Detecting Unsolvability Based on Separating Functions

Remo Christen, Salomé Eriksson, Florian Pommerening, Malte Helmert

University of Basel, Switzerland
{remo.christen,salome.eriksson,florian.pommerening,malte.helmert}@unibas.ch

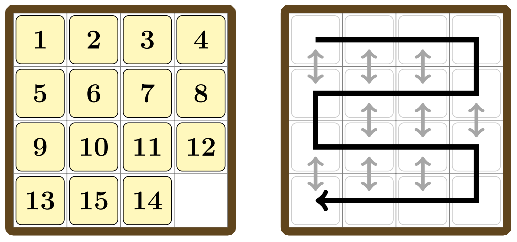
Abstract

While the unsolvability IPC sparked a multitude of planners proficient in detecting unsolvable planning tasks, there are gaps where concise unsolvability arguments are known but no existing planner can capture them without prohibitive computational effort. One such example is the sliding tiles puzzle, where solvability can be decided in polynomial time with a parity argument. We introduce *separating functions*, which can prove that one state is unreachable from another, and show under what conditions a potential function over any nonzero ring is a separating function. We prove that we can compactly encode these conditions for potential functions over features that are pairs, and show in which cases we can efficiently synthesize functions satisfying these conditions. We experimentally evaluate a domain-independent algorithm that successfully synthesizes such separating functions from PDDL representations of the sliding tiles puzzle, the Lights Out puzzle, and Peg Solitaire.

Introduction

On January 24, 1880, dentist Dr. Charles K. Pevey offered a set of teeth worth $25 to anyone who could solve the *14-15 puzzle* (Slocum and Sonneveld 2006). To claim the prize you had to show how 15 wooden tiles arranged as in Figure 1a can be sorted by shifting them inside their box. Larger prizes were offered later on but nobody ever claimed them as the problem happens to be unsolvable (Johnson and Story 1879; Archer 1999).

To understand why the problem is unsolvable, consider the positions in order of the snaking path shown in Figure 1b. We say that a pair of tiles is *ordered incorrectly* if the one that occurs first on this path does not occur first in the goal position. (The blank position does not count as a tile.) The initial position has exactly one incorrectly ordered pair, namely the tiles 14 and 15. Moving any tile along the path does not change the number of incorrectly ordered pairs. Only shifting a tile along one of the gray arrows shown in Figure 1b can change this number. A tile moving along one of these arrows changes its position relative to exactly 2, 4, or 6 tiles. Independent of whether the individual changes are from correct to incorrect order vice versa, the total number of incorrectly ordered pairs is always changed by an even



(a) 14-15 puzzle (b) Ordering path and transitions affecting order

Figure 1: Initial configuration and (un)solvability argument for the 14-15 puzzle.

number. Consider a function that maps every game state to 0 if the number of incorrectly ordered pairs is even, and to 1 if it is odd. Moving any tile in any game situation does not change the value of the function, but the initial state and goal state have different values. In this way, the function *separates* reachable states from unreachable ones.

We formalize the general idea underlying this argument in the area of classical planning to detect if planning tasks are unsolvable. After introducing planning tasks formally, we define *separating functions* as functions that separate two states in the way we saw for the 14-15 puzzle. We then discuss a syntactic restriction of separating functions to weighted sums analogous to potential heuristics (Pommerening et al. 2015). In this form, separating functions can compactly express why some planning tasks are unsolvable without relying on a search.

Separating functions can exist over different rings and relations. We present both necessary and sufficient conditions describing such functions and discuss their properties with respect to certain relations and rings. Furthermore, we show how to synthesize separating functions and so come up with an argument for unsolvability automatically. In general, this is intractable but we show tractable cases in particular for functions over the field $\mathbb{F}_2$, which encode parity arguments as the one for the 14-15 puzzle, and over $\mathbb{R}$. Mutex information can be used to strengthen separating functions and can prove unsolvability in additional cases.

To evaluate this approach empirically, we implemented the synthesis of separating functions, specifically for parity arguments (i.e, over $\mathbb{F}_2$). While there are only few domains where this proves unsolvability, we argue that these domains rely on parity arguments that are hard to capture efficiently by other techniques.

Finally, we compare separating functions to related notions such as traps (Lipovetzky, Muise, and Geffner 2016), inductive sets (Eriksson, Röger, and Helmert 2017), dead-end formulas (e. g. Junghanns and Schaeffer 1998), and invariants (e. g. Blum and Furst 1997).

Background

We define *planning tasks* Π according to the SAS$^+$ formalism (Bäckström and Nebel 1995), where $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_* \rangle$, and require them to be in *transition normal form* (TNF; Pommerening and Helmert 2015) [1].

The set $\mathcal{V}$ contains variables V that can be assigned to values within their finite domain $dom(V)$. An *atom* is a tuple $\langle V, d \rangle$ where V is a variable and $d \in dom(V)$ is its assigned value. We refer to the variable mentioned in atom a by $var(a)$. A *partial state* is an assignment of a subset of variables in $\mathcal{V}$ to a value in their respective domain. Given a partial state s, we refer to the set of variables assigned by s as $vars(s)$ and to the value d that s assigns to variable V by $s[V] = d$. When convenient we treat partial state s as a set of atoms $\{\langle V, d \rangle \mid V \in vars(s), s[V] = d\}$. A *state* is a partial state s with $vars(s) = \mathcal{V}$.

The set $\mathcal{O}$ contains *operators* $o = \langle pre(o), eff(o) \rangle$ where the partial states $pre(o)$ and $eff(o)$ denote the preconditions and effects of o respectively. In TNF, we must have that $vars(pre(o)) = vars(eff(o))$ for all $o \in \mathcal{O}$ and denote this variable set by $vars(o)$. An operator o is *applicable* in state s iff $pre(o)$ is consistent with s, i. e. $pre(o) \subseteq s$. Applying o in s yields the transition $s \xrightarrow{o} s'$, where state s' firstly inherits all atoms in $eff(o)$, and secondly all atoms a from s where $var(a) \notin vars(eff(o))$. We write $s' = s[\![o]\!]$. A state s' is called *reachable* from state s iff there exists a sequence of operators such that their successive application starting in s yields s'.

Lastly, s_0 and s_* refer to the initial and goal state. (In TNF, the goal is always a fully specified state.) Task Π is *solvable* if s_* is reachable from s_0 and *unsolvable* otherwise.

In order to emphasize the generality of the presented concepts, we discuss them in terms of the least restrictive algebraic structures possible. The most basic structure we consider is the *ring*, a set of elements over which associative, commutative addition as well as associative, distributive multiplication is defined. For addition, the identity element 0 and the inverse of every element in the ring exists. We only consinder *nonzero* rings, where the multiplicative identity element $1 \neq 0$ is defined. Furthermore, we consider a special case of rings called *fields*, in which multiplication is commutative and multiplicative inverses are defined. Given prime number p, we denote the finite field containing p elements by $\mathbb{F}_p$.

[1]This does not incur a loss in generality; any planning task can be transformed into TNF in linear time.

Separating Functions

The core mechanic of our approach can be understood as defining a property that holds in all states reachable from a state s_α. If this property does not hold in a state s_ω we know that s_ω is not reachable from s_α. In particular, we define this property based on a relation $\bowtie$ over a set X and a function φ that maps states to values in X.

Definition 1 (separating function)**.** *Let Π be planning task with states $\mathcal{S}$. Further, let X be a set and $\bowtie$ a binary relation over X. A function $\varphi : \mathcal{S} \to X$* separates *two states $s_\alpha, s_\omega \in \mathcal{S}$ if it satisfies*

$$\varphi(s_\alpha) \not\bowtie \varphi(s_\omega) \tag{1}$$

$$\varphi(s_\alpha) \bowtie \varphi(s) \qquad \text{for all states } s \text{ reachable from } s_\alpha. \tag{2}$$

We call φ a separating function *for s_α and s_ω with $\bowtie$ over X. We omit s_α, s_ω, $\bowtie$, and X if they are clear from context.*

In the 14-15 puzzle, the function $\varphi_\#$ that maps every state to the number of incorrectly ordered pairs is a separating function over $\mathbb{Z}$ for the initial state shown in Figure 1a and the goal state with the relation $\{(x, y) \mid x \equiv y \pmod 2\}$. The function φ_{par} that maps states s to the parity of $\varphi_\#(s)$ is a separating function over $\{0, 1\}$ with the equality relation. We can also see $\varphi_{\text{par}} = 1$ as an invariant, i. e. a property that holds in all states reachable from the initial state.

If s_α and s_ω are separated by a function then s_ω cannot be reachable from s_α. In particular, if a separating function for the initial and goal state of a task exists, the task is unsolvable. We can also show that separating functions are powerful enough to capture unsolvability on all tasks:

Theorem 1. *Let Π be a planning task with initial state s_0 and goal state s_*, and let X be a set with $|X| \geq 2$. Then Π is unsolvable if and only if there exists a separating function for s_0 and s_* over X.*

Proof. If Π is solvable, then s_* is reachable from s_0, so if φ were a separating function then we would get $\varphi(s_0) \bowtie \varphi(s_*)$ from (2) but $\varphi(s_0) \not\bowtie \varphi(s_*)$ from (1).

If Π is unsolvable, consider two distinct elements a and b from X and define $\varphi(s) = a$ if s is reachable from s_α and $\varphi(s) = b$ otherwise. Then φ is a separating function with the equality relation: since Π is unsolvable, s_* is not reachable from s_0, so $\varphi(s_0) = a \not\bowtie b = \varphi(s_*)$ and for all states s reachable from s_0, we have $\varphi(s_0) = a \bowtie a = \varphi(s)$. □

Synthesizing such separating functions is in general intractable due to the global condition in equation (2), since deciding whether s is reachable from s_α is as difficult as deciding solvability in the first place. We thus consider sufficient conditions for functions to be separating functions.

Theorem 2. *Given a planning task Π with states $\mathcal{S}$, a set X, and a reflexive transitive binary relation $\bowtie$ over X, every function $\varphi : \mathcal{S} \to X$ with $\varphi(s_\alpha) \not\bowtie \varphi(s_\omega)$ and either*

$$\varphi(s) \bowtie \varphi(s') \qquad \text{for all transitions } s \to s' \text{ reachable from } s_\alpha, \tag{3}$$

or

$$\varphi(s) \bowtie \varphi(s') \qquad \text{for all transitions } s \to s' \tag{3'}$$

is a separating function for s_α and s_ω with $\bowtie$ over X.

Proof. It is sufficient to show that (3) implies (2) because we require (1) explicitly and (3′) implies (3). For any state s reachable from s_α, we know that (3) holds along all transitions of a path from s_α to s. Because $\bowtie$ is reflexive and transitive, we can use induction along this path to show $\varphi(s_\alpha) \bowtie \varphi(s)$. □

Note that the argument for the 14-15 puzzle that we made in the introduction used the reasoning in equation (3′): we argued that shifting any tile to a neighboring position would not affect the parity of incorrectly ordered pairs. This argument was independent of the position of the remaining tiles, i.e., it applied to all transitions $s \to s'$, even those where s is not reachable from s_0.

The local property in equation (3′) no longer argues about reachability but it still poses one condition for every transition in the state space. The number of such transitions can be exponential in the size of the task. In order to synthesize separating functions, we will thus consider cases where these constraints can be represented compactly.

Separating Potential Functions

Potential functions (Pommerening et al. 2015) are a general tool to express functions over states. They depend on a notion of *features*, which are conjunctions of atoms. A feature $f = a_1 \wedge \cdots \wedge a_n$ is *true* in state s, formally $s \models f$, iff $a_i \in s$ for all atoms $1 \leq i \leq n$. The *dimension* of f is given by the number of its conjuncts. The dimension of a potential function is its largest feature dimension. A *weight function* w maps features $\mathcal{F}$ to values, and the potential function then adds the weights of all true features. Using Iverson brackets (Knuth 1992) for indicator functions, we can define the potential function as

$$\varphi(s) = \sum_{f \in \mathcal{F}} w(f)[s \models f].$$

So far, potential functions have used values from $\mathbb{R}$ (e. g., Pommerening et al. 2015) or $\mathbb{Z}$ (e. g., Seipp et al. 2016a), but we adopt a more general view here. Potential functions only use addition and multiplication of the form $a \cdot b$ where a is a value of w and b is a boolean expression. As such, potential functions can be evaluated over any ring and we consider weight functions $w : \mathcal{F} \to R$ for arbitrary (nonzero) rings R. The definition of potential functions remains the same, but addition, multiplication, and the values 0 and 1 returned by the indicator function are interpreted as the respective operations and neutral elements of R.

As an example for a potential function over a ring other than $\mathbb{R}$ or $\mathbb{Z}$, consider the 14-15 puzzle again. We can introduce a feature for each pair of tiles that is true if those tiles are incorrectly ordered. With the weight function $w_{\text{14-15}} : \mathcal{F} \to \mathbb{Z}/2\mathbb{Z}$ that maps all features to 1, the potential function computes the parity of incorrectly ordered pairs. A computation over the quotient ring $\mathbb{Z}/2\mathbb{Z}$, and in particular the isomorphic field $\mathbb{F}_2$, generally makes a *parity argument* and we will consider this special case in more detail later.

Potential functions over sufficiently high-dimensional features can represent any mapping from states to values and therefore do not incur inherent restrictions compared to general functions. In other words, for all nonzero rings R the statement of Theorem 1 also holds when only considering potential functions of arbitrary dimension over R, instead of general functions. Given a ring and an unsolvable task, we can thus always find a set of features and a weight function over the ring to certify that the task is unsolvable. This may however require an exponential number of features and synthesizing the potential function is PSPACE-hard in general, as it decides the plan existence problem. We therefore consider computationally tractable cases, focusing on lower-dimensional potential functions and cases where weights of separating potential functions can be synthesized efficiently.

Our goal is to develop a system of linear constraints of the form $\sum_i a_i x_i \bowtie b$ or $\sum_i a_i x_i \not\bowtie b$ where the solutions correspond to weight functions of separating potential functions. We make one further restriction to the relation used in the argument that allows us to reformulate conditions (3′) and (1). We say the relation is *translation-invariant* if

$$x \bowtie y \text{ implies } (x + z) \bowtie (y + z) \text{ for all } x, y, z \in R.$$

Note that this also implies that $\not\bowtie$ is translation-invariant, otherwise there would exist $x, y, z \in R$ such that $x \not\bowtie y$ and $(x + z) \bowtie (y + z)$. But since $\bowtie$ is translation-invariant, this implies $((x + z) - z) \bowtie ((y + z) - z)$ which contradicts $x \not\bowtie y$.

Since $\not\bowtie$ and $\bowtie$ are translation-invariant, we can rewrite conditions (3′) and (1) so that all occurrences of φ are on the left-hand side, by adding the additive inverse of the right-hand side. Plugging in the definition of a potential function and rearranging the terms (using associativity, distributivity and additive commutativity), brings the condition into the desired form. We summarize this in the following theorem.

Theorem 3. *Let R be a nonzero ring and $\bowtie$ a reflexive, transitive, and translation-invariant relation on R. Let $\mathcal{F}$ be a set of features for a planning task Π. If there exists a weight function $w : \mathcal{F} \to R$ that satisfies*

$$\sum_{f \in \mathcal{F}} w(f)([s_\alpha \models f] - [s_\omega \models f]) \not\bowtie 0 \quad (4)$$

$$\sum_{f \in \mathcal{F}} w(f)([s \models f] - [s' \models f]) \bowtie 0 \quad \textit{for all transitions } s \to s', \quad (5)$$

then the potential function for $\mathcal{F}$ and w separates s_α and s_ω.

To make the method tractable, we need a polynomial method of solving such a system of constraints for the specific ring R and relation $\bowtie$, and the number of constraints and decision variables has to be polynomial in the size of the task. We will first show that for one-dimensional features there is a compact set of constraints that is equivalent to conditions (4)–(5). For two-dimensional features we can in general only derive a compact set of constraints which implies these conditions. However, for the special cases of the relation $=$ on any nonzero ring and the relations $\leq$ on ordered rings, we can show that our compact representation completely classifies potential separating function for these features, i.e., that the reverse implication holds as well. Lastly, we will present efficient solution techniques for different combinations of $\bowtie$ and R.

One-Dimensional Features

The key to attaining a compact system of equations is to express condition (5) in terms of operators instead of transitions. Considering operator o, we ultimately want to find an expression for the change in value of features $f \in \mathcal{F}$ across transitions $s \xrightarrow{o} s'$, formally $([s \models f] - [s[\![o]\!] \models f])$, independent of state s. For tasks in TNF and the feature set containing all one-dimensional features, Pommerening et al. (2015) observe that this change is fully defined by o. To see this, consider an arbitrary operator o and atom $a = \langle V, d\rangle$. If $V \notin vars(o)$, then o does not change a and thus the truth of a does not change between s and $s[\![o]\!]$. Otherwise, we know that V appears both in the precondition and effect of o. Since we have $pre(o) \subseteq s$ and $\mathit{eff}(o) \subseteq s[\![o]\!]$, we can thus directly infer whether $s \models f$ and $s[\![o]\!] \models f$ holds.

Theorem 4. *If $\mathcal{F}$ is the set of all one-dimensional features, then constraints* (4)–(5) *over a nonzero ring R and a reflexive, transitive, and translation-invariant relation $\bowtie$ are equivalent to the following constraints over the decision variables $w(f)$:*

$$\sum_{f\in\mathcal{F}} w(f)([s_\alpha \models f] - [s_\omega \models f]) \not\bowtie 0 \quad (6)$$

$$\sum_{f\in pre(o)\cup \mathit{eff}(o)} w(f)([pre(o) \models f] - [\mathit{eff}(o) \models f]) \bowtie 0 \quad \text{for all operators } o \in \mathcal{O}. \quad (7)$$

Proof. Since conditions (4) and (6) are identical, we only need to show that conditions (5) and (7) are equivalent for any transition $s \xrightarrow{o} s'$, which we do in two steps:

$$\begin{aligned} &\sum_{f\in\mathcal{F}} w(f)([s \models f] - [s' \models f]) \\ \overset{(1)}{=} &\sum_{f\in\{\langle V,d\rangle \mid V\in vars(o)\}} w(f)([s \models f] - [s' \models f]) \\ \overset{(2)}{=} &\sum_{f\in pre(o)\cup \mathit{eff}(o)} w(f)([pre(o) \models f] - [\mathit{eff}(o) \models f]). \end{aligned}$$

Step (1) is justified by the fact that for all $f \in \{\langle V, d\rangle \mid V \notin vars(o)\}$ we have $s' \models f$ iff $s \models f$, which means $w(f)([s \models f] - [s' \models f]) = 0$ and can thus be omitted from the sum.

For step (2) we similarly first observe that, if $f \notin pre(o)$ and $f \notin \mathit{eff}(o)$, then $s' \models f$ iff $s \models f$ and thus all $f \notin pre(o) \cup \mathit{eff}(o)$ can be omitted in the sum. What is left to show is that we can replace $s \models f$ with $pre(o) \models f$ and $s' \models f$ with $\mathit{eff}(o) \models f$. If $f \in pre(o)$ then we obviously have $s \models f$ iff $pre(o) \models f$, and due to the task being in TNF we also know that $\langle V, d'\rangle \in \mathit{eff}(o)$, from which follows that $s' \models f$ iff $\mathit{eff}(o) \models f$. The argument for $f \in \mathit{eff}(o)$ is analogous.

Since the equation holds, we know that for every condition (5) for $s \xrightarrow{o} s'$ there is a corresponding condition (7) for o, and in the other direction every condition (7) for o is covered by at least one condition (5) for $s \xrightarrow{o} s'$. □

Two-Dimensional Features

For two-dimensional potential heuristics, Pommerening, Helmert, and Bonet (2017) show a way to represent admissibility constraints closely resembling constraints (4)–(5) as a compact system of inequalities. We adapt their result to separating function constraints and show that it generalizes to both potential functions over rings and more general relations.

Unlike the one-dimensional case, the change in value of two-dimensional features $f \in \mathcal{F}$ across transitions $s \xrightarrow{o} s'$ is not obviously independent of state s anymore. We thus partition the set of features $\mathcal{F}$ into three subsets based on their overlap with o: *irrelevant* features $\mathcal{F}^{irr}$ that share no variable with $vars(o)$, *context-independent* features $\mathcal{F}^{ind}$ that share both[2] variables with $vars(o)$, and *context-dependent* features $\mathcal{F}^{ctx}$ that share one of two variables with $vars(o)$. Using these notions, we can define a compact system of equations describing separating functions.

Theorem 5. *Constraints* (4)–(5) *over a nonzero ring R and a reflexive, transitive, and translation-invariant relation $\bowtie$ are implied by the following constraints over the decision variables $w(f)$ and X_V^o*

$$\sum_{f\in\mathcal{F}} w(f)([s_\alpha \models f] - [s_\omega \models f]) \not\bowtie 0 \quad (8)$$

$$\sum_{f\in\mathcal{F}^{ind}} w(f)\Delta(o, f) + \sum_{V\in\mathcal{V}_{\bar{o}}} X_V^o \bowtie 0 \quad \text{for all operators } o \in \mathcal{O} \quad (9)$$

$$\sum_{\substack{f\in\mathcal{F}^{ctx} \\ f=a\wedge\langle V,d\rangle}} w(f)\Delta(o, a) \bowtie X_V^o \quad \text{for } o \in \mathcal{O},\ V \in \mathcal{V}_{\bar{o}}, d \in dom(V), \quad (10)$$

where $\Delta(o, f) = ([pre(o) \models f] - [\mathit{eff}(o) \models f])$, $\mathcal{V}_{\bar{o}} = \mathcal{V} \setminus vars(o)$ and a is an atom such that $var(a) \in vars(o)$.

Proof. This proof generalizes the one by Pommerening, Helmert, and Bonet.

Constraint (8) is the same as (4), so we must only show that constraints (9)–(10) imply (5) for a state s and operator o. We independently consider the contributions of irrelevant, context-independent, and context-dependent features.

The value of irrelevant features never changes across operator o, therefore $\Delta(o, f) = 0$ for all $f \in \mathcal{F}^{irr}$, meaning such features do not contribute to the total change. Context-independent features on the other hand are affected, but also completely determined by o. Both variables mentioned in features $f \in \mathcal{F}^{ind}$ are assigned in $pre(o)$ and $\mathit{eff}(o)$, thus the change in value of f across o is given by $\Delta(o, f)$.

This leaves the contribution of context-dependent features $f \in \mathcal{F}^{ctx}$. As the name suggests, their value does not solely depend on o, but also on the context, i. e. the state s. We thus use constraint (10) specifically for the value $s[V]$ to show that the second part of constraint (9) covers the contribution

[2]We can consider all features to be two-dimensional by replacing one-dimensional features a with $a \wedge a$.

of context-dependent features:

$$\begin{aligned}
&\sum_{f \in \mathcal{F}^{ctx}} w(f)([s \models f] - [s' \models f]) \\
=& \sum_{V \in \mathcal{V}_{\bar{o}}} \sum_{\substack{f \in \mathcal{F}^{ctx} \\ f = a \wedge \langle V, d \rangle}} w(f)([s \models f] - [s' \models f]) \\
=& \sum_{V \in \mathcal{V}_{\bar{o}}} \sum_{\substack{f \in \mathcal{F}^{ctx} \\ f = a \wedge \langle V, s[V] \rangle}} w(f)\Delta(o, a) \overset{(10)}{\bowtie} \sum_{V \in \mathcal{V}_{\bar{o}}} X_V^o
\end{aligned}$$

Together with the first two results and constraint (9), this shows that constraint (5) is satisfied. $\square$

Note that the number of constraints and decision variables in (8)–(10) is polynomial in the size of Π. In particular, there are $O(|\mathcal{O}||\mathcal{V}|d)$ constraints and $O((|\mathcal{V}|d)^2 + |\mathcal{O}||\mathcal{V}|)$ decision variables where d is an upper bound for the size of variable domains.

We now show that for two important choices of relation $\bowtie$ constraints (4)–(5) also imply the constraints in Theorem 5 and thus completely classify the separating potential functions described in Theorem 3.

Equality Relation Case

For any ring R the equality relation is reflexive, transitive, and translation-invariant. It can thus always be used as the relation $\bowtie$. In that case we can subtract two constraints from each other (e.g., $a = b$ and $c = d$ implies $a - c = b - d$). Consider constraint (5) for two states s_1 and s_2 that differ only in one variable $V \in \mathcal{V}_{\bar{o}}$. The contribution of all irrelevant features is 0 and the contribution of features that do not depend on V is the same in both constraints and cancels out. Likewise, context-dependent features containing $\langle V, d \rangle$ for $s_1[V] \neq d \neq s_2[V]$ contribute nothing. The remaining constraint is

$$\sum_{\substack{f \in \mathcal{F}^{ctx} \\ f = a \wedge \langle V, s_1[V] \rangle}} w(f)\Delta(o, a) = \sum_{\substack{f \in \mathcal{F}^{ctx} \\ f = a \wedge \langle V, s_2[V] \rangle}} w(f)\Delta(o, a)$$

and shows that the total contribution of context-dependent features is the same independent of the value of V. Choosing this value as X_V^o then shows that constraints (9) and (10) are implied by (5).

Inequality Relation Case

For *ordered rings* like $\mathbb{Z}$, $\mathbb{Q}$, or $\mathbb{R}$, there exist a translation-invariant total order $\leq$ that satisfies our properties. For each operator o and variable $V \in \mathcal{V}_{\bar{o}}$ there then is a value d for which

$$\sum_{\substack{f \in \mathcal{F}^{ctx} \\ f = a \wedge \langle V, d \rangle}} w(f)\Delta(o, a)$$

is maximal according to this order. Choosing this value as X_V^o then shows that constraint (10) is satisfied. To see that constraint (9) is also satisfied, consider constraint (5) for the state $s^{\max}$ where o is applicable and all variables $V \in \mathcal{V}_{\bar{o}}$ take their maximal value.

Solving the Constraints

Now that we have a way to construct compact separating function constraints of desirable form, namely $\sum_i a_i x_i \bowtie b$ and $\sum_i a_i x_i \not\bowtie b$, we discuss possible choices of rings and relations such that the resulting constraints can be solved efficiently. Without loss of generality, we focus on potential functions that separate the initial and goal state in this section, as this allows us to show unsolvability directly. We will denote particular cases as a pair consisting of the chosen ring R and relation $\bowtie$, written as $\langle R, \bowtie \rangle$. In each case, we do not necessarily need a way to compute a solution to the constraints, just a way to check whether a solution exists.

Gaussian Elimination If we restrict R to be a field, we can divide a constraint by a constant (due to multiplicative inverses existing in fields) and thus can use Gaussian elimination (e. g. Strang 2016) to find solutions. If we can express $\sum_i a_i x_i \neq b$ as an equality, this gives us a method to solve our constraints for $\langle \mathbb{F}, = \rangle$ with a field $\mathbb{F}$.

This is particularly useful for the finite fields $\mathbb{F}_p$ over prime number p, for which the equality relation is the only reflexive, transitive, and translation-invariant relation. We cannot directly express the inequality $\sum_i a_i x_i \neq b$ as an equality, but in a finite field with p elements, there are only $p - 1$ choices for a value to be different from b. We can thus construct one system of equations for each value $b' \neq b$, where we replace the inequality with $\sum_i a_i x_i = b'$. If one of the systems is solvable, a weight function defining a potential function that separates s_0 and s_* exists and the task is unsolvable. Note that, in general, this procedure leads to a number of systems that is exponential in the number of inequalities, but the constraints we construct always contain exactly one inequality, making the procedure tractable.

Of particular interest is the case $\mathbb{F}_2$, where only one value $b' \neq b$ exists and thus only a single system of equations has to be considered. Gaussian elimination in this field requires no division or multiplication (all entries are 0 or 1), but only addition of two rows. Rows $\sum_i a_i x_i \bowtie b$ can be efficiently represented as a bit vector where the i-th bit is set iff $a_i = 1$. Adding two such rows corresponds to computing the bitwise XOR of these vectors, which is an efficient operation.

A further notable property of $\mathbb{F}_2$ is that the resulting weight function w essentially defines a set of relevant features that contains every feature f for which $w(f) = 1$ holds. In this view, the unsolvability argument is that the parity of this set's cardinality remains constant along transitions, but differs between initial and goal state.

Linear Programming On ordered rings, a total order $\leq$ exists that satisfies our requirements for $\bowtie$. For $\langle \mathbb{Z}, \leq \rangle$, $\langle \mathbb{Q}, \leq \rangle$, and $\langle \mathbb{R}, \leq \rangle$, linear programming can be used to solve systems of equations efficiently. However, the relation $\not\leq$ is $>$ and cannot be expressed in a linear program. To solve this, we observe that any separating potential function on $\langle \mathbb{Z}, \leq \rangle$, $\langle \mathbb{Q}, \leq \rangle$, or $\langle \mathbb{R}, \leq \rangle$ remains a separating function if all weights are scaled up by a positive constant, since the scaling factor cancels out in the constraints of Theorem 3. We can thus rewrite the inequality $\varphi(s_0) - \varphi(s_*) > 0$ as $\varphi(s_0) - \varphi(s_*) \geq 1$. This is because, if a function φ exists, we can scale it up by the factor $1/(\varphi(s_0) - \varphi(s_*))$.

Mutex Relaxation

In Definition 1 we have seen the necessary and sufficient conditions for a separating function. Realizing that the global nature of condition (3) makes it unsuitable for generating arguments automatically, we replaced it by the stronger condition (3′) which encodes a sufficient but not necessary argument. In terms of the set of transitions considered, these conditions represent two extremes: condition (3) considers only the necessary and therefore minimal set of reachable transitions, while condition (3′) considers the set of all and therefore the maximal set of transitions. Between these two cases, there lies a range of separating arguments that disregard some, but not all, unreachable transitions.

We describe one such argument by taking the compact representation from Theorem 5 and relaxing condition (10) by introducing reachability information in the form of mutexes. A mutex is a set of two atoms $\{a, a'\}$ such that no state reachable from the initial state contains both atoms, and we say a is mutex with a'.

The decision variable X_V^o for a given operator o and variable $V \notin \mathit{vars}(o)$ induces an equation for every value $d \in \mathit{dom}(V)$. Considering one such equation E, its right-hand side consists of a sum over all context-dependent features $f = a \wedge \langle V, d\rangle$ where a is an atom whose variable is mentioned in o. We can see that, within equation E, only features containing the atom $\langle V, d\rangle$ are considered. This means that E only puts restrictions on those transitions $s \xrightarrow{o} s'$ where $\langle V, d\rangle \in s, s'$. Additionally, the operator o provides the knowledge that $s \supseteq \mathit{pre}(o)$ and $s' \supseteq \mathit{eff}(o)$. We can exploit this knowledge to prune equations that only constrain unreachable transitions. If $\langle V, d\rangle$ is mutex with an atom in $\mathit{pre}(o) \cup \mathit{eff}(o)$, we can conclude that there is no reachable transition over o where the atom $\langle V, d\rangle$ holds and consequently that equation E only constrains unreachable transitions. Therefore, we can safely ignore E as all reachable transitions are still sufficiently constrained and condition (3) holds.

The idea of relaxing constraints over potential functions using mutexes has previously been explored by Fišer, Horčík, and Komenda (2020), who applied this idea to admissibility constraints over one-dimensional features. They identify *disambiguations* (Alcázar et al. 2013) to reduce ambiguity in constraints stemming from both a partial goal state and operators o where $\mathit{vars}(\mathit{pre}(o)) \neq \mathit{vars}(\mathit{eff}(o))$. Clearly, these optimizations do not immediately apply to our case, as tasks in TNF do not contain these ambiguities by definition. In our case, ambiguity is instead introduced by two-dimensional, context-dependent features, as seen in the previous paragraph. Despite this difference, our application can also be expressed in terms of disambiguations: for a given decision variable X_V^o, we identify the disambiguation D_V^o over variable V which contains only those atoms $\langle V, d\rangle$ such that $\langle V, d\rangle$ is not mutex with any atom in the preconditions or effects of o.

We can now relax condition (10) and rewrite it as

$$\sum_{\substack{f \in \mathcal{F}^{ctx} \\ f = a \wedge \langle V, d\rangle}} w(f)\Delta(o, a) \bowtie X_V^o \qquad \text{for } o \in \mathcal{O}, V \in \mathcal{V}_{\bar{o}}, \langle V, d\rangle \in D_V^o.$$

Evaluation

We implemented two variants for synthesizing potential functions over $\langle \mathbb{F}_2, =\rangle$ separating the initial and goal state:

- $\mathtt{sp}(1, \mathbb{F}_2)$ uses features of dimension 1, and
- $\mathtt{sp}(2, \mathbb{F}_2)$ uses features of dimension 2 and relaxes the operator constraints with the help of mutexes computed through h^2 (Bonet and Geffner 2001).

Our implementation is built on top of Fast Downward 20.06 (Helmert 2006) and contains our own straightforward version of Gaussian elimination restricted to $\mathbb{F}_2$. The experiments were run on an Intel Xeon Silver 4114 processor running at 2.2 GHz with a 30 minute timeout and a 3.5 GiB memory limit. The evaluation of the experiments was performed with Downward Lab (Seipp et al. 2017).

We tested our implementation on the benchmark set of the unsolvability IPC 2016 and additionally on the puzzle Lights Out, which we encoded in PDDL. We compared our approach against Aidos (Seipp et al. 2016b), the winner of the unsolvability IPC 2016, as well as $\mathtt{sp}(2, \mathbb{R})$, a component of Aidos which can be interpreted as synthesizing two-dimensional potential functions over $\langle \mathbb{R}, \leq\rangle$ separating the initial and goal state, and $\mathtt{sp}(1, \mathbb{R})$, which is the same component but only considers one-dimensional potential functions. We ran these configurations with CPLEX v12.10 as the LP solver. Since most domains are not in TNF, our approach first translates the task into TNF as described by Pommerening and Helmert (2015) with additionally disambiguating the goal state (Alcázar et al. 2013). The code, benchmarks, and experiment data are available online (Christen et al. 2022).

While all domains contain some solvable tasks, we only present results for unsolvable tasks, noting that no configuration incorrectly reported a solvable problem as unsolvable. Table 1 shows how many unsolvable tasks each approach identified as unsolvable. For $\mathtt{sp}(2, \mathbb{F}_2)$ we report two numbers: due to $\mathtt{sp}(2, \mathbb{F}_2)$ computing h^2 for the mutex relaxation step, there are cases where h^2 alone already detects unsolvability, which is denoted by the number in parentheses. The first number is the amount of problems where we find a separating function but h^2 alone cannot show unsolvability, and is the number we will focus on in what follows.

We first compare separating functions over $\mathbb{F}_2$. We see that $\mathtt{sp}(1, \mathbb{F}_2)$ is already sufficient for showing unsolvability in all problems in lights-out and all but two problems in pegsol. An investigation of the found separating functions shows that for Lights Out they correspond to a polynomial time computable argument from Anderson and Feil (1998) that fully decides unsolvability, while for Peg Solitaire they correspond to a parity argument described in Beasley (1985), which is also polynomial time computable but only a sufficient argument for unsolvability. While $\mathtt{sp}(2, \mathbb{F}_2)$ runs out of time in larger lights-out problems since it builds significantly larger constraint systems, it is the only approach to prove unsolvability for all sliding-tiles tasks, suggesting that it can fully capture the parity argument introduced earlier. To show the importance of mutex relaxation we tested a version of $\mathtt{sp}(2, \mathbb{F}_2)$ without it, which could not find any separating functions for sliding-tiles. This

domain	$\mathtt{sp}(1, \mathbb{F}_2)$	$\mathtt{sp}(2, \mathbb{F}_2)$	$\mathtt{sp}(1, \mathbb{R})$	$\mathtt{sp}(2, \mathbb{R})$	Aidos
bag-barman (20)	0	0	0	0	**12**
bag-gripper (25)	0	0	**15**	**15**	**15**
bag-transport (29)	0	0 (+15)	9	25	**26**
bottleneck (25)	0	0 (+10)	0	**25**	**25**
cave-diving (25)	0	0 (+1)	1	2	**8**
chessboard-pebbling (23)	0	0	**23**	**23**	**23**
document-transfer (20)	0	0 (+3)	0	10	**13**
lights-out (74)	**74**	55	0	0	36
over-nomystery (24)	0	0 (+2)	0	5	**14**
over-rovers (20)	0	0 (+3)	0	5	**13**
over-tpp (33)	0	0 (+1)	1	12	**26**
pegsol (24)	22	22	0	4	**24**
pegsol-row5 (15)	1	0 (+2)	12	**15**	**15**
sliding-tiles (20)	0	**20**	0	0	10
tetris (20)	0	0	15	**20**	**20**

Table 1: Problems shown unsolvable by domain. For $\mathtt{sp}(2, \mathbb{F}_2)$, the number in parentheses denotes the amount of tasks which h^2 shows unsolvable.

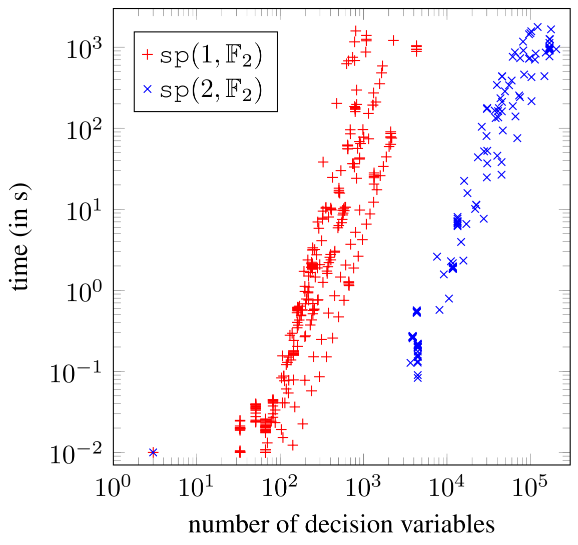


Figure 2: Termination time as a function of decision variables for all problems where the algorithm terminated.

is due to the fact that the parity argument only works for consistent sliding-tiles configurations, i. e. states where each tile is in exactly one location, while the state space considered by Fast Downward's FDR representation allows for a tile to be in several locations. The mutex relaxation lets $\mathtt{sp}(2, \mathbb{F}_2)$ disregard constraints over such inconsistent states.

Looking at $\mathtt{sp}(2, \mathbb{R})$ we see that separating potential functions over $\mathbb{R}$ are successful in more domains, but cannot find separating functions for any lights-out or sliding-tiles problem, and only for a few pegsol problems. This result confirms that separating potential functions over different fields complement each other. The same is not true when varying dimensionality, as $\mathtt{sp}(1, \mathbb{R})$ is dominated by $\mathtt{sp}(2, \mathbb{R})$ with respect to expressive power. Additionally, the tasks shown unsolvable by $\mathtt{sp}(1, \mathbb{R})$ are a subset of those shown by $\mathtt{sp}(2, \mathbb{R})$, which suggests that, over $\mathbb{R}$, expressiveness is more limiting than resource consumption.

Finally, Aidos proves unsolvability in the most problems overall, which is not surprising since it is a portfolio of different approaches which resorts to a breadth-first search once the other approaches fail. But while Aidos can confirm all pegsol problems to be unsolvable, it only achieves this in roughly half the cases for lights-out and sliding-tiles, which $\mathtt{sp}(2, \mathbb{F}_2)$, and in the case of lights-out even $\mathtt{sp}(1, \mathbb{F}_2)$, fully proves to be unsolvable.

One important property of separating functions is that they are not complete, i. e. if they terminate without finding a separating function we do not know if the task is unsolvable or not. However, they are guaranteed to run polynomial in the amount of features they consider. Figure 2 confirms this, since a linear relation between termination and number of decision variables on a log-log plot implies a polynomial relation. This means we can roughly estimate the time required to terminate in advance, which together with their complementary nature makes them interesting to use in portfolios. For example, a simple complete algorithm could first try to find a separating function, and in the case of failure, fall back on a breadth-first search, similarly to Aidos.

Finally we tested $\mathtt{sp}(2, \mathbb{F}_2)$ on 50 unsolvable instances of the 24 puzzle with a more generous time limit. The instances were generated from the (solvable) instances from Korf and Felner (2002) by switching two neighboring tiles in the shared goal state. All problems could be detected as unsolvable with an average termination time of 5862 seconds (roughly 1 hour and 38 minutes) and almost no variance (from 5580 to 6136 seconds), showing that the approach scales well and yields consistent termination times.

Related Work

Unsolvability and more generally unreachability in planning has been studied extensively and many different concepts were introduced for it. One such example is the *invariant*, which is a formula ν over state variables that is true in all states reachable from the initial state. Sometimes invariants are defined in a stronger way, requiring that for any state s satisfying ν, all states reachable from s must satisfy ν as well. We denote this meaning with *strong invariant*. Invariants have been studied extensively with regards to both generation (Fox and Long 1998; Gerevini and Schubert 1998; Rintanen 2000; Helmert 2009) and application (Blum and Furst 1997; Chen, Zhao, and Zhang 2007; Alcázar and Torralba 2015). Mutexes are a prominent example of invariants.

When we are interested in whether or not the goal is reachable, we can turn our attention to *dead-end formulas* δ (Junghanns and Schaeffer 1998; Cserna et al. 2018). These formulas are true only in states from which no goal can be reached. Marking such states has been a byproduct of admissible heuristics for a long time as they assign an infinite heuristic value to states that are found to be dead-ends, but there are more recent approaches specifically tailored to detecting unsolvable states (Steinmetz and Hoffmann 2017; Ståhlberg, Francès, and Seipp 2021).

Traps (Lipovetzky, Muise, and Geffner 2016) generalize the notion of strong invariants by not focusing on the initial state. A formula τ is a trap iff for all states s which satisfy τ, all states reachable from s satisfy τ as well. A closely related concept introduced for creating unsolvability proofs for planning tasks are *inductive sets* (Eriksson, Röger, and Helmert 2017). A set of states S is an inductive set iff for $s \in S$ we have $s' \in S$ for all successor states s' of s.

All these concepts as well as the separating functions we introduce here essentially partition the search space into two parts, one containing the states that have a certain property P and one containing all other states. For traps, invariants and dead-end formulas this property is satisfying a formula, for inductive sets S it is being contained in S, and for separating functions it is being related to $\varphi(s_\alpha)$. Furthermore, we can classify them on whether they require 1) some s_α to have P, 2) some s_ω to not have P and 3) a form of (partial) inductivity.

While only invariants and separating functions require 1), and only dead-end functions and separating functions require 2), all concepts require inductivity in some sense. Inductive sets and traps (and thus by extension strong invariants) use an identical form of *forward inductivity*: if a state has property P, its successors have it as well. This form of inductivity ensures that all states not satisfying P are unreachable from any state satisfying it. Separating functions and (normal) invariants on the other hand require forward inductivity only on the subset of states reachable from s_α, which means we can only conclude that any state not having P is unreachable from s_α[3]. Finally, dead-end formulas use partial backwards inductivitiy, i. e. for all states s from which we can reach s_ω and which do not have P, their predecessors must also not have it. From this follows that s_ω is unreachable from any state having P. Note however that in practice dead-end formulas derived from *consistent* heuristics are also forward inductive, amounting to forward inductivity on dead-ends.

Since traps and inductive sets use the same type of inductivity and have no restrictions on which states have P or not, they are equally expressive, i. e. the set of states satisfying a trap is an inductive set and a formula that is satisfied by all states contained in an inductive set is a trap. In what follows we will thus only consider traps, noting that all results related to traps also apply to inductive sets.

Separating functions differ from traps by using only partial inductivity, and requiring s_α to have P and s_ω to not have P. From this follows that for any trap τ that is neither unsatisfiable nor valid, we can define a separating function over $\langle \mathbb{F}_2, = \rangle$ for some $s_\alpha \models \tau$ and $s_\omega \not\models \tau$ by assigning 0 to all states that satisfy τ and 1 to all states s' that do not satisfy τ. However, there are non-trivial separating functions φ which do not describe a trap, since we might have $\varphi(s_\alpha) \bowtie \varphi(s)$ and $\varphi(s_\alpha) \not\bowtie \varphi(s')$ for a state s and its successor s' if s is not reachable from s_α.

Due to strong invariants being a special case of traps, the above results apply to them as well. Looking at weak invariants however we can see that a separating function φ with $s_\alpha = s_0$ always induces a weak invariant which is true in all states s with $\varphi(s_\alpha) \bowtie \varphi(s)$. This is due to the fact that both concepts use partial forward inductivity.

Finally, dead-end formulas and separating functions are in general incomparable. There are dead-end formulas δ which do not induce a separating function with $s_\omega = s_*$, because there might be states s and s' reachable from any state (and thus from any s_α) such that $s \models \delta$ but $s' \not\models \delta$, as long as s cannot reach s_*. Similarly, a separating function might have $\varphi(s_\alpha) \bowtie \varphi(s')$ for some s' from which a goal can be reached, as long as s' is unreachable from s_α. A formula that is true for all s with $\varphi(s_\alpha) \bowtie \varphi(s)$ is thus not necessarily a dead-end formula.

Conclusion

While many different concepts for showing unsolvability in planning tasks exist, they cannot yet capture certain types of unsolvability arguments. Inspired from parity arguments for the sliding tiles puzzle we introduce *separating functions*, which offer a fresh and principled way of arguing why a state is unreachable from another. Starting out from a general definition based on an arbitrary relation $\bowtie$ and with an arbitrary codomain X, we show step by step how we need to restrict the relation and codomain in order to make the computation of separating functions practically feasible. Our theoretical analysis shows that we can derive polynomial time algorithms for synthesizing separating potential functions over $\langle \mathbb{R}, \leq \rangle$, $\langle \mathbb{Q}, \leq \rangle$, $\langle \mathbb{Z}, \leq \rangle$ and $\langle \mathbb{F}_p, = \rangle$, where $\mathbb{F}_p$ is a finite field for prime number p.

We empirically show that synthesizing separating functions over $\langle \mathbb{F}_2, = \rangle$ is successful in several domains, and covers parity arguments for the sliding tiles puzzle, the Lights Out puzzle, and Peg Solitaire. Furthermore, a comparison against a configuration that can be seen as synthesizing separating functions over $\langle \mathbb{R}, \leq \rangle$ suggests that different types of separating functions complement each other.

We believe that separating functions over finite fields other than $\mathbb{F}_2$ are able to find arguments which our currently evaluated configurations cannot find, and are thus a promising venue for future research. Additionally, investigating which other types of functions are suitable to express separating functions might yield an even wider array of complementary techniques.

Acknowledgments

We have received funding for this work from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement no. 817639). Moreover, this research was partially supported by TAILOR, a project funded by the EU Horizon 2020 research and innovation programme under grant agreement no. 952215.

[3]Note that if the separating function satisfies condition (3′) it effectively requires full forward inductivity, meaning we can conclude that all states not having P are unreachable from any state having P.

References

Alcázar, V.; Borrajo, D.; Fernández, S.; and Fuentetaja, R. 2013. Revisiting Regression in Planning. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2254–2260. AAAI Press.

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 2–6. AAAI Press.

Anderson, M.; and Feil, T. 1998. Turning Lights Out with Linear Algebra. *Mathematics Magazine*, 71(4): 300–303.

Archer, A. F. 1999. A Modern Treatment of the 15 Puzzle. *American Mathematical Monthly*, 106(9): 793–799.

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS$^+$ Planning. *Computational Intelligence*, 11(4): 625–655.

Beasley, J. D. 1985. *The Ins & Outs of Peg Solitaire (Recreations in Mathematics)*. Oxford University Press.

Blum, A.; and Furst, M. L. 1997. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence*, 90(1–2): 281–300.

Bonet, B.; and Geffner, H. 2001. Planning as Heuristic Search. *Artificial Intelligence*, 129(1): 5–33.

Chen, Y.; Zhao, X.; and Zhang, W. 2007. Long-Distance Mutual Exclusion for Propositional Planning. In Veloso, M. M., ed., *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 1840–1845.

Christen, R.; Eriksson, S.; Pommerening, F.; and Helmert, M. 2022. Code, benchmarks, and experiment data for the ICAPS 2022 paper "Detecting Unsolvability Based on Separating Functions". https://doi.org/10.5281/zenodo.6376175.

Cserna, B.; Doyle, W. J.; Ramsdell, J. S.; and Ruml, W. 2018. Avoiding Dead Ends in Real-Time Heuristic Search. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, 1306–1313. AAAI Press.

Eriksson, S.; Röger, G.; and Helmert, M. 2017. Unsolvability Certificates for Classical Planning. In Barbulescu, L.; Frank, J.; Mausam; and Smith, S. F., eds., *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS 2017)*, 88–97. AAAI Press.

Fišer, D.; Horčík, R.; and Komenda, A. 2020. Strengthening Potential Heuristics with Mutexes and Disambiguations. In Beck, J. C.; Karpas, E.; and Sohrabi, S., eds., *Proceedings of the Thirtieth International Conference on Automated Planning and Scheduling (ICAPS 2020)*, 124–133. AAAI Press.

Fox, M.; and Long, D. 1998. The Automatic Inference of State Invariants in TIM. *Journal of Artificial Intelligence Research*, 9: 367–421.

Gerevini, A. E.; and Schubert, L. 1998. Inferring State Constraints for Domain-Independent Planning. In Rich, C.; and Mostow, J., eds., *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI 1998)*, 905–912. AAAI Press.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M. 2009. Concise Finite-Domain Representations for PDDL Planning Tasks. *Artificial Intelligence*, 173: 503–535.

Johnson, W. W.; and Story, W. E. 1879. Notes on the "15" Puzzle. *American Journal of Mathematics*, 2(4): 397–404.

Junghanns, A.; and Schaeffer, J. 1998. Single-Agent Search in the Presence of Deadlocks. In Rich, C.; and Mostow, J., eds., *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI 1998)*, 419–424. AAAI Press.

Knuth, D. E. 1992. Two Notes on Notation. *American Mathematical Monthly*, 99(5): 403–422.

Korf, R. E.; and Felner, A. 2002. Disjoint Pattern Database Heuristics. *Artificial Intelligence*, 134(1–2): 9–22.

Lipovetzky, N.; Muise, C.; and Geffner, H. 2016. Traps, Invariants, and Dead-Ends. In Coles, A.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 211–215. AAAI Press.

Pommerening, F.; and Helmert, M. 2015. A Normal Form for Classical Planning Tasks. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 188–192. AAAI Press.

Pommerening, F.; Helmert, M.; and Bonet, B. 2017. Higher-Dimensional Potential Heuristics for Optimal Classical Planning. In Singh, S.; and Markovitch, S., eds., *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI 2017)*, 3636–3643. AAAI Press.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From Non-Negative to General Operator Cost Partitioning. In Bonet, B.; and Koenig, S., eds., *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI 2015)*, 3335–3341. AAAI Press.

Rintanen, J. 2000. An Iterative Algorithm for Synthesizing Invariants. In Kautz, H.; and Porter, B., eds., *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*, 806–811. AAAI Press.

Seipp, J.; Pommerening, F.; Röger, G.; and Helmert, M. 2016a. Correlation Complexity of Classical Planning Domains. In Kambhampati, S., ed., *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*, 3242–3250. AAAI Press.

Seipp, J.; Pommerening, F.; Sievers, S.; and Helmert, M. 2017. Downward Lab. https://doi.org/10.5281/zenodo.790461.

Seipp, J.; Pommerening, F.; Sievers, S.; Wehrle, M.; Fawcett, C.; and Alkhazraji, Y. 2016b. Fast Downward Aidos. In Muise, C.; and Lipovetzky, N., eds., *Unsolvability International Planning Competition: Planner Abstracts*, 28–38.

Slocum, J.; and Sonneveld, D. 2006. *The 15 Puzzle Book: How it Drove the World Crazy*. Slocum Puzzle Foundation.

Ståhlberg, S.; Francès, G.; and Seipp, J. 2021. Learning Generalized Unsolvability Heuristics for Classical Planning. In Zhou, Z.-H., ed., *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI 2021)*, 4175–4181. IJCAI.

Steinmetz, M.; and Hoffmann, J. 2017. State Space Search Nogood Learning: Online Refinement of Critical-Path Dead-End Detectors in Planning. *Artificial Intelligence*, 245: 1–37.

Strang, G. 2016. *Introduction to Linear Algebra*. Wellesley-Cambridge Press.