Diagnosing Multi-Agent STRIPS Plans

Avraham Natan
Ben-Gurion University of the Negev, Beersheba, Israel

Roni Stern
Ben-Gurion University of the Negev, Beersheba, Israel

Meir Kalech
Ben-Gurion University of the Negev, Beersheba, Israel

William Yeoh
Washington University in St. Louis, MO, USA

Tran Cao Son
New Mexico State University, Las Cruces, NM, USA

Abstract

The increasing use of multi-agent systems demands that many challenges be addressed. One such challenge is diagnosing failed multi-agent plan executions, sometimes in system setups where the different agents are not willing to disclose their private actions. One formalism for generating multi-agent plans is the well-known MA-STRIPS formalism. While there have been approaches for delivering as robust plans as possible, we focus on the plan execution stage. Specifically, we address the problem of diagnosing plans that failed their execution. We propose a Model-Based Diagnosis approach to solve this problem. Given an MA-STRIPS problem, a plan that solves it, and an observation that indicates execution failure, we define the MA-STRIPS diagnosis problem. We compile that problem into a boolean satisfiability problem (SAT) and then use an off-the-shelf SAT solver to obtain candidate diagnoses. We further expand this approach to address privacy by proposing a distributed algorithm that can find these same diagnoses in a decentralized manner. Additionally, we propose an enhancement to the distributed algorithm that uses information generated during the diagnosis process to provide significant speedups. We found that the improved algorithm runs more than 10 times faster than the basic decentralized version and, in one case, runs faster than the centralized algorithm.

2012 ACM Subject Classification Hardware → Bug detection, localization and diagnosis; Computing methodologies → Multi-agent systems; Computing methodologies → Distributed algorithms; Security and privacy → Domain-specific security and privacy architectures

Keywords and phrases Model-based diagnosis, Multi-agent systems, Distributed diagnosis, Privacy

Digital Object Identifier 10.4230/OASIcs.DX.2024.8

1 Introduction

In recent years, Multi-Agent Systems (MAS) have seen a significant increase in applications in different areas such as logistics, transportation, public services, and more. In many MAS applications, the agents are tasked with meeting a set of predefined goals. To achieve this, each agent is given a series of actions to perform. This is called a Multi-Agent Plan (MAP). Figure 1 demonstrates a simple example where the goal is to move a package from a location in city 1 to a location in city 2, where trucks can travel between locations in the cities and airplanes can fly between the cities.

Multi-Agent STRIPS (MA-STRIPS) is a well-established formalism for MAP [3, 4, 2]. Most efforts in the MA-STRIPS formalism focused on various planning algorithms and heuristics. These advances have contributed to generating efficient plans.

When such a plan is executed, however, faults may occur unexpectedly. This may lead to unmet goals. Some approaches address this challenge during the planning phase by

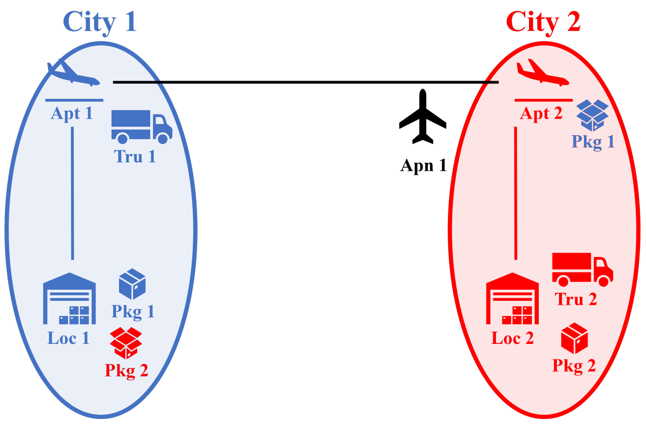

Figure 1 An illustration of a package delivery problem. Every truck can move between the locations of its respective city, and the airplane can move only between airports. The goal is to move the packages between initial locations (closed packages) and goal locations (open packages).

computing robust plans [19]. Execution faults, however, may still occur. In such cases, it is important to understand which actions or agents failed. This is the task of *Fault Diagnosis* [22]. Fault Diagnosis explains the inconsistencies between the expected outcome of a system run and the observed outcome. It uses knowledge about the problem, the plan, and the execution.

Our **first contribution** is to define the problem of diagnosing MA-STRIPS plan executions. Our **second contribution** is a centralized MBD-based algorithm called MAS-DX for solving this problem. Our **third contribution** addresses privacy concerns, allowing each agent to keep private information. The algorithm, called Dec-MAS-DX, is a decentralized MBD-based algorithm that works in three phases. First, the problem is divided into local problems, where each local problem is relevant to a specific agent. Then, problems are solved to generate local diagnoses. Finally, the local diagnoses are combined into the final set of diagnoses. We propose an enhancement to this algorithm that intelligently coordinates the diagnosis process. As the **fourth and final contribution**, we provide empirical evaluation on eight MA-STRIPS domains. We conclude that the centralized algorithm (MAS-DX) has better runtime overall, while the distributed algorithm (Dec-MAS-DX) can address privacy and even outperforms in terms of run-time for one of the domains.

2 Literature Review

2.1 Related Work

Within the MA-STRIPS formalism [3, 4, 27], advances have tackled different problems. Some examples include distributed [20, 2] and heuristic-based [26, 28] planning. A recent work addressed learning the models of the actions [17].

In this work, we assume the plan is given and focus on the diagnosis part. Diagnosis of Multi-Agent Systems has been studied in various settings w.r.t. different assumptions about the agents and how they can fail. Some studies aim to isolate which actions have failed during execution [8, 23, 7, 16, 29, 18]. Others aim to isolate which agents' beliefs made them choose actions that led to coordination failures [6, 11, 12]. We refer to [13] for a comprehensive survey of previous work in this field.

A very recent work has considered diagnosing plans within MA-STRIPS formalism [25]. However, they focused on extending the MA-STRIPS language to support common-sense fluents and how these can be used for diagnosis purposes. We, on the other hand, focus solely on the diagnosis process.

2.2 Background

MA-STRIPS is a formalism for defining multi-agent planning problems in a classical planning setting, i.e., where actions are deterministic, and states are represented by sets of discrete facts.

A planning problem in MA-STRIPS is defined by a tuple $\Pi = \langle F, k, \{A_i\}_{i=1}^{k}, I, G\rangle$ where F is a set of fluents, k is the number of agents we are planning for, A_i is the set of actions agent i can perform, I is the initial state, and G is the desired goal. A fluent $f \in F$ represents a fact that may or may not be true in a given state. The initial state I is the set of all the true fluents in the initial state, and the goal G is the set of fluents we want to achieve. Every action $a \in A_i$ is defined by a tuple $\langle \mathit{name}(a), \mathit{pre}(a), \mathit{eff}(a)\rangle$, where $\mathit{name}(a)$ is the name of the action, $\mathit{pre}(a)$ is the action's preconditions, and $\mathit{eff}(a)$ is the action's effects. The preconditions and effects of an action a are sets of *literals*, where a literal is either a fluent or its negation. An action a is *applicable* in a state s if:

- for every literal $l \in \mathit{pre}(a)$ such that $l = f$, the fluent f is in s.
- for every literal $l \in \mathit{pre}(a)$ such that $l = \neg f$, the fluent f is not in s.

Applying a to a state s results in a state denoted $a(s)$, where:

- s includes fluents f such that $l = f$ is in $\mathit{eff}(a)$.
- s does not include fluents f such that $l = \neg f$ is in $\mathit{eff}(a)$.

In a multi-agent setting, it is common for agents to be able to act concurrently. The concurrent execution of a set of actions is represented by a *joint action*, which is an k-ary vector $\mathbf{a} \in \{A_1 \cup \{\mathit{nop}\}\} \times \{A_2 \cup \{\mathit{nop}\}\} \times \cdots \times \{A_k \cup \{\mathit{nop}\}\}$, where nop represents a no-op, i.e., that the corresponding agent does not perform any action. The preconditions and effects of a joint action can generally be different from the preconditions and effects of its constituent single-agent actions [1, 24]. For simplicity, we will assume in this work that the preconditions and effects of a joint action are the corresponding union of the joint action's constituent single-agent actions. This assumption limits our approach to MA-STRIPS domains, where actions do not require more than one agent to be performed. However, the set of MA-STRIPS domains is still large enough for our approach to be relevant despite this limitation. To avoid ambiguity, we also require that executed joint actions are *well-defined* [5], which means that their effects are consistent.

▶ **Definition 1 (Multi-Agent Plan).** *For an MA-STRIPS problem $\Pi = \langle F, k, \{A_i\}_{i=1}^{k}, I, G\rangle$, A plan π is a sequence of n joint actions $\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n$ such that (1) $\boldsymbol{a}_1$ is applicable in the initial state I, (2) for each $1 < t \leq n$, $\boldsymbol{a}_t$ is applicable in the state $\boldsymbol{a}_{t-1}(...\boldsymbol{a}_1(I)...)$, and (3) $G \subseteq \boldsymbol{a}_n(\boldsymbol{a}_{n-1}(...\boldsymbol{a}_1(I)...))$.*

Table 1 shows an example of a plan that solves the MA-STRIPS problem presented in Figure 1. The trucks and the airplane collaborate in order to deliver the packages to their destinations. Every row in the table outlines a joint action that consists of all the actions in that row. They can be performed together since we assume the effects of these actions correspond to the union of the joint action's constituent single-agent actions. A counter-example for that is action four of the $apn1$ and action three of $tru2$. One of the effects of the action $\mathit{unload}(tru2, p2)$ is that $p2$ can be picked by other vehicles in the future. Only then, $apn1$ can execute its action $\mathit{load}(apn1, p2)$. Hence, those two actions are physically impossible to perform simultaneously and cannot be parts of the same joint action.

Table 1 Example of an MA-STRIPS plan.

#	Airplane 1	Truck 1	Truck 2
1	nop	drive(tru1,loc1)	load(tru2,p2)
2	nop	load(tru1,p1)	drive(tru2,apt2)
3	nop	drive(tru1,apt1)	unload(tru2,p2)
4	load(apn1,p2)	unload(tru1,p1)	nop
5	fly(apn1,apt1)	nop	nop
6	unload(apn1,p2)	nop	nop
7	load(apn1,p1)	load(tru1,p2)	nop
8	fly(apn1,apt2)	drive(tru1,loc1)	nop
9	unload(apn1,p1)	unload(tru1,p2)	nop

3 Problem Definition

Executing a plan π of n joint actions on an initial state I incurs a series of $n+1$ states, where each state essentially differs from the previous one by the effects of the corresponding joint action. We call such a series a *Trajectory*.

▶ **Definition 2 (Trajectory).** *Given an initial state I and a plan $\pi = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n)$, a* trajectory *$T(I,\pi)$ is an alternating sequence of states and joint actions $(I = s_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n, s_n)$, where s_{i-1} and s_i are the states before and after joint action $\boldsymbol{a}_i$, respectively. The triplet $(s_{i-1}, \boldsymbol{a}_i, s_i)$ is referred as the i^{th} transition of the trajectory. In the rest of the paper, unless required for context, we will denote $T(I,\pi)$ simply as T.*

Since MA-STRIPS domains are deterministic, the initial state I and plan π induce a unique trajectory where for each state s_i it holds that $s_i = \mathbf{a}_i(\ldots\mathbf{a}_1(I)\ldots)$. We call this trajectory the *expected trajectory* and denote it as $T_{exp}(I,\pi)$. This trajectory, however, may not represent the actual execution of the plan. We focus on the cases where, at some point, the trajectory differs from $T_{exp}(I,\pi)$. In this work, we assume there are two reasons for this difference. The first reason is an action that was applicable in the current state but was not applied or did not have its intended effects. We refer to such an action as a *faulty action*. The second reason is an action that is not applicable in the current state. We refer to it as a *conflicted action*. Let us formally define them.

▶ **Definition 3 (Faulty Action, Conflicted Action).** *Given a plan π for an MA-STRIPS problem $\Pi = \langle F, k, \{A_i\}_{i=1}^{k}, I, G\rangle$, a trajectory $T(I,\pi)$ and a transition $(s, \boldsymbol{a}, s') \in T(I,\pi)$, a single-agent action $a \in \boldsymbol{a}$ is said to be faulty if it is applicable in state s, but $a(s) \neq s'$, and is said to be conflicted if it is not applicable in state s. We denote the set of faulty and conflicted actions in trajectory T as $F(T)$ and $C(T)$, respectively.*

In this work, we assume that the root causes of conflicted actions are faulty actions caused by an internal error in one of the agents and not by causes related to the environment. Consequently, in every trajectory $T(I,\pi) \neq T_{exp}(I,\pi)$, we assume there must be at least one faulty action.

Multi-agent systems commonly have monitoring mechanisms that collect *observations* during plan execution in order to monitor execution and detect discrepancies between expected and executed trajectories that may lead to system failures. In our work we define observation as states that had their fluents observed.

▶ **Definition 4 (Observation).** *An observation O is an alternating sequence of the form $(o_0, \boldsymbol{a_1}, o_1, \ldots, \boldsymbol{a_n}, o_n)$ where o_i is a set of literals and $\boldsymbol{a_i}$ is a joint action. An observation is* consistent *with a trajectory $T = (s_0, \ldots, s_n)$ if they consist of the same sequence of joint actions and for every i either o_i is null (i.e., not observed) or $o_i = s_i$. Otherwise, we say that the observation is inconsistent with T.*

In this work, we assume that each state is fully observed or not observed. We leave cases where some of the states' fluents are observed as future work.

▶ **Example 5.** For example, consider the third action of $tru2$ in Table 1 ($unload(tru2, p2)$). Observing $p2$ at $apt2$ after the action execution is consistent with the plan since one of the action's effects is that $p2$ is at $apt2$. An example of an inconsistent observation is to observe that $p2$ is inside $tru2$. In that case, the explanation is that $tru2$ could not carry its action to unload $p2$.

In the context of diagnosis, there are 2 main approaches for models used to describe the behaviour of faulty actions: *weak fault model* (WFM) and *strong fault model* (SFM). Weak fault model does not specify the behaviour of a faulty action and is essentially the negation of the normal behaviour of an action. Although simpler to implement, WFM usually means longer run-time for computing a diagnosis due to larger size of different configurations of faulty actions. Strong fault model, on the other hand, specifies exactly how a faulty action should behave, and by that imposing restrictions on the states of other actions, thus contributing to a smaller space of possible faulty action configurations. However, SFM usually requires additional assumptions about the system, and in cases where there is more than one candidate fault model, this can actually contribute to much higher complexity of the diagnosis process.

In this work, we define strong model both for the behaviour of faulty actions and for conflicted actions, and denote them by M_f and M_c, respectively. In the context of MA-STRIPS, given an action a, $M_f(a)$ and $M_c(a)$ are sets of literals in the same way that *eff*(a) is. Applying a faulty action a on s is denoted as $(M_f(a))(s)$ and the resulting state is different than the state $a(s)$. The definition of fault and conflict models allows our approach to be generalized to different MA-STRIPS domains, since then all one has to do is define the specifics of M_f and M_c. The fault and conflict models we will explore in this work are *noeffect* models. It means that if an action is faulty or conflicted, its effect will not occur. In other words, for such models, $M_f = M_c = \emptyset$. Effects that do not occur are regarded as different from those that should have occurred according to the MA-STRIPS specification.

When a fault in the execution of a plan π from starting state I occurs, the observation O, collected during the execution, is inconsistent with the expected trajectory $T_{exp}(I, \pi)$. This means that for at least one transition $(s_i, \mathbf{a}, s_{i+1})$ in $T_{exp}(I, \pi)$, it holds that $s_{i+1} \neq o_{i+1}$. This inconsistency indicates that at least one action was faulty when executing π. In that case, the observation O represents a trajectory T' that is different from $T_{exp}(I, \pi)$. In fact, whenever a set of actions becomes faulty, this corresponds to a trajectory that is different from the expected one. Let us define the set of possible trajectories $\mathcal{T}(I, \pi)$:

▶ **Definition 6 (The set of possible trajectories).** *For a starting state I and a plan π, the set of possible trajectories $\mathcal{T}(I, \pi)$ is the set of every possible trajectory that can result by an execution of the plan π where some (or none) of the actions are faulty.*

Note two things. First, $T_{exp}(I, \pi) \in \mathcal{T}(I, \pi)$ by definition. Second, since this definition is not in the context of fault and conflict models, it includes trajectories that, given a specific fault and conflict models, might be impossible. Therefore, we further define the set of trajectories for fault and conflict models M_f and M_c:

▶ **Definition 7 (The set of trajectories for M_f and M_c).** *For a starting state I, a plan π, and fault and conflict models M_f and M_c, the set of trajectories $\mathcal{T}(I, \pi, M_f, M_c)$ is the set of every trajectory $\mathcal{T}(I, \pi)$ where for every transition $(s_i, \boldsymbol{a}, s_{i+1})$, either $s_{i+1} = o_{i+1}$, $(M_f(a))(s_i) = o_{i+1}$ or $(M_c(a))(s_i) = o_{i+1}$.*

We can now define the MA-STRIPS diagnosis problem. An MA-STRIPS diagnosis problem arises when the observation O is not consistent with the expected trajectory $T_{exp}(I, \pi)$. A diagnosis is then defined to be a set of actions that, if assumed to be faulty or conflicted, make the O consistent with $T_{exp}(I, \pi)$. Formally:

▶ **Definition 8 (MA-STRIPS Diagnosis problem).** *An MA-STRIPS diagnosis problem is defined by a tuple $\langle \Pi, \pi, O, M_f, M_c \rangle$, where $\Pi = \langle F, k, \{A_i\}_{i=1}^{k}, I, G \rangle$ is an MA-STRIPS problem, π is a plan that solves it, O is the observation received by executing π from I, M_f is the fault model and M_c is the conflict model. A diagnosis problem arises when O is not consistent with $T_{exp}(I, \pi)$, or in other words, there exists a transition $(s_i, \boldsymbol{a}, s_{i+1})$ such that $s_{i+1} \neq o_{i+1}$. A diagnosis for an MA-STRIPS diagnosis problem is a set of single agent actions $\omega = \{a_{i_1}, a_{i_2}, ...\}$ from the plan π such that there exists a trajectory $T \in \mathcal{T}(I, \pi, M_c, M_f)$ that is consistent with O where $F(T) = \omega$.*

We continue to describe our approach for solving this problem, which involves modelling the problem as a SAT problem.

4 Centralized diagnosis approach

In this section, we present MAS-DX, a centralized approach for solving a MA-STRIPS diagnosis problems. MAS-DX formulates the MA-STRIPS Diagnosis problem as a classical Model-Based Diagnosis (MBD) problem and then solves it using an off-the-shelf MBD algorithm [22]. First, we provide a brief background on MBD.

4.1 MBD Background

An MBD problem is a tuple $\langle SD, COMPS, OBS \rangle$ where SD, $COMPS$, and OBS denote the system description, the set of components, and the observations, respectively. In a Weak Fault Model (WFM) setting, SD takes into account that some components might be faulty without specifying a fault mode. In that case, the unary predicate $h(\cdot)$ on the components specifies that $h(c)$ is true if component c is healthy, and $\neg h(c)$ is true if c is not healthy. When the assumption that all components are healthy is inconsistent with the system model and observed system behavior, a diagnosis problem arises. This is formalized as follows:

$$SD \; \wedge \bigwedge_{c \in COMPS} h(c) \; \wedge \; OBS \; \vdash \; \bot$$

where $\vdash$ means to entail and $\bot$ means dissatisfaction. By that, $\vdash \; \bot$ means that the formula cannot be satisfied, or in other words, inconsistent.

Diagnosis approaches try to find diagnoses, which are possible ways to explain the above inconsistency by assuming that some components are not healthy, and that adhere to the principle of parsimony. Such approaches define diagnosis as a set of components Δ if:

$$SD \; \wedge \bigwedge_{c \in \Delta} \neg h(c) \; \wedge \bigwedge_{c \notin \Delta} h(c) \; \wedge \; OBS \; \nvdash \; \bot$$

and where for each $\Delta' \subseteq \Delta$,

$$SD \ \wedge \ \bigwedge_{c \in \Delta'} \neg h(c) \ \wedge \ \bigwedge_{c \notin \Delta'} h(c) \ \wedge \ OBS \ \vdash \ \bot$$

This is also known as minimal subset diagnosis.

Strong fault model, on the other hand, assumes a model f that defines the effects of a faulty action. In that case, a diagnosis δ will be a set of components such that:

$$SD \ \wedge \ \bigwedge_{c \in \Delta} f(c) \ \wedge \ \bigwedge_{c \notin \Delta} h(c) \ \wedge \ OBS \ \nvdash \ \bot$$

In this work we assume strong fault models, as discussed in section 3. What this essentially means is that SD extends the healthy behaviour specification with specifications of faulty behaviour [9].

4.2 MA-STRIPS Diagnosis as MBD

MAS-DX formulates a given MA-STRIPS Diagnosis problem as an MBD problem by specifying SD, COMPS, and OBS as follows. *OBS* are the sets of literals $O = (o_0, \boldsymbol{a_1}, o_1, \ldots, \boldsymbol{a_n}, o_n)$ observed in the plan execution observation as defined in Definition 4. The system components are mapped to the single-agent actions in the plan: $COMPS = \{\{a_{i,t}\}_{i=0}^{k}\}_{t=0}^{n}$, where k denotes the number of agents and n the length of the plan. Three health-state predicates $h(\cdot)$, $f(\cdot)$ and $c(\cdot)$ over the action space specify whether an action is healthy, faulty, or conflicted. The system description *SD* specifies how the fluents of a state in time t change given a single-agent action a_t. Fluents that are not in the effects of any single-agent action at time t retain their current values. Fluents that are in the effects of an action change their values according to the health state of the action. If the action is normal, the fluent changes its value according to the effects $\mathit{eff}(a_t)$. If the action is faulty or conflicted, the fluent changes its value according to the fault or conflict model, respectively.

To solve the resulting MBD problem, MAS-DX compiles it into a Boolean satisfiability problem (SAT). A SAT problem is defined by a conjunction of boolean literals, and a solution is an assignment of true or false values to those literals, such that the entire formula is true [10]. In the next section, we show how MAS-DX formulates the different parts of our MBD problem as SAT Boolean formulas.

4.3 MA-STRIPS as SAT

4.3.1 Fluents as SAT variables

Recall that in our MA-STRIPS domain, every state is defined as a set of fluents. Thus, for every time-step t, we define for each fluent $f \in F$ a Boolean variable f_t. This allows us to model every state s as a conjunction of Boolean variables $s_t = \bigwedge_{1 \leq j \leq |F|} f_t^j$. Setting f_t^j to true means that the fluent f^j was part of s_t.

4.3.2 COMPS as SAT variables

Since we want to find the faulty actions, we model COMPS as the set of actions $COMPS = \{\{a_{i,t}\}_{i=0}^{k}\}_{t=0}^{n}$, where k denotes the number of agents and n the length of the plan.

4.3.3 OBS as SAT assignments

We encode every observed fluent by setting its corresponding Boolean variable value to true or false. Setting f_t^j to true represents that fluent f^j was observed as part of state s_t.

4.3.4 SD as a conjunction of SAT formulas

Defining SD essentially means defining how the values of the temporal variables f_t change in every time step t. This is determined by the health state of an action $a_{i,t}$. To that end, we model SD as a conjunction of Boolean formulas.

For each action $a_{i,t}$, we define how each health state entails the values of fluents in the preconditions of the action. Formally:

1. $h(a_{i,t}) \to \bigwedge_{f^j \in pre(a_{i,t})} f_t^j$
2. $f(a_{i,t}) \to \bigwedge_{f^j \in pre(a_{i,t})} f_t^j$
3. $c(a_{i,t}) \to \neg(\bigwedge_{f^j \in pre(a_{i,t})} f_t^j)$

Note that normal and faulty actions entail the same fluent values. This is because what differentiates the two health states is whether the effects occur according to the plan or the fault model.

For each action, the health predicates are mutually exclusive, meaning that exactly one of them must be true:

4. $h(a_{i,t}) \otimes f(a_{i,t}) \otimes c(a_{i,t})$

The second group of formulas we define are the transition formulas. For each variable f_t^j, we specify how its value changes in the next state s_{t+1}. We distinguish between four types of variables in time step t:

- variables not in any action effects at time t.
- variables in action effects of a normal action at time t.
- variables in action effects of a faulty action at time t.
- variables in action effects of a conflict action at time t.

For each variable type, we define the transition formulas. We begin by defining the transition of variables that are not in any action effects. We do not assume any exogenous events in our work and therefore, in that case the values stay the same as the values of the previous state. Formally:

5. $\bigwedge_{f^j \notin \bigcup_{i=1}^{k} eff(a_{i,t})} f_{t+1}^j \leftrightarrow f_t^j$

Values of variables that are in the effects of normal actions are set in the next state to the values according to the actions' effects. Formally:

6. $h(a_{i,t}) \to \bigwedge_{f^j \in eff(a_{i,t})} f_{t+1}^j$

Values of variables included in faulty and conflicted actions are set in the next state to the values according to the respective fault and conflict models. Formally:

7. $f(a_{i,t}) \to \Phi_f(i,t)$

8. $c(a_{i,t}) \to \Phi_c(i,t)$

where $\Phi_f(i,t)$ and $\Phi_c(i,t)$ represent the formulas encoding the fault and conflict models M_f and M_c.

The fault and conflict models are general and can be defined independently from the model of the system description. In this work, we explore the *noeffect* fault and conflict models, defined as such:

9. $\Phi_f(i,t) \rightarrow \bigwedge_{f^j \in \mathit{eff}(a_{i,t})} f_{t+1} \leftrightarrow f_t$

10. $\Phi_c(i,t) \rightarrow \bigwedge_{f^j \in \mathit{eff}(a_{i,t})} f_{t+1} \leftrightarrow f_t$

Using those formulas, we define the complete SD model:

$$\begin{aligned} SD = \bigwedge_{t=1}^{n} \Bigg(\Big(& \bigwedge_{f^j \notin \bigcup_{i=1}^{k} \mathit{eff}(a_{i,t})} f^j_{t+1} \leftrightarrow f^j_t \Big) \\ & \wedge \Big(h(a_{i,t}) \rightarrow \bigwedge_{f^j \in \mathit{eff}(a_{i,t})} f^j_{t+1} \Big) \\ & \wedge \Big(f(a_{i,t}) \rightarrow \Phi_f(i,t) \Big) \\ & \wedge \Big(c(a_{i,t}) \rightarrow \Phi_c(i,t) \Big) \\ & \wedge \Big(h(a_{i,t}) \otimes f(a_{i,t}) \otimes c(a_{i,t}) \Big) \Bigg) \end{aligned}$$

MAS-DX solves the resulting SAT problem using an off-the-shelf SAT solver, and yields a set of boolean assignments that satisfy the SD model. Each one of these assignments can be translated into a diagnosis. Given an assignment ψ and a variable $a_{i,t}$, action a_i that was executed in time t is healthy if $h(a_{i,t}) = \mathit{true}$, faulty if $f(a_{i,t}) = \mathit{true}$, and conflicted if $c(a_{i,t}) = \mathit{true}$. Actions with a faulty health state are the diagnosis of the problem.

▶ **Example 9.** As an example, consider Table 1. Let us assume that action 2 of agent 2 ($a_{2,2} = \mathit{drive}(\mathit{tru}2, \mathit{apt}2)$) failed, causing action $a_{2,3}$ to become conflicted. Because of that, agent 0 (the airplane) will not execute actions $a_{0,4}, a_{0,6}$, since the package $p2$ will not be at the airport. Consequently, agent 1 (truck 1) will fail its actions $a_{1,7}, a_{1,9}$, and the package $p2$ will be observed inside agent 2 at the time of the task execution. In this scenario, MAS-DX will assign a faulty health state to action $a_{2,2}$, since the preconditions will be valid, but the effects will not be as expected. This will be done using formulas 2, 47 in the problem model. MAS-DX will continue assigning health state values in the same manner, returning a solution where action $a_{2,2}$ is faulty and actions $a_{2,3}, a_{0,4}, a_{0,6}, a_{1,7}, a_{1,9}$ are conflicted.

4.3.5 Proofs of soundness and completeness

▶ **Theorem 10** (Soundness). *Every variable assignment returned by MAS-DX corresponds to a valid diagnosis.*

Proof. Let ψ be an assignment of health state values returned by MAS-DX. To prove the theorem, we show that the values of the health states of the actions are consistent with the observations. In ψ, the values of the observations are already set at the time of the problem modelling. Meaning that $\{f^1_0, \ldots, f^{|F|}_0, \ldots, f^1_t, \ldots, f^{|F|}_t, \ldots, f^1_{t'}, \ldots, f^{|F|}_{t'}, \ldots, f^1_n, \ldots, f^{|F|}_n\}$ which are the values of the fluents in the observed states are set (which means some states that are not observed do not have set values).

Following Formula 4, for every time step and for every agent, exactly one of the health state variables $h(a_{i,t}), f(a_{i,t}), c(a_{i,t})$ is true, where $a_{i,t}$ is the action that agent i was planned to execute at time step t. In addition, Formulas $1-3, 6-8$ mean that the fluent variables of the preconditions and the effects are consistent with the chosen health states. These fluents are fluents with indices t and $t+1$ that correspond to states S_t and S_{t+1}. In turn, those equations mean that each value of these fluents is consistent with the actions in time steps $t-1$ and $t+1$. In addition, fluents that are not part of effects or preconditions in time step t, get in time step $t+1$ the same values. This is ensured by Formula 5.

Following the same logic, the values reach consistency with the fluent values that were set as part of the observation, during the modelling of the problem. This means that the health state values are consistent with the observations, which is the definition of a diagnosis. ◀

▶ **Theorem 11** (Completeness). *MAS-DX returns a variable assignment for every valid diagnosis.*

Proof. Suppose a valid diagnosis d for which the assigned action health state values are consistent with the observations, and for which, the observations consist of the following observed states: $\{S_0, \ldots, S_t, \ldots, S_{t'}, \ldots, S_n\}$. MAS-DX encodes them as fluent variables with set values $\{f_0^1, \ldots, f_0^{|F|}, \ldots, f_t^1, \ldots, f_t^{|F|}, \ldots, f_{t'}^1, \ldots, f_{t'}^{|F|}, \ldots, f_n^1, \ldots, f_n^{|F|}\}$.

Suppose that according the d, the first faulty action is action $a_{i,t}$. Throughout the assignment process, MAS-DX assigns variables to actions and state fluents such that all actions before $a_{i,t}$ are healthy and all fluents up to $f_t^1, \ldots, f_t^{|F|}$ change according to formulas $1, 5, 6$. At time t, MAS-DX considers the fluents $f_t^1, \ldots, f_t^{|F|}$ that represent state S_t. Their values, are consistent with the formulas defining the preconditions of action $a_{i,t}$. Thus, this action can not be conflicted, and the other two options are faulty or healthy. Each option influences the values of the fluents $f_{t+1}^1, \ldots, f_{t+1}^{|F|}$ that are part of the effects of $a_{i,t}$. From this point on, MAS-DX continues to set different values to state variables, representing the states $S_{t+1}, \ldots S_n$. Following the same process, MAS-DX will check its value assignment with all of the fluents that represent the observed states, and since d is a valid diagnosis, one of these checks will satisfy all of the value assignments. ◀

5 Distributed Diagnosis Approach

MAS-DX is designed to be executed by a central diagnoser that has full access to the information defining the diagnosis problem. The leading assumption here is that the system has all the relevant information in one place, or, that once a diagnosis process begins, the agents voluntarily submit any private information that is required. Thanks to that assumption, MAS-DX is able to model the entire problem as MBD and solve it.

Relaxing such an assumption allows addressing problems where the system description (e.g., the actions of the agents and the fluents) is not available to a central diagnoser. However, the approach we propose in section 4 is not suitable for such cases. Consider, for example, Figure 1. There could be a case where the trucks do not share information about their private actions with an outside diagnoser. We could try to use a diagnoser per truck, but then the private information of other trucks will not be available, which will lead to incorrect modelling of each one of the per-truck diagnosers.

To address this, we propose Dec-MAS-DX, a distributed diagnosis approach for solving MA-STRIPS diagnosis problems. In Dec-MAS-DX, each agent is associated a diagnoser, that models the information available to the agent (private and public). Dec-MAS-DX consists of two main parts. The first part, where each agent attempts to infer diagnoses based on the public information combined with its private information. That information includes the fluents and actions relevant to that agent. At this point, the computed diagnoses are not yet correct - there are duplicates and some diagnoses contradict each other. In the second part, the agents combine these diagnoses in order to converge to diagnoses that are not contradicting each other, and also do not have duplicates.

5.1 Local MA-STRIPS Diagnosis

Prior work on MA-STRIPS used the notion of private and public fluents and actions for each agent [14, 15]. A fluent is private for agent i if it does not appear in the preconditions or effects of any other agent's action. An action is private for agent i if its preconditions and effects only include fluents that are private for that agent. We draw from this definition to define in a similar manner the notions of *relevant fluents* and *relevant actions.*

Unlike private fluents and actions, relevant fluents and actions are defined with respect to the specific plan the agents attempted to execute. This distinction is important for two reasons. First, it means that the plan π, generated as the solution to an MA-STRIPS problem Π, has a major role in defining how the SAT modelling will look like. Second, it means that different plans will define different sets of actions and fluent as relevant to different agents.

▶ **Definition 12** (**Relevant Fluent**). *A fluent $f \in F$ is defined as relevant to agent i if $\exists a \in A_i : f \in pre(a) \cup eff(a) \vee \neg f \in pre(a) \cup eff(a)$. We denote the set of relevant fluents to agent i as $\mathcal{R}_i(F)$.*

▶ **Definition 13** (**Relevant Action**). *An action a is defined as relevant to agent i if $\exists f \in \mathcal{R}_i(F) : f \in pre(a) \cup eff(a) \vee \neg f \in pre(a) \cup eff(a)$. We denote the set of actions relevant to agent i as $\mathcal{R}_i(A)$, where $A = \bigcup_{i=1}^{k} A_i$ is the union of the agents' actions.*

Considering these definitions, we can look at the parts of Π and π relevant to agent i. We call this the *local view* of agent i. The part of the MA-STRIPS planning problem relevant to agent i is $\Pi_i = \langle \mathcal{R}_i(F), k, \mathcal{R}_i(A), I, G\rangle$, and the plan π_i is defined as a sequence of joint actions $\mathbf{a}_1^i, \mathbf{a}_2^i, ..., \mathbf{a}_n^i$ where each joint action $\mathbf{a}_t^i$ is a subset of the joint action $\mathbf{a}_t$ that includes only actions $a \in \mathbf{a}_t$ that are relevant to agent i. Actions in $\mathbf{a}_t$ that are not in $\mathcal{R}_i(A)$ are replaced by $\{nop\}$ in $\mathbf{a}_t$.

We can now define the local MA-STRIPS Diagnosis problem for agent i using only its relevant fluents and actions.

▶ **Definition 14** (**Local MA-STRIPS Diagnosis Problem**). *A Local MA-STRIPS Diagnosis problem for agent i is a tuple $(\Pi_i, \pi_i, O, M_f, M_c)$ where $\Pi_i = \langle \mathcal{R}_i(F), k, \mathcal{R}_i(A), I, G\rangle$ defines the parts of Π relevant to agent i, π_i is the series of joint actions of plan π that contain only actions relevant to agent i, O_i is the observation received by executing π from I that includes only fluents relevant to agent i, M_f is the fault model and M_c is the conflict model. A local MA-STRIPS diagnosis problem arises when O_i is not consistent with $T_{exp}(I, \pi)$. A diagnosis for that problem is a set of single agent actions $\omega_i = \{a_{j_1}, a_{j_2}, ...\}$ from π_i such that there exists a trajectory $T \in \mathcal{T}(I, \pi, M_c, M_f)$ that is consistent with O_i where $F_i(T) = \omega_i$.*

5.2 Local MA-STRIPS Diagnosis as MBD

Dec-MAS-DX formulates a given local MA-STRIPS diagnosis problem as MBD by specifying SD, COMPS and OBS in a similar way done by MAS-DX, with some important differences.

The first difference is that the components and fluents of the local MA-STRIPS Diagnosis problem for agent i are defined with relation to the local view of agent i. The problem for agent i is modelled as the tuple $(SD_i, OBS_i, COMPS_i)$ where OBS_i are the sets of literals observed in the plan execution that are positive or negative fluents in $\mathcal{R}_i(F)$. The system components are mapped to the actions relevant to agent i: $COMPS_i = \{\{a_{i,t}\}_{i=0}^{k}\}_{t=0}^{n}$ *s.t.* $a_{i,t} \in \mathcal{R}_i(A)$.

The second difference is with the health states defining each component. Here, we need to separate $\mathcal{R}_i(A)$ into two sets - actions that are executed by agent i and actions that are executed by other agents but are relevant to agent i. Formally:

▶ **Definition 15 (Internal Action, External action).** *Given the local diagnosis problem* $(\Pi_i, \pi_i, O, M_f, M_c)$ *for agent* i*, an Internal Action is an action* $a \in \mathcal{R}_i(A)$ *such that* $a \in A_i$*, and external action is an action* $a \in \mathcal{R}_i(A)$ *such that* $a \notin A_i$*. We denote the set of internal and external actions for agent* i *as* $\mathcal{I}_i(A)$ *and* $\mathcal{E}_i(A)$*, respectively. It holds that* $\mathcal{I}_i(A) \cup \mathcal{E}_i(A) = \mathcal{R}_i(A)$ *and* $\mathcal{I}_i(A) \cap \mathcal{E}_i(A) = \emptyset$*.*

The health states for actions in $\mathcal{I}_i(A)$ are defined by the same three health-state predicates as before: $h(\cdot)$, $f(\cdot)$ and $c(\cdot)$. The same predicates cannot be used for actions in $\mathcal{E}_i(A)$. The reason is the fact that such actions are external to agent i, and by definition, only part of the fluents in their preconditions are actually visible to i in its local view (see Definitions 12, 13). To address this, we introduce additional three health state predicates, corresponding to the original three: *external healthy*, *external faulty* and *external conflicted*, and we denote them as $eh(\cdot)$, $ef(\cdot)$ and $ec(\cdot)$, respectively.

Given these changes, SD_i specifies how the fluents of a state change according to the health states of the actions relevant to agent i. In the next section we specify how we encode those changes as SAT boolean formulas.

5.3 Local MA-STRIPS as SAT

5.3.1 Fluents as SAT variables

For every time-step t, we define for each fluent $f \in \mathcal{R}_i(F)$ a boolean variable f_t. This allows us to model every state s as a conjunction of Boolean variables $s_t = \bigwedge_{1 \le j \le |\mathcal{R}_i(F)|} f_t^j$. Setting f_t^j to true means that the fluent f^j was part of s_t.

5.3.2 *COMPS*$_i$ as SAT variables

We model $COMPS_i$ as the set of actions $COMPS_i = \{\{a_{i,t}\}_{i=0}^{k}\}_{t=0}^{n}$ $s.t.$ $a_{i,t} \in \mathcal{R}_i(A)$, where k denotes the number of agents and n the length of the plan.

5.3.3 *OBS*$_i$ as SAT assignments

We encode every observed fluent by setting its corresponding boolean variable value to true or false. Setting f_t^j to true represents that fluent f^j was observed as part of state s_t.

5.3.4 *SD*$_i$ as a conjunction of SAT formulas

We model SD as a conjunction of Boolean formulas in a similar way we modelled MAS-DX, except for some changes.

For each action $a_{i,t} \in \mathcal{I}_i(A)$, we define how healthy, faulty, and conflicted health states entail the values of fluents in the preconditions of the action. Formally:

11. $h(a_{i,t}) \rightarrow \bigwedge_{f^j \in pre(a_{i,t})} f_t^j$
12. $f(a_{i,t}) \rightarrow \bigwedge_{f^j \in pre(a_{i,t})} f_t^j$
13. $c(a_{i,t}) \rightarrow \neg(\bigwedge_{f^j \in pre(a_{i,t})} f_t^j)$

We do not specify such formulas for the actions $a_{i,t} \in \mathcal{E}_i(A)$, since by definition, the agent does not know the full set $pre(a_{i,t})$ of its external actions. In other words, the variables $eh(a_{i,t})$, $ef(a_{i,t})$ and $ec(a_{i,t})$ do not entail anything about the preconditions of the action.

For each temporal action $a_{i,t} \in \mathcal{I}_i(A)$, the health predicates are mutually exclusive, meaning that exactly one of them must be true. The same is done for actions $a_{i,t} \in \mathcal{E}_i(A)$:

12. $h(a_{i,t}) \otimes f(a_{i,t}) \otimes c(a_{i,t})$
13. $eh(a_{i,t}) \otimes ef(a_{i,t}) \otimes ec(a_{i,t})$

We define the transition formulas similarly to MAS-DX, but with relation to the local view of agent i. In addition, we add transition formulas for the innocent and guilty health states.We begin with transition of variables $f^j \in \mathcal{R}_i(F)$ that are not in any action effects:

14. $\bigwedge_{f^j \in \mathcal{R}_i(F) \text{ } s.t. \text{ } f^j \notin \bigcup_{i=1}^{k} \mathit{eff}(a_{i,t})} f^j_{t+1} \leftrightarrow f^j_t$

Values of variables $f^j \in \mathcal{R}_i(F)$ that are in the effects of normal actions are set in the next state to the values according to the actions' effects. Formally:

15. $h(a_{i,t}) \rightarrow \bigwedge_{f^j \in \mathcal{R}_i(F) \text{ } s.t. \text{ } f^j \in \mathit{eff}(a_{i,t})} f^j_{t+1}$

Values of variables included in faulty and conflicted actions are set in the next state to the values according to the respective fault and conflict models. Formally:

16. $f(a_{i,t}) \rightarrow \Phi_f(i,t)$

17. $c(a_{i,t}) \rightarrow \Phi_c(i,t)$

where $\Phi_f(i,t)$ and $\Phi_c(i,t)$ represent the formulas encoding the fault and conflict models M_f and M_c.

We encode similar formulas for actions $a_{i,t} \in \mathcal{E}_i(A)$:

18. $eh(a_{i,t}) \rightarrow \bigwedge_{f^j \in \mathcal{R}_i(F) \text{ } s.t. \text{ } f^j \in \mathit{eff}(a_{i,t})} f^j_{t+1}$

19. $ef(a_{i,t}) \rightarrow \Phi_f(i,t)$

20. $ec(a_{i,t}) \rightarrow \Phi_c(i,t)$

The fault and conflict models are the same as described previously, but here they effect only the relevant fluents.

9. $\Phi_f(i,t) \rightarrow \bigwedge_{f^j \in \mathcal{R}_i(F) \text{ } s.t. \text{ } f^j \in \mathit{eff}(a_{i,t})} f_{t+1} \leftrightarrow f_t$

10. $\Phi_c(i,t) \rightarrow \bigwedge_{f^j \in \mathcal{R}_i(F) \text{ } s.t. \text{ } f^j \in \mathit{eff}(a_{i,t})} f_{t+1} \leftrightarrow f_t$

Using those formulas, we define the complete SD model:

$$
\begin{aligned}
SD_i = \bigwedge_{t=1}^{n} \Bigg(\Big(& \bigwedge_{f^j \in \mathcal{R}_i(F) \text{ } s.t. \text{ } f^j \notin \bigcup_{i=1}^{k} \mathit{eff}(a_{i,t})} f^j_{t+1} \leftrightarrow f^j_t \Big) \\
& \wedge \Big(h(a_{i,t}) \rightarrow \bigwedge_{f^j \in \mathcal{R}_i(F) \text{ } s.t. \text{ } f^j \in \mathit{eff}(a_{i,t})} f^j_{t+1} \Big) \\
& \wedge \Big(f(a_{i,t}) \rightarrow \Phi_f(i,t) \Big) \\
& \wedge \Big(c(a_{i,t}) \rightarrow \Phi_c(i,t) \Big) \\
& \wedge \Big(h(a_{i,t}) \otimes f(a_{i,t}) \otimes c(a_{i,t}) \Big) \\
& \wedge \Big(eh(a_{i,t}) \rightarrow \bigwedge_{f^j \in \mathcal{R}_i(F) \text{ } s.t. \text{ } f^j \in \mathit{eff}(a_{i,t})} f^j_{t+1} \Big) \\
& \wedge \Big(ef(a_{i,t}) \rightarrow \Phi_f(i,t) \Big) \\
& \wedge \Big(ec(a_{i,t}) \rightarrow \Phi_c(i,t) \Big) \\
& \wedge \Big(eh(a_{i,t}) \otimes ef(a_{i,t}) \otimes ec(a_{i,t}) \Big) \Bigg)
\end{aligned}
$$

Solving this model for agent i will result in sets of health state assignments to the actions relevant to agent i. At this point, some assignments are incorrect due to the partial knowledge that agent i has. The next stage in this approach is to combine the diagnoses of the agents to achieve global diagnoses, which means a set of correct health state assignments that can explain the local observations of all of the agents.

Algorithm 1 computeGlobal.

```
   Input: $T$ - plan size
   Input: $k$ - agents number
   Input: $\{\mathcal{LD}_i\}_{i=1}^{k}$ - sets of local diagnoses
   Result: $\mathcal{GD}$ - a set of global diagnoses
1  sort $\{\mathcal{LD}_i\}_{i=1}^{k}$ by local diagnoses number
2  $\mathcal{GD} \leftarrow \mathcal{LD}_1$
3  for $i \in [2, ..., k]$ do
4      $\mathcal{NGD} \leftarrow \emptyset$
5      for $gd \in \mathcal{GD}$ do
6          for $ld \in \mathcal{LD}_i$ do
7              $ngd \leftarrow combine(k, T, gd, ld)$
8              if $ngd \neq None \wedge ngd \notin \mathcal{NGD}$ then
9                  $\mathcal{NGD} \leftarrow \mathcal{NGD} \cup ngd$
10     $\mathcal{GD} \leftarrow \mathcal{NGD}$
11 return $\mathcal{GD}$
```

5.4 Combining the Diagnoses

At this point, each agent i has a set of local diagnoses $\mathcal{LD}_i = \{ld_{i,1}, ld_{i,2}, ...\}$. Each such local diagnosis contains one assigned health state to each action in $\mathcal{R}_i(A)$. At this stage, some agents might have local diagnoses that disagree about the health state of some actions. The next stage is to combine those local diagnosis sets into one set of global diagnoses, where the agents agree on the health states of the actions.

Algorithm 1 presents this process. The algorithm sorts the agents by the number of local diagnoses from lowest to highest (line 1) and initializes the global diagnosis set as the set of local diagnoses of the first agent (line 2). Then, the algorithm iteratively goes over the agents (Line 3). For every agent, it tries to combine every global diagnosis with every local diagnosis of that agent (Lines 5-7). If the global and local diagnoses agree on their health states, the combination is successful, and the algorithm records the updated diagnosis (Lines 8-9). Finally, the algorithm saves the list of updated global diagnoses (line 10). At the end of the iteration of the last agent, the set of global diagnoses is returned.

▶ **Example 16.** As an example, consider Table 1 again. Let us assume that action 2 of agent 2 ($a_{2,2} = drive(tru2, apt2)$) failed, causing action $a_{2,3}$ to become conflicted. Because of that, agent 0 could not execute action $a_{0,4} = load(apn1, p2)$. Agent 0 can not determine whether $a_{2,3}$ is faulty or conflicted because it does not know about action $a_{2,2}$, which is not relevant to agent 0. Hence, agent 0 outputs two diagnoses. In one, $f(a_{2,3})$ is true, and in another $c(a_{2,3})$ is true. Both diagnoses are inserted into the global diagnoses set to be updated. When these diagnoses are examined by agent 2, it removes the diagnosis where $f(a_{2,3})$ is true. The remaining diagnosis ($c(a_{2,3})$) is corresponding to the local diagnosis of agent 2, where $f(a_{2,2})$ and $c(a_{2,3})$ are true.

Algorithm 2 outlines the process of the combination of two diagnoses. For clarity, we denote the different health predicates as $\mathcal{H}(a_{j,t})$. For example $\mathcal{H}_{ld}(a_{j,t})$ denotes the calculated health state of action a of agent j at time t, according to diagnosis ld. In addition, we extend the equality between health states to include equality between internal and external versions of a health state, for the pairs $h(\cdot)$ and $eh(\cdot)$, $f(\cdot)$ and $ef(\cdot)$, $c(\cdot)$ and $ec(\cdot)$. This way, if an agent diagnosed his internal action $a_{i,t}$ as healthy ($h(\cdot)$), and another agent diagnosed the same (now external from his local view) action as externally healthy ($eh(\cdot)$), then the algorithm will get a false value in line 4.

Algorithm 2 combine.

```
Input: $k$ - agent number
Input: $T$ - plan length
Input: $gd$ - global diagnosis
Input: $ld$ - local diagnosis
Result: updated global diagnosis
1 $ngd \leftarrow gd$
2 for $i \in [1, ..., k]$ do
3     for $t \in [1, ..., T]$ do
4         if $\mathcal{H}_{gd}(a_{i,t}) \neq \mathcal{H}_{ld}(a_{i,t})$ then
5             return None
6         else if $\nexists \mathcal{H}_{gd}(a_{i,t}) \wedge \exists \mathcal{H}_{ld}(a_{i,t})$ then
7             $\mathcal{H}_{gd}(a_{i,t}) \leftarrow \mathcal{H}_{ld}(a_{i,t})$
8 return ngd
```

The algorithm iterates over all of the agents and time-steps in the plan (Lines 2-3). For every agent and step, in case the health states in the global diagnosis is different than the health state of the local diagnosis, the combining fails, and the algorithm returns no updated global diagnosis (Lines 4-5). Otherwise, if the global diagnosis does not have a health state concerning an action $a_{i,t}$ and the local diagnosis does have it, then the health state of $a_{i,t}$ in the global diagnosis is updated according to the value in the local diagnosis (Lines 6-7). Once the algorithm finishes iterating (Line 8), it returns the updated global diagnosis.

5.5 Proofs of completeness and privacy

We prove completeness by showing that Dec-MAS-DX returns the same set of diagnoses as MAS-DX. We first define the Projected MA-STRIPS Diagnosis problem:

▶ **Definition 17** (Projected MA-STRIPS Diagnosis problem). *Let $DX = (\Pi, \pi, O, M_f, M_c)$ be a given MA-STRIPS diagnosis problem and $Ag \subseteq \{1, \ldots, k\}$ be a set of agents. We define the Projected MA-STRIPS Diagnosis problem Dx_{Ag} as the projection of DX that includes only fluents and actions relevant to at least one of the agents in Ag.*

The problem Dx_{Ag} can be viewed as a diagnosis problem by itself. Hence, we can model it as a SAT model in the same way we model a local MA-STRIPS diagnosis problem. We denote this model as $DX\text{-}Model_{Ag}$. We next prove the following theorem:

▶ **Theorem 18** (Invariant: Consistency of combined diagnosis). *After agent i finishes combining the set of local diagnoses LD_i with the set of global diagnoses GD_{i-1}, the set of global diagnoses GD_i includes the exact diagnosis set received by solving the SAT model $DX\text{-}Model_{1,\ldots,i}$.*

Proof. By induction on the number of agents.

Base (agent 1): For the first agent the diagnoses in LD_1 are inserted directly GD_1. Hence, GD_1 is the set of diagnoses obtained by solving the SAT model $DX\text{-}Model_1$.

Inductive Hypothesis: Assume for agent i.

Inductive Step: Consider agent $i + 1$, its set of local diagnoses LD_{i+1} and the set of global diagnoses GD_i received from agent i. Let there be a global diagnosis $gd \in GD_i$ and a local diagnosis $ld \in LD_{i+1}$ that agent $i + 1$ currently tries to combine. Consider the health values $\mathcal{H}_{gd}(a_{i',t})$ and $\mathcal{H}_{ld}(a_{i',t})$ of some action for agent i' at some time t according to gd and ld, respectively. We consider three cases:

1. The health variable is $\mathcal{H}(a_{i',t})$ has a value in ld but does not have a value in gd. In this case, action $a_{i',t}$ is not relevant for any of the agents $1,...,i$, and because of that, the global diagnosis does not keep a health state value. By the inductive hypothesis, that means that in *DX-Model*$_{1,...,i}$ there is no variable representing $a_{i',t}$. In the local model of agent $i+1$ there is a variable for action $a_{i',t}$, and there are constraints related to this variable. Hence, when adding the variable and the constraints to *DX-Model*$_{1,...,i}$ one possible diagnosis will be gd with the addition of the value of $\mathcal{H}_{ld}(a_{i',t})$ as set in ld. The *Combine* method of Dec-MAS-DX (given in Algorithm 2) returns the same diagnosis.
2. The health value of $\mathcal{H}(a_{i',t})$ in gd and ld is the same. In this case, the existence of a health value for $a_{i',t}$ both in gd and ld implies that the variable for that action exists both in *DX-Model*$_{1,...,i}$ and the local model of agent $i+1$. Additionally, $\mathcal{H}_{gd}(a_{i',t}) = \mathcal{H}_{ld}(a_{i',t})$ implies that the health state value for action $a_{i',t}$ in gd is consistent with the constraints in the model of agent $i+1$. This means that if the constraints related to action $a_{i',t}$ in the model of agent $i+1$ are added to *DX-Model*$_{1,...,i}$, the diagnosis gd will be generated. In that case, the diagnosis gd is consistent with *DX-Model*$_{1,...,i+1}$.
3. The health value of $\mathcal{H}(a_{i',t})$ in gd and ld is not the same. Similarly to case 2, the existence of a health value for $a_{i',t}$ both in gd and ld implies that the variable for that action exists both in *MA-STRIPS-DX*$_{1,...,i}$ and the local model of agent $i+1$. However, $\mathcal{H}_{gd}(a_{i',t}) \neq \mathcal{H}_{ld}(a_{i',t})$ implies that the health value of $a_{i',t}$ that is consistent with the constraints of *MA-STRIPS-DX*$_{1,...,i}$ is not consistent with the constraints of the local model of agent $i+1$ and hence, the set of values in gd combined with the value $\mathcal{H}_{ld}(a_{i',t})$ are not part of a diagnosis, consistent with *MA-STRIPS-DX*$_{1,...,i+1}$. In that case, the combine method of Dec-MAS-DX does not continue with the combination process and returns *none* instead, indicating that the combination of gd and ld is not valid. ◀

▶ **Theorem 19** (Completeness). *Dec-MAS-DX returns all the diagnoses.*

Proof. We prove completeness by showing that Dec-MAS-DX returns the same set of diagnoses as our centralized algorithm, MAS-DX.

From Theorem 18 it follows that when agent k finishes combining the diagnoses, the global diagnoses it returns is the set of diagnoses corresponding to the set of diagnoses returned by solving *DX-Model*$_{1,...,k}$, which is the same as the centralized model. Hence, the global diagnoses returned by agent k are the same as the diagnoses returned by MAS-DX. ◀

▶ **Theorem 20** (Privacy). *If a fluent f is only relevant to a single agent, then no other agent can infer its value during the diagnosis process.*

This statement about privacy stems from the fact that the value of this fluent is not in the local view of any other agent. Moreover, whether the value of this fluent is true or false has no direct impact on the applicability or health state of any action of any other agents. Otherwise, it would have been relevant. Note that some fluents that are public, in terms of MA-STRIPS, may still be only relevant to one agent due to the concrete plan the agents were trying to execute.

5.6 Enhancing the Distributed Algorithm

In our Dec-MAS-DX algorithm, once the problem is divided, each local problem has fewer variables in its local MBD formulation but also fewer constraints. This expands the search space, especially for agents that have a lot of external relevant actions.

To address this, we propose an enhanced version of Dec-MAS-DX, which we call Dec-MAS-DX+, in which the agents decide on a computation order based on the upper bound of possible diagnoses of each agent. This number can easily be deduced from the number of possible health assignments of each agent's relevant actions. If agent i is the one with the lowest bound, it begins its local diagnosis process. Upon finishing, agent i updates the possible health states of its relevant actions. For example, if all of the local diagnoses state that action $a_{i,t}$ is healthy, then agent i updates the possible health values of action $a_{i,t}$ to exclusively healthy. This update includes external actions that are visible to other agents. Then, the remaining agents use the updated possible health states to compute an updated upper bound for the number of diagnoses and decide which agent to perform the local diagnosis next, and so on. At the end of this process, the agents perform the combination process as shown in Algorithm 1, in order to get the final set of global diagnoses.

6 Evaluation

6.1 Experimental Setup

We evaluated our algorithms on eight MA-STRIPS domains *blocksworld*, *depot*, *diverlog*, *logistics*, *rovers*, *satellite*, *taxi* and *zenotravel* taken from the *Competition of Distributed and Multiagent Planners*.[1] For each domain, we experimented with ten problems of different difficulty levels. For each problem, we used an off-the-shelf planning algorithm to generate a plan.[2] Then, for each such problem, we injected $f \in \{1, 2, 3, 4, 5\}$ faulty actions. For each number of faults, we simulated 10 faulty plan executions. For each instance, we observed $o \in \{1, 10, 20, 100\}$ percentage of states, where 1% denotes that we observed the initial and final states, and 100% denotes that we observed every state. In total, we got 2000 instances for every domain. We used the Java Choco library [21] to model the MBD problems as Constraint Satisfaction Problems (CSP) and solve them. We decided to use this library since it is convenient to use and because internally it translates the CSP to SAT.

We then ran each of the algorithms we proposed. Namely, we run the centralized MAS-DX, as well as the distributed algorithms Dec-MAS-DX and Dec-MAS-DX+ (the enhanced version of Dec-MAS-DX that intelligently prioritizes the agents). We limited every instance to run for no more than 10 seconds. We measured the run-time of solving the problems.

6.2 Results

Table 2 presents the average run-time in milliseconds for each algorithm in each domain. We can observe that MAS-DX significantly outperforms in the domains *blocksworld*, *depot*, *rovers*, *taxi* and *zenotravel* (highlighted in bold) and does not outperform in the *driverlog logistics* and *rovers* domains. The reason is that the level of interaction between the actions of the agents differs among the different domains. In the domains where MAS-DX outperforms, the actions of one agent highly interact with the actions of other agents. This is due to many fluents in the plan that are relevant to a number of agents. Therefore, the separation of the central problem to local problems will result in the local problems becoming under-constrained due to many actions becoming external in the local views of the agents. This expands the diagnosis search space, meaning that more local diagnoses will be found. To demonstrate this, we measured the number of local diagnoses found by the agent with the slowest runtime as shown in Table 3. Results show that the decentralized algorithms performed worse in domains with a significantly large number of local diagnoses.

[1] `http://agents.fel.cvut.cz/codmap/`
[2] `https://api.planning.domains/`

Table 2 Average run-time in milliseconds of the different algorithms for each domain.

Domain	MAS-DX	Dec-MAS-DX	Dec-MAS-DX+
blocksworld	**31**	4656	3001
depot	**782**	2994	2010
driverlog	72	71	67
logistics00	142	136	**13**
rovers	**18**	1819	1567
satellite	21	11	18
taxi	**12**	2249	119
zenotravel	**7**	3104	2307

Table 3 Average number of local diagnoses computed by the slowest agent in each algorithm.

Domain	MAS-DX	Dec-MAS-DX	Dec-MAS-DX+
blocksworld	3	4721	3275
depot	115	2534	1434
driverlog	14	14	14
logistics00	66	177	7
rovers	3	3350	3057
satellite	1	1	1
taxi	5	4274	148
zenotravel	1	5062	4622

At this stage, we emphasize that the purpose of Dec-MAS-DX and Dec-MAS-DX+ is to address privacy. In both algorithms, each agent is familiar only with the actions relevant to it. We assume that information concerning private actions is not known publicly. Hence, since the process of modeling and solving only accesses the actions of the diagnosing agent and other publicly known actions, no private actions are being shared. This makes the decentralized algorithms relevant to applications where a number of self-interested entities collaborate to achieve a common goal that benefits all of them but where disclosing private information is not desirable.

Another thing to be observed is that Dec-MAS-DX+ significantly improves Dec-MAS-DX in almost every domain. In many cases, distributing the problem leads to uneven distribution of external actions among the agents. This means that for some agents, the local diagnosis space is significantly higher than for others. By solving the local problems for agents with a lower number of external actions first and then using the diagnosis information when solving the local problems of other agents, Dec-MAS-DX+ significantly decreases the run-time of the agent with the largest diagnosis space. This decrease in runtime is so large that it compensates for the serial manner in which Dec-MAS-DX+ works. Specifically in the *logistics* domain, this speedup improvement, combined with the fact that the number of external actions is low, leads to Dec-MAS-DX+ outperforming MAS-DX by an order of magnitude.

7 Conclusions and Future Work

In this work, we presented the problem of MA-STRIPS diagnosis and proposed two approaches for solving the problem. The first approach, MAS-DX, works centrally by modeling the system as a single MBD problem and then solving it. The second approach, Dec-MAS-DX, addresses privacy requirements by solving a decentralized version of the problem, where agents do not share private information. It first models the local views of the agents to MBD problems of smaller sizes and concurrently solves them, and then computes the global diagnoses by executing a serial process for combining the local diagnoses. Additionally, we proposed an enhancement called Dec-MAS-DX+ that improves Dec-MAS-DX. Empirical evaluation shows that the centralized approach is faster than the distributed approach in most domains, but the benefit of the distributed approach is that it preserves private information.

For future work, we intend to expand our empirical evaluation to other problems and also propose an analysis of the problems in which the distributed approach may run faster. In addition, we aim to generalize our work to multi-agent plans in richer planning formalisms that support stochastic effects and partial observability.

References

1 Craig Boutilier and Ronen I Brafman. Partial-order planning with concurrent interacting actions. *Journal of Artificial Intelligence Research*, 14:105–136, 2001. doi:10.1613/JAIR.740.

2 Ronen Brafman and Uri Zoran. Distributed heuristic forward search for multi-agent systems. In *ICAPS DMAP workshop*, pages 1–6, 2014.

3 Ronen I Brafman and Carmel Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In *ICAPS*, pages 28–35, 2008. URL: http://www.aaai.org/Library/ICAPS/2008/icaps08-004.php.

4 Ronen I Brafman and Carmel Domshlak. On the complexity of planning for agent teams and its implications for single agent planning. *Artificial Intelligence*, 198:52–71, 2013. doi:10.1016/J.ARTINT.2012.08.005.

5 Matthew Crosby, Anders Jonsson, and Michael Rovatsos. A single-agent approach to multiagent planning. In *ECAI*, pages 237–242, 2014. doi:10.3233/978-1-61499-419-0-237.

6 Matthew Daigle, Xenofon Koutsoukos, and Gautam Biswas. Distributed diagnosis of coupled mobile robots. In *Proceedings 2006 IEEE ICRA, 2006.*, pages 3787–3794. IEEE, 2006. doi:10.1109/ROBOT.2006.1642281.

7 Femke De Jonge, Nico Roos, and Cees Witteveen. Primary and secondary diagnosis of multi-agent plan execution. *AAMAS*, 18(2):267–294, 2009. doi:10.1007/S10458-008-9045-X.

8 Orel Elimelech, Roni Stern, Meir Kalech, and Yedidya Bar-Zeev. Diagnosing resource usage failures in multi-agent systems. *ESWA*, 77:44–56, 2017. doi:10.1016/J.ESWA.2017.01.047.

9 Alexander Feldman, Gregory M Provan, and Arjan JC van Gemund. Solving strong-fault diagnostic models by model relaxation. In *IJCAI*, volume 9, pages 785–790, 2009.

10 Jun Gu, Paul W Purdom, John Franco, and Benjamin W Wah. Algorithms for the satisfiability (sat) problem: A survey. *Satisfiability problem: Theory and applications*, 35:19–152, 1996. doi:10.1090/DIMACS/035/02.

11 Meir Kalech and Gal A Kaminka. Towards model-based diagnosis of coordination failures. In *AAAI*, volume 5, pages 102–107, 2005. URL: http://www.aaai.org/Library/AAAI/2005/aaai05-017.php.

12 Meir Kalech and Gal A Kaminka. Coordination diagnostic algorithms for teams of situated agents: Scaling up. *Computational Intelligence*, 27(3):393–421, 2011. doi:10.1111/J.1467-8640.2011.00386.X.

13 Meir Kalech and Avraham Natan. Model-based diagnosis of multi-agent systems: A survey. In *AAAI*, volume 36, pages 12334–12341, 2022. doi:10.1609/AAAI.V36I11.21498.

14 Shlomi Maliah, Guy Shani, and Roni Stern. Privacy preserving landmark detection. In *ECAI 2014*, pages 597–602. IOS Press, 2014. doi:10.3233/978-1-61499-419-0-597.

15 Shlomi Maliah, Guy Shani, and Roni Stern. Privacy preserving pattern databases. In *Proceedings of the 3rd Distributed and Multiagent Planning (DMAP) Workshop of ICAPS*, volume 15, pages 9–17, 2015.

16 Roberto Micalizio. A distributed control loop for autonomous recovery in a multi-agent plan. In *21st IJCAI*, 2009.

17 Argaman Mordoch, Brendan Juba, and Roni Stern. Learning safe numeric action models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 12079–12086, 2023. doi:10.1609/AAAI.V37I10.26424.

18 Avraham Natan and Meir Kalech. Privacy-aware distributed diagnosis of multi-agent plans. *ESWA*, 192:116313, 2022. doi:10.1016/J.ESWA.2021.116313.

19 Tuan Nguyen, Sarath Sreedharan, and Subbarao Kambhampati. Robust planning with incomplete domain models. *Artificial Intelligence*, 245:134–161, 2017. doi:10.1016/J.ARTINT.2016.12.003.

20 Raz Nissim, Ronen I Brafman, and Carmel Domshlak. A general, fully distributed multi-agent planning algorithm. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1323–1330. International Foundation for Autonomous Agents and Multiagent Systems, 2010. URL: https://dl.acm.org/citation.cfm?id=1838379.

21 Charles Prud'homme and Jean-Guillaume Fages. Choco-solver: A java library for constraint programming. *Journal of Open Source Software*, 7(78):4708, 2022. doi:10.21105/joss.04708.

22 Raymond Reiter. A theory of diagnosis from first principles. *Artificial intelligence*, 32(1):57–95, 1987. doi:10.1016/0004-3702(87)90062-2.

23 Nico Roos and Cees Witteveen. Models and methods for plan diagnosis. *AAMAS*, 19(1):30–52, 2009. doi:10.1007/S10458-007-9017-6.

24 Shashank Shekhar and Ronen I Brafman. Representing and planning with interacting actions and privacy. *Artificial Intelligence*, 278:103200, 2020. doi:10.1016/J.ARTINT.2019.103200.

25 Tran Cao Son, William Yeoh, Roni Stern, and Meir Kalech. Multi-agent planning and diagnosis with commonsense reasoning. In *The Fifth International Conference on Distributed Artificial Intelligence*, pages 1–9, 2023. doi:10.1145/3627676.3627690.

26 Michal Štolba and Antonín Komenda. Relaxation heuristics for multiagent planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 24, pages 298–306, 2014.

27 Michal Štolba, Antonin Komenda, and Daniel Kovacs. Competition of distributed and multiagent planners (codmap). In *AAAI Conference on Artificial Intelligence*, volume 30, 2016.

28 Alejandro Torreno, Oscar Sapena, and Eva Onaindia. Global heuristics for distributed cooperative multi-agent planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 25, pages 225–233, 2015.

29 Gianluca Torta, Roberto Micalizio, and Samuele Sormano. Explaining failures propagations in the execution of multi-agent temporal plans. In *ICAPS*, pages 2232–2234, 2019. URL: http://dl.acm.org/citation.cfm?id=3332068.