Symmetry Detection and Breaking in Linear Cost-Optimal Numeric Planning

Alexander Shleyfman[1], **Ryo Kuroiwa**[2], **J. Christopher Beck**[2]

[1]Department of Computer Science, Bar-Ilan University, Haifa, Israel
[2]Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Canada
alexash@biu.ac.il, ryo.kuroiwa@mail.utoronto.ca, jcb@mie.utoronto.ca

Abstract

One of the main challenges of domain-independent numeric planning is the complexity of the search problem. The exploitation of structural symmetries in a search problem can constitute an effective method of pruning search branches that may lead to exponential improvements in performance. For over a decade, symmetry breaking techniques have been successfully used within both optimal and satisficing classical planning. In this work, we show that symmetry detection methods applied in classical planning, with some effort, can be modified to detect symmetries in linear numeric planning. The detected symmetry group, thereafter, can be used almost directly in the A^*-based symmetry breaking algorithms such as DKS and Orbit Space Search. We empirically validate that symmetry pruning can yield a substantial reduction in the search effort, even if algorithms are equipped with a strong heuristic, such as LM-cut.

Introduction

Deterministic planning is the problem of finding a sequence of actions that brings the actor from a given state to some desired state. While the formalisms to describe this paradigm may vary, where richer models can capture finer aspects, and thus, represent the problem with higher fidelity. For example, in classical planning, the variables of the problem are restricted to finite domains, whereas the numeric variant of planning encompasses both continuous and finite variable ranges. Satisficing planners that can manage numeric fluents were designed at the beginning of the century (Hoffmann 2003), yet, it seems that the progress was slowed due to the theoretical undecidability of even simplest numeric formalisms (Helmert 2002). In recent years, however, there seems to have been a surge of interest in planning with numeric fluents, resulting in the development of multiple heuristics for both optimal and satisficing settings (Aldinger and Nebel 2017; Scala et al. 2016, 2017; Piacentini et al. 2018; Scala et al. 2020; Kuroiwa et al. 2022). Unfortunately, having a good heuristic is not enough to assemble an efficient planner, e.g., A^* can expand an exponential number of states even when equipped with an almost perfect heuristic (Helmert and Röger 2008).

Partially to account for this deficit, pruning methods were developed for classical planning (Fox and Long 2002; Coles and Coles 2010; Nissim, Apsel, and Brafman 2012; Wehrle and Helmert 2012; Holte and Burch 2014), and in the past decade, the use of symmetry-based pruning methods has shown its potential within the context of forward search (Pochter, Zohar, and Rosenschein 2011; Domshlak, Katz, and Shleyfman 2013; Wehrle et al. 2015; Gnad et al. 2017). In particular, symmetry reduction methods such as DKS and Orbit Space Search (OSS) were effectively applied in a wide range of classical planning domains, often substantially reducing the expanded state-space size, with a significant increase in planning performance (Domshlak, Katz, and Shleyfman 2012, 2015). In classical planning, symmetry reduction methods compute equivalence classes of states and states are aggregated based on a precomputed symmetry group. The search exploits these classes by replacing all states in this class with some representative state. Domshlak et al. (2012) have shown that given a "path" where each state is replaced by a representative state, one may efficiently reconstruct a corresponding path in the original state space. Hence, the expanded search tree must contain at most one representative of each class at all times.

In this work, we show that the graph-based symmetry detection method proposed by Pochter et al. (2011) can be adapted for the numeric setting. We extend the notion of structural symmetries proposed by Shleyfman et al. (2015) to account for linear numeric formulas and demonstrate the equivalence of the obtained symmetries. We also established that computing these numeric symmetries is not harder than computing the symmetries for classical planning. By grounding these symmetries to the state space level, we enable the use of both DKS and OSS practically as is. Finally, our experimental evaluation demonstrates that in presence of symmetries in the planning task, the symmetry breaking algorithms compete favorably with A^*, even if equipped with a strong heuristic such as numeric LM-cut.

Preliminaries

We consider a fragment of numeric planning restricted to the FDR formalism (Bäckström and Klein 1991; Bäckström and Nebel 1995; Helmert 2009) with the addition of numeric state variables called *linear numeric planning*. In this formalism, conditions and effects on numeric variables are restricted to linear formulas. Formally, a linear task (LT) is defined as a 4-tuple $\Pi = \langle \mathcal{V}, \mathcal{A}, s_I, G \rangle$, where $\mathcal{V} = \mathcal{V}_p \cup \mathcal{V}_n$, with $\mathcal{V}_p$ is being a finite set of propositional variables, where each propositional variable $v \in \mathcal{V}_p$ is associated with a finite domain $\mathcal{D}(v)$, and $\mathcal{V}_n$ is a set of numeric variables. Numeric variables $v \in \mathcal{V}_n$ have rational values, i.e., $\mathcal{D}(v) = \mathbb{Q}$. If Π does not have any numeric variables ($\mathcal{V}_n = \emptyset$), we have a *classical* (FDR) *planning task* Π_{FDR}. Assignment of a valid value d to a variable $v \in \mathcal{V}$ is called a fact and denoted by $\langle v, d \rangle$. For a subset of variables $V \subseteq \mathcal{V}$ we define its joint domain to be $\mathcal{D}[V] = \times_{v \in V} \mathcal{D}(v)$. The state space is $\mathcal{S} = \mathcal{D}[\mathcal{V}]$. A state $s \in \mathcal{S}$ is a full assignment over all variables, and can be seen as a tuple $\langle s_p, s_n \rangle$, where $s_p \in \mathcal{D}[\mathcal{V}_p]$ and $s_n \in \mathcal{D}[\mathcal{V}_n]$. s_I is a state. Note that s, s_p, and s_n may be presented via vector representation, or as a set of facts $s = s_p \cup s_n$. In the latter case, there are no two facts that involve the same variable, and $|s| = |\mathcal{V}|$. The value of a variable v in s is given by $s[v] = d$, and is equivalent to $\langle v, d \rangle \in s$. A partial state is a subset $s^{pt} \subseteq s$ of some $s \in \mathcal{S}$.

A *linear expression* ξ has the form $\xi = \sum_{v \in V} w_v^\xi v + w_0^\xi$, where $V \subseteq \mathcal{V}_n$, with $\forall v \in V, w_v^\xi \in \mathbb{Q}$, and $w_0^\xi \in \mathbb{Q}$. The value of ξ in a state s is given by the expression $s[\xi] = \sum_{v \in V} w_v^\xi s[v] + w_0^\xi$. We assume that there is always a variable $v_0 \in \mathcal{V}_n$ s.t. $s[v_0] = 1$ for each state s. This assumption allows a more convenient expression $\xi = \sum_{v \in V} w_v^\xi v$. We assume that there are no redundant factors in the condition representation, i.e., for all $v \in V$ it holds $w_v^\xi \neq 0$. Ξ is the set of all linear expressions in Π. The set of all constants that appear in ξ is denoted by $nums(\xi)$, and the set of all variables that appear in ξ is denoted by $vars(\xi)$. By $vars(s^{pt}) \subseteq \mathcal{V}$ we denote the variables assigned in a partial state s^{pt}.

Conditions can be either propositional or numeric. A propositional condition ψ is a partial state over the variables $vars(\psi) \subseteq \mathcal{V}_p$. We say that ψ is satisfied by s, $s \models \psi$, if $\psi \subseteq s_p$. A numeric condition ψ has the form $\xi \trianglerighteq 0$, where ξ is a linear expression and $\trianglerighteq \in \{\geq, >\}$. We say that ψ is satisfied by s if $\xi[s] \trianglerighteq 0$ holds. The set of conditions Ψ is satisfied by s, if for each $\psi \in \Psi$ it holds that $s \models \psi$. The goal condition $G = G_p \cup G_n$ is a union over sets of propositional and numeric conditions, respectively.

Action $a = \langle \mathsf{pre}(a), \mathsf{eff}(a), \mathsf{cost}(a) \rangle \in \mathcal{A}$ has preconditions $\mathsf{pre}(a) = \mathsf{pre}_p(a) \cup \mathsf{pre}_n(a)$, effects $\mathsf{eff}(a) = \mathsf{eff}_p(a) \cup \mathsf{eff}_n(a)$, and cost $\mathsf{cost}(a) \in \mathbb{R}^{0+}$. $\mathsf{pre}_p(a)$ is a partial state over propositional variables and $\mathsf{pre}_n(a)$ is a set of linear numeric conditions. The set of all numeric conditions is denoted by Ψ_n, i.e., $\Psi_n = G_n \cup \bigcup_{a \in \mathcal{A}} \mathsf{pre}_n(a)$. a is applicable to s if $s \models \mathsf{pre}(a)$. Similarly, the propositional effect $\mathsf{eff}_p(a)$ is a partial state on a subset of $\mathcal{V}_p$. The effect $\mathsf{eff}_n(a)$ is a set of numeric effects of the form $(v \mathrel{+}= \xi)$, where $v \in \mathcal{V}_n$ and $\xi \in \Xi$. The assignment effect $v := \xi$ and subtractive effect $v \mathrel{-}= \xi$ are normalized to the additive forms $v \mathrel{+}= \xi - v$ and $v \mathrel{+}= -\xi$, and any action has at most one effect on a given numeric variable. The result of applying a in s is denoted by $s[\![a]\!] = s'_p \cup s'_n$, where the resulting state is defined as $s'_p[v] = \mathsf{eff}_p(a)[v]$ for $v \in vars(\mathsf{eff}_p(a))$, $s[\![a]\!][v] = s[v] + \xi[s]$ if $(v \mathrel{+}= \xi) \in \mathsf{eff}_n(a)$, and $s[\![a]\!][v] = s[v]$ otherwise. $\mathcal{A}(s)$ is the set of all actions applicable to s.

An *s-plan* is a finite action sequence π that can be applied successively in s and results in a goal state $s_* \models G$. A plan for Π is an s_I-plan. The *cost of an s-plan* π is the sum of all its action costs and an *optimal s-plan* has minimal cost among all possible s-plans.

A *state transition graph* is a labeled digraph $\mathcal{T}_\Pi = \langle \mathcal{S}, E \rangle$, whose vertices $\mathcal{S}$ are the states of Π, the set of labeled arcs $E = \{\langle s, s[\![a]\!]; a \rangle \mid s \in \mathcal{S}, a \in \mathcal{A}(s)\}$ is induced by the actions of Π, where $\mathsf{cost}(s, s[\![a]\!]; a) = \mathsf{cost}(a)$. A plan for Π is equivalent to a path from s_I to some s_* in $\mathcal{T}_\Pi$.

Structural Symmetries

This subsection defines the notion of *structural symmetries* (Shleyfman et al. 2015), which captures previously proposed concepts of symmetries in classical planning. In short, structural symmetries relabel a given planning task. Variables are mapped to variables, values to values (preserving the $\langle var, val \rangle$ structure), and actions are mapped to actions. In this work, we follow the definition of structural symmetries for FDR planning tasks as defined by Wehrle et al. (2015). For a planning task $\Pi_{FDR} = \langle \mathcal{V}, \mathcal{A}, I, G \rangle$, let P be the set of facts of Π, and let $P_\mathcal{V} := \{\{\langle v, d \rangle \mid d \in \mathcal{D}(v)\} \mid v \in \mathcal{V}\}$ be the set of sets of facts attributed to each variable in $\mathcal{V}$. We say that a permutation $\sigma : P \cup \mathcal{A} \to P \cup \mathcal{A}$ is a *structural symmetry* if the following holds:

1. $\sigma(P_\mathcal{V}) = P_\mathcal{V}$,
2. $\sigma(\mathcal{A}) = \mathcal{A}$, and, for all $a \in \mathcal{A}$, $\sigma(\mathsf{pre}(a)) = \mathsf{pre}(\sigma(a))$, $\sigma(\mathsf{eff}(a)) = \mathsf{eff}(\sigma(a))$, and $\mathsf{cost}(\sigma(a)) = \mathsf{cost}(a)$.
3. $\sigma(G) = G$.

We define the application of σ to a set X by $\sigma(X) := \{\sigma(x) \mid x \in X\}$, where σ is applied recursively up to the level of action labels and facts. This recursive definition allows us to distinguish between domain values of different variables, i.e., since σ is a permutation of $P_\mathcal{V}$, each set of facts $\{\langle v, d \rangle \mid d \in \mathcal{D}(v)\}$, that is associated with a given variable $v \in \mathcal{V}$ is mapped to a set of facts that is associated with some variable $v' \in \mathcal{V}$. Thus, we can write $\sigma(v) = v'$ and $\sigma(\mathcal{D}(v)) = \mathcal{D}(v')$. This slight abuse of notation allows us to define each σ as a permutation over the set $\mathcal{V} \cup \mathcal{A} \cup \left(\bigsqcup_{v \in \mathcal{V}} \mathcal{D}(v)\right)$, where $\sigma(\mathcal{V}) = \mathcal{V}$ and $\sigma(\mathcal{D}(v)) = \mathcal{D}(\sigma(v))$. Note that we use a disjoint union here,[1] since each value $d \in \mathcal{D}(v)$ is actually given by the ordered pair $\langle v, d \rangle$, and therefore it cannot be that $d \in \mathcal{D}(v)$, $d' \in \mathcal{D}(v)$, and $d = d'$ if $v \neq v'$.

Using this notation, we can apply σ to a partial state s. Since s can be represented as a set of facts, applying σ to s results in a partial state s', s.t. for all facts $\langle v, d \rangle \in s$ it holds that $\sigma(\langle v, d \rangle) = \langle \sigma(v), \sigma(d) \rangle \in \sigma(s)$.

The set of all structural symmetries of an FDR task is a finite set of bijections, thus, it is closed under composition (Herstein 1975). Therefore, structural symmetries form a group over the task Π_{FDR}, denoted by $Aut(\Pi_{FDR})$.

Symmetries and Problem Description Graphs

The problem description graph (PDG) was introduced by Pochter et al. (2011), and later reformulated by Domshlak

[1]Example of disjoint union: $\{5\} \sqcup \{5\} = \{\langle 5, 0 \rangle, \langle 5, 1 \rangle\}$.

et al. (2012) and Shleyfman et al. (2015).

Definition 1. *Let Π_{FDR} be a* FDR *planning task. The* ***problem description graph*** *$PDG_{\Pi_{FDR}}$ is the colored digraph $\langle N, E, \mathsf{col}\rangle$ with nodes*

$$N = N_{\mathcal{V}} \cup \bigcup_{v\in\mathcal{V}} N_{\mathcal{D}(v)} \cup N_{\mathcal{A}}$$

where $N_{\mathcal{V}} = \{n_v \mid v \in \mathcal{V}\}$, $N_{\mathcal{D}(v)} = \{n_v^d \mid d \in \mathcal{D}(v)\}$, and $N_{\mathcal{A}} = \{n_a \mid a \in \mathcal{A}\}$; node colors

$$\mathsf{col}(n) = \begin{cases} 0 & \text{if } n \in N_{\mathcal{V}}, \\ 1 & \text{if } n := n_v^d \in \bigcup_{v\in\mathcal{V}} N_{\mathcal{D}(v)}, \langle v, d\rangle \in G, \\ 2 & \text{if } n := n_v^d \in \bigcup_{v\in\mathcal{V}} N_{\mathcal{D}(v)}, \langle v, d\rangle \notin G, \\ 3 + \mathsf{cost}(a) & \text{if } n := n_a \in N_{\mathcal{A}}. \end{cases}$$

and edges

$$E = \bigcup_{v\in\mathcal{V}} E^v \cup \bigcup_{a\in\mathcal{A}} E_a^{\mathsf{pre}} \cup E_a^{\mathsf{eff}},$$

where $E^v = \{(n_v, n_v^d) \mid d \in \mathcal{D}(v)\}$, $E_a^{\mathsf{pre}} = \{(n_a, n_v^d) \mid \langle v, d\rangle \in \mathsf{pre}(a)\}$, and $E_a^{\mathsf{eff}} = \{(n_v^d, n_a) \mid \langle v, d\rangle \in \mathsf{eff}(a)\}\}$.

Pochter et al. observed that *PDG symmetry* is a symmetry of $\mathcal{T}_\Pi$ that is induced by a graph automorphism of the *PDG* of Π_{FDR}.[2] In what follows, we denote by $Aut(PDG(\Pi_{FDR}))$ the automorphism group of the PDG of the task Π_{FDR}. Shleyfman et al. (2015), in turn showed that every structural symmetry of Π_{FDR} corresponds to a PDG symmetry in the sense that they induce the same transition graph symmetry, i.e., the groups $Aut(PDG(\Pi_{FDR}))$ and $Aut(\Pi_{FDR})$ are isomorphic.

Group *homomorphism f* is a function between two groups $f : \Gamma \to \hat{\Gamma}$ that respects the group operations. i.e., given $\sigma, \sigma' \in \Gamma$ it holds $f(\sigma\circ\sigma') = f(\sigma)\circ f(\sigma')$. Isomorphism is a bijective (one-to-one and onto) homomorphism. The inverse of an isomorphsim is also an isomorphism.

Symmetries of the State Transition Graph

A *symmetry* of a transition graph $\mathcal{T}_\Pi = \langle \mathcal{S}, E\rangle$ with actions $\mathcal{A}$ is a permutation σ of $\mathcal{S} \cup \mathcal{A}$ mapping states to states and actions to actions s.t.

– $\langle s, s'; a\rangle \in E$ iff $\langle \sigma(s), \sigma(s'); \sigma(a)\rangle \in E$,
– $\mathsf{cost}(\sigma(a)) = \mathsf{cost}(a)$, and
– s is a goal state iff $\sigma(s)$ is a goal state

for all states s, s' and actions a. Symmetries are also called *(goal-stable) automorphisms*. They are closed under composition and inverse, forming the *automorphism group* $Aut(\mathcal{T}_\Pi)$ of the transition graph. Each subgroup Γ of symmetries induces an equivalence relation $\sim_\Gamma$ on states $\mathcal{S}$: $s \sim_\Gamma s'$ iff $\sigma(s) = s'$ for some $\sigma \in \Gamma$. States in the same equivalence class are called *symmetric*.

The following (immediate) result is the formal basis for exploiting symmetries for planning.

Theorem 1. *Let Π be a planning task, let s be one of its states, let π be a sequence of actions of Π, and let σ be a symmetry of $\mathcal{T}_\Pi$. Then π is a plan for s iff $\sigma(\pi)$ is a plan for $\sigma(s)$, and the two plans have the same cost.*

[2]The formal proof can be found in Shleyfman (2020).

Note that the definition of symmetries of a state transition graph depends only on the notion of actions applicable to states, $s' = s[\![a]\!]$, and the notion of goal state, thus it fits multiple formalisms that support this dynamic. Particularly, this definition matches not only the transition graph induced by an FDR task but also the one induced by an LT.

A color preserving graph automorphism of *PDG*$_{\Pi_{FDR}}$ induces a symmetry of $\mathcal{T}_{\Pi_{FDR}}$. Moreover, the group $Aut(PDG(\Pi_{FDR}))$ is isomorphic to $Aut(\Pi_{FDR})$ and induces a subgroup of $Aut(\mathcal{T}_{\Pi_{FDR}})$, which in turn defines an equivalence relation over the states $\mathcal{S}$ of Π_{FDR}. In the following section we define a numeric version of PDG and structural symmetries that obey the same relation, i.e., induce a symmetry group of the state transition graph.

Symmetries in Numeric Domains

Shleyfman and Jonnson (2021) show that determining whether two states are symmetric in a transition system induced by an FDR planning task is PSPACE-hard. This result automatically grants PSPACE-hardness for the same problem in LT, since it, trivially, contains the FDR formalism. More troublesome is the fact that the presence of numeric fluents makes the search space infinite, which may lead to a lot of unpleasant properties of its symmetry group. I.e., the group of all permutations of a countable set contains an uncountable number of symmetries, infinitely many of which have an infinite order.

Since identifying the whole symmetry group of the transition task is infeasible, we would like to obtain a manageable subgroup and use it for symmetry breaking in the search over the state space. To this end, we extend the Structural Symmetries and the *PDG* by numeric fluents. Then, we show that Numeric Structural Symmetries (NSS) can be mapped into symmetries of the state transition graph (Thm. 2), and that the symmetry group of the numeric version of *PDG* is isomorphic to the NSS group of the task (Thm. 3). For the flowchart of the proof of the grounding process see Figure 1.

We exploit the grounded, effectively computed subgroup Γ_Π of $Aut(\mathcal{T}_\Pi)$ by plugging it into the symmetry breaking searches DKS (Domshlak, Katz, and Shleyfman 2012) and OSS (Domshlak, Katz, and Shleyfman 2015).

Let us give here a motivational example. Suppose we have a planning task where homogeneous trucks deliver various cargo across a city, where the city map is represented via digraph. A truck T in this task is represented with three variables: $loc(T)$ the locations of the truck (finite domain variable), $fuel(T)$ the amount of fuel in the truck, and $load(T)$ the current load of the truck (both numeric variables). Since the trucks are homogeneous, we would like to have a symmetry σ to capture this information, i.e., given two trucks T_1 and T_2 a map that replaces only the labels of the trucks should induce an automorphism of the transition graph. In what follow we would like to capture this behavior. Instead of saying that T_1 is symmetric to T_2, we want a bijection that switches between the variables associated with T_1 and T_2, as follows: $\sigma(loc(T_1)) = loc(T_2)$, $\sigma(fuel(T_1)) = fuel(T_2)$, $\sigma(load(T_1)) = load(T_2)$, and vice versa.

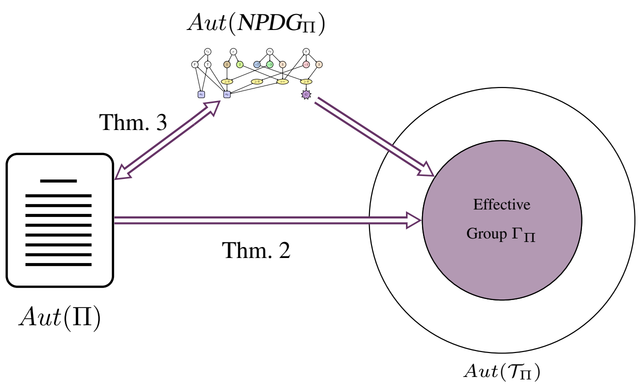

Figure 1: Schematic representation of detection and grounding of Numeric Structural Symmetries (NSS) for linear numeric planning. Thm. 3 shows that the symmetry group of $NPDG_\Pi$ and the group of NSS are the same. Thm. 2 shows that there is a natural injection from NSS to the symmetry group of the transition system induced by Π.

Numeric Structural Symmetries

For an LT $\Pi = \langle \mathcal{V}, \mathcal{A}, I, G\rangle$. We say that a permutation σ over $\mathcal{V} \cup \mathcal{A} \cup \left(\bigsqcup_{v\in\mathcal{V}_p} \mathcal{D}(v)\right)$ is a *numeric structural symmetry (NSS)* if the following **five** conditions hold:

1. $\sigma(\mathcal{V}_p) = \mathcal{V}_p$, $\sigma(\mathcal{V}_n) = \mathcal{V}_n$,
2. for all $v \in \mathcal{V}_p$, $\sigma(\mathcal{D}(v)) = \mathcal{D}(\sigma(v))$,
3. $\sigma(\mathcal{A}) = \mathcal{A}$.

We define an application of σ to a partial state s^{pt} over the propositional variables $vars(s^{pt})$, as $\sigma(s^{pt})[\sigma(v)] := \sigma(s^{pt}[v])$ for all $v \in vars(s^{pt})$. Note that by points 1 and 2 the result of this application, $\sigma(s^{pt})$, is also a partial state.

Let $\xi = \sum_{v\in V} w_v^\xi v$ be a linear formula over the variables $V \subseteq \mathcal{V}_n$. We define the application of σ to ξ to be $\sigma(\xi) = \sum_{v\in V} w_v^\xi \sigma(v)$. The application of σ to condition $\xi \trianglerighteq 0$ is defined as $\sigma(\xi) \trianglerighteq 0$. The application of σ to numeric effect $v \mathrel{+}= \xi \in \mathsf{eff}_n(a)$ is written as $\sigma(v) \mathrel{+}= \sigma(\xi)$, where $v \in \mathcal{V}_n$. Using this notation we establish the following conditions on σ.

4. for all $a \in \mathcal{A}$, $\sigma(\mathsf{pre}_n(a)) = \mathsf{pre}_n(\sigma(a))$, $\sigma(\mathsf{pre}_p(a)) = \mathsf{pre}_p(\sigma(a))$, $\sigma(\mathsf{eff}_p(a)) = \mathsf{eff}_p(\sigma(a))$, $\sigma(\mathsf{eff}_n(a)) = \mathsf{eff}_n(\sigma(a))$, and $\mathsf{cost}(\sigma(a)) = \mathsf{cost}(a)$.
5. $\sigma(G_n) = G_n$ and $\sigma(G_p) = G_p$.

An immediate consequence of this definition is that for any structural symmetry σ, it holds that $\sigma(\Xi) = \Xi$.

Note also that that for two structural symmetries σ_1, σ_2 the composition $\sigma_1 \circ \sigma_2$ is also a structural symmetry.

- This is straightforward for properties 1, 3, and 5: $\sigma_1 \circ \sigma_2(X) = \sigma_1(\sigma_2(X)) = \sigma_1(X) = X$ where $X \in \{\mathcal{V}_n, \mathcal{V}_p, \mathcal{A}, G_n, G_p\}$.
- Property 4 holds for each $a \in \mathcal{A}$ since for each $h \in \{\mathsf{pre}_n, \mathsf{pre}_p, \mathsf{eff}_n, \mathsf{eff}_p,\}$ we have $\sigma_1 \circ \sigma_2(h(a)) = \sigma_1(\sigma_2(h(a))) = \sigma_1(h(\sigma_2(a))) = h(\sigma_1(\sigma_2(a))) = h(\sigma_1 \circ \sigma_2(a))$. By replacing a by $v \in \mathcal{V}_p$ and h by $\mathcal{D}$, we get property 2. The check for cost is trivial and repeats the previous point.

A well-known result in group theory (Herstein 1975) states that a finite set closed under a set of bijective self-maps forms a group. We denote the group of all structural symmetries as $Aut(\Pi)$, which leads us to the observation below.

Theorem 2. *There is a natural injection from $Aut(\Pi)$ to $Aut(\mathcal{T}_\Pi)$, i.e., for each $\sigma \in Aut(\Pi)$ there is a unique $\tilde{\sigma} \in Aut(\mathcal{T}_\Pi)$.*

Proof. Let $\mathcal{T}_\Pi = \langle \mathcal{S}, E\rangle$ be a transition system induced by a LT $\Pi = \langle \mathcal{V}, \mathcal{A}, s_I, G\rangle$, and let $\sigma \in Aut(\Pi)$ be a structural symmetry. For a state $s = s_p \cup s_n \in \mathcal{S}$ we define the application of $\tilde{\sigma}$ to s as $\tilde{\sigma}(s_p \cup s_n) = \tilde{\sigma}(s_p) \cup \tilde{\sigma}(s_n)$, where

$$\tilde{\sigma}(s_p) = \{\sigma(\langle v, d_v\rangle) \mid \langle v, d_v\rangle \in s_p\} = \{\langle \sigma(v), \sigma(d_v)\rangle \mid \langle v, d_v\rangle \in s_p\},$$

$\tilde{\sigma}(s_n) = \{\langle \sigma(v), q_v\rangle \mid \langle v, q_v\rangle \in s_n\}$ and $\tilde{\sigma}(a) = \sigma(a)$ for $a \in \mathcal{A}$. By properties 1-3, we have that $\tilde{\sigma}$ is a permutation over $\mathcal{S} \cup \mathcal{A}$, and the map $\sigma \mapsto \tilde{\sigma}$ is injective.

Let $\langle s, s'; a\rangle \in E$, and let σ be a structural symmetry. We aim to show that $\langle \tilde{\sigma}(s), \tilde{\sigma}(s'); \tilde{\sigma}(a)\rangle \in E$. Let us decouple the states s and s' to their propositional and numeric components $s_p \cup s_n$ and $s'_p \cup s'_n$, respectively. The proof of the propositional part was already done by Shleyfman et al. (2020), but we present it here for the sake of completeness. By definition of $\tilde{\sigma}$ we have $\tilde{\sigma}(s)_p = \sigma(s_p)$. Let ψ be a partial assignment over the propositional variables s.t. $s_p \models \psi$; set wise it can be written as $\psi \subseteq s_p$. By applying σ to both sides we have $\sigma(s_p) \models \sigma(\psi)$, thus $\tilde{\sigma}(s) \models \sigma(\psi)$. This claim grants us: $s \models \mathsf{pre}_p(a)$ implies that $\tilde{\sigma}(s) \models \mathsf{pre}_p(\tilde{\sigma}(a))$, and $s' \models \mathsf{eff}_p(a)$ implies that $\tilde{\sigma}(s') \models \mathsf{eff}_p(\tilde{\sigma}(a))$. Since σ is a permutation over $\mathcal{V}_p$, its application to the unaffected variables $\mathcal{V}_p \setminus vars(\mathsf{eff}_p(a))$ can be written as $\mathcal{V}_p \setminus vars(\mathsf{eff}_p(\tilde{\sigma}(a)))$, and since $s'[v] = s[v]$ for each $v \in \mathcal{V}_p \setminus vars(\mathsf{eff}_p(a))$ we have that $\tilde{\sigma}(s')[v] = \tilde{\sigma}(s)[v]$ for each $v \in \mathcal{V}_p \setminus vars(\mathsf{eff}_p(\tilde{\sigma}(a)))$, resulting in $\tilde{\sigma}(s)[\![\tilde{\sigma}(a)]\!]_p = \tilde{\sigma}(s')_p$. Note, that the fact that for any partial assignment ψ, it holds $s_p \models \psi \iff \tilde{\sigma}(s)_p \models \sigma(\psi)$ which directly implies that $s_p \models G_p \iff \tilde{\sigma}(s)_p \models G_p$.

We aim to show that $\tilde{\sigma}(s)[\![\tilde{\sigma}(a)]\!]_n = \tilde{\sigma}(s')_n$. Let $\psi \in \Psi_n$ be a numeric condition $\psi : \sum_{v\in V} v \cdot w_v^\psi \trianglerighteq 0$ s.t. $s_n \models \psi$. By definition of $\tilde{\sigma}$ we have $s[v] = \tilde{\sigma}(s)[\sigma(v)]$, thus

$$\sum_{v\in V} s[v] \cdot w_v^\psi = \sum_{v\in V} \tilde{\sigma}(s)[\sigma(v)] \cdot w_v^\psi \trianglerighteq 0 \text{ implies that}$$
$$s_n \models \psi \iff \tilde{\sigma}(s)_n \models \sigma(\psi).$$

This statement combined with the previous paragraph has two immediate consequences:
1. $s \models G \iff \tilde{\sigma}(s) \models G$, and
2. $s \models \mathsf{pre}(a) \iff \tilde{\sigma}(s) \models \mathsf{pre}(\sigma(a))$.

Now, to show that $\langle \tilde{\sigma}(s), \tilde{\sigma}(s'); \tilde{\sigma}(a)\rangle \in E$, it is enough ensure that for any $v \in \mathcal{V}_n$ it holds that $s'[v] = \tilde{\sigma}(s')[\sigma(v)]$. Here we have two cases, either v is affected by $\mathsf{eff}_n(a)$ or not. If v is not affected, i.e., $v \notin vars(\mathsf{eff}_n(a))$, then since σ is a permutation it holds that $\sigma(v)$ is not affected by $\mathsf{eff}_n(\sigma(a))$, i.e., $\sigma(v) \notin vars(\mathsf{eff}_n(\sigma(a)))$. Thus, $s'[v] =$

$s[v] = \tilde{\sigma}(s)[\sigma(v)] = \tilde{\sigma}(s')[\sigma(v)]$. Otherwise, let $v \mathrel{+}= \xi$ be the numeric effect of a on v. Let $\xi = \sum_{v' \in V} w^\xi_v v$. By definition we have that $\sigma(v) \mathrel{+}= \sigma(\xi) \in \mathsf{eff}_n(\sigma(a))$. Thus,

$$s'[v] = s[v] + \sum_{v' \in V} w^\xi_v s[v'] = \tilde{\sigma}(s)[\sigma(v)] + \sum_{v' \in V} w^\xi_v \tilde{\sigma}(s)[\sigma(v')] = \tilde{\sigma}(s')[\sigma(v)],$$

since $s[v] = \tilde{\sigma}(s)[\sigma(v)]$ for each $v \in \mathcal{V}_n$ by definition of $\tilde{\sigma}$.

Hence, we have that $\tilde{\sigma}$ is a symmetry of the transition graph $\mathcal{T}_\Pi$, since it satisfies the following three requirements:
- $\langle s, s'; a\rangle \in E$ iff $\langle \tilde{\sigma}(s), \tilde{\sigma}(s'); \tilde{\sigma}(a)\rangle \in E$,
- $\mathsf{cost}(\tilde{\sigma}(a)) = \mathsf{cost}(\sigma(a)) = \mathsf{cost}(a)$, and
- s is a goal state iff $\tilde{\sigma}(s)$ is a goal state. □

This result establishes that structural symmetries induce transition graph symmetries. Next, we show how to compute these symmetries using a variation of the problem description graph modified for numerical planning.

Numeric Problem Description Graph

Usually, the state transition graph $\mathcal{T}_\Pi$ of a planning task Π is too large for explicit representation. Thus, we must deduce its symmetries from a compact description.

Pochter et al. introduced a way for inferring a subgroup of symmetries of the transition system of Π using the *problem description graph* (PDG). Using PDG one can compute this subgroup using off-the-shelf tools for discovering the full automorphism group of a given colored graph, such as *bliss* (Junttila and Kaski 2007). Later, Domshlak et al. (2012) modified the definition, mainly to add support action costs.

In contrast to the propositional FDR, the transition system defined by its numeric counterpart may be infinite. We group numeric elements of conditions and effects into a set of linear formulas. Recall, that Ξ denotes the set of all linear formulas that appear in Π (both in conditions and in effects). For each $v \in \mathcal{V}_n$, we define $\mathcal{W}(v) = \{w^\xi_v \in \mathbb{Q} \mid \exists \xi \in \Xi : w^\xi_v \in nums(\xi)\}$, the set of all numeric coefficients associated with v. Let $\mathcal{C}_\Pi$ be the set of unique costs of actions and constants in numeric variables in the task Π, i.e.,

$$\mathcal{C}_\Pi = \{\mathsf{cost}(a) \mid a \in \mathcal{A}\} \sqcup \bigcup_{v \in \mathcal{V}_n} \mathcal{W}(v).$$

We define a function $\mathsf{ord} : \mathcal{C}_\Pi \to [|\mathcal{C}_\Pi|]$ to be an order-preserving enumeration of elements in $\mathcal{C}_\Pi$, i.e., for $c_1, c_2 \in \mathcal{C}_\Pi$ holds that $c_1 \le c_2$ implies that $\mathsf{ord}(c_1) \le \mathsf{ord}(c_2)$. The properties of the function ord that we are interested in are

1. for $c_1, c_2 \in \mathcal{C}_\Pi$ it holds $c_1 = c_2$ iff $\mathsf{ord}(c_1) = \mathsf{ord}(c_2)$,
2. for each $c \in \mathcal{C}_\Pi$ it holds that $\mathsf{ord}(c) \le |\mathcal{C}_\Pi|$.

This function allows us to distinguish between the task constants while operating with a reasonable number of colors.

Definition 2. *Let $\Pi = \langle \mathcal{V}, \mathcal{A}, s_I, G\rangle$ be a* LT. *The* ***numeric problem description graph*** *($NPDG_\Pi$) of Π is the colored digraph $\langle N, E, \mathsf{col}\rangle$ with nodes (e.g., Figure 2)*

$$N = N_{\mathcal{V}_p} \cup N_{\mathcal{V}_n} \cup N_{\mathcal{A}} \cup N_\Xi \cup \{n_G\} \cup \bigcup_{v \in \mathcal{V}} N_{\mathcal{D}(v)}, \text{ where}$$

$N_{\mathcal{V}_x} = \{n_v \mid v \in \mathcal{V}_x\}$ *for* $x \in \{p, n\}$,
$N_{\mathcal{A}} = \{n_a \mid a \in \mathcal{A}\}$,
$\forall v \in \mathcal{V}_p : N_{\mathcal{D}(v)} = \{n^v_d \mid d \in \mathcal{D}(v)\}$,
$\forall v \in \mathcal{V}_n : N_{\mathcal{D}(v)} = \{n^v_w \mid w \in \mathcal{W}(v)\}$,
$N_{\Psi_n} = \{n^{\trianglerighteq}_\xi \mid \psi : \xi \trianglerighteq 0 \in \Psi_n\}$ *and*
$N_+ = \{n^+_\xi \mid \exists \xi \in \Xi, a \in \mathcal{A} : v \mathrel{+}= \xi \in \mathsf{eff}_n(a)\}$,
with $N_\Xi = N_{\Psi_n} \cup N_+$, *edges*

$$E = E_{\mathcal{V}_p} \cup E_{\mathcal{V}_n} \cup E_\Xi \cup E_{\mathcal{A}} \cup E_G, \text{ where}$$

$E_{\mathcal{V}_p} = \bigcup_{v \in \mathcal{V}_p} \{(n_v, n^v_d) \mid d \in \mathcal{D}(v)\}$,
$E_{\mathcal{V}_n} = \bigcup_{v \in \mathcal{V}_n} \{(n_v, n^v_w) \mid w \in \mathcal{W}(v)\}$,
$E_\Xi = \bigcup_{n_\xi \in N_\Xi} \{(n^v_{w^\xi_v}, n_\xi) \mid v \in \mathcal{V}_n, w^\xi_v \in nums(\xi)\}$,
$E_{\mathcal{A}} = \bigcup_{a \in \mathcal{A}} \left(E^{\mathsf{pre}_n}_a \cup E^{\mathsf{pre}_p}_a \cup E^{\mathsf{eff}_p}_a \cup E^{\mathsf{eff}_n}_a\right)$, *with*

$$\begin{aligned} E^{\mathsf{pre}_n}_a &= \{(n^{\trianglerighteq}_\xi, n_a) \mid \psi : \xi \trianglerighteq 0 \in \mathsf{pre}_n(a)\}, \\ E^{\mathsf{pre}_p}_a &= \{(n^v_d, n_a) \mid \langle v, d\rangle \in \mathsf{pre}_p(a)\}, \\ E^{\mathsf{eff}_n}_a &= \{(n_a, n^+_\xi), (n^+_\xi, n_v) \mid v \mathrel{+}= \xi \in \mathsf{eff}_n(a)\}, \\ E^{\mathsf{eff}_p}_a &= \{(n_a, n^v_d) \mid \langle v, d\rangle \in \mathsf{eff}_p(a)\}, \end{aligned}$$

$E_G = \{(n^{\trianglerighteq}_\xi, n_G) \mid \xi \trianglerighteq 0 \in G_n\} \cup \{(n^d_v, n_G) \mid \langle v, d\rangle \in G_p\}$,
and node colors

$$\mathsf{col}(n) = \begin{cases} \mathsf{ord}(\mathsf{cost}(a)) & \text{if } n := n_a \in N_{\mathcal{A}}, \\ \mathsf{ord}(w) & \text{if } n := n^x_w \in \bigcup_{v \in \mathcal{V}_n} N_{\mathcal{D}(v)}, \\ |\mathcal{C}_\Pi| + \mathsf{rel}(\dagger) & \text{if } n := n^\dagger_\xi \in N_\Xi, \\ |\mathcal{C}_\Pi| + 4 & \text{if } n := n_G, \\ 0 & \text{otherwise}, \end{cases}$$

where $\mathsf{rel} : \{\ge, >, +\} \to [3]$ *is a bijection.*[3]

Theorem 3. *$Aut(NPDG_\Pi)$ and $Aut(\Pi)$ are isomorphic.*

Proof Sketch: Let Π be a numeric planning task, with the corresponding $NPDG_\Pi = \langle N, E, \mathsf{col}\rangle$. We aim to show that there is a bijective map $f : Aut(NPDG_\Pi) \to Aut(\Pi)$.

Let $f(\alpha) = l^{-1} \circ \alpha \circ l \in Aut(\Pi)$, where the map

$$l : \mathcal{V} \cup \mathcal{A} \cup \left(\bigsqcup_{v \in \mathcal{V}_p} \mathcal{D}(v)\right) \to N$$

is given by a bijection $l(x) \mapsto n_x$, with a slight abuse of notation $l(\langle v, d\rangle) = n^v_d$. Note that f is a homomorphism, i.e., $f(\alpha) \circ f(\beta) = l^{-1} \circ \alpha \circ l \circ l^{-1} \circ \beta \circ l = f(\alpha \circ \beta)$.

We need to show that f is **well-defined**, **injective**, and **surjective**. f is **well-defined** if $f(\alpha) \in Aut(\Pi)$, i.e., $f(\alpha)$ is an NSS. Since $\alpha \in Aut(NPDG_\Pi)$ is color-, edge- and degree-preserving we have that the vertex sets $N_{\mathcal{V}_p}$,$N_{\mathcal{V}_n}$,$N_{\mathcal{A}}$, N_{Ψ_n},N_+, $\bigcup_{v \in \mathcal{V}_p} N_{\mathcal{D}(v)}$, $\bigcup_{v \in \mathcal{V}_n} N_{\mathcal{D}(v)}$, and $\{n_G\}$ are all preserved under α. The properties 1-5 hold due to the structure of the $NPDG_\Pi$.

For the **injectivity** of f, it is enough to show that $f(\alpha) = \mathrm{id}_\Pi$ implies $\alpha = \mathrm{id}_{NPDG_\Pi}$. Thus, assume that there is $n \in$

[3]We use the notation $n := n^y_x$ to indicate the type of the graph node in question.

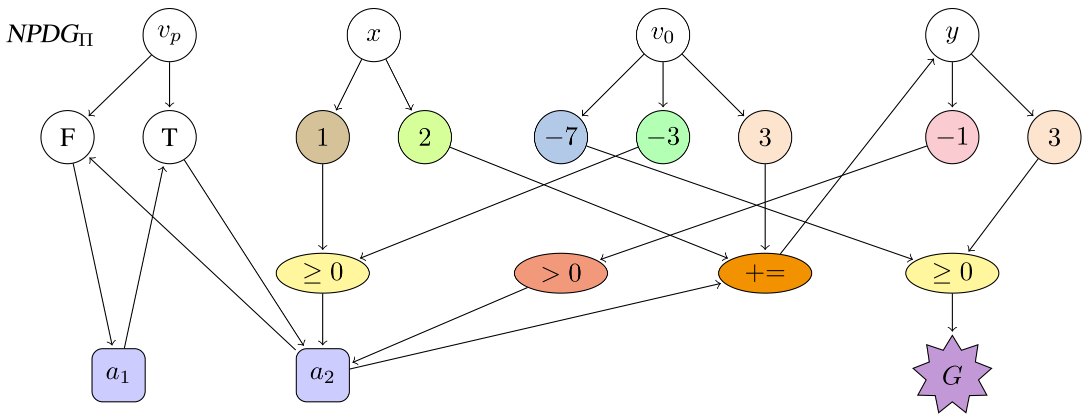

Figure 2: Toy example of a *NPDG*$_\Pi$ graph of a task Π, with the variable $\mathcal{V}_p = \{v_p\}$ and $\mathcal{V}_n = \{x, y\}$, actions $\mathcal{A} = \{a_1, a_2\}$, and the goal condition $G = \{3y \geq 7\}$. The actions a_1 and a_2 have the following form $\mathsf{pre}(a_1) = \{\langle v_p = F\rangle\}$, $\mathsf{pre}(a_2) = \{\langle v_p, T\rangle, x \geq 3, y < 0\}$, $\mathsf{eff}(a_1) = \{\langle v_p, T\rangle\}$, $\mathsf{eff}(a_2) = \{\langle v_p, F\rangle, y \mathrel{+}= 2x + 3\}$, and $\mathsf{cost}(a_1) = \mathsf{cost}(a_2) = 1$. The initial state is not given, since it is not involved in the construction of *NPDG*.

N s.t. $\alpha(n) \neq n$, and use the color- and edge-preservation properties of the authomorphism α to show contradiction.

To show that f is **surjective**, we needs to prove that $f^{-1}(\sigma) \in Aut(\textit{NPDG}_\Pi)$, for any NSS σ. Define the map:

$$f^{-1}(\sigma)(n) = \begin{cases} n_{\sigma(x)} & \text{if } n = n_x \in N_{\mathcal{V}_p} \cup N_{\mathcal{V}_n} \cup N_{\mathcal{A}} \\ n_{\sigma(\langle v,d\rangle)} & \text{if } n = n^v_d = n_{\langle v,d\rangle} \in \bigcup_{v\in\mathcal{V}_p}, \\ n_w^{\sigma(v)} & \text{if } n = n^v_w \in \bigcup_{v\in\mathcal{V}_n}, \\ n^\dagger_{\sigma(\xi)} & \text{if } n = n^\dagger_\xi \in N_\Xi, \\ n_G & \text{if } n = n_G. \end{cases}$$

The extended function $f^{-1}(\sigma)$ is well-defined. By definition σ is a permutation on variables, actions, and $\sigma(\langle v, d\rangle) \in \mathcal{D}(\sigma(v))$. Furthermore, the NSS preserves linear conditions and effects. By construction, f^{-1} is a homomorphism. Thus, all is left is to show that $\alpha := f^{-1}(\sigma)$ is a color-preserving automorphism. The full details of the proof can be found in the Supplementary Materials (Shleyfman et al. 2023). □

To summarise, we proved that the groups $Aut(\textit{NPDG}_\Pi)$ and $Aut(\Pi)$ are isomorphic, i.e., structurally identical, and that there is an injection that maps these groups into Γ_Π – a subgroup of $Aut(\mathcal{T}_\Pi)$. The generators of Γ_Π are used by DKS and OSS for symmetry breaking. The generators of $Aut(\textit{NPDG}_\Pi)$ allow us compute the generators of Γ_Π in linear time. Next, we discuss the computational complexity of computing the generators of $Aut(\textit{NPDG}_\Pi)$.

Computational Complexity

The *graph isomorphism* (GI) decision problem gets as input two finite graphs and determines if these graphs are isomorphic. This problem is neither known to be NP-complete nor tractable. Since many related problems appeared to be polynomial-time equivalent to the GI problem (Mathon 1979), it gave its name to a complexity class. As usual for complexity classes within the polynomial-time hierarchy, a problem X is called GI-hard if there is a polynomial-time reduction from GI to X. A problem X lies in GI if it can be reduced to the GI problem. X is GI-complete, i.e., polynomial-time equivalent, if it both lies in GI and GI-hard.

Shleyfman (2019) showed that computing the generators for the groups of structural symmetries and the automorphisms of *PDG* is GI-hard. Since numeric planning contains FDR in the trivial case when $\mathcal{V}_n = \emptyset$, and in this case *NPDG* and *PDG* are equivalent, computing the *automorphism group of NPDG and NSS is also GI-hard.*

Shleyfman and Jonsson (2021) proved that computing the generators for the automorphism group of a colored digraph is GI-complete. Since *NPDG* is a colored graph, the problem of computing generators of its automorphism group lies in GI. Thus, we have that *the computation of generators of NPDG and NSS is GI-complete*. Note that the linearity of the tasks is required only to produce a canonical representation of the numeric formulas. A more generic formalism that relies on a finite set of relations can be used instead of LT, expanding the scope of symmetry breaking to beyond linear numeric planning. The construction of such a formalism and its integration into the framework presented here is an interesting direction for future work.

While the GI problem is suspected to have a quasi-polynomial computational complexity (Babai 2016), there are some off-the-shelf symmetry detection packages such as *bliss* (Junttila and Kaski 2007) or *Saucy* (Darga, Sakallah, and Markov 2008), that while being a worst-case exponential in time, perform well in practice. For the experimental evaluation of the method proposed in this paper, we chose *bliss*.

Forward Search and Symmetry Breaking

Having discovered a generating set of the group $Aut(\Pi)$, the next step is to determine whether two states are symmetric or not. Unfortunately, this problem is NP-hard (Luks 1991). Pochter et al. suggested a heuristic approach that may produce false negatives in an attempt to determine whether two states are symmetric or not. They construct a procedural mapping $\Sigma : \mathcal{S} \to \mathcal{S}$ from states to states, that is implemented via a heuristic local search in a space with states being our planning task states $\mathcal{S}$, actions corresponding to the generators of $Aut(\Pi)$, and state evaluation being based on a lexicographic ordering of $\mathcal{S}$. Each local minimum in this space defines a canonical state (CS), i.e., two states s and s' are symmetric if they have the same CS, that is $\Sigma(s) = \Sigma(s')$. When two symmetric states have different CSs, the search does not discover the symmetry. For the numeric case we construct Σ using the same lexicographic order, where propositional variables precede the numeric ones.

Both A^*-based symmetry breaking algorithms DKS and OSS rely on this procedure to discover symmetrical states. For further details see Domshalk et al. (2015).

Experimental Evaluation

We implement the symmetry detection method in Numeric Fast Downward (NFD) (Aldinger and Nebel 2017). All experiments are run on an Intel Xeon Gold 6148 processor with a 30-minutes time limit and 4 GB memory limit using GNU parallel (Tange 2011). We evaluate A^*, DKS, and OSS using the blind heuristic, operator-counting heuristic with LM-cut and state equations constraints, $h_{\text{LP}}^{\text{LM-cut, SEQ}}$ (Kuroiwa et al. 2022), for SCT domains, which is a subset of LT, and LM-cut heuristic adapted for LT planning, $h_2^{\text{LM-cut}}$ (Kuroiwa, Shleyfman, and Beck 2022), for LT domains. When computing a CS, the lexicographic order of the variables follows the order used by NFD, where propositional variables are ordered according to the causal graph. In NPDG, we normalize a linear formula ξ by scaling coefficients so that $|w_v^\xi| = 1$ where v is the first variable in $vars(\xi)$ in the lexicographic order. We only show instances where symmetries are detected, since the overhead of detecting no symmetries is usually less than 5 sec.

OSS vs. DKS One of the curious behaviors that can be spotted in Table 1, is that OSS solves as many problems or more as DKS in all domains. Which is not the case on the classical domains, where the algorithms performed on average on par. We attribute this difference to the fact that DKS stores both original and canonical states, while OSS operates only on CSs, and thus requires less memory. The states of DKS are twice as large as those used by A^* and OSS. This difference is amplified in the numeric setting, since numeric states require significantly more memory.

Propositional vs. Numeric Symmetries Note that structural symmetries used for classical planning are not enough to detect symmetries in numeric planning tasks. Consider a logistics problem where trucks with capacity limits deliver weighted packages to some given locations. Two trucks with different capacities and no goal locations would be considered symmetric in terms of propositional symmetries, but not in terms of numeric ones. Therefore, to measure the relative impact of numeric symmetries, we construct two NPDGs with different colorings. For numeric symmetries, we use the standard NPDG defined above (*NPDG*$^{\text{full}}$). To detect strictly propositional symmetries, we fix all nodes in $N_{\mathcal{V}_n}$ by coloring each of them in a unique color (*NPDG*$^{\text{prop}}$).

Almost everywhere the searches that use the full symmetry group outperform the ones that use the propositional subgroup. The interesting domains where this is not the case are SAILING-SAT and TPP-METRIC. Both these domains have no propositional symmetries. A preliminary study shows that the overhead in these domains comes from the symmetries between different connected components of the transition graphs. Thus, DKS and OSS do not benefit from state pruning, but still, have some overhead while determining whether two states are symmetric or not.

New Domains Symmetries in classical planning vary between domains, and where the GRIPPER has a group that is exponential in the number of balls in the rooms, the BLOCKSWORLD domain has no symmetries at all due to an order imposed on the blocks in the goal of a task. The large increase in coverage shown by symmetry-breaking techniques can be attributed to the vast variety of classical planning domains. That was partially a byproduct of the International Planning Competitions (IPC) that were held during the last three decades. In addition to domains in the numeric tracks of IPC and previous work (Scala et al. 2020; Li et al. 2018; Leofante et al. 2020), we introduce six new domains.

As a sanity check, we performed some preliminary experiments on the GRIPPER domain, where the balls have homogeneous weights and the gripper-robot has a maximum load. The numeric approaches were consistent with the classical ones. From the 20 GRIPPER problems A^*, DKS, OSS equipped with blind heuristics solved $7, 20, 20$, respectively. The same algorithms equipped with numeric LM-cut solved $6, 20, 20$, in the same order. These results are in line with these reported by Domshlak et al. for classical planning. These results are not included in Table 1, since the domain is too simplistic. Instead, we introduce a new domain DELIVERY, where multiple robots, equipped with multiple arms, and a tray deliver objects on a map, represented by a digraph.

We introduce a DEPOTS-SYM domain, a variation of the previously existing DEPOTS SCT domains, where the weights of the packages and the capacities of the trucks are "standardized" by replacing what seems to be arbitrarily chosen numbers by some standard upper bounds. For example, the packages were all given weights in the set $\{4, 6, 8, 10, 12\}$, while the truck got capacities in the set $\{10, 20, 40\}$. This change allowed us to obtain a more symmetric domain while still having a meaningful numeric behavior. One may even argue that DEPOTS-SYM is closer to reality than DEPOTS due to standardization.

CVRP is the capacitated vehicle routing problem studied in operations research, where a set of n customers are visited by k vehicles with capacity q. In our PDDL domain, for each vehicle x, we have a numeric variable l_x representing the load. An action visits customer j using vehicle x from customer i with cost c_{ij} and increases l_x by the demand d_j, which is applicable if $l_x + d_j \leq q$. We generate 20 instances following the method used by Uchoa et al. (2017) using uniform distributions for coordinates and demands of customers with $n = 9, 10, 11, 12, 13$ and $r = 2, 3, 4, 5$, where r is a parameter to control k s.t. $k \simeq \frac{n}{r}$.

PETRINET is the reachability problem for Petri nets: whether from the given initial configuration there exists a sequence of valid execution steps that reaches the given fi-

	A^*			DKS (*NPDG*$^{\text{prop}}$)			DKS			OSS (*NPDG*$^{\text{prop}}$)			OSS		
	c.	t.	#gen.	c.	t.	#gen.	c.	t.	#gen.	c.	t.	#gen.	c.	t.	#gen.
SCT							Blind								
GARDENING (63)	63	4.0	1205997	63	7.6	1205997	63	4.3	**694083**	63	4.0	1205997	63	**2.3**	**694083**
GARDENING-SAT (51)	10	61.0	12693663	10	113.6	12546397	**11**	72.3	**7871534**	10	60.8	12546397	**11**	**38.3**	7897880
ROVER (19)	4	5.3	1773861	4	4.3	**764752**	4	4.2	**764752**	4	**2.4**	**764752**	4	**2.4**	**764752**
SAILING (20)	0	-	-	0	-	-	0	-	-	0	-	-	**1**	-	-
CVRP (20)	8	25.8	6402903	8	46.0	6402903	9	9.2	**1138274**	8	25.9	6402903	**13**	**5.1**	**1138274**
DELIVERY (20)	2	22.9	12032971	4	0.9	233222	4	0.6	**144849**	4	0.6	233222	**6**	**0.3**	145698
DEPOTS-SYM (20)	4	21.4	8907468	4	19.8	4338217	**5**	12.4	**2598405**	**5**	11.7	4337185	**5**	**6.9**	2606740
PETRINETS (20)	2	522.4	49590610	2	1013.8	49590610	3	184.5	8703352	2	523.0	49590610	**5**	**93.3**	**8703318**
TOTAL (269)	93	-	-	95	-	-	99	-	-	96	-	-	**108**	-	-
Linear							Blind								
ROVER-METRIC (10)	4	7.3	2463299	4	6.6	**1141856**	4	6.5	**1141856**	4	**3.6**	**1141856**	4	**3.6**	**1141856**
TPP-METRIC (40)	5	2.0	272368	5	3.8	272368	5	3.8	**270475**	5	**1.9**	272368	5	6.8	878884
ZENOTRAVEL-LINEAR (9)	3	**21.7**	5548179	3	39.9	**5470567**	3	43.5	5953172	3	22.0	5482235	3	22.7	5472508
BARMAN (15)	2	0.6	150255	2	0.6	80910	3	0.2	**20226**	3	0.3	80872	**4**	**0.1**	20330
BARMAN-UNIT (15)	2	4.3	895338	2	4.2	478390	**3**	0.7	**64560**	**3**	2.3	478353	**3**	**0.3**	64945
FO-COUNTERS-SYM (20)	1	95.4	35812449	1	168.9	35812449	2	32.3	**6167774**	1	100.7	35812449	**4**	**16.2**	**6167774**
TOTAL (125)	17	-	-	17	-	-	20	-	-	19	-	-	**23**	-	-
SCT							$h_{\text{LP}}^{\text{LM-cut,SEQ}}$								
DEPOTS (6)	1	1093.1	1692368	1	815.2	1364638	1	**797.7**	**1364544**	1	800.2	1369977	1	812.1	1369977
GARDENING (63)	63	3.8	118204	63	4.2	118202	63	2.5	72081	63	3.9	118200	63	**2.2**	**71940**
GARDENING-SAT (51)	12	67.5	1922720	12	75.8	1922708	12	45.7	**1189704**	12	66.0	1922678	12	**39.8**	1189810
ROVER (19)	4	65.2	1671782	4	28.1	**720103**	4	27.7	**720103**	4	25.8	**720103**	4	**25.5**	**720103**
SAILING (20)	20	**0.4**	**2667**	20	**0.4**	**2667**	20	**0.4**	**2667**	20	**0.4**	**2667**	20	**0.4**	2854
SAILING-SAT (30)	**7**	**0.5**	3723	**7**	**0.5**	3723	6	**0.5**	3723	**7**	**0.5**	3723	6	**0.5**	3723
CVRP (20)	11	139.7	453570	11	140.5	454206	14	**24.2**	**71604**	11	140.6	459676	**15**	24.4	71951
DELIVERY (20)	2	0.1	200	**6**	**0.0**	**184**	**6**	**0.0**	**184**	**6**	**0.0**	**184**	**6**	0.1	233
DEPOTS-SYM (20)	6	281.6	851717	6	202.4	781956	6	**118.4**	**452598**	6	258.4	1014985	6	156.2	623928
PETRINETS (20)	8	512.4	9045923	8	731.8	9047092	9	163.8	1945630	8	508.5	9050844	**10**	**110.9**	**1931552**
TOTAL (269)	134	-	-	138	-	-	141	-	-	138	-	-	**143**	-	-
Linear							$h_2^{\text{LM-cut}}$								
FO-SAILING (16)	**1**	630.8	7734153	**1**	676.4	7734153	0	-	-	**1**	641.9	7734153	**1**	**587.9**	**7439946**
ROVER-METRIC (10)	6	38.7	754115	6	34.7	687783	6	32.5	682699	6	30.7	**678101**	6	**30.5**	704527
TPP-METRIC (40)	5	**7.2**	**78121**	5	11.3	84035	5	12.2	83804	5	11.3	83959	5	29.6	204424
ZENOTRAVEL-LINEAR (9)	8	174.4	193432	8	126.7	140621	8	**124.8**	**140584**	8	128.1	147781	8	131.3	147159
BARMAN (15)	2	2.1	138859	2	1.3	74823	3	**0.3**	**19247**	3	1.1	74786	**4**	**0.3**	19340
BARMAN-UNIT (15)	2	165.7	40962	2	122.5	22766	2	0.7	**11198**	2	123.3	22696	2	**0.6**	11342
FO-COUNTERS-SYM (20)	6	60.9	4375453	6	112.2	4375453	9	6.8	**717955**	6	65.4	4375453	**11**	**4.4**	**717955**
TOTAL (125)	30	-	-	30	-	-	33	-	-	31	-	-	**37**	-	-

Table 1: 'c.' is coverage, 't.' is time, and '#gen' is the number of generated states. For 't.' and '#gen.', we took the average over instances solved by all search algorithms except for $h_2^{\text{LM-cut}}$ in FO-SAILING, where only one method solves no instance.

nal configuration. This problem is known to be exponential.

For LT, we introduce a linear numeric version of the BARMAN domain. This domain comes with unit-cost actions, and actions that have costs only if a drink was poured from the dispenser. Interestingly, since the BARMAN domain has a lot of delete-effects, on the unit-cost version of the domain the blind version of OSS outperforms all other configurations.

FO-COUNTERS-SYM is a variant of FO-COUNTERS (Li et al. 2018) in LT. Numeric variables x_i and r_i for $i \in [n]$ are initialized with zero, and an action increases/decreases r_i by 1 or increases/decreases x_i by r_i. While the goal condition is $x_i + 1 \leq x_{i+1}$ for $i \in [n]$ in the original domain, we use $\sum_{i=1}^{n} x_i \geq m$ in FO-COUNTERS-SYM to introduce symmetry. We generate 20 instances with $n = 3, 4, 5, 6$ and $m = 200, 300, 400, 500, 600$.

Conclusion

In this paper, we extended the notions of structural symmetries and problem description graphs from classical to numeric linear planning. This extension allowed us to use symmetry-breaking forward search techniques in the numeric setting. Our experiments demonstrate that these techniques outperform the state-of-the-art both in terms of expanded nodes and coverage. In the future, one may consider looking for groups that are larger than structural symmetries.

Acknowledgements

This work is partially supported by the Natural Sciences and Engineering Research Council of Canada. Alexander Shleyfman is partially supported by the Israel Academy of Sciences and Humanities program for Israeli postdocts.

References

Aldinger, J.; and Nebel, B. 2017. Interval Based Relaxation Heuristics for Numeric Planning with Action Costs. In *KI*, 15–28.

Babai, L. 2016. Graph isomorphism in quasipolynomial time [extended abstract]. In *STOC*, 684–697.

Bäckström, C.; and Klein, I. 1991. Planning in polynomial time: the SAS-PUBS class. *Comp. Intell.*, 7(3): 181–197.

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

Coles, A. J.; and Coles, A. 2010. Completeness-Preserving Pruning for Optimal Planning. In *ECAI*, 965–966.

Darga, P. T.; Sakallah, K. A.; and Markov, I. L. 2008. Faster Symmetry Discovery Using Sparsity of Symmetries. In *DAC*, 149–154.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2012. Enhanced Symmetry Breaking in Cost-Optimal Planning as Forward Search. In *ICAPS*, 343–347.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2013. Symmetry Breaking: Satisficing Planning and Landmark Heuristics. In *ICAPS*, 298–302.

Domshlak, C.; Katz, M.; and Shleyfman, A. 2015. Symmetry breaking in deterministic planning as forward search: Orbit space search algorithm. Technical Report IS/IE-2015-03, Technion, Israel.

Fox, M.; and Long, D. 2002. Extending the Exploitation of Symmetries in Planning. In *AIPS*, 83–91.

Gnad, D.; Torralba, Á.; Shleyfman, A.; and Hoffmann, J. 2017. Symmetry Breaking in Star-Topology Decoupled Search. In *ICAPS*, 125–134.

Helmert, M. 2002. Decidability and Undecidability Results for Planning with Numerical State Variables. In *AIPS*, 303–312.

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

Helmert, M.; and Röger, G. 2008. How Good is Almost Perfect? In *AAAI*, 944–949.

Herstein, I. N. 1975. *Topics in algebra*. Xerox College Pub.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *JAIR*, 20: 291–341.

Holte, R. C.; and Burch, N. 2014. Automatic Move Pruning for Single-Agent Search. *AI Comm.*, 27(4): 363–383.

Junttila, T.; and Kaski, P. 2007. Engineering an efficient canonical labeling tool for large and sparse graphs. In *ALENEX07@SIAM*, 135–149.

Kuroiwa, R.; Shleyfman, A.; and Beck, J. C. 2022. LM-Cut Heuristics for Optimal Linear Numeric Planning. In *ICAPS*, 203–212.

Kuroiwa, R.; Shleyfman, A.; Piacentini, C.; Castro, M. P.; and Beck, J. C. 2022. The LM-Cut Heuristic Family for Optimal Numeric Planning with Simple Conditions. *JAIR*, 75: 1477–1548.

Leofante, F.; Giunchiglia, E.; Ábrahám, E.; and Tacchella, A. 2020. Optimal Planning Modulo Theories. In *IJCAI*, 4128–4134.

Li, D.; Scala, E.; Haslum, P.; and Bogomolov, S. 2018. Effect-Abstraction Based Relaxation for Linear Numeric Planning. In *IJCAI*, 4787–4793.

Luks, E. M. 1991. Permutation Groups and Polynomial-Time Computation. In *DIMACS (vol. 11)*, 139–175.

Mathon, R. 1979. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8: 131–132.

Nissim, R.; Apsel, U.; and Brafman, R. I. 2012. Tunneling and Decomposition-Based State Reduction for Optimal Planning. In *ECAI*, 624–629.

Piacentini, C.; Castro, M. P.; Ciré, A. A.; and Beck, J. C. 2018. Compiling Optimal Numeric Planning to Mixed Integer Linear Programming. In *ICAPS*, 383–387.

Pochter, N.; Zohar, A.; and Rosenschein, J. S. 2011. Exploiting Problem Symmetries in State-Based Planners. In *AAAI*, 1004–1009.

Scala, E.; Haslum, P.; Magazzeni, D.; and Thiébaux, S. 2017. Landmarks for Numeric Planning Problems. In *IJCAI*, 4384–4390.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramírez, M. 2020. Subgoaling Techniques for Satisficing and Optimal Numeric Planning. *JAIR*, 68: 691–752.

Scala, E.; Ramírez, M.; Haslum, P.; and Thiébaux, S. 2016. Numeric Planning with Disjunctive Global Constraints via SMT. In *ICAPS*, 276–284.

Shleyfman, A. 2019. On Computational Complexity of Automorphism Groups in Classical Planning. In *ICAPS*, 428–436.

Shleyfman, A. 2020. *Symmetry Breaking and Operator Pruning in Classical Planning and Beyond*. Ph.D. thesis, Technion, Israel.

Shleyfman, A.; and Jonsson, P. 2021. Computational Complexity of Computing Symmetries in Finite-Domain Planning. *JAIR*, 70: 1183–1221.

Shleyfman, A.; Katz, M.; Helmert, M.; Sievers, S.; and Wehrle, M. 2015. Heuristics and Symmetries in Classical Planning. In *AAAI*, 3371–3377.

Shleyfman, A.; Kuroiwa, R.; and Beck, J. C. 2023. Supplement for Symmetry Detection and Breaking in Cost-Optimal Numeric Planning. https://zenodo.org/record/7741888. Accessed: 2023-03-16.

Tange, O. 2011. GNU Parallel - The Command-Line Power Tool. *;login: The USENIX Magazine*, 36: 42–47.

Uchoa, E.; Pecin, D.; Pessoa, A.; Poggi, M.; Vidal, T.; and Subramanian, A. 2017. New benchmark instances for the Capacitated Vehicle Routing Problem. *EJOR*, 257: 845–858.

Wehrle, M.; and Helmert, M. 2012. About Partial Order Reduction in Planning and Computer Aided Verification. In *ICAPS*, 297–305.

Wehrle, M.; Helmert, M.; Shleyfman, A.; and Katz, M. 2015. Integrating Partial Order Reduction and Symmetry Elimination for Cost-Optimal Classical Planning. In *IJCAI*, 1712–1718.