Diagnosis of Multi-Robot Coordination Failures Using Distributed CSP Algorithms

Meir Kalech[1] and **Gal A. Kaminka**[1] and **Amnon Meisels**[2] and **Yehuda Elmaliach**[1]
Computer Science Departments
[1]**Bar Ilan University, Israel** {*kalechm,galk,elmaley*}*@cs.biu.ac.il*
[2]**Ben Gurion University of the Negev, Israel** *am@cs.bgu.ac.il*

Abstract. With increasing deployment of systems involving multiple coordinating agents, there is a growing need for diagnosing coordination failures in such systems. Previous work presented centralized methods for coordination failure diagnosis; however, these are not always applicable, due to the significant computational and communication requirements, and the brittleness of a single point of failure. In this paper we propose a distributed approach to model-based coordination failure diagnosis. We model the coordination between the agents as a constraint graph, and adapt several algorithms from the distributed CSP area, to use as the basis for the diagnosis algorithms. We evaluate the algorithms in extensive experiments with simulated and real Sony Aibo robots and show that in general a trade-off exists between the computational requirements of the algorithms, and their diagnosis results. Surprisingly, in contrast to results in distributed CSPs, the asynchronous backtracking algorithm outperforms stochastic local search in terms of both quality and runtime.

1 Introduction

With increasing deployment of systems involving multiple coordinating agents or robots, there is growing need for diagnosing coordination failures. Coordination failures often lie at the boundaries between the agents and their environment, including other agents. For instance, a robot may send a message that another robot, due to an intermittent radio failure, did not receive. As a result, the two agents come to disagree on an action to be taken.

Previous work in diagnosis of coordination failures has focused on centralized methods for such diagnosis [5, 8, 1, 4]. Unfortunately, centralized methods suffer from key limitations: First, they can be computationally expensive in practice, in terms of communications and run-time. Second, they rely on a single diagnoser, and thus risk a single point of failure. Moreover, this assumes no communication limitations, e.g., range. Finally, they do not not necessarily exploit the different knowledge of different agents; e.g., an intended receiver faces difficulty detecting that a message to it was lost, where the sender may do it more readily. However, distributed methods that have been proposed, e.g., [10] do not address coordination failures.

This work takes a first step towards *distributed* model-based diagnosis of coordination (inter-agent) failures. Following [4] we model the coordination between the agents as a graph of concurrence and mutual-exclusion constraints on agents' actions. The basic idea of the diagnosis process is to compare the current observed actions of the agents to those that satisfy the coordination constraints. Deviations which cause the constraints to be violated lead to suspecting agents of being at fault. The diagnosis output includes the agents that deviate from the expected coordination (i.e., a minimal set of *abnormal* agents). Modeling the coordination as a constraint graph brings to bear solution methods from distributed constraint satisfaction (DisCSP) literature, as solutions to the constraint graph form the basis for diagnoses.

We present four distributed model-based diagnosis algorithms to compute the diagnosis, based on DisCSP algorithms. While the reasoning behind all is the same as outlined above, the algorithms differ from each other with respect to their expected run-time (based on DisCSP literature) and their completeness of the diagnoses (based on whether they find all or a single DisCSP solution). Two of the algorithms (based on *synchronous backtracking*) are expensive, but compute a complete set of minimal diagnoses . One algorithm, (*asynchronous backtracking*) is expected to be computationally cheaper, and guarantees computing a single diagnosis (though not necessarily minimal). The last algorithm (*distributed stochastic search*) is a local search algorithm that is not guaranteed to find a diagnosis, but is known to be highly effective (and cheapest of the above) in solving DisCSPs in practice [14, 15].

We evaluated the use of these algorithms in comprehensive experiments with a team of physical and simulated Sony Aibo robots, experiencing systematic coordination failures. We examined the computational requirements of the algorithms (i.e., their run-time and bandwidth usage), and the correctness of the diagnoses produced. We find that in general, synchronous backtracking methods that compute the entire space of minimal diagnoses are naturally more expensive than others, though they produced better diagnosis results. However, a surprising result is that, the local search algorithm (which typically outperforms asynchronous backtracking methods in DisCSPs) shows only mediocre results, both in terms of quality of the diagnosis, as well as in terms of computational requirements.

2 Related Work

Pencolé et al. [9] and Lamperti and Zanella [5] use a fault-model approach, where the distributed system is modeled as a discrete event system, and the faults are modeled in advance. The diagnoser infers unobservable fault events by computing possible paths in the discrete event system that match observable events. Horling et al. [3] and Micalizio et al. [8] use causal models of failures and diagnoses to detect and respond to multi-agent and single-agent failures. In contrast to these, we compute the diagnosis in a distributed fashion, and use model-based diagnosis with no fault models. In addition, a common theme in all of these is that they require pre-enumeration of faulty interactions among system entities. However, in multi-agent systems, these are not necessarily known in advance since they depend on the specific run-time conditions of the environment, and the actions taken by the agents.

Ardissono et al. [1] divide the system to sub-systems where every agent is responsible to its own sub-system. Instead of letting the agents compute the global diagnosis by exchanging information, the agents send only necessary information to a central diagnostic service by request. Kalech and Kaminka [4] propose centralized consistency-based and abductive diagnosis methods for diagnosis of coordination faults. In contrast, the methods we report on here are all distributed, and thus avoid the shortcomings of centralized methods. Moreover, in contrast to [4], we present here empirical results, where the previous work has only provided a theoretical analysis.

Roos et al. [10] presented model-based diagnosis methods for spatially distributed, where a set of n agents are responsible for diagnosing n sub-systems, respectively. Every agent makes a local diagnosis to its own sub-system and then all agents compute a global diagnosis. In order to build a global diagnosis set, each agent should consider the correctness of those inputs of its subsystem that are determined by other agents. Unlike our work, they assume that there are no conflicts between the knowledge of the different agents, i.e., that no coordination faults occur.

To date, only a few researchers use CSP methods to practically diagnose a system. Wotawa [12] makes use of the corresponding representation of the environmental models as constraint satisfaction problems. He shows how this representation can be used directly to derive explanations and diagnoses. To this goal, he models the system using cause-effect model, such that different solutions to the CSP are actually different explanations of the system, and the diagnoses are derived from them. Sachenbacher and Williams [11] extends this model to cope with constraint optimization problems over lattices, and with semiring-CSPs. Here again a satisfaction of constraints signifies an explanation to a fault. In contrast, we use constraints to model the ideal coordination relationships. Thus the diagnosis algorithm goal is to diagnose the violated constraints. In addition, previous systems are not distributed and the diagnosis is computed centrally, in contrast to our work.

3 The Social Diagnosis Problem

To present the distributed methods we develop in this paper, we first begin by briefly describing the model-based coordination diagnosis problem. We refer the reader to [4] for a detailed discussion and explanation.

Let T be a group of n agents, where each agent has a single action variable (easily extended to more variables) with domain d—the actions that can be selected by the agent (the actions are transformed to a set of boolean variables by applying completeness and mutual-exclusion formulas). The coordination between the agents is defined by constraints on the values of the agents' actions: A pair $\langle Agent1 = Value1, Agent2 = Value2\rangle$, represents a constraint between the action $Value1$ of $Agent1$ and the action $Value2$ of $Agent2$.

There are two kinds of constraints, which restrict the joint action selected by agents: *Concurrence* constraints (CCRN) signify that the agents must select their respective specific action values jointly, at the same time. *Mutual-exclusion* constraints (MUEX) signify that the specific actions must never be selected jointly, at the same time (example below). We define CG to represent the set of the constraints between the agents, and $CG(A_{ix}, A_{jy})$ to denote the constraint relating $\langle Agent_i = Value_x, Agent_j = Value_y\rangle$.

Given a set of constraints CG and a team T, we can define a multi-agent system description as a set of implications from the normality of the agents to the satisfaction of the coordination constraints.

Definition 1. A *multi agent system description (MASD)* is a set of implications from the normality of agents in a team T to CG. The meaning of the predicate $AB(\cdot)$ is that the corresponding agent is considered abnormal (failing).

$$\begin{aligned} MASD = \quad & \{\neg AB(A_i) \wedge \neg AB(A_j) \Rightarrow CG(AS_{ix}, AS_{jy}) \\ & |CG(AS_{ix}, AS_{jy}) \in CG \wedge A_i, A_j \in T\} \end{aligned}$$

Given a set S of the actions of the agents (set of boolean variables) and a set CG of the constraints between them, and assuming that all the agents did not fail (in terms of model-based diagnosis, are not abnormal), the system is inconsistent if the constraints are violated.

Definition 2. *Coordination Diagnosis Problem.* Given $\{T, MASD, S\}$ where T is a team of agents $\{A_1...A_n\}$, $MASD$ is a multi agent system description defined over T, and S is the set of the actions of the agents, then the *coordination diagnosis problem (CDP)* arises when

$$MASD \cup \{\neg AB(A_i) | A_i \in T\} \cup S \vdash \bot$$

This can imply a failure in the joint action selection, i.e., in coordination. In that case, the goal of the social diagnosis process is to find a **minimal diagnosis** set of abnormal agents that account for the failure; i.e., agents whose action selection we can change to cause the system to become consistent (in terms of CSP—that enable the satisfaction of all constraints). [1]. We seek a set of *minimal diagnoses*, where no proper subset of any of them is also a diagnosis.

To illustrate, assume a group of robotic space explorers, whose goal of is to slowly creep on a newly-discovered alien. To capture the alien, they must approach it from all sides in alternating steps: A bit from the left, then from the right, then again from the left, etc (Figure 1). To do this in coordinated manner, the robots divide into sub-teams of three that spread around the alien, each with a leader (C_i) and two followers (D_i), that move in formation using cameras to maintain distances and angles. A mission commander (B) directs the sub-team leaders, alternating commands for them to go and stop, as needed.

The robots must coordinate all through their mission. The sub-team leaders are coordinated with each other via the mission commanders' commands; and each sub-team's leader is coordinated with its followers using vision. Once a coordination failure(s) is detected, the mission must be suspended, in order to diagnose the failed robotic soldiers and then reestablish collaboration. A coordination failure could happen due to intermittent communication failures (between the mission commander and sub-team leaders) or due to a vision failure (a sub-team leader and its followers).

Focusing on a case with a single team, $T = \{B, C_1, D_1, D_2\}$, where D_1, D_2 are followers, C_1 sub-team leader, and B mission commander, we define the domain of the agents to be the actions go (g) or stop (s), $d = \{g, s\}$. The coordination constraints between the agents are:

CCRN: $\langle C_1 = g, D_1 = g\rangle, \langle C_1 = g, D_2 = g\rangle$
MUEX: $\langle B = s, C_1 = g\rangle$

[1] Note that we focus here on the abductive version of social diagnosis.

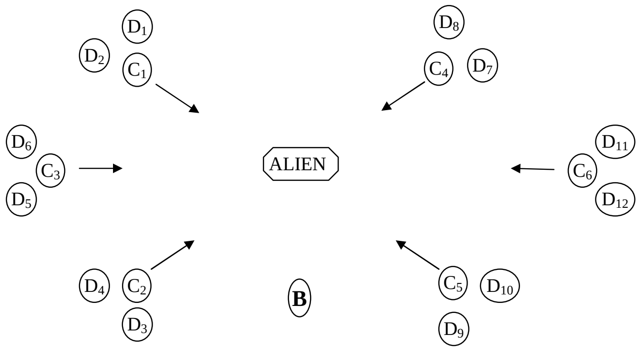

Figure 1. Example: Robotic space explorers tackling an alien.

We can represent the constraints between the agents in a coordination graph (Figure 2), where the vertices represent the values of the agents' variables, and the edges represent the constraints (solid edges mark concurrence constraints; dashed edges mark mutual exclusion).

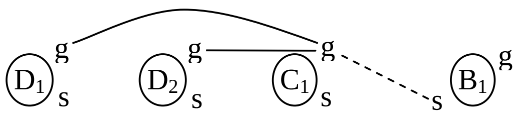

Figure 2. Coordination graph for mission commander B, sub-team leader C_1, and the two followers.

Assume follower D_1 thinks, due to a vision failure, that the sub-team leader C_1 stopped, selecting the action s, then the actual assignments are: $S = \{B = g, C_1 = g, D_1 = s, D_2 = g\}$, i.e. the constraint: $\langle C_1 = g, D_1 = g\rangle$ is violated. By finding solutions to the constraint graph, and comparing these to S, the agent can generate two minimal diagnoses: $\Delta_1 = \{D_1\}, \Delta_2 = \{B, C_1, D_2\}$. Δ_1 corresponds to the possibility that D_1 is wrong in its belief, Δ_2 corresponds to the possibility that everyone else is wrong.

4 Distributed Social Diagnosis

We present a distributed approach, where the agents find the satisfaction(s) and compute the diagnosis by exchanging information with each other. As in the centralized approach, computing the diagnosis is done by finding the satisfactions of the coordination graph, and contrasting these with actual values. As far as we know, there is no existing algorithm which finds a minimal satisfaction (in terms of minimal diagnosis), where no proper subset of the changed values could also satisfy the constraints. Thus the minimality goal is preserved only for some algorithms (see below).

In the next two subsections we propose four distributed algorithms to find the satisfactions and compute the diagnosis. All the algorithms use communication, therefore they work only in nonpermanent communication breakdowns. In permanent communication breakdowns neither distributed nor centralized approach will work. In the first subsection we present two algorithms for computing the complete set of minimal diagnoses, and in the following subsection we present two algorithms for computing an incomplete diagnosis which is not guaranteed to be minimal. As we shall see, these can offer an attractive alternative, despite their lack of guarantees.

4.1 Algorithms for Complete Minimal Diagnoses

In order to compute a complete set of minimal diagnoses, the agents must compute the whole satisfaction space of the system. We use a synchronous backtracking algorithm (SBT) to compute the satisfactions [13]. This algorithm is based on a distributed depth-first search. The agents are arranged in a static order. Every agent sends its possible values to its next agent. The receiving agent checks the compatibility of the former assignments with every value of its domain, separately. It returns backward a $nogood$ message upon inconsistency, or the partial assignments to the next agent, upon consistency.

In a system where the constraints between the agents are static, i.e. they do not change dynamically, the agents could compute all the satisfactions in advance (offline). During run-time, every agent keeps a copy of all solutions, using them to compute the diagnosis. We denote this method *SBT_OFF*. On the other hand, in systems where the constraints can change dynamically, the agents must compute the satisfactions, as well as the diagnosis, online. We denote this *SBT_ON*.

During diagnosis, every agent reports to the other agents the indexes of the satisfaction database in which that agent found an inconsistency. Every agent collects this information from the others and computes the diagnoses by dividing the agents according to the reported indexes. So as to produce minimal diagnoses, if a diagnosis set is a superset of another diagnosis, it is dropped.

In the previous example the values that satisfy the team variables are: $s_1 = (g, g, g, g)$ and $s_2 = (s, s, s, s)$ (corresponding to the order of the agents (B, C_1, D_1, D_2)). Assume that follower D_1 failed due to a failure in its vision, which caused it to select the action stop (s). The agents exchange the satisfaction indexes in which they found an inconsistency. B sends index 2 since its current value is g which is not equal to its expected value in satisfaction s_2. In the same manner, C_1 sends index 2, D_1 sends index 1 and D_2 sends index 2. Once an agent accepts this information from all the others, it divides them according to the indexes, to form two diagnoses: $\Delta_1 = \{D_1\}$ and $\Delta_2 = \{B, C_1, D_2\}$.

In the same manner, we take care also of cases which the minimal diagnosis is a subset of another diagnosis. For example, assume the values $s_3 = (g, s, s, s)$ also satisfies the team variables. Then the indexes sent by the agents are as the following:

$B : 2$
$C_1 : 2, 3$
$D_1 : 1$
$D_2 : 2, 3$

Dividing the agents according to the indexes produces the following diagnoses: $\Delta_1 = \{D_1\}$, $\Delta_2 = \{B, C_1, D_2\}$ and $\Delta_3 = \{C_1, D_2\}$. However, $\Delta_3 \subseteq \Delta_2$ so Δ_2 is not a minimal diagnosis. The final minimal diagnosis sets are: $\Delta_1 = \{D_1\}$ and $\Delta_3 = \{C_1, D_2\}$.

The first stage, of building the satisfaction database, involves an exponential number of messages and its computation is also exponential in the number of agents. However, the diagnosis process itself entails only the exchanging of the indexes of the satisfactions in which the agents found an inconsistency. The rest of the computation is linear in the number of agents and polynomial in the size of the satisfaction database. It only divides the group according to the indexes. In systems where the constraints are static, these costs are mostly delegated to offline processes. However, where constraints change dynamically, the agents must compute all the satisfactions dynamically, and these computational costs are incurred during runtime.

Indeed, distributed CSP literature recognizes the computational costs of SBT, and offers cheaper alternatives [14]. These are examined below.

4.2 Non-Minimal Diagnosis

One alternative taken by many distributed CSP algorithms is to settle for computing only one solution to a given CSP. However, for diagnosis, this means that the results are not guaranteed to be minimal. Moreover, since only one of possibly many diagnoses would be produced, the result may not even be correct. Once a satisfaction is found, the agents compute the diagnosis by comparing their current values to the expected values in the satisfaction. The deviant agents are suspected as the abnormal agents.We examine two distributed CSP algorithms: Asynchronous backtracking and distributed stochastic search.

4.2.1 Asynchronous Backtracking (ABT).

In ABT, the priority order of agents' variables is fixed, and each agent communicates its value assignment to neighboring agents via $ok?$ messages. Each agent maintains an $agentview$, the current value assignment of other agents. An agent changes its assignment if its current value assignment is not consistent with the assignments of higher priority agents. If there exists no value that is consistent with the higher priority agents, the agent generates a new constraint (called a $nogood$), and communicates the $nogood$ to a higher priority agent, thus the higher priority agent changes its value.

ABT is complete in terms of CSP. It always finds a solution if one exists, and terminates if no solution exists, so we are guaranteed to find one diagnosis. However, still it has three drawbacks, first, we cannot be sure in advance which agents will communicate with each other, since an agent that detects a $nogood$ constraint with non-neighboring agent adds communication channel to it. Second, in contrast to SBT, here at the end of the diagnosis process, each agent may have only a portion of the diagnosis, related only to its $agentview$. Third, once a satisfaction is found, the agents do not continue to look for it, but on the other hand, they do not know that the search was completed. The next algorithm copes with some of these drawbacks.

4.2.2 Distributed Stochastic Search Algorithm (DSA).

In contrast to ABT, DSA is synchronous in that all processes proceed in synchronized steps. The agents go through a sequence of steps until a termination threshold is met (for example, limited number of cycles). In each step, an agent sends its current variable value to its neighboring agents, and concurrently receives the values from the neighbors. It then decides stochastically, whether to keep its current value or change to a new one. This is done based on a pre-defined strategy that depends on the possibility to reduce violated constraints. The most critical step of DSA is for an agent to decide the next value, based on its current state and its perceived states of the neighboring agents. The decision strategy we utilized is the following: If the agent cannot find a new value to improve its current state (reduces violations), it will not change its current value; if there exists such a value that improves its state, the agent may change to the new value with probability p, or keep the current value unchanged with probability $1 - p$. This continues until a termination threshold is reached (i.e., a certain number of cycles).

DSA is incomplete, so it may return no solution even when one exists. However, it copes with some disadvantages of ABT. First we know in advance the communication channels of every agent (neighboring agents). Second, if an agent is diagnosed as abnormal, this diagnosis is known to the abnormal agent and its neighboring agents. Third, the termination threshold is known to all the agents.

5 Experiments and Discussion

This section evaluates the distributed diagnosis algorithms we presented, in terms of computation and communication. In addition, we examine, for every algorithm, the trade-off between its computational costs and its ability to produce correct diagnosis.

We created laboratory versions of the space exploration example described previously. We evaluated every algorithm in different size groups: 4 robots, 7 robots and 10 robots. In the experiments for 4 robots, the group consisted of a mission commander and a sub-team consisting of one sub-team leader and two followers. The group of 7 robots consisted of a mission commander and two sub-teams, and the group of 10 robots consisted of a mission commander and three sub-teams.

In order to evaluate the algorithms on a representative and diverse set of problems, a wide set of combination of potential failures was selected. First, we generated all single-faults possible (1–7 in the list below). Note that we assume all followers/leaders are the same, so it does not matter which follower/leader has failed. Then we created double-fault combinations (8–12), and a quadruple failure (13):

1. a follower thinks that the leader stops, although the leader continues to go.
2. a follower thinks that the leader started to go although it actually did not.
3. a sub-team leader thinks that it got a message from the mission commander to stop, although the message was not sent.
4. a sub-team leader thinks that it got a message from the mission commander to go, but the message was not sent.
5. the mission commander sent a message to the sub-team leaders, but only some of them received it.
6. a follower stops because of an individual technical problem (nothing to do with coordination).
7. a leader stops because of an individual technical problem (nothing to do with coordination).
8. failure 1 above, in two different followers.
9. failure 2 above, in two different followers.
10. failures 3 and 4 above (one in each sub-team).
11. failure 5 above for two sub-team leaders.
12. failures 2 and 6 above.
13. failure 2 above (twice, for two different followers), and failure 5 above (twice, for two different sub-team leaders).

Failures 6 and 7 reflect a local fault but not a coordination fault, since the action values of the robots in the group remain the same. In particular, although the robot stopped, it did not select the "stop" action; it believes that its current action is "go". For these failures, we expect the diagnosis process to find that the agents' values satisfy the constraints and therefore the agents will continue to diagnose the fault locally. This process is beyond the scope of this paper.

To evaluate the performance of the algorithms from a computational perspective, two independent measures of performance were used. We measured communication load in terms the total number of messages sent [6]. We also measured runtime in terms non-

Figure 3. Sony Aibo robots capturing a mock alien.

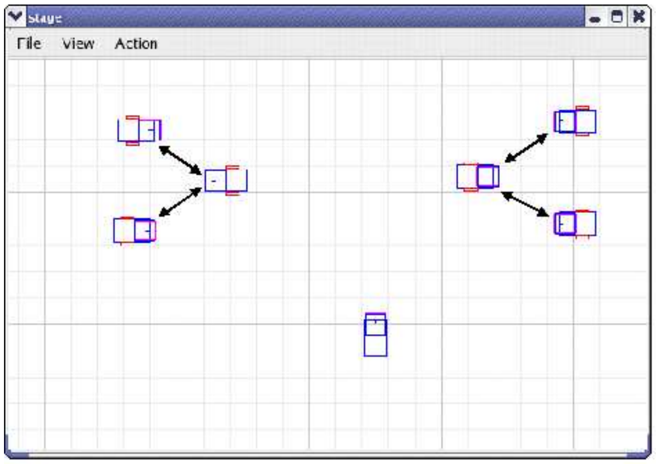

Figure 4. Screen shot of Stage simulator in action.

concurrent constraint checks (cycles) [7]. Each of the test-case failures is different, and for all algorithms other than DSA, a single run is sufficient to determine the results, since no randomization takes place, and no noise is involved in the observations or deterministic decisions of the algorithms. However, for DSA (which is a stochastic algorithm), results may change between runs, even starting with the same initial conditions. For DSA, we therefore run every experiment 30 times and takes the average. The termination threshold for DSA was set to the number of robots in the team (below we will present results using a lower—fixed—termination threshold).

Experiments with 4 robots were carried out on physical Sony Aibo robots (Figure 3). These experiments were then repeated using the Player/Stage software package [2] simulator, a popular and practical development tool for robotics (Figure 4). We verified that the results of the physical and simulated robots (group of 4) were identical, and then continued the experiments in larger groups in simulation. Also, experiments using the DSA were all carried out using the simulator (because of the need for a significant number of repeated trials).

The results of the communication load and the runtime are presented in Figure 6 and Figure 5, respectively. The x axis shows the diagnosis algorithm and the y axis presents the total number of messages sent (Figure 6) and the runtime (Figure 5). For each algorithm, three bars are shown, one for each of the group sizes. Each bar represents the average results across the different failures.

As expected, computing all the satisfactions online (SBT_ON) is expensive in terms of both communication as well as computation. Obviously, computing the satisfactions offline (SBT_OFF) and then online the diagnosis, significantly improves the efficiency. SBT_OFF is even better than the local search algorithm, DSA, although it computes a complete set of diagnoses and not a single one. The reason for this is that in SBT_OFF the agents communicate only the indexes of the inconsistent satisfactions, and do not search online for CSP solutions.

Surprisingly, ABT outperforms DSA. These results are surprising in light of previous research that showed that the stochastic search algorithm is more efficient than ABT [15]. This has to do with the likely state of a multi-agent system after a coordination failure. In a team that was in coordination and then failed, the selected actions of most agents are likely going to be close to the satisfaction. This enables ABT to find a satisfaction in only a few steps. On the other hand, in DSA the search may proceed towards a different part of the space; also, the termination threshold may cause DSA to continue running needlessly (see below for experiments with a reduced threshold).

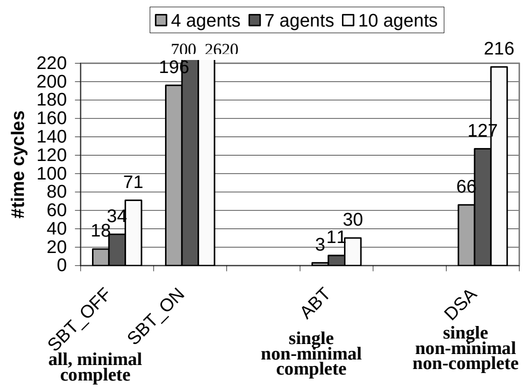

Figure 5. Average number of cycles in different diagnosis methods.

In order to further evaluate the diagnosis algorithms we examine also the correctness of the diagnoses they produce. SBT_ON and SBT_OFF produce a complete set of minimal diagnoses. However, the other algorithms produce only a single diagnosis. This diagnosis is not guaranteed to be minimal and thus to correctly explain the fault(s). In this sense, ABT is better than DSA, since it is complete and so guaranteed to find a diagnosis if one exists (although its minimality is not guaranteed).

We examine three factors in diagnosis correctness (Table 1): (i) the percentage of robots (out of the group) that failed to find a solution to the DisCSP, even if some of their peers did (here the diagnosis did not completely fail); (ii) the percentage of experiments in which the group failed to compute a diagnosis; and (iii) the percentage of

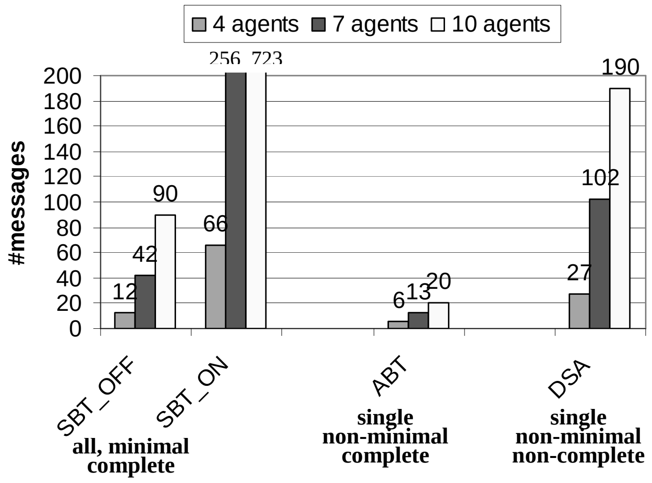

Figure 6. Average number of messages in different diagnosis methods.

experiments in which the computed diagnosis did not match the correct explanation of the failure(s). Obviously, ABT always succeeds to compute a diagnosis, because it is complete, and therefore the number of failed robots and failures in computing the diagnosis is zero. DSA is based on local search and is incomplete; some robots failed to compute a diagnosis in 8% of cases, and all failed to compute even a single diagnosis in 33%. Both of the algorithms generate diagnoses that do not match the correct explanation (ABT: 28%, DSA: 46%), since they compute only a single diagnosis and not a complete set of all the diagnoses.

Diagnosis	% failed robots	% diagnosis failures	% incorrect diagnosis
ABT	0	0	28
DSA	8	33	46

Table 1. Diagnosis failures and correctness measures, for ABT and DSA.

The results of DSA are affected by the termination threshold, which determines how long the stochastic search runs. To evaluate the effect of this factor, we reran the above experiments for DSA with a threshold of two cycles. Table 2 summarizes the results of the number of messages and runtime. Comparing these results to the results presented in Figures 6 and 5, shows a significant improvement especially in terms of runtime cycles. However, compared to running with non-fixed threshold, diagnosis quality has deteriorated further: 23% robot failure cases, 49% diagnosis failure cases, and 56% of diagnoses incorrect.

	4 agents	7 agents	10 agents
# messages	25	30	46
runtime	23	23	23

Table 2. DSA with a threshold of 2 cycles: Number of messages and runtime in cycles. Each data point is an average of 30 trials.

One lesson—expected to some degree—is that there exists trade-off between the effectiveness of the algorithms in terms of communication and computation and the correctness of the diagnosis that the algorithms produce. Algorithms that produce only a single diagnosis cannot always provide the correct diagnosis (ABT: in 28% of experiments, DSA: 46%).

However, there are two surprises. First, ABT outperforms DSA in running time and communications, in contrast to results in distributed CSP. We believe that this is a general result in the use of ABT for coordination diagnosis, because when there are only few failures at a time, ABT determines in a few steps a close solution to the CSP (and based on it, a diagnosis), compared to the stochastic behavior of DSA. Second, ABT outperforms DSA in terms of the diagnosis results: ABT provides a guarantee to find a diagnosis (DSA does not), and empirically returns the correct diagnosis much more often than DSA.

6 Summary and Future Work

To counter limitations of centralized coordination diagnosis methods, we presented an empirical investigation of distributed diagnosis algorithms, using distributed CSP algorithms as a basis. Two algorithms compute all minimal diagnoses: SBT_OFF (suitable for systems where the coordination is static), and SBT_ON (for dynamic coordination). One algorithm guarantees a single diagnosis (ABT), and one algorithm utilizes a local search approach and therefore does not guarantee any solution (DSA).

We evaluated the algorithms with real and simulated robots, and concluded that there is a trade-off between the effectiveness of the algorithms in terms of communication and computation and the correctness of the diagnosis that the algorithms produce. However, The ABT algorithm provides a surprise: It runs faster, communicates less, and provides better diagnoses than the stochastic local search algorithm—in contrast to lessons in the distributed CSP literature [15]. However, ABT has three disadvantages: (i) The diagnosis is known only to some of the agents; (ii) the agents do not know that the diagnosis process is complete; and (iii) the diagnosis is not guaranteed to be minimal. We hope to address these difficulties in the future.

REFERENCES

[1] Liliana Ardissono, Luca Console, Anna Goy, Giovanna Petrone, Claudia Picardi, Marino Segnan, and Daniele Theseider Duprpé, ‘Cooperative model-based diagnosis of web services’, in *16th International Workshop on Principles of Diagnosis (DX 05)*, pp. 125–130, (2005).

[2] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard, ‘The player/stage project: Tools for multi-robot and distributed sensor systems’, in *Proceedings of the International Conference on Advanced Robotics*, pp. 317–323, Coimbra, Portugal, (Jul 2003).

[3] Bryan Horling, Brett Benyo, and Victor Lesser, ‘Using Self-Diagnosis to Adapt Organizational Structures’, *Proceedings of the 5th International Conference on Autonomous Agents*, 529–536, (June 2001).

[4] Meir Kalech and Gal A. Kaminka, ‘Towards model-based diagnosis of coordination failures’, in *American Association for Artificial Intelligence (AAAI-05)*, (2005).

[5] Gianfranco Lamperti and Marina Zanella, *Diagnosis of Active Systems*, Kluwer Academic Publishers, 2003.

[6] Nancy A. Lynch, *Distributed Algorithms*, Morgan Kaufmann, 1996.

[7] A. Meisels, E. Kaplansky, I. Razgon, and R. Zivan, ‘Comparing performance of distributed constraints processing algorithms’, in *Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-02)*, (2002).

[8] R. Micalizio, P. Torasso, and G. Torta, ‘On-line monitoring and diagnosis of multi-agent systems: a model based approach’, *in Proceeding of European Conference on Artificial Intelligence (ECAI 2004)*, **16**, 848–852, (2004).

[9] Y. Pencolé, M.O. Cordier, and L. Rozé, ‘Incremental decentralized diagnosis approach for the supervision of a telecommunication network’, *IEEE Conference on Decision and Control (CDC’02)*, 435–440, (December 2002).

[10] Nico Roos, Annette ten Teije, and Cees Witteveen, ‘A protocol for multi-agent diagnosis with spatially distributed knowledge’, in *Proceedings of Autonomous Agents and Multi Agent Systems (AAMAS-03)*, pp. 655–661, (July 2003).

[11] Martin Sachenbacher and Brian Williams, ‘Diagnosis as semiring-based constraint optimization’, in *ECAI-2004*, (2004).

[12] F. Wotawa, *e-Environement: Progress and Challenge*, volume 11 of *Research on Computing Science*, 334–347, México, 2004.

[13] Makoto Yokoo, Edmund H. Durfee, Toru Ishida, and Kazuhiro Kuwabara, ‘The distributed constraint satisfaction problem: Formalization and algorithms’, *IEEE Trans. Knowl. Data Eng.*, **10**(5), 673–685, (1998).

[14] Makoto Yokoo and Katsutoshi Hirayama, ‘Algorithms for distributed constraint satisfaction: A review’, *Autonomous Agents and Multi-Agent Systems*, **3**(2), 185–207, (2000).

[15] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg, ‘Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks.’, *Artificial Intelligence*, **161**(1-2), 55–87, (2005).