Planning Domain Repair as a Diagnosis Problem

Songtuan Lin, Alban Grastien, Pascal Bercher

Planning Domain Repair as a Diagnosis Problem

Songtuan Lin and **Alban Grastien** and **Pascal Bercher**
School of Computing, The Australian National University
firstName.lastName@anu.edu.au

Abstract

Techniques for diagnosis have been used in many applications. We explore the connection between diagnosis and AI planning in this paper and apply the diagnosis algorithm to repair a flawed planning domain. In particular, the scenario we are concerned with is that we are given a plan which is supposed to be a solution to a planning problem, but it is actually not due to some flaws in the planning domain, and we want to repair the domain to turn the plan into a solution. For this, we will first frame this problem as a diagnosis problem and then solve it via using diagnosis algorithms.

1 Introduction

Diagnosis refers to the task of finding a set of faulty components in a system. The theory for diagnosis developed by [1; 2] being model based, its principles are independent from the specific problem of diagnosis. Consequently, diagnosis intersects with a wide range of applications.

In this paper, we are interested in the application of diagnosis techniques in AI planning [3; 4], which is the task of generating a course of *actions* automatically which achieve some certain *goals*. In particular, we are concerned with exploiting the diagnosis algorithm to repair *planning domains* that are *not* adequately engineered.

Concretely, modeling the domain of a planning problem, known as domain engineering [5], is the process of abstracting practical (planning) problems as mathematical models, which can then be solved by exploiting automated planning techniques. This is unfortunately not a trivial process, and the competition for this, i.e., International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS), has been held for many years. Therefore, errors occur quite often in engineered planning domains, and hence, how to provide assistance for correcting a flawed planning domain is regarded as a critical problem [6].

The scenario we consider for repairing a flawed planning domain is as follows. We are given a plan, i.e., a sequence of actions, and a planning domain in which the given plan is *not* a solution due to some existing flaws in the domain. Our goal here is repairing the domain, i.e., finding possible flaws in the domain, so that the plan will be a solution. The scenario is initially studied by [7]. However, the authors there only investigated the computational complexity of the problem and did not propose practical approaches for solving the problem. Thus, we present such an approach here which first frames the problem of repairing a planning domain as the diagnosis problem and then plug in the diagnosis algorithm for solving it.

Specifically, we view each atomic repair as a component in the domain, where this component is abnormal if the atomic repair should be performed. We then formulate the repair problem as a diagnosis one, in which a (minimal) diagnosis can be mapped to a (minimal) repair. Diagnosis algorithms can then be used to efficiently compute the repair.

The paper is organized as follows. We start by introducing the frameworks of *model based diagnosis* and *planning*. Afterward, we formalize the problem of repairing planning domains. After that, we present how to formulate a domain repair problem as a diagnosis problem and some practical realizations for exploiting the diagnosis algorithm to solve the domain repair problem. We then present the experimental results about our approach, and lastly, we discuss some related works centered in repairing and modifying planning domains.

2 Preliminaries

For the purpose of illustrating how to exploit diagnosis techniques to repair flawed planning domains, we shall first introduce the diagnosis framework and the planning framework employed in this paper.

2.1 Model Based Diagnosis

We start by introducing the diagnosis framework employed in the paper. The framework is based on the classical definitions of *model based diagnosis* [1], which we recap here.

A system is *modeled* by a pair $(\texttt{Comps}, \texttt{SD})$ in which $\texttt{Comps}$ is a (finite) set of *components* and $\texttt{SD}$ (the *system description*) is a logical statement. Importantly, $\texttt{SD}$ relies on the unary predicate $\texttt{Ab}(c)$ where $c \in \texttt{Comps}$ is a component; $\texttt{Ab}(c)$ is a boolean proposition that evaluates to *true* whenever c behaves in a faulty (*abnormal*) manner.

An *observation* $\texttt{Obs}$ is a logical statement on some of the variables mentioned by the system description.

A *diagnosis problem* is a triple $(\texttt{Comps}, \texttt{SD}, \texttt{Obs})$ where $(\texttt{Comps}, \texttt{SD})$ is a system model and $\texttt{Obs}$ an observation.

A *(diagnosis) candidate* is a subset $\delta \subseteq \texttt{Comps}$ of components that are assumed to be faulty; by default, the other components are assumed to be behaving normally. The logical interpretation of candidate δ, written $\varphi_{\texttt{Comps}}(\delta)$ or $\varphi(\delta)$ when the set of components is obvious from the context, is defined as follows:

$$\bigwedge_{c \in \delta} \texttt{Ab}(c) \wedge \bigwedge_{c \in \texttt{Comps} \setminus \delta} \neg \texttt{Ab}(c).$$

A diagnosis candidate δ is called a *diagnosis* of some diagnosis problem $(\texttt{Comps}, \texttt{SD}, \texttt{Obs})$ if it is logically consistent with the model and the observations:

$$\texttt{SD}, \texttt{Obs}, \varphi(\delta) \not\models \bot.$$

We write Δ the set of diagnoses. A diagnosis is *minimal* if none of its strict subsets is a diagnosis:

$$\nexists \delta' \in \Delta.\ \delta' \subset \delta.$$

A diagnosis is *cardinality-minimal* if no other diagnosis involves fewer components:

$$\nexists \delta' \in \Delta.\ |\delta'| < |\delta|.$$

Cardinality-minimal diagnoses are minimal diagnoses.

Solving diagnosis problems The theory of model based diagnosis developed by [1] and [2] relies heavily on the notion of conflicts which we now review.

Given a diagnosis problem $(\texttt{Comps}, \texttt{SD}, \texttt{Obs})$, a *conflict* C is a subset of components one of which must be faulty. This is formally expressed as follows:

$$\texttt{SD}, \texttt{Obs}, \bigwedge_{c \in C} \neg \texttt{Ab}(c) \models \bot.$$

As a consequence, each diagnosis δ must intersect with (i.e., "hit") each conflict C: $\delta \cap C \neq \emptyset$. Actually, every minimal diagnosis is a minimal hitting set of the set of (subset-)minimal conflicts, where a hitting set is defined as follows:

Let $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a collection of sets; a hitting set of $\mathcal{C}$ is a set H that intersects every one of its sets:

$$\forall i \in \{1, \ldots, k\}.\ C_i \cap H \neq \emptyset.$$

A simple diagnosis algorithm is given in Alg. 1. The algorithm maintains a collection $\mathcal{C}$ of (possibly non-minimal) conflicts (initially empty). At each iteration, it computes a minimal hitting set for the collection and tests whether the candidate is a diagnosis. If it is, the algorithm returns this minimal diagnosis. Otherwise, a conflict is extracted from the inconsistency proof which is added to $\mathcal{C}$.

Alg. 1 relies on two operators. The first operator verifies whether a given diagnosis candidate is a diagnosis, and if it is not, the operator returns a conflict. We call this operator an *oracle*. Its implementation is problem-dependent, and we show in Sec. 4 how it is implemented for our problem. The second operator computes hitting sets from a collection of conflicts; we provide more details on this tool in the experimental section. Notice that if this operator returns minimal cardinality hitting sets, then the procedure returns a minimal cardinality diagnosis, as proved by [1], by [2], and by [8].

Additionally, one remark is that the correctness of Alg. 1 is based upon the set Δ of all diagnoses of a diagnosis problem being *monotonic*, that is, for any $\delta \in \Delta$, if $\delta \subseteq \delta'$ for some diagnosis candidate δ', then $\delta' \in \Delta$.

Algorithm 1 Diagnosis algorithm

Input: A diagnosis problem $(\texttt{Comps}, \texttt{SD}, \texttt{Obs})$
Output: A minimal diagnosis

▷ Collection of known conflicts
$\mathcal{C} \leftarrow \emptyset$
loop
  $\delta \leftarrow$ a minimal hitting set of $\mathcal{C}$
  if $\texttt{SD}, \texttt{Obs}, \varphi(\delta) \not\models \bot$ **then**
    return δ
  $C \leftarrow$ a conflict C with $C \subseteq \texttt{Comps} \setminus \delta$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$

2.2 Planning Formalism

We move on now to introduce the planning framework used in the paper. The planning formalism we are concerned with is the *grounded* $\mathcal{STRIPS}$ formalism [9]. By grounded, we mean that the formalism is defined in terms of *propositional logic*, i.e., without variables. A generalization of it, called the *lifted* $\mathcal{STRIPS}$ formalism, is defined in terms of *first order logic*, i.e., with variables, which will only be considered in our future work. Hence, for convenience, unless otherwise specified, we will use the $\mathcal{STRIPS}$ formalism to refer to the grounded version. We start by presenting the definition of $\mathcal{STRIPS}$ planning problems, and afterward we will explain each element in a planning problem.

A $\mathcal{STRIPS}$ planning problem $\mathcal{P}$ is defined as a tuple $(\mathcal{F}, \mathcal{A}, \alpha, s_I, g)$ where $\mathcal{F}$ is a set of propositions, $\mathcal{A}$ is a set of *actions*, $\alpha : \mathcal{A} \rightarrow 2^{\mathcal{F}} \times 2^{\mathcal{F}} \times 2^{\mathcal{F}}$ is a function, and $s_I \in 2^{\mathcal{F}}$ and $g \subseteq \mathcal{F}$ are called the initial state and the goal description of $\mathcal{P}$, respectively. In particular, $\mathcal{F}$ together with $\mathcal{A}$ and α is called the *domain* of $\mathcal{P}$.

In the $\mathcal{STRIPS}$ planning formalism, a set of propositions $s \in 2^{\mathcal{F}}$ is called a *state*, and applying an action $a \in \mathcal{A}$ in a state leads to a new state. Concretely, an action a is mapped to the respective *precondition* $prec(a) \subseteq \mathcal{F}$, *positive effects* $\mathit{eff}^+(a) \subseteq \mathcal{F}$, and *negative effects* $\mathit{eff}^-(a) \subseteq \mathcal{F}$ by the function α, written as $\alpha(a) = (prec(a), \mathit{eff}^+(a), \mathit{eff}^-(a))$. Applying an action a in some state s will lead to a new state s' such that $s' = (s \setminus \mathit{eff}^-(a)) \cup \mathit{eff}^+(a)$, written $s \rightarrow_a s'$. Given a sequence of actions $\pi = \langle a_1 \cdots a_n \rangle$, we write $s \rightarrow_\pi^* s'$ to indicate that the state s' is obtained by applying the action sequence π in s, that is, there exists a state sequence $\langle s_0 \cdots s_n \rangle$ such that $s_0 = s$, $s_n = s'$, and for each $1 \leq i \leq n$, $s_{i-1} \rightarrow_{a_i} s_i$. Further, an action a is *applicable* in a state s if $prec(a) \subseteq s$ (i.e., the precondition of a is satisfied in s).

Having presented the definition of planning problems, we now give the solution criteria for a planning problem which is a sequence of actions achieving the goal description in the planning problem. Concretely, let $\mathcal{P}$ be a planning problem. A solution to $\mathcal{P}$ is an action sequence $\pi = \langle a_1 \cdots a_n \rangle$ such that applying π in the initial state s_I results in the state sequence $\overline{s} = \langle s_0 \cdots s_n \rangle$, i.e., $s_0 = s_I$ and $s_0 \rightarrow_\pi^* s_n$, where $g \subseteq s_n$ and for each $1 \leq i \leq n$, a_i is applicable in s_{i-1}.

2.3 Planning Domain Repair Problem

Having presented the planning framework we will use, now we would like to introduce the problem we want to solve, namely the problem of repairing a flawed planning domain. The problem is formally introduced by [7] and proved to be $\mathbb{NP}$-complete. We reproduce their definition in this section.

Recall that the basic configuration of the problem is that we are given a planning problem $\mathcal{P}$ and a plan π which is *not* a solution to $\mathcal{P}$. We want to repair (i.e., change) the domain of $\mathcal{P}$ so that π will be a solution.

For the purpose of formulating the problem precisely, we first define repairs that are allowed to be used in correcting a planning domain. In the paper, we only consider repairing actions' preconditions and effects and will *not* add or delete propositions from $\mathcal{F}$. Specifically, repairs we are concerned with are restricted to removing propositions from actions'

preconditions, adding propositions to actions' positive effects, and removing propositions from actions' negative effects. The reason for making such a restriction is that other changes (e.g., adding propositions to actions' preconditions) only increase the chance of a plan not being executable. We formally define the sets of all atomic repairs respectively targeted at an action and a planning problem as follows:

Definition 1. *Let $a \in \mathcal{A}$ be an action. The set F_a of all atomic repairs targeted at the action a is defined as $F_a = F_a^p \cup F_a^+ \cup F_a^-$ in which*

- $F_a^p = \{\langle F_a|_f^p\rangle \mid f \in prec(a)\}$
- $F_a^+ = \{\langle F_a|_f^+\rangle \mid f \in \mathcal{F}\}$
- $F_a^- = \{\langle F_a|_f^-\rangle \mid f \in \mathit{eff}^-(a)\}$

Consequently, given a planning problem $\mathcal{P}$, the set $F_\mathcal{P}$ of all atomic repairs for $\mathcal{P}$ is $\bigcup_{a\in\mathcal{A}} F_a$.

Intuitively speaking, the semantics of $\langle F_a|_f^p\rangle$ is removing the proposition f from the precondition of a, and similarly, $\langle F_a|_f^+\rangle$ and $\langle F_a|_f^-\rangle$ respectively adds f to the positive effects of a and removes f from the negative effects of a. We formally define the semantics of these repairs in terms of the consequence of applying them to a planning domain.

Definition 2. *Let $\mathcal{P} = (\mathcal{F}, \mathcal{A}, \alpha, s_I, g)$ be a planning problem and $F'_\mathcal{P}$ a subset of $F_\mathcal{P}$, i.e., $F'_\mathcal{P} \subseteq F_\mathcal{P}$. The consequence of applying $F'_\mathcal{P}$ to $\mathcal{P}$ is a new planning problem $\mathcal{P}'$ such that $\mathcal{P}' = (\mathcal{F}, \mathcal{A}, \alpha', s_I, g)$, and for every action $a \in \mathcal{A}$ with $\alpha(a) = (prec(a), \mathit{eff}^+(a), \mathit{eff}^-(a))$, $\alpha'(a) = (prec(a)\backslash P, \mathit{eff}^+(a)\cup E^+, \mathit{eff}^-(a)\backslash E^-)$ in which*

- $P = \{f \mid \langle F_a|_f^p\rangle \in F'_\mathcal{P}\}$
- $E^+ = \{f \mid \langle F_a|_f^+\rangle \in F'_\mathcal{P}\}$
- $E^- = \{f \mid \langle F_a|_f^-\rangle \in F'_\mathcal{P}\}$

We write $\mathcal{P} \Rightarrow_{F'_\mathcal{P}} \mathcal{P}'$ to indicate that $\mathcal{P}'$ is obtained by applying a set $F'_\mathcal{P}$ of repairs to $\mathcal{P}$.

Another interpretation of a repair in $F_\mathcal{P}$ is to view it as a *flaw* that need to be repaired. For instance, the repair $\langle F_a|_f^+\rangle$ can be interpreted as the flaw that the positive effects of a lack the proposition f. One remark is that, as we will see in the later section, such an interpretation is in line with the semantics of a component being *abnormal* when formulating a domain repair problem as a diagnosis problem.

After defining the set of atomic repairs for a planning problem together with the respective semantics, we now formulate the problem of finding a set of repairs that turns a non-solution action sequence into a solution.

Definition 3. *Given a planning problem $\mathcal{P}$ and an action sequence $\pi = \langle a_1 \cdots a_n\rangle$ with $a_i \in \mathcal{A}$ for each $1 \le i \le n$, a domain repair problem is the tuple $\Pi = (\mathcal{P}, \pi)$ that is to find a minimal cardinality subset $F^*_\mathcal{P} \subseteq F_\mathcal{P}$ such that $\mathcal{P} \Rightarrow_{F^*_\mathcal{P}} \mathcal{P}^*$ for some $\mathcal{P}^*$, and π is a solution to $\mathcal{P}^*$.*

Notice that here we demand that a repair set found for the domain repair problem must have the *minimal cardinality*, because otherwise, we can *always* find the repair set which empties the precondition of *every* action in the plan and adds all propositions in the goal description to the last action in the plan. Further, we emphasize that we again distinct the notion of a *minimal* repair set from that of a *minimal cardinality* repair set. More specifically, let Δ be the set of all repair sets $F^\dagger_\mathcal{P} \subseteq F_\mathcal{P}$ such that $\mathcal{P} \Rightarrow_{F^\dagger_\mathcal{P}} \mathcal{P}^\dagger$ and π is a solution to $\mathcal{P}^\dagger$. A set $F^\dagger_\mathcal{P} \in \Delta$ is said to be minimal if for any subset $F'_\mathcal{P} \subseteq F^\dagger_\mathcal{P}$, $F'_\mathcal{P} \notin \Delta$, and $F^\dagger_\mathcal{P}$ is of the minimal cardinality if $|F^\dagger_\mathcal{P}| \le |F'_\mathcal{P}|$ for any $F'_\mathcal{P} \in \Delta$.

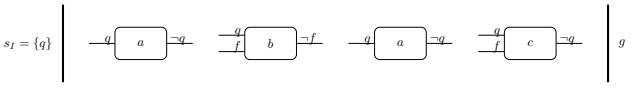

Figure 1: An example of the domain repair problem where the planning problem is $\mathcal{P} = (\mathcal{F}, \mathcal{A}, \alpha, s_I, g)$ with $\mathcal{F} = \{q, f\}$ and $\mathcal{A} = \{a, b, c\}$, and the plan is $\pi = \langle a\ b\ a\ c\rangle$. Propositions placed before an action in the figure are propositions in the precondition of the action, and propositions placed after an action are those in the effects. A proposition without the symbol $\neg$ in front of it is in the positive effects of an action, otherwise, it is in the negative effects. The solution to the problem is the set of repairs $\{\langle F_a|_q^-\rangle, \langle F_a|_f^+\rangle\}$.

Fig. 1 illustrates an example of the domain repair problem. Consider the planning problem $\mathcal{P} = (\mathcal{F}, \mathcal{A}, \alpha, s_I, g)$ in which $\mathcal{F} = \{q, f\}$, $\mathcal{A} = \{a, b, c\}$, $s_I = \{q\}$, $g = \emptyset$, and α is the function such that

- $\alpha(a) = (\{q\}, \emptyset, \{q\})$
- $\alpha(b) = (\{q, f\}, \emptyset, \{f\})$
- $\alpha(c) = (\{q, f\}, \emptyset, \{q\})$

The plan $\pi = \langle a\ b\ a\ c\rangle$ in Fig. 1 is not executable because the action b in the plan is not applicable in the state obtained by applying a in the initial state, and because the propositions q and f demanded by the actions a and c, respectively, are removed from the states after applying the actions a and b. The domain repair problem is to find a minimal cardinality subset $F^*_\mathcal{P}$ of $F_\mathcal{P}$ such that π will be a solution to the planning problem $\mathcal{P}^*$ obtained by applying $F^*_\mathcal{P}$. In this specific setting, we have $F_\mathcal{P} = \mathcal{F}_a \cup \mathcal{F}_b \cup \mathcal{F}_c$ in which

- $F_a = \{\langle F_a|_q^p\rangle, \langle F_a|_q^+\rangle, \langle F_a|_f^+\rangle, \langle F_a|_q^-\rangle\}$
- $F_b = \{\langle F_b|_q^p\rangle, \langle F_b|_f^p\rangle, \langle F_b|_q^+\rangle, \langle F_b|_f^+\rangle, \langle F_b|_f^-\rangle\}$
- $F_c = \{\langle F_c|_q^p\rangle, \langle F_c|_f^p\rangle, \langle F_c|_q^+\rangle, \langle F_c|_f^+\rangle, \langle F_c|_q^-\rangle\}$

One set of repairs which can make the plan π be a solution is $F'_\mathcal{P} = \{\langle F_a|_q^p\rangle, \langle F_b|_q^p\rangle, \langle F_b|_f^p\rangle, \langle F_c|_q^p\rangle, \langle F_c|_f^p\rangle\}$, i.e., removing the preconditions of a, b and c. This is however *not* a solution to the domain repair problem because it is not optimal. The optimal one is $F^*_\mathcal{P} = \{\langle F_a|_q^-\rangle, \langle F_a|_f^+\rangle\}$ which is thus the solution to the domain repair problem.

3 Formulating Domain Repair Problem as Diagnosis Problem

In this section, we show how to formulate a domain repair problem as a diagnosis problem which can then be solved by exploiting the generic diagnosis algorithm given in Alg. 1.

For this, given a domain repair problem $\Pi = (\mathcal{P}, \pi)$, we first specify the set of components in the respective diagnosis problem. We can regard the set $F_\mathcal{P}$ of all repairs as the component set, i.e., each atomic repair is viewed as a component. The interpretation of this analogy is straightforward, that is, a component is *abnormal* if the respective *flaw* exists, i.e., the corresponding repair *should* be applied to the planning domain in order to make the plan become a solution. Naturally, the set of observations in the respective diagnosis problem consists solely of one logical statement which is evaluated to *true iff* the plan is a solution. We will introduce how this logical statement looks like after presenting the most complicated part in our formulation – the system description.

Generally speaking, the system description shall be able to describe how the system functions. In our context, this refers to 1) the state trajectory produced by applying π in the initial state, and 2) how the status of each component (i.e., whether it is normal or abnormal) affects the state trajectory.

Suppose $\overline{s} = \langle s_0 \cdots s_n \rangle$ is the state trajectory obtained by applying π in the initial state s_I, that is, $s_0 = s_I$, and $s_0 \rightarrow_\pi^* s_n$. The core aspect of encoding $\overline{s}$ as a logic statement is to use a proposition f_i to indicate whether a planning proposition $f \in \mathcal{F}$ holds in the state s_i ($0 \leq i \leq n$) and encode the effects of each action in terms of those extra propositions. More concretely, we define the augmented set of propositions:

$$\mathcal{F}^* = \{f_i \mid f \in \mathcal{F}, 0 \leq i \leq n\}$$

For some $f \in \mathcal{F}$ and $0 \leq i \leq n$, f_i is evaluated to *true* if $f_i \in s_i$, i.e., f_i indicates whether the proposition f holds in the state s_i. Consequently, the initial state can be expressed by the following formula:

$$\tau(s_0) = \left(\bigwedge_{f \in s_I} f_0 \right) \wedge \left(\bigwedge_{f \notin s_I} \neg f_0 \right)$$

The remaining states in $\overline{s}$ are affected by the effects of each action in the plan together with the state of each component. Specifically, the value of f_i is computed iteratively from f_{i-1}, the definitions of a_i, and the status of the components $\langle F_{a_i}|_f^+ \rangle$ and $\langle F_{a_i}|_f^- \rangle$. For each $f \in \mathcal{F}$, it is in some state s_i ($1 \leq i \leq n$) if one of the following conditions hold:
1) $f \in \mathit{eff}^+(a_i)$, or
2) f is in the state s_{i-1} and will *not* be deleted by a_i.
Conversely, the proposition f *cannot* be in the state s_i if one of the following two conditions hold:
1) $f \notin s_{i-1}$, $f \notin \mathit{eff}^+(a_i)$, and $\langle F_{a_i}|_f^+ \rangle$ is *not* abnormal (i.e., a_i is supposed *not* to add f to the state), or
2) $f \in s_{i-1}$, $f \in \mathit{eff}^-(a_i)$, and $\langle F_{a_i}|_f^- \rangle$ is *not* abnormal (i.e., a_i is supposed to remove f from the state).

In order to encode these criteria, we need to consider four different cases depending on the definitions of a_i:

1. If $f \in \mathit{eff}^+(a_i)$ and $f \notin \mathit{eff}^-(a_i)$, then
$$\tau(f_i) = \quad f_i$$
2. If $f \notin \mathit{eff}^+(a_i)$ and $f \notin \mathit{eff}^-(a_i)$, then
$$\tau(f_i) = \quad \left(\neg f_{i-1} \wedge \neg \mathtt{Ab}\left(\langle F_{a_i}|_f^+ \rangle \right) \right) \rightarrow \neg f_i$$
3. If $f \notin \mathit{eff}^+(a_i)$ and $f \in \mathit{eff}^-(a_i)$, then
$$\tau(f_i) = \quad \left(\neg f_{i-1} \vee \neg \mathtt{Ab}\left(\langle F_{a_i}|_f^- \rangle \right) \right) \rightarrow \neg f_i$$
4. If $f \in \mathit{eff}^+(a_i)$ and $f \in \mathit{eff}^-(a_i)$,[1] then
$$\tau(f_i) = \quad f_i$$

We then define $\tau(s_i) = \bigwedge_{f \in \mathcal{F}} \tau(f_i)$ for $i \in \{1, \ldots, n\}$, and the system description is thus

$$\mathtt{SD} = \bigwedge_{0 \leq i \leq n} \tau(s_i).$$

[1]This latter case is generally considered impossible in practice, but it can occur as a consequence of modelling mistakes or when the model is an abstraction of a more refined model.

Lastly we present the encoding of the observation $\mathtt{Obs}$ on top of the system description which asserts that the plan must be a solution. The solution criteria for a planning problem demand that the precondition of each action in the plan together with the goal description of the planning problem must be satisfied, that is, for each a_i with $1 \leq i \leq n$, if $f \in \mathit{prec}(a_i)$ and the component $\langle F_{a_i}|_f^p \rangle$ is *not* abnormal (i.e., a_i demanding f is *not* a flaw), then f must be in s_{i-1}. Consequently, the formula encoding that every action's precondition is satisfied is as follows:

$$\tau(\pi) = \bigwedge_{\substack{1 \leq i \leq n \\ f \in \mathit{prec}(a_i)}} \neg \mathtt{Ab}\left(\langle F_{a_i}|_f^p \rangle \right) \rightarrow f_{i-1}$$

The constraint over the goal description is also trivial:

$$\tau(g) = \bigwedge_{f \in g} f_n$$

Hence, the observation is the following:

$$\mathtt{Obs} = \tau(\pi) \wedge \tau(g)$$

Having introduced the transformation from a domain repair problem into a diagnosis problem, we formally prove the correctness of the procedure.

Theorem 1. *Given a domain repair problem $\Pi = (\mathcal{P}, \pi)$ and the respective diagnosis problem Π^*, a set of repairs $F_\mathcal{P}^*$ is a diagnosis to Π^* iff π is a solution to $\mathcal{P}^*$ in which $\mathcal{P} \Rightarrow_{F_\mathcal{P}^*} \mathcal{P}^*$.*

Proof. ($\Longleftarrow$): Let $F_\mathcal{P}^*$ be a set of repairs such that $\mathcal{P} \Rightarrow_{F_\mathcal{P}^*} \mathcal{P}^*$ and π is a solution to $\mathcal{P}'$. We assume that, without loss of generality, $\pi = \langle a_1 \cdots a_n \rangle$ ($n \in \mathbb{N}$), and $\overline{s} = \langle s_0 \cdots s_n \rangle$ is the state sequence obtained by applying π in the initial state of $\mathcal{P}^*$ (i.e., π is applied in the *updated* domain). One can verify that the formula $\mathtt{SD} \wedge \mathtt{Obs} \wedge \varphi(F_\mathcal{P}^*)$ is evaluated to *true* under the truth assignment where for each $f \in \mathcal{F}$, f_i is assigned *true* if $f \in a_i$ for each $0 \leq i \leq n$.

($\Longrightarrow$): For the other direction, the key observation here is again that for each $0 \leq i \leq n$, if f_i is assigned *true*, then $f \in s_i$. The basis for this argument is that the formula $\tau(s_i)$ simulates how the state s_i is computed in terms of a_i's effects and the repairs applied to a_i. Further, the truthhood of $\mathtt{Obs}$ asserts that the plan π is a solution to $\mathcal{P}^*$. □

We mentioned earlier that in order to use the diagnosis algorithm given in Alg. 1, the set of all diagnoses for a diagnosis problem must be monotonic. Here, we prove this property holds for the domain repair problem.

Theorem 2. *Let $\Pi = (\mathcal{P}, \pi)$ be a domain repair problem, $F'_\mathcal{P} \subseteq F_\mathcal{P}$ a set of repairs, if π is a solution to $\mathcal{P}'$ with $\mathcal{P} \rightarrow_{F'_\mathcal{P}} \mathcal{P}'$, then for any set $F^*_\mathcal{P}$ of repairs with $F'_\mathcal{P} \subseteq F^*_\mathcal{P}$, π is a solution to $\mathcal{P}^*$ with $\mathcal{P} \rightarrow_{F^*_\mathcal{P}} \mathcal{P}^*$.*

Proof. Let $\overline{s}' = \langle s'_0 \cdots s'_n \rangle$ and $\overline{s}^* = \langle s^*_0 \cdots s^*_n \rangle$ be the state sequences which are obtained by executing π in $\mathcal{P}'$ and $\mathcal{P}^*$, respectively. Since $F'_\mathcal{P} \subseteq F^*_\mathcal{P}$, we have $s'_i \subseteq s^*_i$ for each $0 \leq i \leq n$. Thus, π is also a solution to $\mathcal{P}^*$. □

4 Oracle for Domain Repair Problem

As mentioned earlier, for using the diagnosis algorithm, we have to implement the oracle targeted specifically at the domain repair problem which can 1) determine whether a diagnosis candidate can serve as a diagnosis and 2) compute a

conflict given an invalid diagnosis candidate. We introduce here how this oracle works.

We start with the procedure for deciding whether a diagnosis candidate is a diagnosis. This is trivial because, given a diagnosis candidate, which is a subset $F'_{\mathcal{P}}$ of $F_{\mathcal{P}}$ for some planning problem $\mathcal{P} = (\mathcal{F}, \mathcal{A}, \delta, s_I, g)$, we only need to accomplish the following two steps:

1) apply the diagnosis candidate (i.e., the set of repairs) to the domain, and
2) check whether the solution criteria are satisfied in the updated domain by the given plan.

One can easily verify that the time complexity for the entire procedure is $\mathcal{O}(|F'_{\mathcal{P}}| + n|\mathcal{F}|)$ in which n is the length of the input plan, and the term $n|\mathcal{F}|$ is the time for checking whether the solution criteria are satisfied because the precondition of an action can have up to $|\mathcal{F}|$ many propositions. In particular, we have $|F'_{\mathcal{P}}| \leq |F_{\mathcal{P}}|$, and the right-hand side is bounded by the polynomial number:

$$|F_{\mathcal{P}}| \leq 3 \times |\mathcal{F}| \times |\mathcal{A}|$$

The key observation here is that for each action $a \in \mathcal{A}$, the total number of repairs for its precondition, positive effects, or negative effects cannot exceed $|\mathcal{F}|$. Consequently, the procedure for determining whether a diagnosis candidate is a diagnosis has polynomial time complexity.

Next we introduce the procedure for computing a conflict provided a diagnosis candidate which is not a diagnosis. A generic template of such a procedure is given by [8], according to which, given a diagnosis candidate $F'_{\mathcal{P}}$ in our context, a conflict is a subset $F^{\dagger}_{\mathcal{P}}$ of $F_{\mathcal{P}}$ such that $F^{\dagger}_{\mathcal{P}} \cap F'_{\mathcal{P}} = \emptyset$, and the given plan is *not* a solution to the updated planning problem after applying the set of repairs $F_{\mathcal{P}} \setminus F^{\dagger}_{\mathcal{P}}$. For the reason why this result holds, we refer to the work by [8].

Given a diagnosis candidate $F'_{\mathcal{P}}$, a set $F^{\dagger}_{\mathcal{P}}$ satisfying the above criteria can be found as follows. Let $\mathcal{P}'$ be the planning problem with $\mathcal{P} \Rightarrow_{F'_{\mathcal{P}}} \mathcal{P}'$. According to our hypothesis, the input plan $\pi = \langle a_1 \cdots a_n \rangle$ is not a solution to $\mathcal{P}'$ (because $F'_{\mathcal{P}}$ is not a diagnosis). Therefore, there must exist an action a_j $(1 \leq j \leq n)$ which has a proposition q in its precondition that is not satisfied. We randomly pick such a_j and q and find the *largest* index i with $1 \leq i < j \leq n$ such that $q \in \mathit{eff}^-(a_i)$, and for each k with $i < k < j$, $q \notin \mathit{eff}^-(a_k)$. Notably, for each such a_k, it is impossible to have $q \in \mathit{eff}^+(a_k)$ because otherwise, q will be satisfied. Given i and j, the set $F^{\dagger}_{\mathcal{P}}$ is thus

$$F^{\dagger}_{\mathcal{P}} = \{\langle F_{a_i}|^-_q \rangle, \langle F_{a_j}|^p_q \rangle\} \cup \{\langle F_{a_k}|^+_q \rangle \mid i \leq k < j\}$$

Clearly, $F^{\dagger}_{\mathcal{P}} \cap F'_{\mathcal{P}} = \emptyset$, because if it is not the case, then q will be satisfied. Further, the plan π will *not* be a solution after applying the repair set $F_{\mathcal{P}} \setminus F^{\dagger}_{\mathcal{P}}$. The key observation for this is that the proposition q demanded by a_j will still be deleted by a_i because the repair $\langle F_{a_i}|^-_q \rangle$ is not allowed, and q will not be added by any a_k with $i \leq k < j$ because $\langle F_{a_k}|^+_q \rangle$ is also forbidden. Moreover, since $\langle F_{a_j}|^p_q \rangle$ is also not allowed, q cannot be removed from $\mathit{prec}(a_j)$.

The procedure for computing a conflict clearly has polynomial time complexity, more precisely, $\mathcal{O}(n)$ with n being the length of the input plan because, at the worst case, we pick the last action in the given plan whose precondition is not satisfied, and the action with the largest index deleting the respective proposition is the first action in the plan.

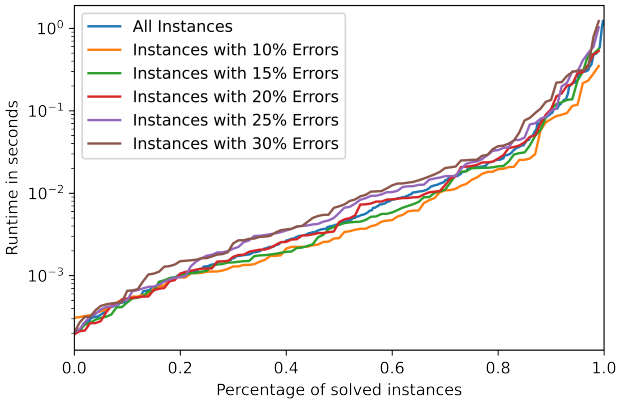

Figure 2: Runtimes against percentages of solved instances.

5 Experimental Results

Thus far, there exist no benchmark sets of flawed planning domains. Therefore, for the empirical evaluation, we have to create our own benchmark set. For this, we choose 100 planning problems from 10 planning domains in the *fast downward benchmark collection*[2] each of which contains 10 planning problems. For each planning problem, we first invoke the *fast downward* planning system [10] to find a solution plan to it, and afterward, for each such pair of planning problem and solution plan, we create 5 domain repair problem instances by randomly introducing 10%, 15%, 20%, 25%, and 30% errors to the actions in the plan, that is, we randomly select 10%, 15%, 20%, 25%, and 30% actions from the plan, and for each selected action, we make one of the following three changes: 1) adding a proposition to the action's precondition, 2) adding a proposition to the action's negative effects, or 3) removing a proposition from the action's positive effects.

We ran the experiments on an Intel i7-10700 CPU and recorded the cardinality of the solution (i.e., a minimal cardinality set of repairs) found for each domain repair problem instance as well as the respective time for finding the solution. Further, as said in the introduction, the performance of our method relies on a minimal hitting set solver. In this paper, find a minimal hitting set is done by invoking the respective procedure implemented by [11] which encodes a minimal hitting set problem as a weighted SAT problem and solves it by the state-of-the-art MaxSAT solver [12].

The metric we are concerned with for evaluating the performance of our approach is the runtime required for solving a domain repair problem instance. Fig. 2 depicts the percentages of the solved instances against the runtimes which are created by respectively introducing 10%, 15%, 20%, 25%, and 30% errors, i.e., the percentages of instances (the x-axis) that can be solved with a specific time (the y-axis). As we can see from the figure, *all* instances can be solved in *one second*. Consequently, we believe that our diagnosis-based domain repair approach is efficient.

Further, Tab. 1 lists *some* of our experiment results in which we present the minimal cardinalities of the diagnoses found for the respective domain repair problem instances. Each row is a planning problem. The first column indicates the name of the domain of each planning problem. The last column is the length of the plan which is supposed

[2]https://github.com/aibasel/downward-benchmarks

Domain	Minimal Cardinality Diagnosis for 10% Error Rate		15% Error Rate		20% Error Rate		25% Error Rate		30% Error Rate		Plan Length
PIPESWORLD	4	(0.6667)	6	(0.6667)	7	(0.5833)	8	(0.5333)	10	(0.5556)	62
	4	(0.3333)	12	(0.6667)	13	(0.5200)	20	(0.6452)	24	(0.6486)	125
	6	(0.8571)	5	(0.5000)	9	(0.6429)	11	(0.6111)	15	(0.7143)	72
	8	(0.7273)	9	(0.5625)	16	(0.7273)	22	(0.8148)	21	(0.6364)	110
	4	(0.5714)	5	(0.4545)	9	(0.6000)	9	(0.5000)	15	(0.6818)	75
VISITALL	3	(0.6000)	3	(0.4286)	5	(0.5000)	3	(0.2308)	6	(0.4000)	52
	10	(0.3704)	19	(0.4750)	17	(0.3148)	21	(0.3088)	38	(0.4691)	272
	5	(0.2273)	13	(0.3824)	21	(0.4667)	20	(0.3509)	27	(0.3971)	228
	20	(0.4545)	30	(0.4545)	28	(0.3182)	49	(0.4414)	54	(0.4060)	444
	5	(0.3125)	13	(0.5417)	19	(0.5938)	20	(0.4878)	24	(0.4898)	164
TPP	14	(0.4667)	25	(0.5556)	39	(0.6500)	32	(0.4267)	42	(0.4667)	302
	7	(0.4375)	12	(0.5000)	20	(0.6250)	16	(0.4000)	23	(0.4792)	163
	9	(0.4737)	15	(0.5172)	15	(0.3846)	24	(0.5000)	34	(0.5862)	195
	4	(0.4000)	9	(0.6000)	7	(0.3333)	15	(0.5769)	15	(0.4839)	105
	18	(0.6207)	23	(0.5227)	30	(0.5085)	44	(0.5946)	42	(0.4719)	299

Table 1: A partial summary of the experiment results listing the minimal cardinality diagnoses found for the domain repair problems instances created by introducing 10%, 15%, 20%, 25%, and 30% errors, respectively. Each fraction in a parenthesis indicates the ratio between the respective minimal diagnosis cardinality and the number of random modifications introduced.

to be a solution to the respective planning problem. The five columns in the middle present the minimal cardinalities of the diagnoses found for the *domain repair problem instances* created by randomly introducing 10%, 15%, 20%, 25%, and 30% errors to the domain, respectively. Specifically, each fraction in a parenthesis indicates the ratio between the respective minimal diagnosis cardinality and the number of random modifications introduced for creating the problem instance. For instance, for the first row, the planning problem is from the domain PIPESWORLD, and the length of the plan that is supposed to be a solution to this planning problem is 62. The minimal cardinality of the diagnosis for the domain repair problem instance created by introducing 10% errors is 4. Further, introducing 10% errors means that we applied $62 \times 10\% \approx 6$ modifications. Thus, the ratio between the minimal cardinality of the diagnosis and the number of modifications is 0.6667.

The reason that we only show some results in Tab. 1 is that since our approach guarantees to find a minimal cardinality diagnosis, the statistical data reported in the table cannot be used as a metric to evaluate the performance of our approach. Thus, the main purpose of the table is to simply provide some intuitions about the minimal number of repairs required to fix a planning domain.

6 Discussion and Related Works

Diagnosis techniques have been proposed to find discrepencies between models and real world. Most notably, Belard et al. [13] introduced the idea of *meta-diagnosis*, where the goal is to diagnose the model. Earlier, Crow and Rushby [14] and Stumptner and Wotawa [15] proposed to use diagnosis techniques to reconfigure a system, which is an alternative view of the domain repair problem. Further, our work is also in line with the idea of redesigning an almost correct system via diagnosis [16].

Our work also bears some similarities with software debugging in which a faulty piece of code (akin to our planning domain) is provided alongside some test cases (akin to our valid plans). The task is then to identify the faulty lines in the code and, ideally, possible corrections. Software debugging is a very difficult task—undecidable in general—and successful solutions revolve around using statistical analysis [17]. The repairs available in our case are much narrower, which explains why we are able to compute a complete repair.

To our best knowledge, this paper is the first that turns a non-solution plan into a solution. It is however certainly *not* the first one considering modifying a planning domain (or a planning problem itself). Many of those works modify planning domains for the purpose of building Explainable AI Planning (XAIP) systems [18] among which one notable framework is called *model reconciliation* [19; 20] sharing a lot of similarities with ours.

The configuration for model reconciliation is as follows. It has three inputs: 1) a plan π that is observed by a user, 2) a human mental model $\mathcal{P}_h^R$, i.e., a planning problem which, *according to a user's assumption*, is deployed in some robot, and 3) a robot model $\mathcal{P}^R$ i.e., a planning problem that is *actually* deployed in the robot. Specifically, the plan π is not a solution (or not an optimal solution) to $\mathcal{P}_h^R$, but it is a(n) (optimal) solution to $\mathcal{P}^R$. The objective is to modify the domain of $\mathcal{P}_h^R$ such that π will be an *optimal* solution to the updated problem and the changes applied should be *compatible* to $\mathcal{P}^R$.

Compared with our domain repair framework which only demands one input model (i.e., an input planning problem), the model reconciliation framework demands two (i.e., two input planning problems). Further, the model reconciliation framework demands that the given plan must be an optimal one in the updated model, which raise the complexity of the respective problem to Σ_p^2 [20]. In contrast, our framework does not require that the given plan should be optimal in the updated model, and hence the domain repair problem is also computationally cheap (i.e., $\mathbb{NP}$-complete [7]).

Apart from frameworks that modify a planning domain to turn a non-solution plan into a(n) (optimal) solution, several works have attempted to turn an *unsolvable* planning problem into a solvable one by modifying the planning do-

main (or the planning problem). For instance, the work by [21] modify the *initial state* of a planning problem for this purpose (note that changing the initial state of a planning problem is *not* equivalent changing the domain). Recently, an extension of this work done by [22] modifies the domain of an unsolvable planning problem in order to make it solvable. Compared with our configuration, the problem studied in [22] only has one input which is an unsolvable planning problem, and the authors there only allowed adding propositions to actions' positive effects.

7 Extensions

In this section, we introduce some possible extensions of our work. The first one is to repair *lifted* planning domains instead of grounded domains. Compared with grounded domains which are formulated in propositional logic, lifted domains are formulated in terms of first-order logic. This extension is of great importance for our future work because a great many planning problems (domains) are lifted in practice. Such an extended domain repair problem can also be solved via formulating it as a diagnosis problem in which we could define each component as a lifted one.

Another extension is to introduce prior probability distributions over whether a component is flawed. More concretely, given a diagnosis problem which is a formulation of a domain repair problem and in which $F_{\mathcal{P}}$ is the set of components, we define a probability distribution P over $F_{\mathcal{P}}$ such that $P(x)$ for each $x \in F_{\mathcal{P}}$ indicates the probability of the component x being flawed. In this configuration, our objective is thus to find a diagnosis $F^*_{\mathcal{P}}$ such that $\prod_{x \in F^*_{\mathcal{P}}} P(x)$ is maximal, i.e., the component set $F^*_{\mathcal{P}}$ has the maximal probability of being flawed. One remark is that this setting can be applied to both the grounded and lifted versions of the domain repair problem and solve the respective diagnosis problem via diagnosis algorithms.

8 Conclusion

In this paper, we proposed a method for solving the domain repair problem in which we are given a plan and a planning problem in which the plan is not a solution to the planning problem, and the goal is to repair the domain of the planning problem so that the plan will be a solution. The method formulates a domain repair problem as a diagnosis problem by viewing each atomic repair as a component and the planning problem together with the plan as a system.

References

[1] Raymond Reiter. A theory of diagnosis from first principles. *AIJ*, 32(1):57–95, 1987.

[2] Johan de Kleer and Brian C. Williams. Diagnosing multiple faults. *AIJ*, 32(1):97–130, 1987.

[3] Malik Ghallab, Dana S. Nau, and Paolo Traverso. *Automated Planning and Acting*. Cambridge University Press, 2016.

[4] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 2020.

[5] Thomas Leo McCluskey, Tiago Stegun Vaquero, and Mauro Vallati. Engineering knowledge for automated planning: Towards a notion of quality. In *K-CAP 2017*, pages 14:1–14:8. ACM, 2017.

[6] Alan Lindsay, Santiago Franco, Rubiya Reba, and Thomas Leo McCluskey. Refining process descriptions from execution data in hybrid planning domain models. In *ICAPS 2020*, pages 469–477. AAAI, 2020.

[7] Songtuan Lin and Pascal Bercher. Change the world - how hard can that be? on the computational complexity of fixing planning models. In *IJCAI 2021*, pages 4152–4159. IJCAI, 2021.

[8] John Slaney. Set-theoretic duality: A fundamental feature of combinatorial optimisation. In *ECAI 2014*, pages 843–848. IOS, 2014.

[9] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *AIJ*, 2(3–4):189–208, 1971.

[10] Malte Helmert. The fast downward planning system. *JAIR*, 26:191–246, 2006.

[11] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT 2018*, pages 428–437. Springer, 2018.

[12] António Morgado, Alexey Ignatiev, and João Marques-Silva. MSCG: robust core-guided maxsat solving. *JSAT*, 9(1):129–134, 2014.

[13] Nuno Belard, Yannick Pencolé, and Michel Combacau. A theory of meta-diagnosis: reasoning about diagnostic systems. In *IJCAI 2011*, pages 731–737, 2011.

[14] Judith Crow and John M. Rushby. Model-based reconfiguration: Toward an integration with diagnosis. In *AAAI 1991*, pages 836–841. AAAI, 1991.

[15] Markus Stumptner and Franz Wotawa. Reconfiguration using model-based diagnosis. In *DX 1999*, pages 266–271, 1999.

[16] Johan de Kleer, Alexander Feldman, and Ion Matei. The duality of design and diagnosis. In *DX 2018*, pages 259 – 265, 2018.

[17] Rui Abreu, Peter Zoeteweij, Rob Golsteijn, and Arjan J.C. van Gemund. A practical evaluation of spectrum-based fault localization. *JSS*, 82(11):1780–1792, 2009.

[18] Tathagata Chakraborti, Sarath Sreedharan, and Subbarao Kambhampati. The emerging landscape of explainable automated planning & decision making. In *IJCAI 2020*, pages 4803–4811. IJCAI, 2020.

[19] Sarath Sreedharan, Tathagata Chakraborti, and Subbarao Kambhampati. Foundations of explanations as model reconciliation. *AIJ*, 301:103558, 2021.

[20] Sarath Sreedharan, Pascal Bercher, and Subbarao Kambhampati. On the computational complexity of model reconciliations. In *IJCAI-ECAI 2022*. IJCAI, 2022.

[21] Moritz Göbelbecker, Thomas Keller, Patrick Eyerich, Michael Brenner, and Bernhard Nebel. Coming up with good excuses: What to do when no plan can be found. In *ICAPS 2010*, pages 81–88. AAAI, 2010.

[22] Alba Gragera, Ángel García-Olaya, and Fernando Fernández. Repair suggestions for planning domains with missing actions effects. In *XAIP 2022*, 2022.