previously-published behavior recognition algorithm [17], presented here briefly as a reminder. Under the assumption that each agent has knowledge of all the possible behavior paths available to each team-member, i.e., their behavior path library (an assumption commonly made in plan recognition), each observing agent creates a copy of the fully-expanded behavior hierarchy for each of its teammates. It then matches observed actions with the actions associated with each behavior. If a behavior matches, it is tagged. All tagged behaviors propagate their tags up the hierarchy to their parents (and down to their children) such as to tag entire matching paths:

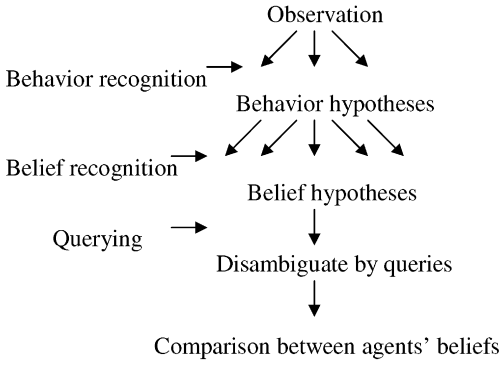


Fig. 3. Querying process for a single agent.

These signify the behavior recognition (plan recognition) hypotheses that are consistent with the observed actions of the team-member.

Example 3. In Example 1 (Figs. 1 and 2), a scout and two attackers are teammates carrying out a mission. They had flown in formation (*Fly Flight* behavior) for a while, when the attackers, say A_1 and A_2, landed (*Wait Point* behavior), while the scout continued to fly since it still did not find the way-point and so continued to execute the *Fly Flight* behavior.

Suppose A_1 monitors A_2 and the scout. It recognizes that the scout's speed is 200, so it can hypothesize (according to the above behavior recognition algorithm) that the scout is executing either: {*Execute Mission, Fly Flight, Fly Route*} or {*Execute Mission, Wait Point, Fly Route*}. In addition, A_1 concludes that A_2 is executing {*Execute Mission, Wait Point, Just Wait*} since its speed is 0. A_1 can not detect the fault, since its own behavior matches A_2's behavior and one of the behavior hypotheses of the scout. On the other hand, once the scout monitors A_1 and A_2, it recognizes that their speed is 0, so it concludes that they are executing {*Execute Mission, Wait Point, Just Wait*}, in contrast to its own behavior {*Execute Mission, Fly Flight, Fly Route*}. It can conclude that there is a fault: A_1 and A_2 selected {*Execute Mission, Wait Point, Just Wait*}, while it selected {*Execute Mission, Fly Flight, Fly Route*}.

The next phase is to identify the reasons for the difference in the selection of the behaviors. The diagnosing agent should disambiguate the correct behavior path of each teammate among its behavior path hypotheses, and the beliefs that account for its selection. These steps are discussed in the next section (Section 4.2.2).

4.2.2. Belief recognition

Once the hypotheses for the selected behavior path of an agent are known to the observer, it may infer the possible beliefs of the observed agent by examining the pre-conditions and the termination conditions of each hypothesized behavior path. To do this, the observer must keep track of the last known behavior path(s) hypothesized to have been selected by the observed agent. As long as the behaviors remain the same, the only general conclusion the observer can make is that the termination conditions for the selected behavior paths have not been met. Thus it can infer that the observed agent currently believes the negation of the termination conditions of selected behavior paths.

When the observer recognizes a transition from one behavior path to another, it may conclude (for the instance in which the transition occurred) that the termination conditions of the previous behavior path, and the pre-conditions of the new behavior path are satisfied. In addition, the termination conditions of the new behavior path must not be satisfied; otherwise this new behavior path would not have been selected. Therefore, the beliefs of the observed agent (at the moment of the transition) are: (*termination conditions of last behavior path*) $\wedge$ (*pre-conditions of current behavior path*) $\wedge \neg$(*termination conditions of current behavior path*).

We use V_i^t to denote the set of behavior path hypotheses of agent a_i at time t. We use $PRE(V_{i_j}^t)$ to denote the set of precondition propositions and their truth value. We use $TER(V_{i_j}^t)$ to denote the set of termination propositions and their truth value. F_i^t denotes a set of *belief* hypotheses of agent a_i at time t.

The procedure `BELIEF_RECOGNITION` (Algorithm 2) receives as input the current-time V_i^t (as generated by the behavior recognition process), and the previous behavior path hypothesis set V_i^{t-1} and returns the belief hypotheses set F_i of the same agent (a_i). As mentioned above, the observer has knowledge of the behavior-hierarchy of the observed agents, so the procedure could get as input the behavior path hypotheses of the observed agents.

Algorithm 2. BELIEF_RECOGNITION
(**input**: V_i^t, V_i^{t-1}
output: belief hypotheses set F_i)
1: $F_i^t \leftarrow \emptyset$
2: **for all** $v \in V_i^t$ **do**
3: **for all** $r \in V_i^{t-1}$ **do**
4: $F_i \leftarrow F_i \cup TER(r) \cup PRE(v) \cup \neg TER(v)$
5: return F_i

In the first line the set of the belief hypotheses is initialized as empty set. In line 2–3 the diagnosing agent goes over the current behavior hypotheses against the previous behavior hypotheses. In line 4 it generates the belief hypothesis

as a result of the union of the termination conditions of the previous behavior hypothesis and the pre-conditions of the current behavior hypothesis and the termination conditions of the current behavior hypothesis.

Example 4. Continuing Example 3, agent A_1 can infer the beliefs of the other agents as follows. As shown above, the behavior path hypotheses of the scout are: $V^t_{scout} = \{\{Execute\ Mission, Fly\ Flight, Fly\ Route\}, \{Execute\ Mission, Wait\ Point, Fly\ Route\}\}$. As a result of belief recognition process we obtain its beliefs hypotheses: $F_{scout} = \{\{way\ point\ found = false\}, \{way\ point\ found = true, battle\ point\ scouted = false\}\}$. The first belief hypothesis is derived from the first behavior path hypothesis while the second belief hypothesis is derived from the second behavior path hypothesis. In reference to agent A_2, its behavior path hypothesis is $V_{A_2} = \{Execute\ Mission, Wait\ Point, Just\ Wait\}$, therefore its belief is: $F_{scout} = \{way\ point\ found = true, battle\ point\ scouted = false\}$.

Let us analyze the runtime and communication complexity of Algorithm 1. As mentioned above an observer agent infers the beliefs of the other agents by a belief recognition process. The runtime complexity of this process depends on the number of the agents' beliefs in a single path, the number of behavior path hypotheses, and the number of agents:

(1) *The number of agent's beliefs in a single path*. The number of agent's beliefs per behavior path in the worst case has already been shown to be $O(bm)$, where m is the number of behaviors and b is the number of beliefs per behavior. But through belief recognition we combine the termination conditions of the previous behavior path with the pre-conditions and termination conditions of the current behavior path thus the number of beliefs is $O(2bm) = O(bm)$. Each belief proposition may be true or false, therefore the number of possible belief combinations per behavior path in the worst case is $O(2^{2bm}) = O(m^{2b})$.
(2) *The number of behavior path hypotheses*. Suppose r denotes the number of behavior path hypotheses in the behavior hierarchy k-ary tree, (where k designates the branching factor, i.e., the number of children of each behavior). Then the number of possible paths is limited by the number of leaves. The number of leaves is at most $(m - m/k)$. It is likely that $r \ll m - m/k$ since only a few of the path possibilities of the hierarchy are indeed possible paths for a certain recognized behavior.
(3) *The number of agents*. This process is repeated for each observed agent so the runtime complexity in the worst case is: $O(nrm^{2b})$.

The belief recognition process does not involve any communication, so we do not present the communication complexity for this process.

4.2.3. Targeting disambiguation queries

Once the belief hypotheses are known, the agent can send targeted queries to specific agents in order to disambiguate the hypotheses. The queries are selected in a manner that minimizes the expected number of queries. Intuitively, the agent prefers to ask first about propositions whose value, when known with certainty, will approximately split the hypotheses space.

For instance, suppose there are four hypotheses: $H1 = \{a, b, c, \neg h\}$, $H2 = \{a, b, d, \neg k\}$, $H3 = \{a, e, \neg m\}$, $H4 = \{f, g, \neg p\}$. a occurs in three of the four hypotheses, therefore if the value of a is queried and the response is $a = true$, then the three hypotheses that contain a are still active. On the other hand, b appears only in two of the hypotheses, and so it splits the hypotheses space. If $b = true$ then hypotheses $H1$ and $H2$ are active, and if $b = false$ the two other hypotheses are active. The other beliefs have one occurrence, therefore, like a, they divide the space to two unequal parts. In the best case only one hypothesis will be active, but in the worst case three hypotheses will be active.

Let us analyze the minimal number of queries necessary to disambiguate the hypotheses. A brute-force approach would have us evaluate the consequences of any sequence of queries to determine the optimal number of queries, but the computational complexity of this procedure is combinatorial in the number of beliefs.

Instead, we use a greedy one-step look ahead strategy based on entropy, similarly to its use in the "twenty questions" problem. The entropy function is taken from the information theory [31]:

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

Entropy(S) calculates the entropy of belief S. c represents the number of values of belief S, and p_i is the proportion of S belonging to value i. The entropy function varies between 0 and $\log_2 c$. The entropy is close to the minimum (0), when the distribution of the values of belief S is not uniform. The more the entropy is close to the maximum, the more the distribution is uniform.

In our case each belief proposition has three possible values: *true*, *false*, *don't care*, and the maximal entropy is $\log_2 3 = 1.58$ (when the hypotheses space is distributed uniformly by a belief query). In each step we want to query as to the belief whose value will split the hypotheses space as uniformly as possible to different classes. Then, every remaining hypothesis is equally likely, which means that the next query is expected to leave only $1/3$ of the hypotheses. In theory, if all queries equally divide remaining hypotheses to three groups, there will be only $O(\log x)$ queries, where x is the number of belief hypotheses. In the worst case, a total of $x - 1$ queries would be necessary, and in the best case, only one.

Example 5. In Example 4, when agent A_1 models the others it recognizes that A_2 has only a single belief hypothesis, so its beliefs are known to A_1 without any query. The scout has two belief hypotheses, therefore only one query is required to disambiguate between them. The two hypotheses are: {*way point found* = *false*} and {*way point found* = *true*, *battle point scouted* = *false*}. The probability of *way point found* = *false* as well as of *way point found* = *true* is 0.5 since they occur one time in two hypotheses, the probability of *way point found* = *don't care* is 0, therefore, $E(\textit{way point found}) = -(-(0.5\log_2 0.5) - (0.5\log_2 0.5) - (0\log_2 0)) = 1$. The probability of *battle point scouted* = *false* is 0.5 as well as the probability of *battle point scouted* = *don't care* (since in the first hypothesis this belief does not appear), and the probability of *battle point scouted* = *true* is 0. Therefore, $E(\textit{battle point scouted}) = -(-(0.5\log_2 0.5) - (0.5\log_2 0.5) - (0\log_2 0)) = 1$. Both of the belief queries have the same entropy and therefore one of them is selected as a query to the scout arbitrarily. Assume *way point found* was selected and the response of the scout is *way point found* = *true*, then A_1 can conclude that the correct hypothesis of the scout is {*way point found* = *true* $\wedge$ *battle point scouted* = *false*}. Now it can find the diagnosis by comparing between the beliefs.

Once the belief hypotheses were disambiguated by querying, the diagnosing agent should compare between the beliefs of the agents. So we should add the runtime complexity of the comparisons between the beliefs as computed in Section 4.1: $O((nbm)^2)$. Thus overall runtime complexity of the querying algorithm in the worst case is: $O((nrm^{2b}) + ((nbm)^2)$.

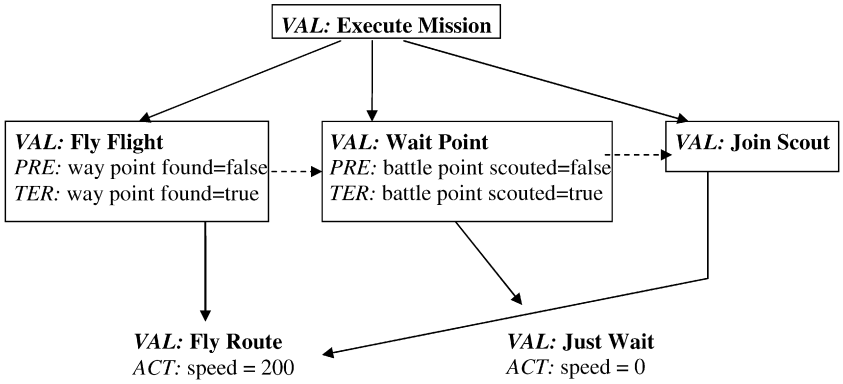
The communication complexity is influenced by sending targeted queries to specific agents in order to disambiguate their hypotheses. As described above the worst-case complexity of the number of queries to one observed agent is the number of beliefs, $O(bm)$. The queries are sent to all the observed agents, so the messages transmission complexity in the worst case is: $O(nbm)$.

This complexity is similar to that of reporting algorithm (Section 4.1) where each agent sends its beliefs to the diagnosing agent ($O(nbm)$). But while in the reporting algorithm this complexity is $O(nb\log m)$ in the best case, here the average case can be expected to have a reduced number of messages, and in the best case it could be even 1 message.

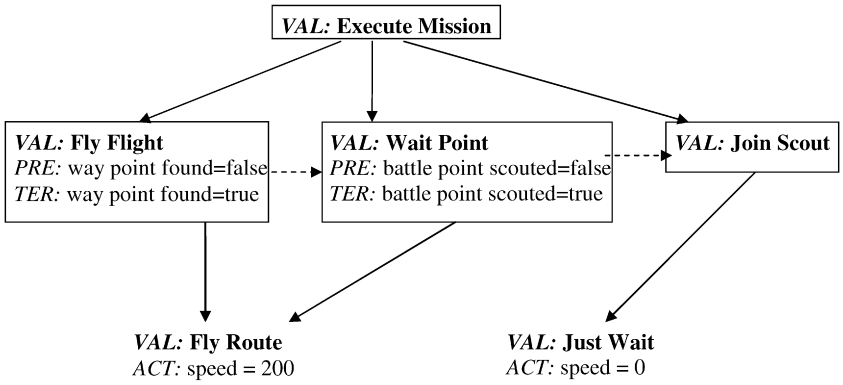
4.2.4. Precedence between behaviors

The analysis above assumes any behavior can potentially be selected at any point. However, in some domains there may be known temporal orderings between behaviors. These eliminate hypotheses from being generated, if they do not agree with the temporal order in which behaviors can be selected.

Example 6. In the ModSAF domain (see Example 1) suppose the permissible transitions are from *Fly Flight* to *Wait Point* and then to *Join Scout*, as seen in Figs. 4(a) and 4(b) (the dashed lines represent possible transitions). Let us examine the following case: An attacker and a scout fly in formation in *Fly Flight* behavior, when a fault is detected. Suppose the scout makes the diagnosis. The attacker's speed is 200, so the scout can conclude, according to the behavior recognition process, that observed behavior paths are either {*Execute Mission*, *Fly Flight*, *Fly Route*} or {*Execute Mission*, *Join Scout*, *Fly Route*}. However, the transition from *Fly Flight* to *Join Scout* is impossible because the attacker could not have gone from *Fly Flight* to *Join Scout* directly without passing through *Wait Point*. So, with



(a) *Attacker*



(b) *Scout*

Fig. 4. Agent behavior hierarchies, with ordering information.

certainty it concludes that attacker's behavior path is {*Execute Mission, Fly Flight, Fly Route*}. Obviously, it is better to disambiguate the beliefs of the attacker when we have only one behavior path hypothesis.

To summarize, we presented two algorithms for generating and disambiguating the social diagnosis hypotheses: *reporting*, in which all the agents send their beliefs that are associated with their selected behavior paths to the diagnosing agent, and *querying*, in which the diagnosing agent models the others by using the belief recognition process, and disambiguates their beliefs by querying them about certain beliefs. In the next section we will examine the question of who makes the diagnosis.

5. Selecting a diagnosing agent

Let us now turn to the first phase of social diagnosis, in which the agents that will carry out the diagnosis are selected. Several techniques are available. First, a design-time selection of one of the agents is the most trivial approach. Since the pre-selected agents do not necessarily know when a failure is detected (a different agent may have detected the failure), a failure state must be declared by the agents that have detected the failure, and communicated to the pre-selected agent, such that the pre-selected agents know when to begin their task. A second technique that circumvents this need is to leave the diagnosis in the hands of those agents that have detected the failure, and allow them to proceed with the diagnosis without necessarily alerting the others unless absolutely necessary.

We present a third approach, in which selection of the diagnosing agent is based on its team-members' estimate of the number of queries that it will send out in order to arrive at a diagnosis, i.e., the number of queries that it will send out in the disambiguation phase of the diagnosis (previous section). The key to this approach is for each agent

to essentially simulate its own reasoning in the second phase, as well as that of its teammates. Agents can then jointly select the agent with the best simulated results (i.e., the minimal number of queries).

Surprisingly, all agents can make the same selection without communicating, using a recursive modeling technique in which each agent models itself through its model of its teammates. This proceeds as follows. First, each agent uses the belief recognition algorithm to generate the belief hypotheses space for each team-member other than itself. To determine its own hypothesis space (as it appears to its peers), each agent uses recursive modeling, putting itself in the position of each one of its teammates and running the belief recognition process described above with respect to itself.

Under the assumption that all agents utilize the same algorithm, and have access to the same observations, an agent's recursive model will yield the same results as the modeling process of its peers. At this point, each team-member can determine the agent with the minimal number of expected queries by using the strategy discussed in Section 4.2. In order to guarantee an agreement on the selected agent, each team-member has an ID number, which is determined and known in advance. In case there are two agents or more with the same minimal number of expected queries, the agent with the minimal ID is selected. This entire process is carried out strictly based on team-members' observations of one another, with no communications other than an announcement of a disagreement.

Although these assumptions may seem too limiting, they hold in many cases. The algorithm assumes that all agents have access to the same observations of each other (i.e., to the actions selected by others). Obviously, such full observability is more likely in small teams, working closely together. However, this is really a question of the particular sensors used. For instance, a radar has a range of some kilometers, which allows determining the altitude and speed of otherwise unobservable agents. And the results of the radar would not necessarily differ between observers. Indeed, in the ModSAF domain, in which we evaluated our algorithms, these assumptions hold, and previous works exploited them [16,17].

The procedure `ESTIMATED_OPTIMAL` (Algorithm 3) gets the behavior path hypotheses of all the agents in a team T: $V^t = \{V_i^t\}$ (that is obtained by behavior recognition process), and their previous behavior path $V^{t-1} = \{V_i^{t-1}\}$. For each observed agent it calls to `BELIEF_RECOGNITION` algorithm which returns the belief hypotheses set of the observed agent (lines 1–2). For each belief hypotheses the algorithm calculates its estimated number of queries N_i (line 5), and then returns the agent that has the minimal number of queries.

Algorithm 3. ESTIMATED_OPTIMAL
(**input**: V^t, V^{t-1}
output: agent a_i)
1: **for all** Agents i **do**
2: $F_i \leftarrow$ BELIEF_ RECOGNITION(V_i^t,V_i^{t-1})
3: $F \leftarrow F \cup F_i$
4: **for all** $F_i \in F$ **do**
5: $N_i \leftarrow$ number of queries based on maximal information gain
6: $N \leftarrow N \cup N_i$
7: return $\{a_i \in T \mid \min(N_i)\}$

For instance, suppose there are three agents A, B and C. To determine the diagnosing agent, A models itself from B's perspective and considers the belief hypotheses that B has about A and C, given A's and C's observable actions. A also uses the belief recognition process described earlier to determine the number of belief hypotheses available about B's beliefs, C's beliefs, etc. It now simulates selecting queries by each agent, and selects the agent (say, C) with the minimal number of expected queries. B, and C also run the same process, and under the assumption that each agent's actions are equally observable to all, will arrive at the same conclusion.

Example 7. In Example 4, the scout models the attackers, and models the attackers modeling the scout (itself). The belief hypotheses of each one of the attackers are: $\{\textit{way point found} = \textit{true} \wedge \textit{battle point scouted} = \textit{false}\}$, so no query is requested. The result of belief recognition (by recursive modeling) of the scout on itself is: $\{\textit{way point found} = \text{false}\}$ or $\{\textit{way point found} = \textit{true} \wedge \textit{battle point scouted} = \textit{false}\}$, only a single query is needed. These process is carried out by the attackers too, and they into the same results. Obviously, in this case the scout is selected to diagnose, since it is expected to send no queries, while if one of the attackers would be selected it would send one query.

Table 1
Summary of the diagnosis methods in the design space

	Query	Report
Pre-selected	Method 5	Method 1, Method 3
Detectors	Method 2	N/A
Minimal queries	Method 4	Method 6

6. Social diagnosis methods

We have presented above a space of social diagnosis algorithms: Each algorithm operates in two phases, and we presented alternative techniques for each phase. For the selection of the diagnosing agent, we have the following methods: (i) rely on pre-selection by the designer; (ii) let the agents that detected the fault do the diagnosis; or (iii) choose the agent most likely to reduce communications (using the distributed recursive modeling technique described in Section 5). In terms of computing the diagnosis, two choices are available: Either have all agents communicate their beliefs to the selected agents (Section 4.1), or allow the diagnosing agents to actively query agents as to the state of their beliefs, while minimizing the number of queries as described above (Section 4.2).

These design alternatives define a space of diagnosis methods, corresponding to different combinations of the alternative algorithms in each phase. These are described in detail below. Table 1 summarizes the diagnosis possible methods in this space. The first column represents the algorithms of the selection of the diagnosing agent. The first row presents the algorithms of computing the diagnosis. The other cells in the table present the relevant methods according to the diagnosis space in the column and the row.

Method 1. The first design choice corresponds to arguably the most trivial diagnosis method, in which *all* agents are pre-selected to carry out the diagnosis. When a failure is detected (and is made known to all agents) each agent communicates all its relevant beliefs to the others so that each and every team-member has a copy of all beliefs, and therefore can do the diagnosis itself.

Method 2. Arguably, only a single agent really needs to have the final diagnosis in order to begin a recovery process. Thus in Method 2, the agents that detected the disagreement automatically take it upon themselves to carry out the diagnosis, unbeknown to each other, and their teammates (who did not detect the disagreement). Because their teammates may not know of the disagreement, the diagnosing agents cannot rely on their teammates to report their beliefs without being queried (in phase 2). Instead, they use the querying algorithm discussed in the previous section. Thus even other diagnosing agents will be queried.

Method 3. The next design choice corresponds to a diagnosis method in which the designer pre-selects one of the agents, arbitrarily. When a failure is discovered (and is made known to all agents), all team-members immediately communicate all their relevant beliefs to this pre-selected agent. While in Method 1 all the agents make the diagnosis and report their beliefs to each other, here only a single pre-defined agent makes the diagnosis.

Method 4. The fourth method attempts to reduce the communications. It uses the recursive modeling technique to have all team-members agree on which agent is to carry out the diagnosis (this requires the detection of the disagreement to be made known). Once the agent is selected (with no communications), it queries its teammates as needed.

Method 5. In this method the diagnosing agent is selected in advance by the designer, in contrast to Method 4. It uses the querying method in order to make the diagnosis.

Method 6. This method uses the selection of the most likely agent to reduce the communication as basis. However, once this agent is selected, the other agents do not wait for its queries, and instead report to it.

In principle, there is one more method in which the beliefs of all the agents are reported to the fault-detecting agents, which make the diagnosis (the empty box in Table 1). But, we do not experiment with this algorithm, since we already examine methods where the agents report their beliefs to the diagnosing agent(s): in Method 1 they report to

Table 2
Summary of evaluated diagnosis methods and their runtime and communication worst-case complexity

#	Selection	Disambiguation	Runtime	Communication
1	pre-selected (N)	$N \rightarrow N$ reporting	$O((nbm)^2)$	$O(n^2bm)$
2	detectors ($K \leqslant N$)	$K \rightarrow N$ querying	$O(nrm^{2b} + (nbm)^2)$	$O(n^2bm)$
3	pre-selected (1)	$N \rightarrow 1$ reporting	$O((nbm)^2)$	$O(nbm)$
4	minimal queries (1)	$1 \rightarrow N$ querying	$O(nrm^{2b} + (nbm)^2)$	$O(nbm)$
5	pre-selected (1)	$1 \rightarrow N$ querying	$O(nrm^{2b} + (nbm)^2)$	$O(nbm)$
6	minimal queries (1)	$N \rightarrow 1$ reporting	$O(nrm^{2b} + (nbm)^2)$	$O(nbm)$

all the other agents (all agents are pre-selected), and in Method 3 they report to a single pre-selected agent. A method where all agents report to some diagnosing agents (here, fault detecting) will be bounded from above and from below by Methods 1 and 3.

Table 2 summarizes the phases and the worst-case complexity of the different methods. Each method is presented in a different row. The first column shows the method (by #). The next two columns correspond to the different phases of the diagnosis process. The choice of algorithm is presented in each entry, along with a marking that signifies the number of agents that execute the selected technique for the phase in question. The last two columns present the worst-case complexity of the runtime and communication, correspondingly.

For instance row 2 should be read as follows: In Method 2, the agents selected to perform the disambiguation are those who detected the disagreement. K such agents exist (where K is smaller or equal than the total number of agents in the team, N), and they each execute the querying algorithm, such that K agents query N agents. In contrast, row 3 indicates that a single pre-selected agent executes the diagnosis, and it relies on reports from all agents to carry out the diagnosis, such that N agents report their beliefs to 1 agent.

We can see that the communication complexity of Methods 1 and 2 has an n^2 factor, in contrast to the other methods, since the diagnosis is carried out by several agents (up to n).

The runtime complexity divides between Methods 1 and 3, and the others. In Methods 1 and 3 the diagnosing agent(s) do not model other agents, but they obtain their beliefs by reporting, and disambiguate the diagnosis only by comparing between these beliefs, thus the complexity is polynomial in the number of agents and beliefs. On the other hand, in Methods 2, 4 and 5 the beliefs of the other agents are not reported to the diagnosing agent, so it infers them by a belief recognition process, which is exponential in the number of beliefs, and then diagnoses by querying.

In Method 6 the diagnosing agent makes the diagnosis by reporting an algorithm (whose complexity is similar to that of Methods 1 and 3). The selection of the diagnosing agent uses the `MINIMAL_QUERIES_DIAGNOSING_AGENT` algorithm. As shown above, this algorithm uses belief recognition, a process whose complexity is exponential in the number of beliefs.

7. Empirical evaluation

We now turn to an empirical evaluation of the diagnosis methods in two domains: One inspired by a real-world application (ModSAF), and one artificial (TEST).

7.1. Real-world domain

We examined the diagnosing methods on teams of behavior-based agents in a simulation of the ModSAF domain. We performed experiments in which Methods 1–6 were systematically tested on different failure cases, while we varied the number of agents, the roles of the agent and the disagreements between the agents.

(1) Number of agents: teams of two to thirty six agents.
(2) Roles of agent: for each n agents (1) one attacker and $n-1$ scouts; (2) $n-1$ attackers and one scout; (3) $n/2$ attackers and $n/2$ scouts.
(3) Disagreements: We systematically checked all possible disagreement cases for all team behavior paths. Thus overall, each method was tested 33,000 times, or, on average, 950 times for every team size.

Table 3
Results of diagnosing a specific failure case: one scout and nineteen attackers fly in formation in *Fly Flight* behavior, when eight of the attackers, A1–A8, transition to the *Wait Point* behavior while the other attackers and the scout continue to fly in formation

Method	Messages		Runtime(msec)		
	Belief	Failure	A1–A8, A10–A19	A9	Scout
1	912	380	2	2	2
2	608	0	23	23	23
3	46	380	0	2	0
4	30	380	20	25	20
5	31	380	0	25	0
6	46	380	21	21	21

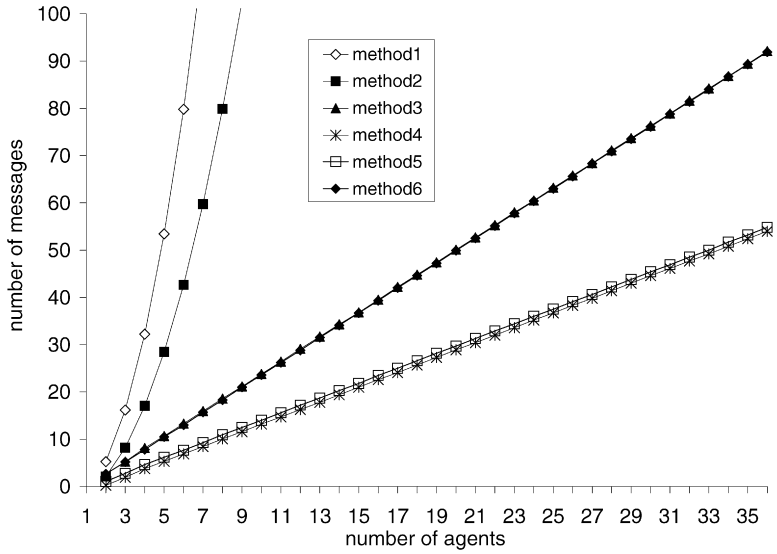


Fig. 5. ModSAF: Average number of messages per failure case.

We tested Methods 1–6 on all failure cases. In each failure case we recorded the number of messages sent by all the agents, and the runtime of the diagnosis process.

For example, in Table 3 we present the results of a single failure case, where one scout and nineteen attackers fly in formation in a *Fly Flight* behavior, when eight of the attackers, A1–A8, transition to the *Wait Point* behavior while the other attackers and the scout continue to fly in formation. The diagnosis is that A1–A8 detected the *way-point* (their belief is: *way point found* = *true*), while the other agents do not detect it (their belief is: *way point found* = *false*).

The first column in Table 3 reports the method used. The second column presents the number of messages sent reporting on beliefs, or querying about their truth (one message per belief). The third column reports the number of messages sent informing agents about the existence of failures (we assume point-to-point communications). The last columns summarize the runtime of each agent in milliseconds.

For instance, the number of messages reporting on beliefs for Methods 3 is 46, and 380 failure messages were sent (i.e., all the agents that detected the failure informed the others). The runtime of all the teammates for Methods 3 is 0 milliseconds, except for A9, which was selected in advance to disambiguated the beliefs in this case, and therefore took 2 milliseconds. On the other hand, the runtime of all the teammates for Method 4 was 20 milliseconds, except for A9, which disambiguated the beliefs in this case, and therefore took an additional 5 milliseconds (for a total of 25 milliseconds). In this example failure case, in Methods 3 and 5 we selected A9 in advance to make the diagnosis, since it was also the agent selected by the recursive-modeling process in Methods 4 and 6. Thus we can show the difference in runtime between these methods.

Figs. 5 and 6 summarize the results of the experiments. In both figures, the horizontal axis shows the number of agents in the diagnosed team. Fig. 5 presents the average number of belief messages Each data point (team size) is an

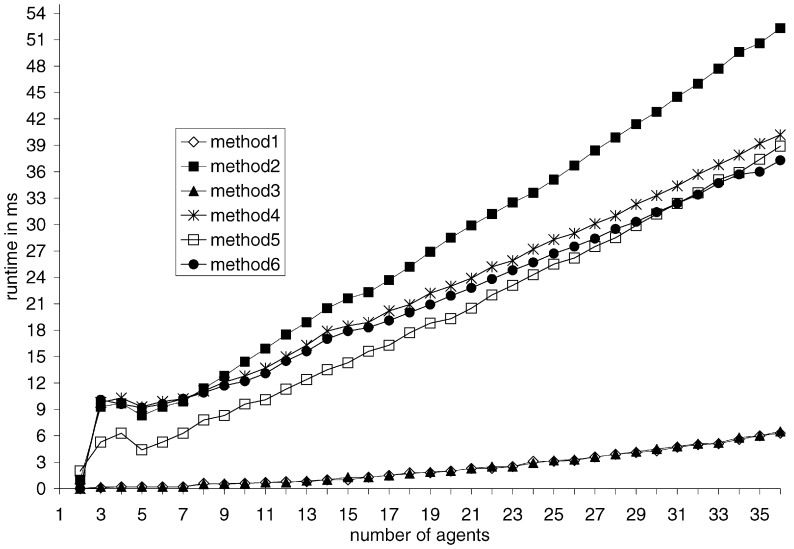


Fig. 6. ModSAF: Average runtime in milliseconds per failure case.

average over 950 runs (failure messages were ignored in the figure, since their effect is negligible). Fig. 6 presents the average runtime (in milliseconds) of those same tests. The runtime of each test was taken as the maximum of any of the agents in the test.

Both figures show grouping of the evaluated techniques. In Fig. 5 (number of messages), Methods 3–6 show a slow, approximately-linear growth (methods 3 and 6 cover each other), while Methods 1 and 2 show a much faster non-linear growth. In Fig. 6 (runtime), the grouping is different: Methods 1 and 3 grow significantly slower than the other methods (they overlap).

According to Fig. 5 the graphs of Methods 4 and 5 grow slower than the graphs of Methods 3 and 6, in contrast to their runtime performance (shown in Table 2). The reason for this is that Methods 4 and 5 use the querying algorithm, while Methods 3 and 6 use reporting algorithm. The communication complexity of the reporting algorithm in the best case is equal to the worst case: O(nbm), since each agent always sends its own beliefs. On the other hand, while the complexity of the worst-case of the querying algorithm is O(nbm), but in the best case it is O(1). Averaged over thousands of tests the results of the querying algorithm are better than the reporting algorithm. There are very small differences in Fig. 5 between Method 4 and Method 5, as well as between Method 3 and Method 6, despite the fact that in Methods 4 and 6 the minimal queries agent makes the diagnosis. The reason for this is that the number of messages is almost the same (when the pre-defined agent, or the minimal queries agent makes the diagnosis) is that in the ModSAF domain all the agents have almost the same behavior hierarchy. As a result, the number of belief hypotheses of the minimal queries diagnosing agent is almost the same as the other agents, and so the benefits of the minimal queries diagnosing agent are not recognizable. In Section 7.3, we will examine the benefit of the minimal queries diagnosing agent in more depth.

The first conclusion we draw from these figures is that runtime is mainly affected by the choice of a belief recognition process (Fig. 6, Table 2). Methods (here, Methods 1 and 3) that rely on the agents to report their relevant beliefs do not reason about the hypothesized beliefs of others. Therefore, their runtime is much smaller than methods (here, Methods 2, 4 to 6) which hypothesize about the beliefs of others (Methods 2, 4 and 5 use belief recognition in querying algorithm and Method 6 uses belief recognition in selecting the minimal-queries diagnosing agent). However, as Fig. 5 shows, the goal of reducing communications is actually achieved, as Methods 4 to 6 do indeed result in less communications then Method 3. The question of whether the cost in runtime is worth the reduction in communications is dependent on the domain.

We draw a second conclusion from Fig. 5. Despite the additional savings provided by the minimal queries diagnosing agent algorithm, the choice of a centralized diagnosing agent is the main factor in qualitatively reducing the number of messages sent, as well as in shaping the growth curve as the number of agents is scaled up. These results

contrast sharply with previous work in disagreement detection, in which distributed algorithms reduce communications [17].

7.2. Evaluation in controlled settings

The experiments above were constrained to the parameters of the original ModSAF domain, and thus limit the variance in the complexity of the agents to that found in the original application. In contrast, this section examines the diagnosis methods in settings where the number of beliefs (b, in Table 2), and the number of behavior path hypotheses (r) are controlled directly.

We created an artificial domain, TEST, in which we directly controlled the parameters influencing the diagnosis. Like ModSAF, this domain simulates teamwork in which the agents should agree on the selection of some pre-defined behaviors, concurrently. During the teamwork task the agents transition between the behaviors. The application simulates faults by controlling the selection of the behaviors by the agents, and causing disagreement in regard to the selected behavior. However, unlike the experiments in the MODSAF domain, here we control the number of beliefs and the number of behavior path hypotheses.

In the first set of experiments, we examine the influence of the number of beliefs (b in Table 2) on the runtime and the communication, of the diagnosis methods. For this goal, we performed experiments with a fixed number of agents (fifteen) and behavior path hypotheses (two), while the number of beliefs per behavior is varied from two to eight. The experiments tested with representative failure cases in a total number of 42.

The results of the communications in these experiments are presented in Fig. 7 and that of the runtime are presented in Fig. 8. Each data point is an average over six trials. The results in the graphs agree with the presented complexity analysis in Table 2. The communication complexity of all methods is approximately linear in the number of beliefs as shown in Fig. 7. However, the growth of Methods 4 and 5 is the slowest, since the agent selected to carry out the diagnosis sends a minimal number of queries (querying algorithm), while the graph of Method 1 grows much faster, since all the agents communicate with each other. On the other hand, Fig. 8 shows that the runtime of Methods 1 and 3 grows slowly and linearly (their runtime is closed to zero) while the other methods grow exponentially, in the number of beliefs, as predicted in Table 2.

In a second set of experiments, we examine the influence of the number of behavior path hypotheses (r in Table 2) on the runtime and the communication. In these tests the number of agents is fixed (thirty) as well as the number of

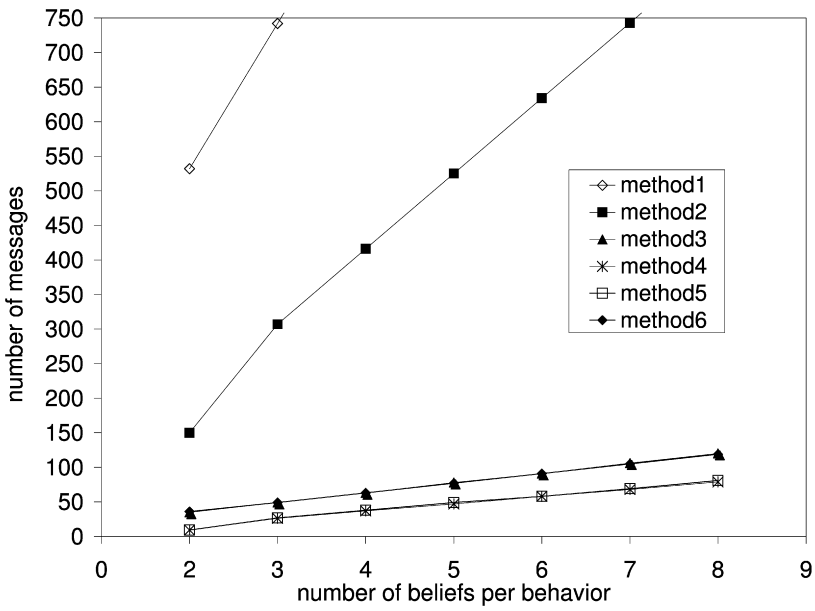


Fig. 7. TEST domain: Average number of messages per failure case when varying number of beliefs. The number of agents is fixed at fifteen, and the number of behavior paths hypotheses is fixed at two.

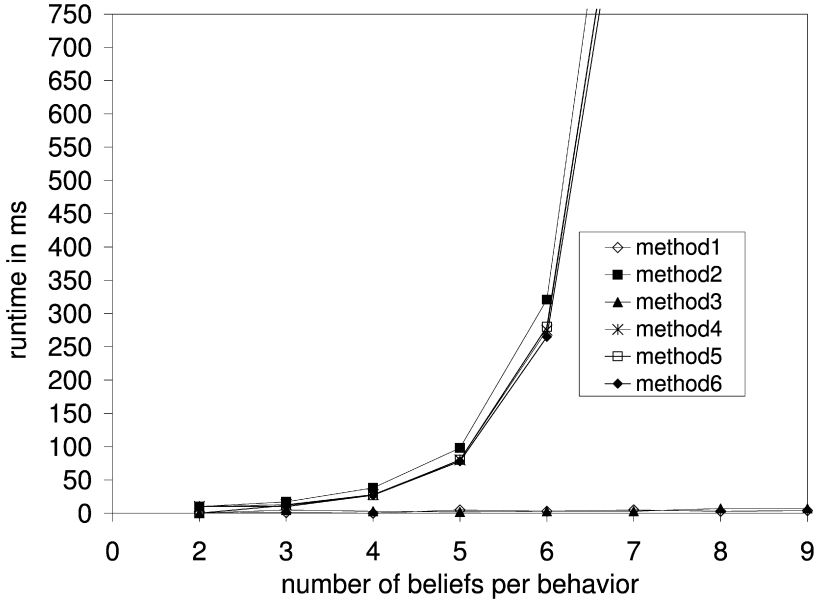


Fig. 8. TEST domain: Average runtime per failure case when varying number of beliefs. The number of agents is fixed at fifteen, and the number of behavior paths hypotheses is fixed at two.

beliefs per behavior (three), while the number of behavior path hypotheses is varied from two to ten. The experiments tested with representative failure cases in a total number of 315.

The results of the communications in the first tests are presented in Fig. 9. Each data point is an average over approximately 35 trials. Here again, the results in the graphs agree with the complexity analysis in Table 2. The number of behavior path hypotheses have no influence on Methods 1, 3 and 6, since in these methods the agents do not use behavior recognition (The graph of Method 1 is out of the scope of the y axis and it is constant at 3132 messages). However, Methods 2, 4 and 5 are affected by the number of behavior path hypotheses. According to the complexity in Table 2 we expect to obtain a linear curve in the number of messages (Fig. 9), and indeed in practice the graphs' growth is linear. The reason is that the graphs' growth depends on the partition of the belief hypotheses space. As discussed in Section 4.2, the selected query by the diagnosing agent divides this space. It is bounded from above with the number of beliefs which influences on the graph to be *linear* with the number of beliefs. On the other hand, it is bounded from below with the constant one which influences on the graph to be *constant* with the number of beliefs. Therefore, the graphs of Methods 2, 4 and 5 involve constant, linearly and logarithmic growths.

Fig. 10 shows the runtime results in these experiments. As expected, the runtime complexity of Methods 2 and 4 to 6 grow quickly as the number of the behavior path hypotheses grows since they use the querying algorithm where the runtime is affected by the number of hypotheses, while the graphs of Methods 1 and 3 are approximately fixed at close to 0 milliseconds, since they use the reporting algorithm which does not depend on the number of behavior path hypotheses.

7.3. Minimal-queries diagnosing agent

In a final set of experiments, we examine the efficiency of the selection of the minimal queries diagnosing agent, by comparing between Methods 4 and 5. Both of these methods use the querying algorithm to make the diagnosis, but while the diagnosing agent in Method 4 is expected to ask the minimal queries, in Method 5 it is selected in advance.

The comparison of these methods in the ModSAF domain yields very little difference, since the number of the involved beliefs in all behavior path hypotheses of the agents is almost the same. As a result, any difference between the number of queries of the minimal queries diagnosing agent and the other agents is very small.

In contrast, in the following experiments, we control the number of beliefs of the behaviors of each agent separately. The number of agents is fixed (four) as well as the number of behavior path hypotheses (two), while the number of beliefs per behavior is varied from two to ten *only* for a single random agent while it is fixed (three) for the other

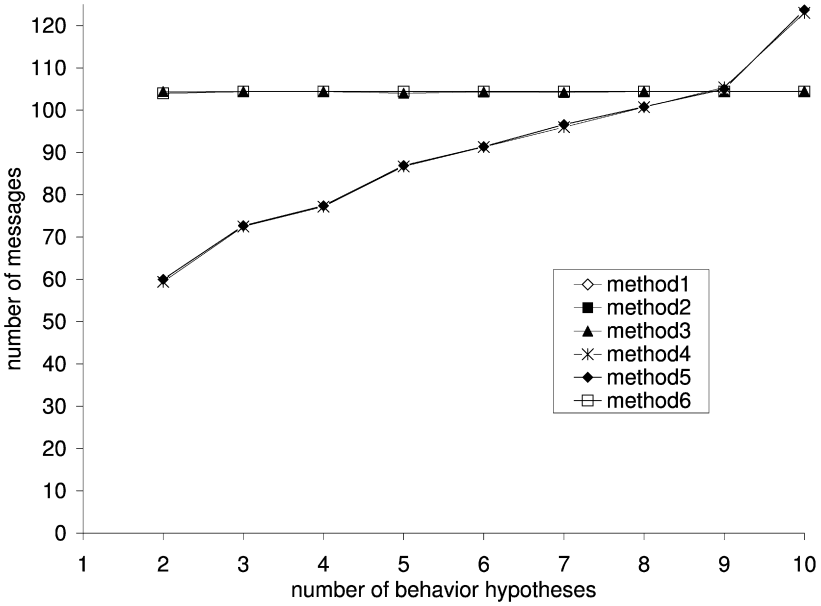


Fig. 9. TEST domain: Average number of messages per failure case when varying number of behavior path hypotheses. The number of agents is fixed at thirty, and the number of beliefs is fixed at three.

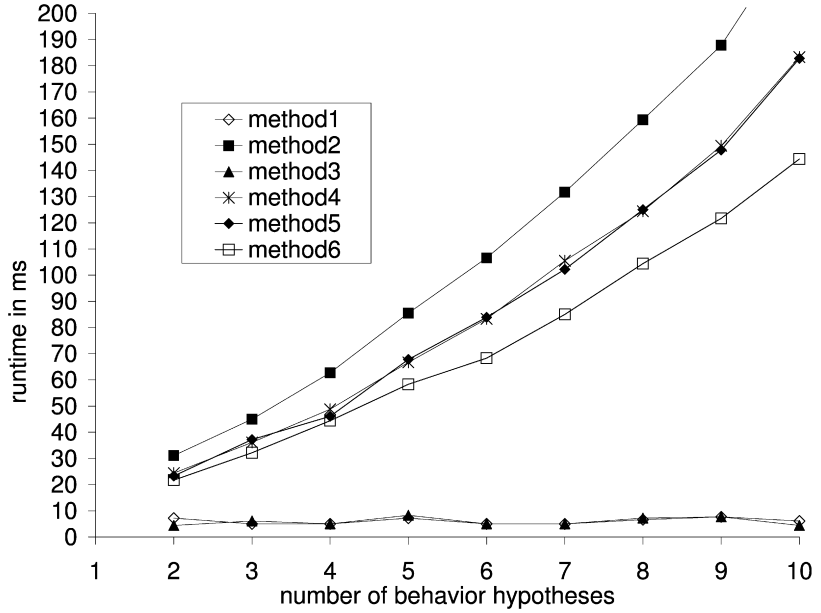


Fig. 10. TEST domain: Average runtime per failure case when varying number of behavior path hypotheses. The number of agents is fixed at thirty, and the number of beliefs is fixed at three.

agents. We expect that in Method 4, this agent will be selected to make the diagnosis according to the selection of the `MINIMAL_QUERIES_DIAGNOSING_AGENT` algorithm (Algorithm 3).

Fig. 11 summarizes the communication results in the experiments. Each data point is an average over six trials. The graph of Method 4 shows a fixed number of messages, independently of the growth of the number of beliefs. The reason is that in this method the agent with the most number of beliefs is selected to make the diagnosis, while the other agents are queried for their beliefs (they have the same number of beliefs in all trials). On the other hand, the graph of Method 5 is dependent on the number of beliefs of the random agent, since the diagnosing agent is selected

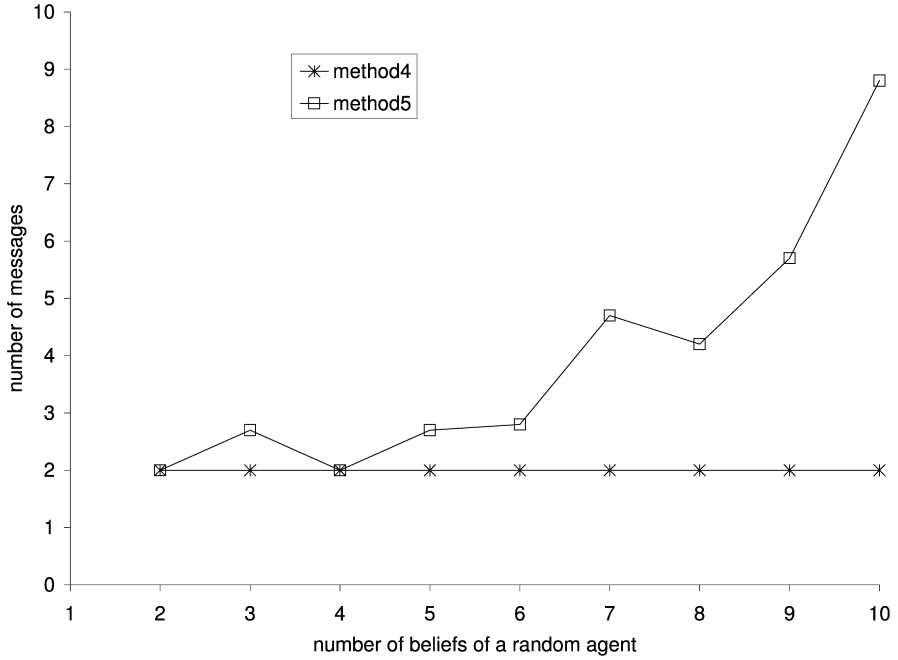


Fig. 11. TEST domain: Average number of messages per failure case when varying number of beliefs per behavior for a single random agent. The number of agents is fixed at four, and the number of behavior path hypotheses is fixed at two.

randomly and it may be not the minimal queries diagnosing agent, so the number of sent messages grows. We can see that although the general tendency of the graph of Method 5 is growing up, it decreases in the points of four and eight beliefs. The reason is that incidentally the diagnosing agent who was randomly selected has the most number of beliefs so it is predicted to query the minimal number of queries similarly to the minimal queries diagnosing agent in Method 4.

Fig. 12 summarizes the runtime results. We can see, as predicted, that both the runtime of Method 4 as well as of Method 5 grow exponentially, since both of them use belief recognition algorithm in the querying algorithm.

8. Summary and future work

In this paper we presented a novel design space for methods of social diagnosis. Each such method operates in two phases, and we have presented alternative techniques for each phase. For the selection of the diagnosing agent, we have the following methods: (i) *pre-selected agent*(*s*), relying on pre-selection by the designer; (ii) *fault detecting agent*(*s*), letting the agents that detected the fault do the diagnosis; or (iii) *minimal queries diagnosing agent*, choosing the agent most likely to reduce communications (using distributed recursive modeling). In terms of computing the diagnosis, two choices are available: (i) Reporting: Have all agents communicate their beliefs to the selected agents, (ii) Querying: Allow the diagnosing agents to actively query agents as to the state of their beliefs, while minimizing the number of queries. For each one of the methods we evaluated the complexity in terms of their communications and computation overheads.

The combination between these methods defines a space of six algorithms of diagnosing a team of behavior-based agents. We empirically and systematically evaluate the different combinations to draw general conclusions about the design of diagnosis algorithms.

A first conclusion is that centralizing the diagnosis disambiguation task is critical in reducing communications. The second conclusion is that techniques where agents reason explicitly about the beliefs of their peers are computationally inferior (in runtime) to techniques where agents do not reason about others. However, such computation does result in a slight reduction in communications.

Much work remains for future research. All methods find only the contradictions between agent beliefs, where the beliefs are derived directly from the hypothesized behavior paths. But in complex behavior-based control systems,

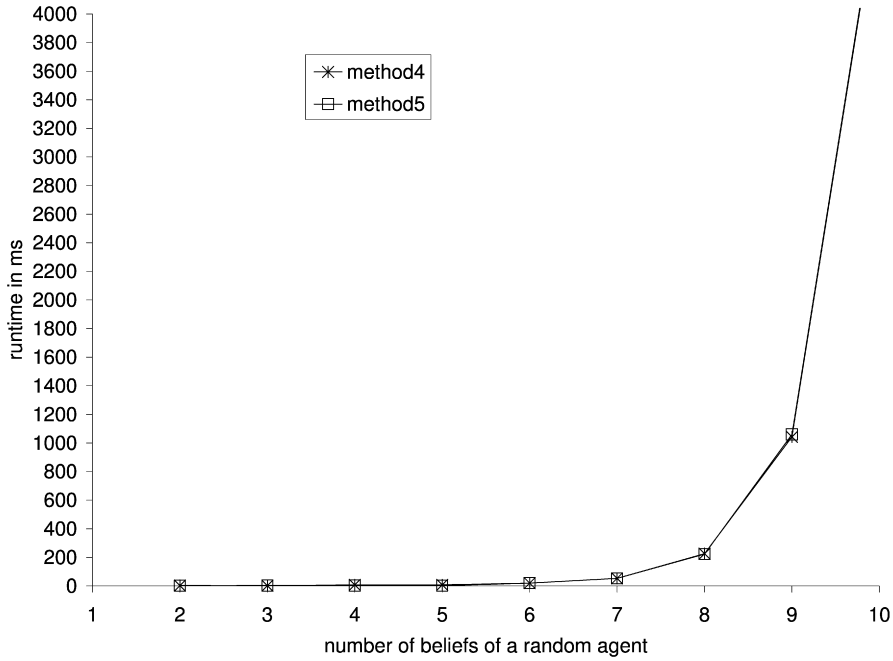


Fig. 12. TEST domain: Average runtime per failure case when varying number of beliefs per behavior for a single random agent. The number of agents is fixed at four, and the number of behavior path hypotheses is fixed at two.

chains of inference may lead from one belief to the next. Our system is currently not able to back chain through such inference pathways, and thus is unable to draw conclusions beyond the beliefs that are immediately tied to pre-conditions and termination conditions. We plan to tackle this challenge in the future.

Another challenge for the future is finding a diagnosis algorithm that combines between the most effective computation algorithm and the algorithm which uses minimal communication. In this paper, we have analyzed the selection of the minimal queries diagnosing agent and the querying algorithm in disambiguating the agent's hypotheses as contributed in reducing communications, however, they use a belief recognition process whose runtime is very high (Method 4). In the future we plan to improve the runtime of the belief-recognition process, and to achieve an efficient method both in terms of communication and computation.

Acknowledgements

This paper extends and builds on an IJCAI-2003 paper by the authors, titled "On the Design of Social Diagnosis Algorithms for Multi-Agent Teams" [11]. This research was supported in part by BSF grant #2002401. We thank Shmuel Tomi Klein and Moshe Koppel for helping with the minimal queries algorithm. We thank the anonymous reviewers for their useful suggestions on how to improve the paper. As always, thanks to K. Ushi and K. Ravit.

References

[1] P.R. Cohen, H.J. Levesque, Teamwork, Nous 25 (4) (1991) 487–512.
[2] R. Davis, W.C. Hamscher, Model-based reasoning: Troubleshooting, In: A.E. Shrobe (Eds.), Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence, 1988, pp. 297–346.
[3] J. de Kleer, B.C. Williams, Diagnosing multiple faults, Artificial Intelligence 32 (1) (1987) 97–130.
[4] C. Dellarocas, M. Klein, An experimental evaluation of domain-independent fault-handling services in open multi-agent systems, in: Proceedings of the Fourth International Conference on Multiagent Systems (ICMAS-00), 2000, pp. 95–102.
[5] R.J. Firby, An investigation into reactive planning in complex domains, in: American Association for Artificial Intelligence (AAAI-87), 1987, pp. 196–201.
[6] P. Fröhlich, I. de Almeida Mora, W. Nejdl, M. Schröder, Diagnostic agents for distributed systems, in: ModelAge Workshop, 1997, pp. 173–186.
[7] B.J. Grosz, S. Kraus, Collaborative plans for complex group actions, Journal of Artificial Intelligence Research 86 (1996) 269–358.

[8] W. Hamscher, L. Console, J. de Kleer (Eds.), Readings in Model-Based Diagnosis, Morgan Kaufmann Publishers, San Mateo, CA, 1992.
[9] B. Horling, V.R. Lesser, R. Vincent, A. Bazzan, P. Xuan, Diagnosis as an integral part of multi-agent adaptability, Technical Report CMPSCI Technical Report 1999-03, University of Massachusetts/Amherst, January 1999.
[10] N.R. Jennings, Controlling cooperative problem solving in industrial multi-agent systems using joint intentions, Artificial Intelligence Journal 75 (2) (1995) 195–240.
[11] M. Kalech, G.A. Kaminka, On the design of social diagnosis algorithms for multi-agent teams, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03), 2003, pp. 370–375.
[12] M. Kalech, G.A. Kaminka, Towards model-based diagnosis of coordination failures, in: American Association for Artificial Intelligence (AAAI-05), 2005.
[13] M. Kalech, G.A. Kaminka, Diagnosis of multi-robot coordination failures using distributed csp algorithms, in: American Association for Artificial Intelligence (AAAI-06), 2006.
[14] G.A. Kaminka, Y. Elmaliach, I. Frenkel, R. Glick, M. Kalech, T. Shpigelman, Towards a comprehensive framework for teamwork in behavior-based robots, in: IAS-8. 2004.
[15] G.A. Kaminka, I. Frenkel, Flexible teamwork in behavior-based robots, in: American Association for Artificial Intelligence (AAAI-05), 2005, pp. 1355–1356.
[16] G.A. Kaminka, M. Tambe, What's wrong with us? Improving robustness through social diagnosis, in: American Association for Artificial Intelligence (AAAI-98), Madison, WI, 1998, pp. 97–104.
[17] G.A. Kaminka, M. Tambe, Robust multi-agent teams via socially-attentive monitoring, Journal of Artificial Intelligence Research 12 (2000) 105–147.
[18] M. Klein, C. Dellarocas, Exception handling in agent systems, in: Proceedings of the Third International Conference on Autonomous Agents, May 1999, pp. 62–68.
[19] S. Kraus, S. Katia, A. Evenchik, Reaching agreements through argumentation: a logical model and implementation, Artificial Intelligence 104 (1–2) (1998) 1–69.
[20] M. Mataric, Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior, Trends in Cognitive Science 2 (3) (1998) 82–87.
[21] R. Micalizio, P. Torasso, G. Torta, On-line monitoring and diagnosis of multi-agent systems: a model based approach, in: Proceedings of European Conference on Artificial Intelligence (ECAI 2004), vol. 16, 2004, pp. 848–852.
[22] A. Newell, Unified Theories of Cognition, Harvard University Press, Cambridge, MA, 1990.
[23] L.E. Parker, ALLIANCE: An architecture for fault tolerant multirobot cooperation, IEEE Transactions on Robotics and Automation 14 (2) (1998) 220–240.
[24] D. Poutakidis, L. Padgham, M. Winikoff, Debugging multi-agent systems using design artifacts: The case of interaction protocols, in: Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-02), 2002, pp. 960–967.
[25] D.V. Pynadath, M. Tambe, N. Chauvat, L. Cavedon, Toward team-oriented programming, in: Proceedings of the Agents, Theories, Architectures and Languages (ATAL'99) Workshop, in: Intelligent Agents, vol. V, Springer-Verlag, 1999, pp. 77–91.
[26] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 (1) (1987) 57–96.
[27] N. Roos, A. ten Teije, A. Bos, C. Witteveen, Multi-agent diagnosis: an analysis, in: Proceedings of the Belgium-Dutch Conference on Artificial Intelligence (BNAIC-01), 2001, pp. 221–228.
[28] N. Roos, A. ten Teije, A. Bos, C. Witteveen, Multi-agent diagnosis with spatially distributed knowledge, in: Proceedings of the Belgium–Dutch Conference on Artificial Intelligence (BNAIC-02), 2002, pp. 275–282.
[29] N. Roos, A. ten Teije, C. Witteveen, A protocol for multi-agent diagnosis with spatially distributed knowledge, in: Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-03), July 2003, pp. 655–661.
[30] N. Roos, A. ten Teije, C. Witteveen, Reaching diagnostic agreement in multi-agent diagnosis, in: Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-04), 2004, pp. 1254–1255.
[31] C.E. Shannon, A mathematical theory of communication, Bell System Technical Journal 27 (1948) 379–693.
[32] M. Tambe, Implementing agent teams in dynamic multi-agent environments, Applied Artificial Intelligence 12 (2–3) (1998) 189–210.
[33] M. Tambe, Towards flexible teamwork, Journal of Artificial Intelligence Research 7 (1997) 83–124.