A Constraint Formulation for Domain Repair with Ground or Lifted Test Plans

Nika Beriachvili[1], **Arthur Bit-Monnot**[1]

[1]LAAS-CNRS, Université de Toulouse, INSA, CNRS, Toulouse, France
nika.beriachvili@laas.fr, abitmonnot@laas.fr

Abstract

Engineering automated planning domains is a meticulous and time-consuming process. In recent years, much attention has been given to techniques assisting domain designers in both writing new domains and ensuring their correctness.

This paper addresses the problem of repairing a flawed planning domain to comply with a given example problem and plan. The goal is to suggest a minimal set of repairs, which can either be the removal of a condition or effect, or the addition of a new effect in some of the domain's action schemas.

The cases of fully specified (ground) and partially specified (lifted) plans, whose actions' parameters can be variables, are handled separately in previous approaches. We introduce an alternative method that covers both ground and lifted cases in a unified way by repurposing an existing encoding of planning as constraint satisfaction. Despite its simplicity, we show the approach to be competitive with the ground case baseline solver and substantially outperforming the lifted one.

Introduction

Automated planning is a branch of artificial intelligence studying action plans that achieve specific goals within a set of given constraints. The formal models used by automated planning systems are often complex to design and verify, as their engineering process is prone to errors and usually requires input from both planning and domain experts. As such, providing modeling support and debugging assistance to users is essential for making planning tools and techniques more accessible.

Model repair, as defined by Bercher, Sreedharan, and Vallati (2025), refers to the process of refining a model to meet a given set of *constraints*, in contrast to *model acquisition*, which focuses on embedding it with existing domain *knowledge*. In this paper, we focus on *domain repair*: a restricted form of model repair dealing with the part of a planning model typically known as its domain. It specifies the actions available to solve a planning problem, by describing the conditions under which they can be executed (preconditions) and the changes to the world state that result from their execution (effects). Domain repair is not concerned with a planning problem's initial state (which specifies the available objects and the facts known to be true or false before plan execution) nor its goal state.

Specifically, the domain repair setting we consider features a *whitelist* (or *positive*) test plan (or witness plan) that the user knows to be a correct solution even though it is rejected by their current, flawed, planning domain (Lin, Grastien, and Bercher 2023). The objective then is to suggest *corrections* or *repairs* to the domain – i.e. changes (additions or removals) of action preconditions and effects – so that it accepts this plan as solution. This plan can be both *ground* (meaning all its actions' parameters are fixed) or *lifted*. In the latter case, the plan is partially specified with placeholder variables in the actions' parameters, and the repairs must ensure at least one grounding of the plan is accepted (Bavandpour et al. 2025). The motivation is to allow users to focus only on the critical parts of their test plan.

Clearly, not all repairs are equally desirable and one would certainly not agree to a set of repairs that would blindly remove all preconditions of the actions, meaning that the selected repairs should be optimised with respect to some preference metric. Like previous work, we consider as a metric the number of selected repairs, which would minimize the number of changes made to the original domain.

It is also worth noting that the task of domain repair with a whitelist plan is closely related to that of explaining this plan's invalidity (in the flawed domain), but still different from it. Indeed, it might be sufficient to address the latter by pointing to one of the falsified preconditions in the plan. This suggests the domain could be repaired by changing that precondition or by adding an effect that could support it, but does not directly compute nor select such repairs.

In this paper, we address the domain repair problem with a ground or lifted whitelist test plan by leveraging the constraint-based planning encoding of Bit-Monnot (2018) to compute cardinality-minimal repair sets, in the form of minimal correction subsets (MCS) of preconditions or effects to remove, or effects to remove/add. We experimentally compare the performance of our approach with the most recent baselines – for the ground case (Lin, Grastien, and Bercher 2023) and for the lifted one (Bavandpour et al. 2025) – and show our approach to be competitive with the former, while substantially outperforming the latter. Other benefits include the unification of the ground and lifted settings, as well as its extensibility potential.

Related Work

The domain repair setting considered in this paper is first addressed by Lin, Grastien, and Bercher (2023) for one or multiple ground whitelist plans, with repair sets composed of atomic repairs (precondition removals, effect removals/additions). Their formalism and algorithms are inspired by model-based diagnosis and (minimal) hitting set duality of conflicts and diagnoses. A minimal repair set is framed as a minimal diagnosis, obtained after repeatedly taking candidate minimal hitting sets for the currently known conflicts, and then computing new conflicts nonintersecting the previously considered minimal hitting set, until the repairs in a new candidate hitting set result in the whitelist plan being accepted as solution (i.e. correspond to a minimal repair set).

Domain repair for totally ordered HTN domains with one ground whitelist plan was addressed by Lin, Höller, and Bercher (2024). The investigated domain flaws are not restrictive preconditions and restrictive/missing effects – they can be dealt with as in non-hierarchical domains – but missing primitive actions in tasks' (i.e. compound actions') decomposition methods.

Lin et al. (2025) extend the approach of Lin, Grastien, and Bercher (2023) to take into account one or multiple ground *blacklist* (or *negative*) plans in addition to ground whitelist plans. This requires an additional type of atomic repair (precondition addition). A blacklist plan is a valid solution for the flawed domain but is known by the user to be incorrect (starting from a specific action) and must be disallowed in the repaired domain.

More recently, the case of one *lifted* whitelist plan was addressed by Bavandpour et al. (2025), which is the closest work to ours. Their approach and implementation exploits the previous algorithm for the ground case (Lin, Grastien, and Bercher 2023). The domain must be repaired such that there exists a grounding of the whitelist plan that is a solution under the repaired domain. They propose a search algorithm where nodes store a ground prefix of the whitelist plan and a (ground) repair set such that the repaired domain accepts that prefix. The algorithm terminates when a node's prefix grows to a full plan meeting the goal state. The final result is obtained by lifting the ground repair set in that node.

However, it should be mentioned this is not the only possible aim. Indeed, Gigante et al. (2025) use an LTL_f specification rather than a whitelist plan – and allow repairs to the initial state in addition to the domain – where one of the considered cases is when *all* valid solutions of the planning problem are required to comply with the LTL_f specification.

The setting considered by Welt et al. (2025) features an unsolvable problem (and therefore no test plans). They compute repairs to both the domain and initial state, whose application renders the problem satisfiable. A similar setting is addressed by Gragera et al. (2023), with the addition of missing effects being the focus of repairs. To compute these, they compile the domain repair problem into a new planning problem with an extended type hierarchy and action definition. Their method was subsequently extended by Gragera and Muise (2024) to account for multiple unsolvable problems.

Formal Problem Statement

In this section, we formalize the domain repair problem after some preliminary definitions.

Lifted Planning Problem

We define a lifted classical planning problem as a tuple $\Pi = (\mathcal{P}, \mathcal{A}, \mathcal{O}, s^I, s^G)$ where $(\mathcal{P}, \mathcal{A})$ is its domain. $\mathcal{P}$ is a set of predicates $P = \mathit{pred}(\chi_1|\tau_1, \ldots, \chi_n|\tau_n)$ where *pred* is a predicate name and χ_i are parameters of type τ_i. $\mathcal{O}$ is a set of objects, each associated to a type. We denote as $\mathcal{O}_\tau \subseteq \mathcal{O}$ the set of objects of type τ.

A fact $\mathbf{f} = \mathit{pred}(\alpha_1, \ldots, \alpha_n) = \sigma_\mathbf{f}(P)$ is obtained by applying a substitution $\sigma_\mathbf{f}$ to P, which replaces its parameters with arguments α_i of type τ_i. Each argument α_i is either an object in $\mathcal{O}_{\tau_i}$ or a *variable* whose allowed values must be within $\mathcal{O}_{\tau_i}$, A fact $\mathbf{f}$ is *ground* when all α_i are constants (in which case it may be denoted f) and *lifted* otherwise.

We denote as Ω the set of all ground facts. A *state* $s \subseteq \Omega$ is a set of facts that are true at a given point in time.

$\mathcal{A}$ is the set of *action schemas* $A = \mathit{act}(\chi_1|\tau_1, \ldots, \chi_n|\tau_n)$ available in the domain, where *act* is an action name and χ_i are parameters of type τ_i. An action schema A is defined with its positive (negative) preconditions $\mathit{cond}^+(A)$ ($\mathit{cond}^-(A)$) as well as its positive (negative) effects $\mathit{eff}^+(A)$ ($\mathit{eff}^-(A)$). These are sets of lifted facts, in which the arguments of predicates are parameters of the action schema.

An *action* $\mathbf{a} = \mathit{act}(\alpha_1, \ldots, \alpha_n) = \sigma_\mathbf{a}(A)$ is obtained by applying a substitution $\sigma_\mathbf{a}$ to A, which replaces its parameters χ_i with arguments α_i of type τ_i. This substitution is applied to obtain the conditions and effects of $\mathbf{a}$, e.g., $\mathit{cond}^\pm(\mathbf{a}) = \{\ \sigma_\mathbf{a}(\mathbf{f}) \mid \mathbf{f} \in \mathit{cond}^\pm(A)\ \}$.

An action $\mathbf{a}$ is said to be *ground* when all its arguments are objects – in which case it may be denoted a – which implies that all facts appearing in its conditions and effects are ground as well. Otherwise, if at least one of its arguments is a variable, an action is said to be *lifted*. A *ground* action a is applicable in a state s if $\mathit{cond}^+(a) \subseteq s$ and $\mathit{cond}^-(a) \cap s = \emptyset$. Its application in a state s yields a new state $a(s) = (s \setminus \mathit{eff}^-(a)) \cup \mathit{eff}^+(a)$.

Finally, the problem definition contains an initial state $s^I \subseteq \Omega$ and a goal state $s^G \subseteq \Omega$ that respectively denote ground facts that are initially true and those that are required to hold at the end of the plan.

A *plan* for $\Pi = (\mathcal{P}, \mathcal{A}, \mathcal{O}, s^I, s^G)$ is a sequence of ground actions $\pi = [a_1, \ldots, a_n]$. A plan is associated with its *trace*: the sequence of states $[s_1, s_2, \ldots, s_{n+1}]$ where $s_1 = s^I$ and $s_{i+1} = a_i(s_i)$ $(\forall i \in [1, n])$. The plan π is a solution to Π if each action a_i is applicable in s_i and the goals hold in the final state ($s^G \subseteq s_{n+1}$).

Domain Repair

The *domain repair problem* assumes one is given a flawed planning problem $\Pi = (\mathcal{P}, \mathcal{A}, \mathcal{O}, s^I, s^G)$ and a *test plan* that is known to be a solution to the true planning problem but is not a solution to Π (Lin, Grastien, and Bercher 2023). The objective is thus to determine a new set of actions schemas $\mathcal{A}'$ such that π is a solution to $\Pi' = (\mathcal{P}, \mathcal{A}', \mathcal{O}, s^I, s^G)$. Obviously, there may be many such actions schemas, some of

which may drastically depart from the original domain definition. Lin, Grastien, and Bercher (2023) thus further require that the new action schemas $\mathcal{A}'$ be obtained by a minimal set of transformations (repairs) of the original one $\mathcal{A}$.

Atomic Repairs Given a domain $(\mathcal{P}, \mathcal{A})$, a repair is of the form $[\![A, \mathbf{f}, c, op]\!]$ where $A \in \mathcal{A}$ is the action schema it applies to, $\mathbf{f}$ is a fluent, $c \in \{ \mathit{eff}^+, \mathit{eff}^-, \mathit{cond}^+, \mathit{cond}^- \}$ denotes the component the repair operates on and $op \in \{ \mathit{add}, \mathit{rm} \}$ denotes whether the fluent is added or removed.

Possible Repairs The set of $\mathcal{R}_\Pi$ of possible repairs associated to a planning domain $\Pi = (\mathcal{P}, \mathcal{A})$ is the set of atomic repair composed of:

- a repair $[\![A, \mathbf{f}, c, \mathit{rm}]\!]$ for each action $A \in \mathcal{A}$, $c \in \{ \mathit{eff}^+, \mathit{eff}^-, \mathit{cond}^+, \mathit{cond}^- \}$, $\mathbf{f} \in c(A)$, denoting the removal of any effect or condition from any action schema,
- repairs $[\![A, \mathbf{f}, \mathit{eff}^+, \mathit{add}]\!]$ and $[\![A, \mathbf{f}, \mathit{eff}^-, \mathit{add}]\!]$ where for each $A \in \mathcal{A}$, and lifted fact $\mathbf{f}$ whose arguments are appropriately typed parameters of A

Note that, as in the original definition of Lin, Grastien, and Bercher (2023), repairs do not consider the possibility of adding conditions to actions, as those can never contribute to turning the test plan into a solution.

Repair Set A repair set $\Delta \subseteq \mathcal{R}_\Pi$ is a subset of the possible atomic repairs. Applying it to a set of actions schema $\mathcal{A}$, yields a new set $\mathcal{A}' = \Delta(\mathcal{A})$ where each schema $A \in \mathcal{A}$ is mapped to a new schema A' with updated conditions and effects:

$$\mathit{cond}^\pm(A') = \mathit{cond}^\pm(A) \setminus \{ \mathbf{f} \mid [\![A, \mathbf{f}, \mathit{cond}^\pm, \mathit{rm}]\!] \in \Delta \}$$
$$\mathit{eff}^\pm(A') = \mathit{eff}^\pm(A) \setminus \{ \mathbf{f} \mid [\![A, \mathbf{f}, \mathit{eff}^\pm, \mathit{rm}]\!] \in \Delta \}$$
$$\cup \{ \mathbf{f} \mid [\![A, \mathbf{f}, \mathit{eff}^\pm, \mathit{add}]\!] \in \Delta \}$$

Applying a repair set Δ to a planning problem Π yields a new planning problem $\Pi' = \Delta(\Pi)$ in which the actions schemas are substituted by their repaired versions ($\mathcal{A}_{\Pi'} = \Delta(\mathcal{A}_\Pi)$).

Definition 1 (Ground Domain Repair Problem (Bavandpour et al. 2025))**.** *A ground domain repair problem is a pair* (Π, π) *where* Π *is a planning problem and* π *is ground plan. Its solution is a repair set* $\Delta \in \mathcal{R}_\Pi$ *such that* π *is a solution to* $\Delta(\Pi)$.

Definition 2 (Lifted Domain Repair Problem (Bavandpour et al. 2025))**.** *A lifted domain repair problem is a pair* (Π, γ) *where* Π *is a planning problem and* γ *is a lifted plan, i.e., a sequence of lifted actions* $[\mathbf{a_1}, \ldots, \mathbf{a_n}]$. *Its solution is a repair set* $\Delta \in \mathcal{R}_\Pi$ *such that there exists a ground instantiation* π *of* γ *that is a solution to* $\Delta(\Pi)$.

As in previous work, we are interested in finding the *smallest* repair set that is a solution to the domain repair problem.

Approach

In this section, we present our approach to solve the ground and lifted domain repair problems. We start by introducing an existing approach for encoding planning problems as constraint satisfaction problems, as well as an assumption under which its semantics aligns with the one of classical planning presented earlier.

We then show how to repurpose this approach, first to decide plan validity and then to solve the ground and lifted domain repair problems.

Well-formedness Assumption

Let us start with an initial assumption regarding the structure of actions. The following defines as *ill-structured* a ground action a whose instantiation from its action schema A yields multiple delete or add effects of the same ground fact.

Definition 3 (Ill-defined action)**.** *Given an action schema* A*, a ground action* $a = \sigma_a(A)$ *is ill-defined if either: (i)* $\mathit{eff}^+(A) \cap \mathit{eff}^-(A) \neq \emptyset$*, or (ii) there exist two lifted fluents* $\mathbf{f_1}$ *and* $\mathbf{f_2}$ *in* $\mathit{eff}^+(A) \cup \mathit{eff}^-(A)$*, such that* $\mathbf{f_1} \neq \mathbf{f_2}$ *and* $\sigma_a(\mathbf{f_1}) = \sigma_a(\mathbf{f_2})$

As an example of an ill-defined action consider an action schema $move(?o, ?d)$ with positive effect $at(?d)$ and negative effect $at(?o)$. The action $move(l_1, l_1)$ would be ill-defined as it both adds and deletes the fact $at(l_1)$. It would similarly be ill-defined if the move action schema had both $at(?o)$ and $at(?d)$ appearing in the positive/negative effects.

For the remainder of this section, we will assume that an ill-defined action is *not applicable*, which is in general a reasonable thing to assume and greatly simplifies the presentation of the approach. In the next section, we will present a pre-processing step that allows relaxing this assumption when the strict set-based semantics are desirable.

Preliminaries: Planning as Timetabling

As a preliminary, we introduce the representation of Bit-Monnot (2018) of bounded planning problems. In a nutshell, from a finite set of possible (lifted) actions, the author propose to represent the associated planning problem as a set of *effect tokens*, capturing all state changes that may occur and a set of *condition tokens* capturing all requirements that may be placed on the state trajectory.

The state trajectory is considered over a sequence of time instants $T = [0, \epsilon, 2\epsilon, 3\epsilon, \ldots, H]$, where 0 denotes the temporal origin, ϵ is a fixed time-discretization step and H is the temporal horizon. Given an instant $t \in T$, we denote as s_t the state at that instant t.

An **effect token** e_i is of the form $\langle p_i : [t_i]\ \mathbf{f_i} \leftarrow v_i \rangle$ where p_i is a boolean *variable*, indicating whether the effect is *active*, $t_i \in T$ is the instant at which the effect applies, $\mathbf{f_i}$ is a fact that may be ground or lifted, v_i is the value asserted by the effect. For classical planning, v_i would be either the constant *true* ($\top$) or *false* ($\bot$), respectively denoting that the fact will be present or absent in the next state.

A ground effect token, $\langle p : [t]\ f \leftarrow \top \rangle$ states that when p is true (the effect is *active*) then f will be true at the next instant ($f \in s_{t+\epsilon}$). Similarly, a ground token $\langle p : [t]\ f \leftarrow \bot \rangle$ states that, if p is true, then f will not hold at the next instant ($f \notin s_{t+\epsilon}$).

A **condition token** c_j is of the form $\langle p_j : [t_j]\ \mathbf{f_j} = v_j \rangle$ where p_j is a boolean variable denoting whether the condition is active, $t_j \in T$ is the instant at which the condition is

required to hold, $\mathbf{f_j}$ denotes the (lifted) fact that the condition refers to, and v_j is the required value associated to the fact.

A ground condition token $\langle p\!:\![t]\ f = \top\rangle$ states that, when p is true (the condition is active), f is required to hold at instant t ($p \Rightarrow f \in s_t$). A ground condition token $\langle p\!:\![t]\ f = \bot\rangle$ states that, when p is true, the fact f should be false at the instant t ($p \Rightarrow f \notin s_t$).

In the original work, focused on temporal plan generation, condition and effect tokens would be generated for each potential action and their *activity* tied to the presence of this action in the plan. The author describes a decision procedure to determine whether such sets of token is satisfiable which would imply plan existence. Here, we present this procedure for the classical planning context considered in this paper.

Given a set E_Ψ of effect tokens and a set C_Ψ of condition token, we refer to $\Psi = (E_\Psi, C_\Psi)$ as a timetable problem (timetable for short). The intuition behind this name is that each effect token would affect a fact at a given point in time, providing a time-indexed view of events for which one should ensure that all active conditions tokens are satisfied.

Satisfiability Consider a timetable $\Psi = (E_\Psi, C_\Psi)$. First note that it contains variables that appear within a token's fluent or as its activity.[1] Let V_Ψ be the set of all such variables. To determine whether a timetable is satisfiable, we are essentially interested in determining whether there is an instantiation of these variables such that the timetable is coherent and all conditions are supported.

To define these notions of coherence and support, Bit-Monnot (2018) introduce for each effect $e_i \in E_\Psi$ an additional *mutex* variable m_i whose initial domain is the set of all possible instants (T). The purpose of this variable is to ensure that the value established by e_i persists *at least* until m_i, effectively enforcing a mutex period $(t_i, m_i]$ on the fact $\mathbf{f_i}$ it modifies. We denote M_Ψ the set of all mutex variables.

We say that an effect $e_i \in E$ *supports* a condition c_j, noted $supports(e_i, c_j)$, iff:

$$p_i \wedge p_j \wedge t_i < t_j \leq m_i \wedge \mathbf{f_i} = \mathbf{f_j} \wedge v_i = v_j \tag{1}$$

which requires that both the condition and the effect be active ($p_i \wedge p_j$), that the effect precedes the condition ($t_i < t_j$), persists at least until the condition ($t_j \leq m_i$) and establishes or deletes the fact required by the condition ($\mathbf{f_i} = \mathbf{f_j} \wedge v_i = v_j$). Here $\mathbf{f_i} = \mathbf{f_j}$ is a shorthand notation to say that they are piecewise equal, i.e. that they share the same predicate name and arguments:

$$\mathbf{f_i} = \mathbf{f_j} \Leftrightarrow pred_i = pred_j \wedge \bigwedge_k \alpha_i^k = \alpha_j^k$$

where α_i^k denotes the k^{th} argument of f_i.

We say that two effects e_i and e_j (with $i \neq j$) are coherent, noted $coherent(e_i, e_j)$, iff they never affect the same fact at the same time:

$$\neg p_i \vee \neg p_j \vee t_i \leq m_j \vee t_j \leq m_j \vee \mathbf{f_i} \neq \mathbf{f_j} \tag{2}$$

which is true when either: one of the effect is inactive ($\neg p_i \vee \neg p_j$), their activity periods do not overlap ($t_i \leq m_j \vee t_j \leq m_j$) or when they do not refer to the same fact ($\mathbf{f_i} \neq \mathbf{f_j}$).

[1] The original work also allowed variables in tokens' timepoints (time instants) and values, which is not considered in this paper.

Definition 4. *A timetable problem $\Psi = (E_\Psi, C_\Psi)$ is satisfiable iff there is an assignment to all its variables such that:*

1. *for each condition $c_i \in C$, there exist an effect $e_j \in E$ that* supports *it,*
2. *any two distinct effects $e_i, e_j \in E^2$, are* coherent*, and*
3. *each active effect $e_i \in E$, has a non-empty activity period ($\neg p_i \vee t_i < m_i$).*

In the following sections, we show how this decision problem can be used to decide *(i)* plan validity, *(ii)* ground domain repair and *(iii)* lifted domain repair.

Plan Validation

As a first step, let us show how one can determine plan validity by encoding it as a timetable problem. This will be helpful in establishing the fundamentals of the approach and constitute an interesting special case of the original method where the set of actions is fixed, with constant parameters and execution time.

We consider a planning problem Π, along with a (ground) plan $\pi = [a_1, \ldots, a_n]$.

We define a set of initial effect tokens E_I, with one positive effect for each fact f true in the initial state and one negative effect for any other fact. These effects are placed at time 0, which by convention denotes the temporal origin.

$$E_I = \{\ \langle \top\!:\![0]\ f \leftarrow \top\rangle \mid f \in s^I\ \} \tag{3}$$

$$\cup\ \{\ \langle \top\!:\![0]\ f \leftarrow \bot\rangle \mid f \in \Omega \setminus s^I\ \} \tag{4}$$

We define a set C_G of goal conditions tokens, that require each goal fact to hold at the temporal horizon $H = n + 1$.

$$C_G = \{\ \langle \top\!:\![n+1]\ f = \top\rangle \mid f \in s^G\ \} \tag{5}$$

For each action a_i in the plan, we define a set of effects E_{a_i} and conditions C_{a_i} tokens. Given $A \in \mathcal{A}$, the action schema from which the action is instantiated ($a_i = \sigma_{a_i}(A)$), the tokens are obtained by substituting the parameters of A by the arguments of a_i in the positive/negative conditions and effects of the schema. The tokens are placed at the time instant i corresponding to the index of the action in the plan.

$$E_{a_i} = \{\ \langle \top\!:\![i]\ \sigma_{a_i}(\mathbf{f}) \leftarrow \top\rangle \mid \mathbf{f} \in \mathit{eff}^+(A)\ \} \tag{6}$$

$$\cup\ \{\ \langle \top\!:\![i]\ \sigma_{a_i}(\mathbf{f}) \leftarrow \bot\rangle \mid \mathbf{f} \in \mathit{eff}^-(A)\ \} \tag{7}$$

$$C_{a_i} = \{\ \langle \top\!:\![i]\ \sigma_{a_i}(\mathbf{f}) = \top\rangle \mid \mathbf{f} \in \mathit{cond}^+(A)\ \} \tag{8}$$

$$\cup\ \{\ \langle \top\!:\![i]\ \sigma_{a_i}(\mathbf{f}) = \bot\rangle \mid \mathbf{f} \in \mathit{cond}^-(A)\ \} \tag{9}$$

Let $\Psi = (E, C)$ where $E = E_I \cup \bigcup_i E_{a_i}$ and $C = C_G \cup \bigcup_i C_{a_i}$, the timetable aggregating conditions and effects from the initial state, the goals and the actions.

Proposition 1. *Ψ is satisfiable if and only if π is a solution to Π with no ill-defined actions.*

Proof. We first establish that satisfiability implies well-formedness. Assume the opposite, i.e. that the plan contains an ill-formed action a_i. Then equations eqs. (6) and (7) would generate two always active effects tokens at instant i and an identical fact, which would not fulfill the coherence requirement, making the problem unsatisfiable.

Assume now that Ψ is satisfied for an assignment Γ to the mutex variables. For each condition c_j, the support constraint ensures that there is at least one effect e_i prior to t_j whose mutex time m_i is at or after t_j ($t_j \leq m_i$) and that establishes the desired fact f (or its negation). The coherence constraint ensures that there is no other effect on f in $(t_i, t_j]$. Thus the fact f established at $t_i + 1$ still holds at t_j.

Assuming the plan is valid and well-formed. Then any action or goal condition must have an enabler effect (initial or from an action). The support requirements would be trivially satisfied by assigning each mutex m_i to the maximum of all t_j for which e_i enables the condition c_j (or $t_i + 1$ if there is no such condition). Well-formedness ensures that all effects are coherent, which otherwise could only be invalidated by effects of the same action. □

Ground Repairs

We now turn our attention to the ground repair problem (Π, π) where $\pi = [a_1, \ldots, a_n]$ is a ground test plan. The key idea will be to provide conditions and effect tokens whose activity is tied to their selection or deactivation by the repairs. For each atomic repair $r \in \mathcal{R}_\Pi$, we introduce a boolean variable δ_r that will be true iff the repair appears in the solution.

First, we introduce a notation that ties each potential condition and effect to the atomic repair that may enable or disable it. Given an action schema A, we define:

- $\widehat{cond^{\pm}(A)}$ as the set of $(r, \mathbf{f})$ pairs where $\mathbf{f} \in cond^{\pm}(A)$ is a positive/negative condition of A and r is the atomic repair that would *remove* it.
- $\widehat{eff^{\pm}(A)}$ as the set of $(r, \mathbf{f})$ pairs where $\mathbf{f} \in eff^{\pm}(A)$ is a positive/negative effect of A and r is the atomic repair that would remove it.
- $\widehat{eff^{\pm}_{\text{add}}(A)}$ as the set of $(r, \mathbf{f})$ pairs where r is the atomic repair that *adds* the fluent $\mathbf{f}$ to the positive/negative effects of A.

In the context of a domain repair problem, we define as $E^{rep}_{a_i}$ the effect tokens associated to an action instance $a_i \in \pi$, with their activity conditioned by the application δ_r of the corresponding repair r, either for removal or addition.

$$E^{rep}_{a_i} = \{\, \langle \neg\delta_r : [i]\ \sigma_{a_i}(\mathbf{f}) \leftarrow \top \rangle \mid (r, \mathbf{f}) \in \widehat{eff^{+}(A)} \,\} \quad (10)$$

$$\cup \{\, \langle \neg\delta_r : [i]\ \sigma_{a_i}(\mathbf{f}) \leftarrow \bot \rangle \mid (r, \mathbf{f}) \in \widehat{eff^{-}(A)} \,\} \quad (11)$$

$$\cup \{\, \langle \delta_r : [i]\ \sigma_{a_i}(\mathbf{f}) \leftarrow \top \rangle \mid (r, \mathbf{f}) \in \widehat{eff^{+}_{\text{add}}(A)} \,\} \quad (12)$$

$$\cup \{\, \langle \delta_r : [i]\ \sigma_{a_i}(\mathbf{f}) \leftarrow \bot \rangle \mid (r, \mathbf{f}) \in \widehat{eff^{-}_{\text{add}}(A)} \,\} \quad (13)$$

Similarly, the conditions tokens of the action are redefined as $C^{rep}_{a_i}$ to capture the fact that each condition is active only when the corresponding repair of removing the condition from the action schema is *not* applied.

$$C^{rep}_{a_i} = \{\langle \neg\delta_r : [i]\ \sigma_{a_i}(\mathbf{f}) = \top \rangle \mid (r, \mathbf{f}) \in \widehat{cond^{+}(A)}\} \quad (14)$$

$$\cup \{\langle \neg\delta_r : [i]\ \sigma_{a_i}(\mathbf{f}) = \bot \rangle \mid (r, \mathbf{f}) \in \widehat{cond^{-}(A)}\} \quad (15)$$

Let $\Psi^{rep}_{\Pi,\pi} = (E^{rep}, C^{rep})$ where $E^{rep} = E_I \cup \bigcup_i E^{rep}_{a_i}$ and $C^{rep} = C_G \cup \bigcup_i C^{rep}_{a_i}$.

Proposition 2. *If $\Psi^{rep}_{\Pi,\pi}$ has a satisfying assignment Γ, then $\Delta = \{\, r \mid \Gamma \vdash \delta_r \,\}$ is valid repair set.*

Proof. First, notice that an effect or condition token whose activity variable is assigned to $\bot$ plays no role in the decision problem: being inactive, such a condition would always be supported and an effect always coherent. Assigning its activity to $\bot$ is thus equivalent to removing it from the set of effects or conditions. Replacing each δ_r variable in $\Psi^{rep}_{\Pi,\pi}$ by its value in Γ thus creates a timetable problem that is equisatisfiable to the one of deciding whether the plan is valid for $\Delta(\Pi)$. Γ provides a satisfying assignment for this problem, in which only the mutex variables remain. □

Lifted Case Repairs

For lifted repairs, the only difference is that the lifted test plan $\gamma = [\mathbf{a_1}, \ldots, \mathbf{a_n}]$ is now composed of lifted actions. It thus contains a finite set of variables x_i, each with a type τ_i defining its domain $\mathcal{O}_{\tau_i}$.

While this does not require any change to the prior definitions that were agnostic to the presence of variables, let us discuss the consequences. First, the codomain of the action substitutions $\sigma_{\mathbf{a_i}}$ is now composed of both domain objects and plan variables. As a result, plan variables will appear in the now lifted facts of the effect and condition tokens $E^{rep}_{\mathbf{a_i}}$ and $C^{rep}_{\mathbf{a_i}}$.

A satisfying assignment to $\Psi^{rep}_{\Pi,\gamma} = (E^{rep}, C^{rep})$ would assign *(i)* the plan variables appearing in the tokens, *(ii)* the repair application variables, and *(iii)* the assignment to mutex variables.

Proposition 3. *Given a lifted domain repair problem (Π, γ) where γ is a lifted plan. If $\Psi^{rep}_{\Pi,\gamma}$ has a satisfying assignment Γ, then $\Delta = \{\, r \mid \Gamma \vdash \delta_r \,\}$ is valid repair set.*

Proof. Let π be the ground plan obtained by replacing all plan variables in γ by their value in Γ. Replacing all plan variables in $P^{rep}_{\Pi,\gamma}$ by their assignment in Γ results in a timetable problem identical to $\Psi^{rep}_{\Pi,\pi}$, the one of the ground repair problem (Π, π). Γ provides a satisfying assignment for this problem. □

Constraint Encoding and Solving

In this section, we provide an encoding of the satisfaction of a timetable (E, C) into a constraint satisfaction problem.

Constraint Satisfaction Problem

Figure 1 provides this encoding in a conjunctive normal form (CNF), assuming finite integer domain. The variables of the CSP match closely the ones from the previous section, with m_i variables being the mutex variables associated with each effect, the δ_r variables encoding whether the r atomic repair is used, and variables x_i representing the variables in the lifted plan (if any). All variables have an integer domain, with the special case of the boolean variables where the values 0 and 1 respectively represent $\bot$ and $\top$. Each object $o \in \mathcal{O}$ is associated to a unique integer value, such that $dom(\mathcal{O}_\tau)$ is a set of integer values representing the objects of type τ. Each additional variable s_{ij} is a boolean variable

Variables	$m_i \in [t_i + \epsilon, H]$	$\forall e_i \in E$
	$\delta_r \in [0, 1]$	$\forall$ repair $r \in \mathcal{R}_\Pi$
	$x_i \in \mathcal{O}_{\tau_i}$	$\forall$ variable x_i with type τ_i in the lifted plan
	$s_{ij} \in [0, 1]$	$\forall$ condition $c_i \in C$ and effect $e_j \in E$
Constraints	$\neg p_i \vee \bigvee_{e_j \in E} s_{ij}$	$\forall c_i \in C$
	$\neg s_{ij} \vee p_j$	$\forall c_i \in C, e_j \in E$
	$\neg s_{ij} \vee t_j < t_i$	$\forall c_i \in C, e_j \in E$
	$\neg s_{ij} \vee t_i \leq m_j$	$\forall c_i \in C, e_j \in E$
	$\neg s_{ij} \vee \mathbf{f}_i = \mathbf{f}_j$	$\forall c_i \in C, e_j \in E$
	$\neg s_{ij} \vee v_i = v_j$	$\forall c_i \in C, e_j \in E$
	$\neg p_i \vee \neg p_j \vee m_i \leq t_j \vee m_j \leq t_i \vee \mathbf{f}_i \neq \mathbf{f}_j$	$\forall$ distinct $e_i, e_j \in E^2$
Objective	minimize $\Sigma_{r \in \mathcal{R}_\Pi} \delta_r$	

Figure 1: Constraint satisfaction problem for deciding the satisfiability of a timetable $\Psi = (E, C)$. Variables are defined in terms of the original domain repair problem.

representing whether a condition $c_i \in C$ is supported by the effect $e_j \in E$.

In our domain repair problem, each activity variable (p_i) is assigned to either $\top$, $\bot$, a repair variable δ, or its negation. Each time instant variable (t_i) is assigned to a constant integer denoting, the temporal origin, a plan step or the horizon. And each value (v_i) is assigned to either $\top$ or $\bot$. A fact f_i is a term $pred_i(\alpha_i^1, \ldots, \alpha_i^n)$ where $pred_i$ is the name of the predicate and associated to its n arguments. Each argument α_i^k is either an integer constant associated to an object, or a plan variable.

Constraints are defined in terms of the token of the timetable (E, C). The first constraint ensures that at least one effect is selected as a support (definition 4.1) and the following five ensure the requirement for a support hold whenever it is selected (eq. (1)). The last one enforces the coherence requirement.

Recalling that a fact $\mathbf{f_i}$ is of the form $pred_i(\alpha_i^1, \ldots, \alpha_i^n)$, the expression $\mathbf{f_i} = \mathbf{f_j}$ would be simplified to $\bot$ if $pred_i \neq pred_j$ and otherwise expanded to $\bigwedge_k \alpha_i^k = \alpha_j^k$. Likewise, $\mathbf{f_i} \neq \mathbf{f_j}$ would be extended to a disjunction of unequality constraints.

Encoding Properties

The encoding uses $\mathcal{O}(|C| \times |E|)$ variables, dominated by the support variables. It introduces in the order of $\mathcal{O}(|C| \times |E| + |E|^2)$ constraints, each as a disjunction of boolean variables or (in)equality expressions. The integration of an inequality expression into the disjunction may in turn introduce new variables reifying it, depending on the combinatorial solver.

Note that each of the inequalities are of the form $x \leq y$, where x and y can be either a constant or an integer variable, which is a term in integer difference logic (IDL, Armando et al. 2005). Assuming the equalities are expanded to inequalities, the formula could thus be solved with an SMT solver with IDL (De Moura and Bjørner 2008), which is the setting used in the original work of Bit-Monnot (2018). In such SMT solvers, any difference expression involving at least one variable would be reified into a new boolean variable visible by the internal solver. Instead, scheduling-oriented CP solvers with *lazy clause generation* such as Tempo (Hebrard 2025) or Aries (Bit-Monnot 2023) would natively support the integration of *bound expressions* into disjunctive constraints. As a result they would only reify difference logic expressions involving two variables, which in our model may only occur between plan variables in the lifted domain repair. In our implementation, we use the Aries CP solver (Bit-Monnot 2023), which was previously used for similar encodings (Bit-Monnot and Godet 2025).

Objective & Search

The objective is to minimize the size of the repair set, encoded as simple linear sum. As a result an optimal solution is a smallest repair set, matching the optimality requirement of Lin, Grastien, and Bercher (2023). As in their work, we do not consider weights for the repair, but they could be integrated into the objective to capture a priori preferences.

Search can be performed by regular branch-and-bound which is available in any CP solver. For this problem, where the objective is expected to be very low, we instead use a simple assume-and-relax strategy in which the objective value is iteratively upper-bounded by a sequence $[0, 1, \ldots, |\mathcal{R}_\Pi|]$. As a result, plan validity is first checked (0 repairs allowed), then a single repair is allowed, etc. The next step is triggered when the solver proves unsatisfiability for the bound and the iteration proceeds until a satisfiable formula is found.

Another option worth mentioning would be to use search algorithms for maximum satisfiability that rely on unsatisfiable core extraction (Morgado, Dodaro, and Marques-Silva 2014; Gange et al. 2020). Much like, the algorithm for ground domain repair of Lin, Grastien, and Bercher (2023), these exploit a hitting set of the unsatisfiable cores to guide the search.

Relaxing Well-Formedness Assumption

At last, let us discuss how our well-formedness assumption can be relaxed to align with the usual semantics of classical planning, as notably found in PDDL.

The classical planning semantics define a successor state of a ground action a in state s as $a(s) = (s \setminus \mathit{eff}^-(a)) \cup \mathit{eff}^+(a)$. This has two problematic implications. First, the fact that $\mathit{eff}^{\pm}(a)$ are sets, means that they contain no duplicates. This is very easy to verify when given a ground action, but when given a lifted action, it means that the number of effects (size of the set) may depend on the instantiation of its variables, which will only be determined during search. Second, any negative effect may be overridden by a positive effect on the same fact.

We tackle this issue with a simple preprocessing step on the set of effect tokens $E_{\mathbf{a}}$ (or equivalently $E^{rep}_{\mathbf{a}}$) associated to an action $\mathbf{a}$ in the test plan. An effect $e_i \in E_{\mathbf{a}}$ is rewritten with a new activity variable p'_i that is constrained to be true iff the original token was active (p_i) and there is no other effect overriding it:

$$p'_i \Leftrightarrow p_i \wedge \bigwedge_{e_j \in E_{\mathbf{a}}} \neg overrides(e_j, e_i)$$

$$overrides(e_j, e_i) = \begin{cases} p_j \wedge \mathbf{f}_i = \mathbf{f}_j \wedge j < i & \text{if } v_i = v_j \\ p_j \wedge \mathbf{f}_i = \mathbf{f}_j & \text{if } v_j \wedge \neg v_i \\ \bot & \text{if } \neg v_j \wedge v_i \end{cases}$$

where an effect e_j overrides e_i if it is active (p_j), they refer to the same fact ($\mathbf{f}_i = \mathbf{f}_j$), and either *(i)* they are both positive or both negative and e_j appears earlier in $E_{\mathbf{a}}$ ($j < i$), or *(ii)* e_j is positive and e_i is negative.

This simple reformulation would ensure that for each action, at most one effect token is active for each ground fact, aligning it with the set semantics of classical planning while retaining the compatibility with the coherence requirements of the timetable problem.

Experiments

Our baseline for the experimental evaluation of our approach is (Lin, Grastien, and Bercher 2023) for the ground repair problem and (Bavandpour et al. 2025) for the lifted one. All experiments were run with 3.5 GB RAM using Intel E5-2695 CPUs restricted to one core. All resources for the experiments and additional material are available online.[2]

Benchmarks

The original benchmarks for the ground repair problem were created by Lin, Grastien, and Bercher (2023) on top of the *fast downward problem suite*. The flawed domains are obtained by introducing one error (precondition or effect addition, or effect removal) in a fraction (0.1, 0.3, and 0.5) of the (correct) reference domains' action schemas. Each flawed domain is paired with a ground whitelist plan (together with initial and goal states) accepted by the reference domain.

We use the same set of benchmarks for our evaluation, only excluding those where the plan is not a valid test plan (not generated for the problem) or that fail to parse (due to an equality in the effect of an action). In total, 4655 instances are included in our benchmarks for the ground case, with 1499 being satisfiable (i.e. requiring no repairs), 2991 requiring 1 to 3 repairs, and 167 instances requiring 4 or more.

For the lifted repair problem, we use the same flawed domains and generate lifted whitelist plans by lifting a fraction (0.33, 0.66, and 1.00) of each action's arguments in the ground whitelist plans, based on the implementation of Bavandpour et al. (2025). Reproducing their experimental setting, we only consider plans of length up to 15, leaving 441 instances for each considered lifted arguments fraction (0.33, 0.66, and 1.00) and a total of 1323. Out of these 440 instances, 207 are satisfiable, 228 require 1-2 repairs, 4 require 3 repairs, and 1 requires 5.

One of the limitations of (Bavandpour et al. 2025) is that they interpret each lifted argument as a new independent plan variable (even if it shares its name with another one in the plan specification). This means a lifted argument appearing multiple times in the lifted plan is allowed to be grounded differently in each occurrence, as each of them effectively corresponds to a different variable. To ensure a fair comparison, we used a modified parser in the reported experiments that deduplicates all variables, so that they are seen as independent by the solver.

Planner result	Solved (optimal)	Timeout
Baseline	4623	32 (0.68%)
Aries	**4648**	**7** (0.15%)
Number of instances	4655	4655

Table 1: Number of ground repair problems solved in 60s.

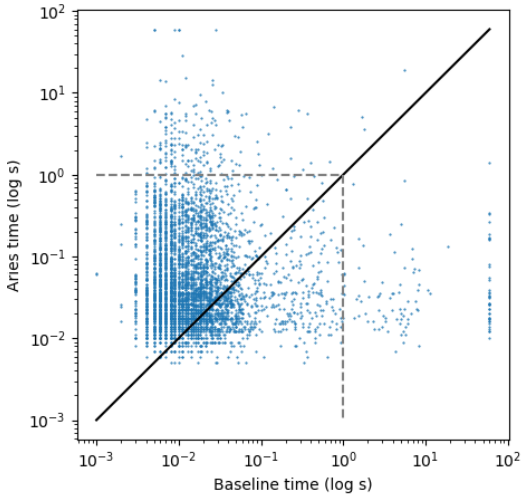

Figure 2: Runtimes of solved ground repair problems.

Results

A first observation is that our method agrees with the baseline on the size of the optimal repair set. Without the preprocessing to align the formalism with the set-based semantics of PDDL, our method would occasionally require one or two additional repairs to remove duplicated or inconsistent effects in the action schema.

[2] https://github.com/plaans/resources-icaps-26-domrep

Lifted Arguments Proportion	0.33	0.66	1.00
Baseline (UCS, Exhaustive)	275	197	102
Baseline (UCS, RelaxPre)	288	287	202
Aries	**440**	**440**	**437**
Number of instances	440	440	440

Table 2: Number of lifted repair problems solved in 900s.

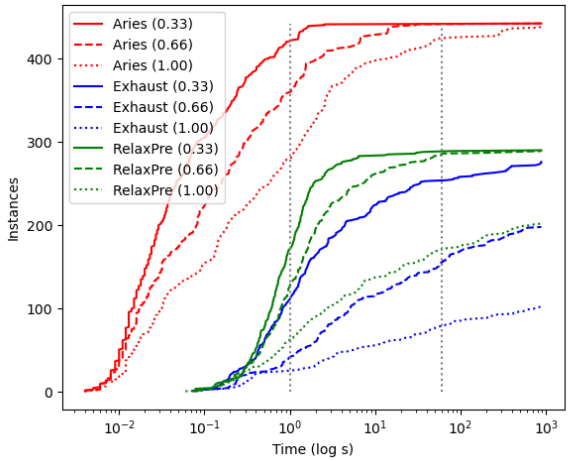

Figure 3: Lifted repair problems solved over time, for whitelist plans (of length 15 at most) with a proportion of 0.33, 0.66, and 1.00 lifted arguments in each action.

Ground Repair As shown in Table 1, an optimal ground repair is found within 60 seconds for more than 99% of instances by both the baseline and our approach. Our method however tends to more reliably solve the problems with only 7 timeouts compared to 32 for the baseline. This advantage is counter-balanced by a generally larger solving time (in 88% of instances) which can be observed in Figure 2. Despite that, the large majority of problems (95%) is solved in less than one second. It should be noted that our method does not treat ground problems any differently from the lifted ones, leaving many software optimizations on the table.

Lifted Repair Table 2 and Figure 3 depict the number of lifted repair problem instances solved within 900 seconds using our approach and two variants of the baseline method. Blind search (UCS – Uniform Cost Search) is used for both variants of the baseline as it is shown to perform the best in (Bavandpour et al. 2025). The *Exhaust* configuration of the baseline method corresponds to a complete, exhaustive exploration of the repair sets (equivalent to ours). On the other hand, *RelaxPre* is a faster but incomplete configuration, effective when there are missing positive preconditions and a small total number of possible predicate groundings. The results clearly show our approach significantly outperforms not only the complete *Exhaust* configuration, but also the faster but incomplete *RelaxPre* configuration.

It should also be mentioned that our approach scales well with plan length as it remains performant for plans longer than 15 actions, which were not considered in the original experimental setting of the baseline. Moreover, when using our original parser and enforcing that all occurrences of the same lifted argument are grounded identically, our performance is slightly better still, as the satisfiability space is smaller and the CP solver tends to require less search effort to prove unsatisfiability.

Discussion and Future Work

Beyond Classical Planning While this paper focused on classical planning to match the existing domain repair definitions, it is worth noting that the encoding that we used was initially defined for temporal planning (Bit-Monnot 2018) with multi-valued state variables. Although the formalism may differ in some places, it is our belief that the method could be readily applied to repair temporal planning domains with durative actions and parallel plans. An extension of the encoding was recently proposed for numeric planning (Godet, Bit-Monnot, and Lesire-Cabaniols 2025), relying on the same mechanism of activating effect and condition tokens. While the method could certainly be adapted there, it would raise the interesting question of the choice of numeric values in the added effects, for which several strategies may be reasonable, including delegating this choice to the constraint solver.

Alternative Repair Problems A current limitation of the approach presented is that it only handles a single whitelist plan. Extending the encoding to more than one whitelist plan should be straightforward by encoding several repair subproblems in the same constraint model, such that only repair variables are shared among the subproblems. Another fairly reasonable extension would be the handling of repairs for the initial state (Göbelbecker et al. 2010) and goal state, that could rely on a similar mechanism of activating effect and condition tokens. A more challenging and interesting problem would be to also consider in the encoding the blacklist plans of Lin et al. (2025). This would not only require considering the addition of new preconditions – similarly to new effects in this paper – but also change the solution criteria to having at least one precondition violated (notably in the first inapplicable action, if it is specified by the user). While the encoding would be similar to ours, this fairly drastically changes the problem addressed by the CP solver, making it hard to extrapolate any expectations from the current results.

Conclusion

Enabling modeling support and assistance for users, particularly domain designers, is essential for making automated planning tools and techniques more accessible. In this paper, we proposed a constraint-based approach to address the problem of domain repair with a ground or lifted test plan, in which a user wants to correct their domain to accept this plan as a solution. Compared to the most recent baselines, it is shown to be very performant in both the ground and lifted cases. Moreover, its simplicity and the very few assumptions made in the encoding open a range of possible extensions to more complex settings.

Acknowledgments

The authors would like to thank the conference reviewers for their comments and feedback. This research was partly funded by the Agence Nationale de la Recherche (ANR) under project HumFleet (ANR-23-CE33-0003-01) and benefited from the AI Interdisciplinary Institute ANITI. ANITI is funded by the France 2030 program under the Grant agreement n°ANR-23-IACL-0002.

References

Armando, A.; Castellini, C.; Giunchiglia, E.; and Maratea, M. 2005. A SAT-Based Decision Procedure for the Boolean Combination of Difference Constraints. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing (SAT)*.

Bavandpour, N. K.; Lauer, P.; Lin, S.; and Bercher, P. 2025. Repairing Planning Domains Based on Lifted Test Plans. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI)*.

Bercher, P.; Sreedharan, S.; and Vallati, M. 2025. A Survey on Model Repair in AI Planning. In *Proceedings of the 34th International Joint Conference on Artificial Intelligence (IJCAI)*.

Bit-Monnot, A. 2018. A Constraint-Based Encoding for Domain-Independent Temporal Planning. In *Proceedings of the 24th International Conference on Principles and Practice of Constraint Programming (CP)*.

Bit-Monnot, A. 2023. Enhancing Hybrid CP-SAT Search for Disjunctive Scheduling. In *Proceedings of the 26th European Conference on Artificial Intelligence (ECAI)*.

Bit-Monnot, A.; and Godet, R. 2025. Towards Canonical and Minimal Solutions in a Constraint-Based Plan-Space Planner. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI)*.

De Moura, L.; and Bjørner, N. 2008. Z3: An efficient SMT Solver. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*.

Gange, G.; Berg, J.; Demirović, E.; and Stuckey, P. J. 2020. Core-Guided and Core-Boosted Search for CP. In *Proceedings of the 17th International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR)*.

Gigante, N.; Leofante, F.; and Micheli, A. 2025. Counterfactual Scenarios for Automated Planning. In *Proceedings of the 22nd International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

Godet, R.; Bit-Monnot, A.; and Lesire-Cabaniols, C. 2025. When Quality Matters: Constraint Programming for Automated Temporal and Numeric Planning. In *Proceedings of the 37th International Conference on Tools with Artificial Intelligence (ICTAI)*.

Gragera, A.; Fuentetaja, R.; García-Olaya, A.; and Fernández, F. 2023. A Planning Approach to Repair Domains with Incomplete Action Effects. In *Proceedings of the 33rd International Conference on Automated Planning and Scheduling (ICAPS)*.

Gragera, A.; and Muise, C. 2024. One Repair to Rule Them All: Repairing a Broken Planning Domain Using Multiple Instances. In *ICAPS 2024 Workshop on Reliable Data-Driven Planning and Scheduling (RDDPS)*.

Göbelbecker, M.; Keller, T.; Eyerich, P.; Brenner, M.; and Nebel, B. 2010. Coming up with Good Excuses: What to Do When No Plan Can Be Found. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*.

Hebrard, E. 2025. Disjunctive Scheduling in Tempo. In *Proceedings of the 31st International Conference on Principles and Practice of Constraint Programming (CP)*.

Lin, S.; Grastien, A.; and Bercher, P. 2023. Towards Automated Modeling Assistance: An Efficient Approach for Repairing Flawed Planning Domains. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence*.

Lin, S.; Grastien, A.; Shome, R.; and Bercher, P. 2025. Told You That Will Not Work: Optimal Corrections to Planning Domains Using Counter-Example Plans. *Proceedings of the 39th AAAI Conference on Artificial Intelligence*.

Lin, S.; Höller, D.; and Bercher, P. 2024. Modeling Assistance for Hierarchical Planning: An Approach for Correcting Hierarchical Domains with Missing Actions. In *Proceedings of the 17th International Symposium on Combinatorial Search (SoCS)*.

Morgado, A.; Dodaro, C.; and Marques-Silva, J. 2014. Core-Guided MaxSAT with Soft Cardinality Constraints. In *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming (CP)*.

Welt, M.; Lodemann, A.; Olz, C.; Bercher, P.; and Glimm, B. 2025. Calculating Optimal Corrections for Unsolvable Planning Problems. In *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI)*.