Diagnosing Faults in a Temporal Multi-Agent Resource Allocation (Extended Abstract)

Yedidya Bar-Zev
Information Systems
Department in Ben-Gurion
University of the Negev
ybz1984@gmail.com

Roni Stern
Information Systems
Department in Ben-Gurion
University of the Negev
roni.stern@gmail.com

Meir Kalech
Information Systems
Department in Ben-Gurion
University of the Negev
kalech@bgu.ac.il

General Terms
Algorithms, Experimentation

Keywords
Planning and Reasoning (single and multi-agent), Multi-robot systems, Social simulation

1. INTRODUCTION
In this research we study multi-agent systems that use a *Temporal Multi-Agent Resource Allocation* (TMARA), which is a planned allocation of resources to agents over time. Every resource has a *capacity*, which is the maximal agents that can use that resource concurrently. In a *proper* TMARA resources are allocated to agents such that no resource is allocated to more agents than its capacity. In reality agents may malfunction due to software or hardware failures and use resources not according to the TMARA. This faulty behavior may cause the multi-agent system to fail.

As an example, consider a set of mobile robots moving around between rooms and performing tasks. Assume that the doorways connecting rooms are only wide enough to allow a single robot to pass. A TMARA can be used to schedule when each robot is allowed to pass through the doorway. A faulty robot that does not obey the TMARA may attempt to pass through a doorway simultaneously with another robot, causing a collision.

In this research we aim to diagnose such failed executions, identifying the faulty agents that caused the failure (by using resources not allocated to them in the TMARA). We name this problem *TMARA-Diag*. A diagnosis of such problem is a set of agents that the assumption they are faulty is consistent with the observation and the TMARA. Our goal is to find all minimal sets of agents that are a diagnosis. Identifying the faulty agents can shorten the recovery time of the multi-agent system by fixing or replacing only the faulty agents.

There are some work on diagnosis of multi-agent plans [1, 2, 3], but in TMARA-Diag we do not assume a plan is available, only a plan of the resource allocation (the TMARA).

2. MBD for TMARA-Diag
We model TMARA-Diag as a model-based diagnosis (MBD) problem by formalizing it in propositional logic.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

Given the MBD formalization of TMARA-Diag, one can solve TMARA-Diag with standard MBD solvers.

2.1 SAT-Based Approach
First, we attempted to apply a standard SAT-based MBD approach using the standard SAT encoding of an MBD problem, introduced by Smith et. al. [4]. The logical description of the system behavior (SD) and the observation (OBS) are compiled to a SAT formula $\alpha = SD \wedge OBS$. The variables of α are the health variables $h(\cdot)$, which indicates if an agent is faulty or not, and the resources used by the agents before the observation time t_{obs}. A satisfying assignment of α represents a diagnosis, where $h(a_i) = false$ means that a_i is assumed to be faulty.

The main limitation of the SAT-based approach to TMARA-Diag is the size of the propositional formula. Although current SAT solvers can solve instances with millions of clauses, this approach is not a scalable solution, as every possible resource allocation needs to be encoded. According to our analysis, in a case with 15 agents, a 30×30 grid cells as resources, and a time horizon (t_{obs}) of 50 time steps. The expected number of clauses is 9,112,500,000. Furthermore, the analysis is given in clauses stated in propositional logic. Standard SAT solvers usually compile the input clauses to CNF, which causes the number of CNF clauses to be even larger.

2.2 Conflict-Directed Approach
An alternative class of MBD algorithms uses *conflicts* to find diagnoses. It has been shown that diagnoses are hitting-sets of conflicts [5]. Conflict-directed diagnosis algorithms are built on this observation. Outlined by Williams and Rango [6], first, a set of conflicts is found using a conflict detection algorithm such as ATMS [7] or LTMS. Then, a minimal hitting set of these conflicts is found using a hitting set algorithm. If this hitting set is found to be a consistent diagnosis it is added to the set of diagnoses. Otherwise, a new conflict is generated and added to the set of conflicts to hit. Conflict-directed MBD algorithms vary in the implementation of this framework. If the initial set of conflicts contains all minimal conflicts, then every minimal hitting set is guaranteed to be consistent [5].

Generating minimal conflicts may be computationally expensive [7]. We present a fast polynomial algorithm to detect conflicts in a TMARA-Diag problem. This algorithm considers the *reservation time* of each agent - the earliest time in the TMARA that an agent planned to use its observed resources (the resources it was observed to be using at time t_{obs}). The reservation time of agent a_i is denoted by rt_i and the set of agents observed sharing a resource

with a_i at t_{obs} is denoted by $N(a_i)$. The relation between reservation times and conflicts is given in Lemma 1 below.

Lemma 1 For any agent $\boldsymbol{a_i}$, if $|\boldsymbol{N(a_i)| > 0}$ $\boldsymbol{and}$ $\boldsymbol{rt_i < \min_{a_j \in N(a_i)} rt_j}$, then $\{\boldsymbol{a_i}\} \cup \boldsymbol{N(a_i)}$ is a conflict.

Proof. Given a proper TMARA and an agent $\boldsymbol{a_i}$ that shared resources in the observation. If $\boldsymbol{a_i}$ was faulty, then $\{\boldsymbol{a_i}\} \cup \boldsymbol{N(a_i)}$ is a conflict due to $\boldsymbol{a_i}$. Assume that $\boldsymbol{a_i}$ is not faulty. This means that $\boldsymbol{a_i}$ assigned its observed resources exactly at time $\boldsymbol{rt_i}$. Thus, agent $\boldsymbol{a_i}$ shared a resource at time $\boldsymbol{rt_i}$ with at least one of the agents in $\boldsymbol{N(a_i)}$. Let $\boldsymbol{a_j}$ denote that agent. Since $\boldsymbol{rt_i < rt_j}$, this means that $\boldsymbol{a_j}$ had assigned the resource before its reserved time $\boldsymbol{rt_j}$, and thus $\boldsymbol{a_j}$ was faulty.■

Simply put, Lemma 1 states that an agent that has the earliest reservation time among the agents that share resources with it in the observation ($N(a_i)$), is a conflict together with those agents. The conflict detection algorithm traverses all agents, checks if the lemma conditions hold and if so a conflict is defined. This conflict detection algorithm is computationally efficient, requiring time that is square in the number of agents, which is reasonable even for very large number of agents. While we do not have a proof that this algorithm returns all minimal conflicts, we use our conflict detection algorithm in the context of CDA* [6], a diagnosis algorithm that does not require all minimal conflicts to find all minimal diagnoses. Instead, CDA* iteratively generating more conflicts and performs consistency checks to verify diagnosis soundness. Thus, the resulting algorithm is complete.

3. Evaluation

We evaluated the performance of the SAT-based and conflict-directed algorithms described above in a synthetic resource allocation setting and the Automatic Intersection Manager (AIM) domain [8]. For the synthetic resource allocation setting, we developed a *TMARA simulator*. The simulator accepts as input the time horizon (T), # agents ($|A|$), and # resources ($|R|$).

Table 1, Results for the conflict-directed algorithm

#Agents	15	20	25	30	35	40
Runtime(ms)	0	6	62	1,219	33,488	137,925
#Diagnoses	66	219	585	2,138	9,790	21,535
#Conflicts	3.81	4.21	4.52	4.63	4.87	5.11

Given these parameters, the simulator creates a TMARA by allocating all the resources randomly to the agents in every time step *t*. The simulator then injects faults according to a *fault rate* (FR) parameter, which sets the probability that an agent is faulty. During execution, the simulator assumes that agents that are not faulty follow the TMARA, using only the resources allocated them. Faulty agents, at each time step, choose randomly one of the bundled resources allocated to them at some time step. This simulates early and late allocation faults. If two or more agents attempt to use the same resource, they halt without releasing the resources that they are currently using. These agents then remain stuck in all subsequent time steps.

The output of a simulator is a randomly generated TMARA, and the observed resources allocation at the end of the time horizon. This is given to our conflict directed TMARA-Diag solver as input to start the diagnosis process.

Table 1 shows a subset of the results from our experiments using the TMARA simulator. For this representative subset of results, we set FR=0.2, time horizon=30, number of resources = $\{15,20,..,40\}$ and unbounded bundle size. Every point is an average over 180 random instances. As expected, increasing the number of agents results in higher runtime. More agents mean more potential conflicts, and since diagnoses are hitting sets of conflicts, more diagnoses and longer runtime. This relation between the number of agents, the number of conflicts, diagnosis and runtime is seen clearly in Table 1.

We also experimented the proposed *SAT*-based method using the *SAT4J* solver [9]. On average, problems with five agents required 2.5 GB of memory and for eight agents, 10.6 GB. Obviously this algorithm is not feasible even for small systems.

In addition, we evaluated the algorithms on problems created by the AIM simulator we modified, by injecting faults that cause accidents. Similarly to the synthetic simulator, results showed that the conflict-directed algorithm outperformed the SAT method.

4. CONCLUSION

We study how to diagnose agent failures given a temporal multi agent resource allocation (TMARA). This problem was formalized as an MBD problem, where the model of the system is the TMARA and the observations are the observed resource usage after the failure. This allowed solving TMARA-Diag using a SAT compilation or a conflict directed approach. While the SAT complication does not scale, we propose a novel efficient conflict detection algorithm that, coupled with a standard conflict directed MBD algorithm can be used to solve TMARA-Diag Efficiently.

5. Acknowledgments

We thank the Kamin program for partly funding this project.

6. References

[1] N. Roos and C. Witteveen, "Models and methods for plan diagnosis," *Autonomous Agents and Multi-Agent Systems,* vol. 19, pp. 30-52, 2009.

[2] F. De Jonge, N. Roos and C. Witteveen, "Primary and secondary diagnosis of multi-agent plan execution," *Autonomous Agents and Multi-Agent Systems,* vol. 18, pp. 267-294, 2009.

[3] M. Kalech, "Diagnosis of coordination failures: a matrix-based approach," *Autonomous Agents and Multi-Agent Systems,* vol. 24, pp. 69-103, 2012.

[4] A. Smith, A. Veneris, M. F. Ali and A. Viglas, "Fault diagnosis and logic debugging using Boolean satisfiability," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 24, pp. 1606-1621, 2005.

[5] J. De Kleer and B. C. Williams, "Diagnosing multiple faults," *Artificial intelligence,* vol. 32, pp. 97-130, 1987.

[6] B. C. Williams and R. J. Ragno, "Conflict-directed A* and its role in model-based embedded systems," *Discrete Applied Mathematics,* vol. 155, pp. 1562-1595, 2007.

[7] J. De Kleer, "An assumption-based TMS," *Artificial intelligence,* vol. 28, pp. 127-162, 1986.

[8] S. Peter, A. Tsz-Chiu and M. Hausknecht, "AIM: Autonomous Intersection Management," 2004. [Online]. Available: http://www.cs.utexas.edu/~aim/.

[9] D. Le Berre and A. Parrain, "The Sat4j library, release 2.2 system description," *Journal on Satisfiability, Boolean Modeling and Computation,* vol. 7, pp. 59-64, 2010.