Graph-Based Factorization of Classical Planning Problems

Martin Wehrle and **Silvan Sievers** and **Malte Helmert**
University of Basel, Switzerland
{martin.wehrle,silvan.sievers,malte.helmert}@unibas.ch

Abstract

In domain-independent planning, dependencies of operators and variables often prevent the effective application of planning techniques that rely on "loosely coupled" problems (like factored planning or partial order reduction). In this paper, we propose a generic approach for *factorizing* a classical planning problem into an equivalent problem with fewer operator and variable dependencies. Our approach is based on variable factorization, which can be reduced to the well-studied problem of graph factorization. While the state spaces of the original and the factorized problems are isomorphic, the factorization offers the potential to exponentially reduce the complexity of planning techniques like factored planning and partial order reduction.

1 Introduction

Automated planning is a computationally hard task [Bylander, 1994], and various directions to tackle this task have been proposed in the last decade. A popular direction is based on exploiting the independence of variables or operators in the formulation of the given planning problem. This direction has been (and still is) investigated for planning in many different variants and contexts, e.g., representing the core idea of factored planning [Amir and Engelhardt, 2003; Brafman and Domshlak, 2006; Kelareva *et al.*, 2007; Brafman and Domshlak, 2008; Fabre *et al.*, 2010] and approaches based on partial order reduction applied to planning [Alkhazraji *et al.*, 2012; Wehrle and Helmert, 2012; Nissim *et al.*, 2012; Wehrle and Helmert, 2014; Holte *et al.*, 2015]. Factored planning solves subproblems individually and finally combines the resulting local solutions to a global plan. Partial order reduction explores only representatives of "permutation equivalent" sequences of independent operators. Both directions are most beneficial in "loosely coupled" problems with a high degree of variable and operator independence.

Although much research has been devoted to develop approaches that exploit independence in planning problems in a *given* formulation, surprisingly little research has been done on the investigation of (automated) techniques to *equivalently reformulate* a planning problem such that the reformulation offers fewer variable and operator dependencies. Such reformulation techniques appear to be attractive as they could render a given planning problem potentially more amenable to all approaches that benefit from "loosely coupled" problems, including the ones mentioned above. The most related paper that has addressed the task of equivalently reformulating a given planning problem in a more "factored" way is the paper by Haslum [2007]. Haslum states that *"a problem formulation with less 'coupling' (fewer mutual dependencies) between variables is generally better"*. However, he concludes (and leaves as open questions) that how to instantiate his technique *"to arrive at a better problem formulation is not [...] easy, and automating [...] even more difficult"*. Apparently, finding automatic techniques appears to be challenging.

In this paper, we propose a novel direction for reformulating planning problems based on *variable factorization*. Planning problems are usually formalized with the help of variables to describe the states of the world. We provide the theoretical basis for a generic approach to algorithmically decide how a given variable can be factorized into several independent variables and corresponding operators. We show that factorizing variables can be reduced to *graph factorization*, which is a well-studied problem in discrete mathematics. The resulting factorized formulation of the planning problem is equivalent to the original problem in the sense that their state spaces are isomorphic. However, due to fewer operator and variable dependencies, the factorization offers the potential to exponentially reduce the complexity of planning techniques that rely on "loosely coupled" problems.

2 Preliminaries

For a finite set of finite-domain state variables $\mathcal{V}$ with finite domain $dom(v)$ for all $v \in \mathcal{V}$, we define a *partial state* s as a mapping from a subset $vars(s) \subseteq \mathcal{V}$ to values in $dom(v)$ for all $v \in vars(s)$. In other words, $vars(s)$ denotes the set of variables for which the partial state s is defined. For $v \notin vars(s)$, we say that the value of v in s is *undefined*, which we denote by $\bot$. In a given partial state s, the value of v in s is $s[v]$. A partial state s *complies* with a partial state s' if $s[v] = s'[v]$ for all $v \in (vars(s) \cap vars(s'))$. A partial state s is called a *state* if $vars(s) = \mathcal{V}$. We sometimes denote (partial) states by sets of variable/value assignments.

A SAS$^+$ planning problem $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ is defined in terms of a finite set of finite-domain state variables $\mathcal{V}$, a

finite set of operators $\mathcal{O}$, an initial state s_0, and a partial goal state $s_\star$. Operators $o = \langle pre(o), \mathit{eff}(o), cost(o)\rangle \in \mathcal{O}$ consist of a precondition $pre(o)$, an effect $\mathit{eff}(o)$, and a function $cost(o)$. Both $pre(o)$ and $\mathit{eff}(o)$ are partial states, and $cost(o)$ assigns a non-negative cost value to o. An operator o is *applicable* in a state s if $pre(o)$ complies with s. Applying an applicable operator o in s yields the *successor state* s', which is obtained from s by setting the values of all $v \in vars(\mathit{eff}(o))$ to $\mathit{eff}(o)[v]$, and retaining the values of the other variables from s. We say that an operator $o \in \mathcal{O}$ *reads* a variable v if $v \in vars(pre(o))$, and that it *writes to* v if $v \in vars(\mathit{eff}(o))$. For technical reasons (and without loss of generality), we require variables v that occur in both the precondition and the effect of o to not read v and write to v with the same value, i.e., if $v \in vars(pre(o)) \cap vars(\mathit{eff}(o))$, then $pre(o)[v] \neq \mathit{eff}(o)[v]$.

A *plan* $\pi = o_1, \ldots, o_n$ is a sequence of operators that is sequentially applicable in the initial state, and applying this sequence yields a state s that complies with $s_\star$. A plan π is *optimal* if the summed cost values of the operators in π are minimal among all plans in Π.

For each operator $o \in \mathcal{O}$ and variable $v \in \mathcal{V}$, we define the partial state $pre_{\setminus v}(o)$ as $pre_{\setminus v}(o)[v'] := pre(o)[v']$ for all $v' \in vars(pre(o)) \setminus \{v\}$, and $pre_{\setminus v}(o)[v']$ as undefined for all other variables v'. Informally, $pre_{\setminus v}(o)$ represents the partial state obtained from $pre(o)$ when setting the value of v to undefined. Accordingly, we define $\mathit{eff}_{\setminus v}(o)$ for $\mathit{eff}(o)$ as the partial state obtained from $\mathit{eff}(o)$ when setting the value of v to undefined.

For a planning problem $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star\rangle$, the *state space* of Π is defined as the *transition system* $\mathcal{T}^\Pi = (S, L, T, s_0, S_\star)$, where S is the set of states of Π, L is the set $\{l(o) \mid o \in \mathcal{O}\}$ of operator labels induced by $\mathcal{O}$, $T \subseteq S \times L \times S$ is the set of transitions with $(s, l(o), s') \in T$ if o is applicable in s and yields the successor state s', s_0 is the initial state of Π, and $S_\star$ is the set of goal states that comply with $s_\star$. For a variable $v \in \mathcal{V}$, we define the *atomic transition system* $\mathcal{T}^v$ of v as the homomorphic projection of $\mathcal{T}^\Pi$ to v via the abstraction function $\alpha(s) := s[v]$, i.e., as the directed graph $\mathcal{T}^v = \langle S^v, L, T^v, s_0^v, S_\star^v\rangle$ with $S^v = dom(v)$, $T^v = \{(\alpha(s), l(o), \alpha(s')) \mid (s, l(o), s') \in T\}$, $s_0^v = \alpha(s_0)$, and $S_\star^v = \{\alpha(s) \mid s \in S_\star\}$.

2.1 Graph Factorization

In discrete mathematics, the *factorization* of graphs has been studied intensively since the 1960s, both for undirected and directed graphs, and for different kinds of products. A factorization of a graph G is given by graphs $G_1, \ldots, G_n$ such that the product of $G_1, \ldots, G_n$ yields G. More specifically, it has been shown that undirected and directed graphs without self-loops have a unique factorization into *prime graphs* with respect to the *Cartesian product*, and that this factorization can be computed in polynomial time [Sabidussi, 1960; Winkler, 1987; Aurenhammer *et al.*, 1992; Imrich and Peterin, 2007]. Recently, a linear factorization algorithm for directed graphs without self-loops has been proposed [Crespelle and Thierry, 2015]. For directed graphs $G_1 = \langle V_1, E_1\rangle, \ldots, G_m = \langle V_m, E_m\rangle$, the *Cartesian product* of $G_1, \ldots, G_m$ is defined as $G^\times := \langle V^\times, E^\times\rangle$, with $V^\times := V_1 \times \cdots \times V_m$. There is an edge $\langle\langle v_1, \ldots, v_m\rangle, \langle w_1, \ldots, w_m\rangle\rangle \in E^\times$ if and only if there is $i \in \{1, \ldots, m\}$ with $\langle v_i, w_i\rangle \in E_i$ and $v_j = w_j$ for all $j \neq i$. We remark that, in contrast to *synchronized products*, edges in Cartesian products are not synchronized across different graphs, in the sense that every edge in the product $G^\times$ corresponds to exactly one vertex change in one graph G_i. In the following, we propose a factorization approach for planning problems based on a reduction to Cartesian graph factorization.

3 A Motivating Example

Assume that a truck is supposed to drive from its initial location 1 to its goal location 4, where the goal location can be reached via the intermediate locations 2 or 3. A straightforward formulation in STRIPS will include four variables `at-i` with $i \in \{1, 2, 3, 4\}$ to represent the location of the truck. In SAS$^+$, one variable `pos` with domain $dom(\texttt{pos}) = \{1, 2, 3, 4\}$ will typically be used. Figure 1 shows the atomic transition system of the SAS$^+$ version (left).

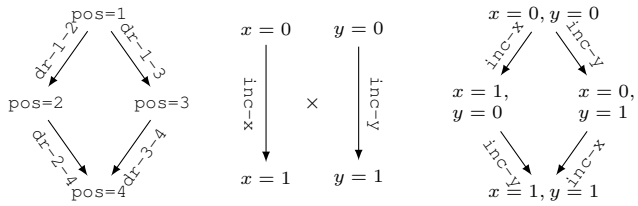

Figure 1: Atomic transition system of original formulation (left), graphs describing factorized formulation of x and y (middle), Cartesian product of factorized formulation (right).

Both the STRIPS and the SAS$^+$ formulation are "tightly coupled" in the sense that, e.g., partial order reduction will not provide any reduction: Partial order reduction is a pruning method that eliminates permutations of sequences of *independent* operators that lead to goal states, such that at least one permutation thereof is preserved. In the example, partial order reduction does not prune anything because the two alternative paths from locations 1 to 4 over 2 or 3 are not spanned by the same set of operators, i.e., there is no equivalent permutation of operator sequences.

However, there exists a *factorization* of the SAS$^+$ variable `pos` into SAS$^+$ variables x and y with $dom(x) = dom(y) = \{0, 1\}$, with operators `inc-x` and `inc-y` that set x and y from 0 to 1, respectively. Figure 1 shows the corresponding factorized graphs representing x and y in the middle. Extending Cartesian products to labeled graphs in the straight forward way (we will formalize this below), we observe that the Cartesian product of the two factorized graphs (shown on the right in Figure 1) is structurally isomorphic to the atomic transition system in the original formulation, leaving operator names aside for the moment. This means that we can capture the semantics of the original formulation with the factorized one. At the same time, the factorized formulation yields independent operators than can be permuted arbitrarily. Hence, one of the two permutations of the plans (`inc-x, inc-y`

and `inc-y`, `inc-x`) can be eliminated by the use of partial order reduction. To summarize, we observe that the factorization can yield a decoupled, but equivalent semantics of the original planning problem, and there is a 1:1 correspondence of the plans in the two formulations.

4 Factorization of Planning Problems

In the following, we will generalize the concept of Cartesian graph factorization of unlabeled graphs to transition systems. The following definitions consider transition systems without self-loops – we will discuss how to handle self-loops (as they occur in atomic transition systems of planning problems) below. We start by defining Cartesian products. To keep things simple, we restrict the definitions to two graphs (the generalization to an arbitrary number is straight forward).

Definition 1 (Cartesian Product)**.** *Let* $G^1 = (S^1, L, T^1, s_0^1, S_\star^1)$ *and* $G^2 = (S^2, L, T^2, s_0^2, S_\star^2)$ *be transition systems with common label set* L *and without self-loops (i.e., without transitions of the form* (s, l, s) *for any state* s *and any label* l*). The* Cartesian product $G^1 \times G^2$ *of* G^1 *and* G^2 *is the transition system* $G^\times := (S^\times, L, T^\times, s_0^\times, S_\star^\times)$*, where* $S^\times = \{(s^1, s^2) \mid s^1 \in S^1, s^2 \in S^2\}$ *is the Cartesian product of* S^1 *and* S^2*, and there is an edge* $((s^1, s^2), l, (t^1, t^2))$ *in* $T^\times$ *iff* $(s^1, l, t^1) \in T^1$ *and* $s^2 = t^2$*, or* $s^1 = t^1$ *and* $(s^2, l, t^2) \in T^2$*. Analogously to* $S^\times$*,* $s_0^\times$ *is defined as* (s_0^1, s_0^2)*, and* $S_\star^\times$ *as the Cartesian product of* $S_\star^1$ *and* $S_\star^2$*.*

Definition 2 (Cartesian Factor)**.** *Let* $G = (S, L, T, s_0, S_\star)$ *be a transition system without self-loops. The transition systems* $G_1 = (S^1, L, T^1, s_0^1, S_\star^1)$ *and* $G_2 = (S^2, L, T^2, s_0^2, S_\star^2)$ *are* Cartesian factors *of* G *iff* G *is isomorphic to the Cartesian product* $G^\times = (S^\times, L, T^\times, s_0^\times, S_\star^\times)$ *of* G_1 *and* G_2*, i.e., iff there is a bijective function* $\varphi : S \to S^\times$ *such that* $\langle s, l, t\rangle \in T$ *iff* $\langle \varphi(s), l', \varphi(t)\rangle \in T^\times$ *for* $l, l' \in L$*.*

We remark that in Def. 2, two transition systems are called "isomorphic" if their graph structures are isomorphic in the usual sense, whereas labels (that correspond to operator names when used for planning problems) are allowed to be different.

We now apply the concept of graph factorization to planning. As atomic transition systems characterize the behavior of variables, values, and value changes, a factorization of such an atomic transition system $\mathcal{T}^v$ of variable v corresponds to a factorization of v itself. Furthermore, the factorized variables of v also yield a factorization of the operators that read or write to v. To formally define these factorized operators, let us come back to the question how to handle self-loops in atomic transition systems. We observe that, apart from the trivial case where an operator does not mention variable v at all, self-loops in an atomic transition system $\mathcal{T}^v$ can either be induced by an operator that reads v, but does not write to v, or vice versa does not read v, but writes to v. In the following definition of factorized operators, these cases are handled separately from the case where an operator both reads and writes to v. For a given atomic transition system $\mathcal{T}^v = \langle S^v, L, T^v, s_0^v, S_\star^v\rangle$, we define the atomic transition system without self-loops $\overline{\mathcal{T}}^v = \langle S^v, L, \overline{T}^v, s_0^v, S_\star^v\rangle$, which is obtained from $\mathcal{T}^v$ by removing all self-loops from T^v, i.e., $\overline{T}^v := \{(s, l, t) \in T^v \mid s \neq t\}$.

Let $\mathcal{O}^v := \{o \in \mathcal{O} | v \in (vars(pre(o)) \cup vars(\mathit{eff}(o)))\}$ be the operators that "work on v". In the following, we again restrict our definitions to two graphs to keep things simple.

Definition 3 (Factorized Operators)**.** *Let* $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star\rangle$ *be a planning problem, let* $v \in \mathcal{V}$ *be a variable with atomic transition system* $\mathcal{T}^v = \langle S^v, L, T^v, s_0^v, S_\star^v\rangle$*, and let* $\overline{\mathcal{T}}^v = \langle S^v, L, \overline{T}^v, s_0^v, S_\star^v\rangle$ *be the corresponding atomic transition system without self-loops. Let* $\mathcal{T}^{v_1} = \langle S^{v_1}, L, T^{v_1}, s_0^{v_1}, S_\star^{v_1}\rangle$ *and* $\mathcal{T}^{v_2} = \langle S^{v_2}, L, T^{v_2}, s_0^{v_2}, S_\star^{v_2}\rangle$ *be Cartesian factors of* $\overline{\mathcal{T}}^v$ *and let* φ *be the bijection between* $\mathcal{T}^v$ *and* $\overline{\mathcal{T}}^{v_1} \times \overline{\mathcal{T}}^{v_2} = (S^\times, L, T^\times, s_0^\times, S_\star^\times)$*.*

Let $o = \langle pre(o), \mathit{eff}(o), cost(o)\rangle$ *be an operator with* $o \in \mathcal{O}^v$*. The* factorized operator o^f *of* o *is defined as follows. We distinguish three cases.*

1. $v \in vars(pre(o)) \cap vars(\mathit{eff}(o))$*, i.e.,* o *both reads and writes to* v*. Then* o *induces a transition* $(pre(o)[v], l, \mathit{eff}(o)[v]) \in \overline{T}^v$*, which, via* φ*, corresponds to a transition* $t^\times = ((s_1, s_2), l', (t_1, t_2)) \in T^\times$ *for some* $l' \in L$*, where* $(s_1, s_2) = \varphi(pre(o)[v])$ *and* $(t_1, t_2) = \varphi(\mathit{eff}(o)[v])$*. By the definition of Cartesian products, the transition* $t^\times$ *is induced by exactly one transition* $(s_i, l', t_i) \in T^{v_i}$ *for some* $i \in \{1, 2\}$*. The factorized operator of* o *is defined as*

   $$o^f := \langle pre_{\setminus v}(o) \cup \{v_i \mapsto s_i\}; \mathit{eff}_{\setminus v}(o) \cup \{v_i \mapsto t_i\}; c_o\rangle$$

   with $c_o := cost(o)$*.*

2. $v \in vars(pre(o)) \setminus vars(\mathit{eff}(o))$*, i.e.,* o *only reads* v*, but does not write to* v*. Let* $\varphi(pre(o)[v]) = (s_1, s_2)$*. The factorized operator of* o *is defined as*

   $$o^f := \langle pre_{\setminus v}(o) \cup \bigcup_{i=1}^{2} \{v_i \mapsto s_i\}; \mathit{eff}_{\setminus v}(o); cost(o)\rangle,$$

   i.e., the condition $\{v \mapsto pre(o)[v]\}$ *is replaced by the corresponding conditions on the factorized variables and values obtained from* φ*.*

3. $v \in vars(\mathit{eff}(o)) \setminus vars(pre(o))$*, i.e.,* o *only writes to* v*, but does not read* v*. Let* $\varphi(\mathit{eff}(o)[v]) = (t_1, t_2)$*. The factorized operator of* o *is defined as*

   $$o^f := \langle pre_{\setminus v}(o); \mathit{eff}_{\setminus v}(o) \cup \bigcup_{i=1}^{2} \{v_i \mapsto t_i\}; cost(o)\rangle,$$

   i.e., the effect $\{v \mapsto \mathit{eff}(o)[v]\}$ *is replaced by the corresponding effects on the factorized variables and values obtained from* φ*.*

Bullet point 1 in Def. 3 refers to operators that both read and write to v and hence induce a non-self-loop transition in the atomic transition system $\mathcal{T}^v$ (recall that we assume $pre(o)[v] \neq \mathit{eff}(o)[v]$ for o). Such operators are mapped to factorized operators that only read and write to only one factorized variable, and retain the precondition and effects on the remaining variables. Bullet points 2 and 3 in Def. 3 cover the remaining cases where o only reads v or writes to

v, respectively. Such operators can only induce self-loops, hence they are not captured by the factorization (which ignores self-loops). The corresponding factorized operator is defined based on the variable/value combination of the factorized variables and the bijective mapping provided by φ.

We note that, due to the definition of Cartesian products, there are generally several operators that are mapped to the same factorized operator. For example, in the planning problem in Fig. 1, the two operators that drive from position 1 to 2 and drive from position 3 to 4, are mapped to the single operator that sets x from 0 to 1.

We are now ready to define factorized planning problems (for simplicity, again restricted to the case of two factors).

Definition 4 (Factorized Planning Problem)**.** *Let $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ be a planning problem, let $v \in \mathcal{V}$ be a variable with atomic transition system $\mathcal{T}^v$, and let v_1, v_2 be factorized variables of v, i.e., let $\mathcal{T}^{v_i} = \langle S^{v_i}, L, T^{v_i}, s_0^{v_i}, S_\star^{v_i} \rangle$ for $i \in \{1,2\}$ be Cartesian factors of $\overline{\mathcal{T}}^v$ and let φ be the bijection between $\overline{\mathcal{T}}^v$ and $\mathcal{T}^{v_1} \times \mathcal{T}^{v_2}$. The* factorized planning problem *$\Pi^f = \langle \mathcal{V}^f, \mathcal{O}^f, s_0^f, s_\star^f \rangle$ with respect to v_1, v_2 is defined as follows:*

1. *$\mathcal{V}^f := \mathcal{V} \setminus \{v\} \cup \{v_1, v_2\}$ with $\{v_1, v_2\} \cap \mathcal{V} = \emptyset$, where $dom(v_i) := S^{v_i}$ for $i \in \{1,2\}$.*
2. *$\mathcal{O}^f := \{o^f$ factorized operator of $o \mid o \in \mathcal{O}^v\} \cup \mathcal{O} \setminus \mathcal{O}^v$*
3. *Let $s_0[v] = x$ and $\varphi(x) = (x_1, x_2)$. Then $s_0^f[v_i] := x_i$ for $i \in \{1,2\}$, and $s_0^f[w] := s_0[w]$ for all $w \in \mathcal{V} \setminus \{v\}$.*
4. *If $v \notin vars(s_\star)$, then $s_\star^f[v_i] := \bot$ for $i \in \{1,2\}$. Otherwise, let $s_\star[v] = x$ and $\varphi(x) = (x_1, x_2)$. Then $s_\star^f[v_i] := x_i$ for $i \in \{1,2\}$. In both cases, $s_\star^f[w] := s_\star[w]$ for all $w \in \mathcal{V} \setminus \{v\}$.*

We remark that the definition of factorized planning problems is solely based on the bijective mapping from variable assignments $\{v \mapsto x\}$ to factorized assignments $\{v_1 \mapsto x_1, v_2 \mapsto x_2\}$ provided by φ (i.e. $\varphi(x) = (x_1, x_2)$). As a consequence, the state space graphs of original and factorized problems are isomorphic (modulo operator renaming), which we show formally in the following theorem. We again only consider the case of two factors for simplicity.

Theorem 1. *Let $\Pi = \langle \mathcal{V}, \mathcal{O}, s_0, s_\star \rangle$ be a planning problem, $v \in \mathcal{V}$, and v_1, v_2 be factorized variables of v induced by an isomorphism φ in the sense of Def. 2. Let $\Pi^f = \langle \mathcal{V}^f, \mathcal{O}^f, s_0^f, s_\star^f \rangle$ be the factorized planning problem with respect to v_1, v_2. Then the state space graphs of Π and Π^f are isomorphic (where operators can have different names).*

Proof. Let $\overline{\mathcal{T}}^v$, $\mathcal{T}^{v_1}$ and $\mathcal{T}^{v_2}$ be the atomic transition systems (without self-loops) of v, v_1, v_2. Let $\mathcal{S}$ and $\mathcal{S}^f$ be the set of states of Π and Π^f. In the following, for a tuple $\varphi(x) = (x_1, x_2)$, we denote the ith component x_i of $\varphi(x)$ by $\varphi(x)[i]$, $i \in \{1,2\}$. We will show that the state spaces of Π and Π^f are isomorphic via the function $\psi : \mathcal{S} \cup \mathcal{O} \to \mathcal{S}^f \cup \mathcal{O}^f$, which is defined by

$$\psi(s) := (s \setminus \{v \mapsto s[v]\}) \cup \{\bigcup_{i=1}^{2} (v_i \mapsto \varphi(s[v])[i])\}$$

for states, by $\psi(o) := o^f$ for $o \in \mathcal{O}^v$ and $\psi(o) := o$ for $o \in \mathcal{O} \setminus \mathcal{O}^v$. For brevity, we only consider the non-trivial case for operators $o \in \mathcal{O}^v$. Assume there is a state transition from state s to t by applying such o in the state space of Π. We will show that there is a corresponding state transition from $\psi(s)$ to $\psi(t)$ by applying $\psi(o) = o^f$ in the state space of Π^f.

1. $v \in vars(pre(o)) \cap vars(\mathit{eff}(o))$. To simplify notation, let $x := s[v]$ and $y := t[v]$, with $x \neq y$. As o sets v from x to y, there is a transition $\langle x, l, y \rangle$ in $\overline{\mathcal{T}}^v$. As φ is an isomorphism between $\overline{\mathcal{T}}^v$ and the Cartesian product $\mathcal{T}^\times$ of $\mathcal{T}^{v_1}$ and $\mathcal{T}^{v_2}$, there is a transition $t^\times := \langle \varphi(x), l', \varphi(y) \rangle$ in $\mathcal{T}^\times$ for some $l' \in L$. By the definition of Cartesian products, the transition $t^\times$ is induced by a transition in exactly one Cartesian factor $i \in \{1,2\}$, say factor i. Hence there is a transition $\langle \varphi(x)[i], l', \varphi(y)[i] \rangle$ in $\mathcal{T}^{v_i}$. This is the defining transition of operator o^f according to Def. 3: $o^f = \langle pre_{\setminus v}(o) \cup \{v_i \mapsto \varphi(x)[i]\}; \mathit{eff}_{\setminus v}(o) \cup \{v_i \mapsto \varphi(y)[i]\}; cost(o) \rangle$. It follows that o^f is applicable in $\psi(s)$ and leads to $\psi(t)$: o^f sets v_i from $\varphi(x)[i]$ to $\varphi(y)[i]$, and o and o^f have the same preconditions and effects on other variables than v (hence $pre(o^f)$ and $\mathit{eff}(o^f)$ comply with $\psi(s)$ and $\psi(t)$, respectively).
2. $v \in vars(pre(o)) \setminus vars(\mathit{eff}(o))$. As o is applicable in s, $pre(o)[v] = s[v]$. By the definitions of o^f and ψ, $pre(o^f)[v_i] = \varphi(s[v])[i] = \psi(s)[v_i]$ for $i \in \{1,2\}$, and the preconditions of o^f on other variables than v_1, v_2 are the same as those of o. Hence, o^f is applicable in $\psi(s)$. As o does not modify v, o^f does not modify v_1, v_2 either, and the effects of o^f on other variables than v_1, v_2 are the same as those of o. Hence, because o leads from s to t, the application of o^f in $\psi(s)$ leads to $\psi(t)$.
3. $v \in vars(\mathit{eff}(o)) \setminus vars(pre(o))$. As v does not occur in the precondition of o, o^f does not mention v_1, v_2 in its precondition either, and its preconditions on other variables than v_1, v_2 are the same as those of o. Hence, because o is applicable in s, o^f is applicable in $\psi(s)$. Because the application of o in s yields t, it follows that $t[v] = \mathit{eff}(o)[v]$. By the definitions of o^f and ψ, $\mathit{eff}(o^f)[v_i] = \varphi(t[v])[i] = \psi(t)[v_i]$ for $i \in \{1,2\}$, and the effects of o^f on other variables than v_1, v_2 are the same as those of o. Hence the application of o^f in $\psi(s)$ leads to $\psi(t)$.

Showing that for all state transitions in Π^f there is a corresponding transition in Π is done analogously. □

Corollary 1. *Factorizing planning problems via isomorphism φ preserves plan existence and optimal plan cost.*

Proof. Plan existence is preserved because the state spaces of the planning problems are isomorphic according to Theorem 1, and because φ preserves the initial state and goal states: goal states are preserved because v is a goal variable in the original planning problem with goal value $s_\star[v] = x$ iff the factorized variables v_1, v_2 of v are goal variables in the factorized planning problem with goal values $s_\star[v_i] = \varphi(x)[i]$ for $i \in \{1,2\}$. Analogously, φ preserves the initial

state. Optimal plan cost are preserved because factorized operators have the same cost as the corresponding operators in the original problem. □

The mapping from operators $\mathcal{O}$ to factorized operators $\mathcal{O}^f$ (Def. 3) allows us to transform plans π^f in the factorized planning problem Π^f to plans π in the original problem Π by scanning π^f, and "mapping back" all factorized operators in $\mathcal{O}^f$ to operators in $\mathcal{O}$. As mentioned above, assuming v to be a variable factorized into $v_1, \ldots, v_n$, and assuming o^f to be a factorized operator, there are several operators in the original problem that correspond to o^f, hence the inverse function is not uniquely defined. However, as the state spaces of the original and the factorized problem are isomorphic, there is exactly one original operator that corresponds to o^f.

5 Properties of Factorized Planning Problems

We now study some properties of factorized planning problems. Brafman and Domshlak [2006] presented a *factored* planning algorithm whose runtime is exponential only in the treewidth of the (undirected) causal graph [Knoblock, 1994] of the planning problem and its *local width*, the smallest number k such that a plan that writes to each variable at most k times exists. We show that factorization can be very beneficial for factored planning.

Theorem 2. *Let Π be a planning problem and Π^f be a factorized planning problem of Π. The runtime bound for factored planning in Π^f can be exponentially smaller than for Π.*

Proof. Consider a family of planning problems Π_n ($n \in \mathbb{N}$) obtained by generalizing the example in Fig. 1. Let Π_n consist of one variable pos with $dom(\texttt{pos}) = \{1, \ldots, 2^n\}$ such that the corresponding factorized planning problem Π_n^f consists of n variables $v_1, \ldots, v_n$. For each v_i, there is exactly one operator in Π_n^f that sets v_i from 0 to 1 (and hence, there are n atomic transition systems with two locations, as shown in the middle in Fig. 1 for $n = 2$). The initial and the goal states are given by $v_i = 0$ and $v_i = 1$ for all $i \in \{1, \ldots, n\}$, respectively. For all $n \in \mathbb{N}$, there is a plan in Π_n^f that affects every variable v_i only once (local width 1), whereas every plan in Π_n affects pos at least n times (local width n). The tree width of the causal graph is equal to 1 in both Π_n and Π_n^f. Hence factored planning provides an exponential runtime bound for Π_n but a polynomial one for Π_n^f. □

In the following, we show that similar results hold for *strong stubborn sets* [Valmari, 1989; Alkhazraji *et al.*, 2012] and *sleep sets* [Godefroid, 1996; Holte *et al.*, 2015].

For a given state s, strong stubborn sets can prune permutations of operator sequences that consist of *independent* operators and lead from s to a goal state, with the guarantee that at least one of these permutations is preserved in s. Operators o and o' are denoted as independent if o does not disable o' (i.e., there is no variable v such that $\mathit{eff}(o)[v] \neq \mathit{pre}(o')[v]$), and o' does not disable o, and o, o' do not have conflicting effects (i.e., there is no variable v such that $\mathit{eff}(o)[v] \neq \mathit{eff}(o')[v]$).

Theorem 3. *Let Π be a planning problem and Π^f be a factorized planning problem of Π. The size of the reachable state space with strong stubborn sets in Π^f can be exponentially smaller than the corresponding size in Π.*

Proof. Consider again the problems Π_n and factorized problems Π_n^f from the proof of Theorem 2. In every state of Π_n^f, strong stubborn sets contain only one of the operators of Π_n^f (all factorized operator pairs are independent). This yields a reachable state space of size $n + 1$. In contrast, strong stubborn sets do not prune in Π_n because all operators are *not* independent with each other. Hence, Π_n has 2^n reachable states. □

Sleep sets are usually applied in tree search algorithms to avoid exploring equal states reached via different paths in the state space. Sleep sets can prune state transitions induced by *commutative* operators. Operators o and o' are denoted as commutative if o and o' are independent, and additionally, o and o' do not enable each other (i.e., there is no variable v such that $\mathit{eff}(o)[v] = \mathit{pre}(o')[v]$, and vice versa).

Theorem 4. *Let Π be a planning problem and Π^f be a factorized planning problem of Π. The number of generated nodes with iterative deepening search and sleep sets can be exponentially smaller in Π^f than in Π.*

Proof. Consider the problems Π_n and factorized problems Π_n^f from the proof of Theorem 2. Optimal plans in Π_n and Π_n^f have length n, and therefore iterative deepening search explores all paths of length $n - 1$ in the second last iteration. Without pruning, Π_n and Π_n^f have $n!$ paths of length $(n - 1)$ from the initial state.

Sleep set pruning in Π_n does not prune any search node because there is no pair of operators that is commutative. Hence $n!$ search nodes are generated at depth $n - 1$.

In contrast, sleep set pruning in Π_n^f prunes all but one path to every reachable state: Because operators in this problem are commutative, sleep sets assume an (arbitrary) total order $<$ on the set of operators, and it is easy to see that under this restriction, every state can only be reached by a single path. This results in a runtime bound of $O(2^n)$ when searching Π_n^f, an exponential reduction compared to $n!$ for Π_n. □

6 Discussion

Our approach provides the basic theory of a novel direction for problem reformulation. As shown in the previous section, techniques that rely on "loosely coupled" problems can benefit from factorization. However, we also observe that the current theory for factorizing atomic transition systems such that the resulting state spaces are isomorphic is stronger than needed, as we only need the *reachable* part of the product to be equal to the original. Consider the example in Fig. 2, an extension of the example in Fig. 1.

The example shows an atomic transition system $\mathcal{T}$, where a truck is supposed to drive from location 1 to 5 via one of the locations 2 or 3. Again, typical partial order reduction methods do not fire in this problem formulation because all pairs of operators interfere (i.e., they are not independent). However, although $\mathcal{T}$ cannot be factorized into non-trivial Cartesian factors, there exists a factorization into variables x, y, z and corresponding operators $o_1 = \langle \{x \mapsto 0\}; \{x \mapsto 1\} \rangle$,

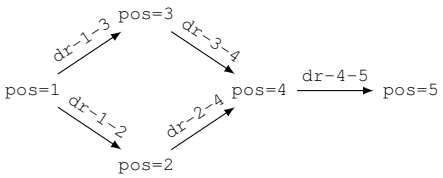

Figure 2: Example atomic transition system.

$o_2 = \langle\{y \mapsto 0\}; \{y \mapsto 1\}\rangle$ and $o_3 = \langle\{x \mapsto 1, y \mapsto 1\}; \{z \mapsto 1\}\rangle$ such that the *reachable* state space (with the initial state $\{x \mapsto 0, y \mapsto 0, z \mapsto 0\}$) induced by o_1, o_2, o_3 is isomorphic to $\mathcal{T}$. We observe that o_1, o_2, o_3 are independent, which can be exploited by, e.g., partial order reduction. How to compute such "relaxed factorizations" remains an open question.

Factorizing variables into an equivalent problem formulation can be viewed as a variant of "inverse fluent merging". Fluent merging [van den Briel *et al.*, 2007; Seipp and Helmert, 2011] merges several variables to a joint new variable with corresponding variable domain. As a result, the merged formulation can be more amenable to the computation of accurate heuristics. On the other hand, fluent merging can increase coupling in the problem formulation.

7 Towards a Planning Algorithm

Factorization algorithms for planning need to handle labeled transitions, while existing graph factorization algorithms do not handle transition labels (see Section 2.1). A straightforward, yet rather restrictive way to still make use of these algorithms is based on a try-and-check approach: Given a planning problem Π and a variable v of Π, such an algorithm considers the atomic transition system $\mathcal{T}^v$, and computes a factorization of the underlying directed graph without self-loops $\overline{\mathcal{T}}^v$ where labels are ignored. To obtain a correct factorization according to our theory, the algorithm then needs to check if operators that are supposed to be mapped to the same factorized operator as defined by the Cartesian factors have the same preconditions and effects on other variables than v in Π. If this is the case, the factorized planning problem Π^f can be computed. The Cartesian graph factorization can be achieved in linear time [Crespelle and Thierry, 2015]. The remaining part of checking the conditions on the factorized operators can be done in low-order polynomial time in the compact size of the planning problem, yielding an overall low-order polynomial time complexity. In addition, atomic transition systems for typical planning problems often have rather small sizes, hence the overhead of checking for Cartesian factors will mostly be negligible.

Currently, we do not have a fully automatic implementation. A preliminary analysis of planning domains from the international planning competitions up to 2014 revealed that some domains (Floortile, Grid, Sokoban, Tetris and Visitall) contain atomic transition systems whose unlabeled graph structure can be factorized. However, they do not pass the try-and-check approach, i.e., cannot be factorized into non-trivial Cartesian factors according to our theory.

As a proof of concept, we slightly modified the IPC Visitall domain such that "moving" and "marking as visited" are represented by different operators. We applied the Fast Downward planner [Helmert, 2006] to this domain and its factorization, using A* with strong stubborn sets as well as IDA* with sleep sets. While the factorization does not yield additional pruning with strong stubborn sets (as "moving" can still disable "mark-as-visited"), it does when applied with sleep sets. We compare IDA* with and without sleep sets, using Fast Downward's blind heuristic and Fast Downward's implementation of iPDB [Haslum *et al.*, 2007; Sievers *et al.*, 2012]. Table 1 shows the resulting number of generated nodes (without the last f-layer to avoid tie-breaking issues) and the runtime in seconds.

	original formulation				factorized formulation			
	#nodes		runtime		#nodes		runtime	
size	nop	prune	nop	prune	nop	prune	nop	prune
	blind							
6	3853	3853	0.0	0.0	3853	987	0.0	0.0
9	2.5e+6	2.5e+6	12.2	14.4	2.5e+6	93613	12.2	0.6
12	—	—	—	—	—	1.1e+7	—	86.0
	iPDB							
12	81042	81042	0.7	0.8	72721	19102	4.5	4.2
16	1.9e+7	1.9e+7	112.6	125.5	1.3e+7	1.5e+6	107.7	22.9
20	—	—	—	—	—	1.6e+8	—	1710.8

Table 1: IDA* without pruning (nop) and with sleep sets pruning (prune); dashes mean out of time (> 1800 seconds)

Focusing on the results with the blind heuristic (to measure the pure pruning ability on the factorization) displayed in the upper part, we observe that the factorization allows sleep sets to cut down the number of generated nodes by avoiding the generation of duplicates. The smaller search space translates into lower runtime. When using iPDB (lower part), we observe that the factorization causes iPDB to compute a more accurate heuristic (i.e., fewer nodes are generated on the factorized formulation also when no pruning is applied), which results in a slightly higher precomputation time. More importantly, we again observe that the factorization can be beneficial when applied with sleep sets, which can in turn result in fewer generated nodes and lower runtime.

8 Conclusions

We have proposed a theory for equivalently reformulating planning problems based on Cartesian graph factorization. Factorized planning problems offer the potential to yield exponentially smaller state spaces when factored planning or partial order reduction is applied. In the future, it will be important to turn the theory into practice: As we have observed, the requirement for the resulting state spaces to be isomorphic is stronger than needed, because we only need the reachable part of the product to be equal to the original. Furthermore, it will be important to investigate more sophisticated factorization algorithms for labeled graphs.

Acknowledgments

This work was supported by the Swiss National Science Foundation (SNSF) as part of the project "Automated Reformulation and Pruning in Factored State Spaces (ARAP)".

References

[Alkhazraji *et al.*, 2012] Yusra Alkhazraji, Martin Wehrle, Robert Mattmüller, and Malte Helmert. A stubborn set algorithm for optimal planning. In *Proc. ECAI 2012*, pages 891–892, 2012.

[Amir and Engelhardt, 2003] Eyal Amir and Barbara Engelhardt. Factored planning. In *Proc. IJCAI 2003*, pages 929–935, 2003.

[Aurenhammer *et al.*, 1992] Franz Aurenhammer, Johann Hagauer, and Wilfried Imrich. Cartesian graph factorization at logarithmic cost per edge. *Computational Complexity*, 2:331–349, 1992.

[Brafman and Domshlak, 2006] Ronen I. Brafman and Carmel Domshlak. Factored planning: How, when and when not. In *Proc. AAAI 2006*, pages 809–814, 2006.

[Brafman and Domshlak, 2008] Ronen I. Brafman and Carmel Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In *Proc. ICAPS 2008*, pages 28–35, 2008.

[Bylander, 1994] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, 1994.

[Crespelle and Thierry, 2015] Christophe Crespelle and Eric Thierry. Computing the directed cartesian-product decomposition of a directed graph from its undirected decomposition in linear time. *Discrete Mathematics*, 338(12):2393–2407, 2015.

[Fabre *et al.*, 2010] Eric Fabre, Loïg Jezequel, Patrik Haslum, and Sylvie Thiébaux. Cost-optimal factored planning: Promises and pitfalls. In *Proc. ICAPS 2010*, pages 65–72, 2010.

[Godefroid, 1996] Patrice Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems – An Approach to the State-Explosion Problem*, volume 1032 of *Lecture Notes in Computer Science*. Springer-Verlag, 1996.

[Haslum *et al.*, 2007] Patrik Haslum, Adi Botea, Malte Helmert, Blai Bonet, and Sven Koenig. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proc. AAAI 2007*, pages 1007–1012, 2007.

[Haslum, 2007] Patrik Haslum. Reducing accidental complexity in planning problems. In *Proc. IJCAI 2007*, pages 1898–1903, 2007.

[Helmert, 2006] Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

[Holte *et al.*, 2015] Robert C. Holte, Yusra Alkhazraji, and Martin Wehrle. A generalization of sleep sets based on operator sequence redundancy. In *Proc. AAAI 2015*, pages 3291–3297, 2015.

[Imrich and Peterin, 2007] Wilfried Imrich and Iztok Peterin. Recognizing cartesian products in linear time. *Discrete Mathematics*, 307(3-5):472–483, 2007.

[Kelareva *et al.*, 2007] Elena Kelareva, Olivier Buffet, Jinbo Huang, and Sylvie Thiébaux. Factored planning using decomposition trees. In *Proc. IJCAI 2007*, pages 1942–1947, 2007.

[Knoblock, 1994] Craig A. Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68(2):243–302, 1994.

[Nissim *et al.*, 2012] Raz Nissim, Udi Apsel, and Ronen I. Brafman. Tunneling and decomposition-based state reduction for optimal planning. In *Proc. ECAI 2012*, pages 624–629, 2012.

[Sabidussi, 1960] Gert Sabidussi. Graph multiplication. *Mathematische Zeitschrift*, 72:446–457, 1960.

[Seipp and Helmert, 2011] Jendrik Seipp and Malte Helmert. Fluent merging for classical planning problems. In *ICAPS 2011 Workshop on Knowledge Engineering for Planning and Scheduling*, pages 47–53, 2011.

[Sievers *et al.*, 2012] Silvan Sievers, Manuela Ortlieb, and Malte Helmert. Efficient implementation of pattern database heuristics for classical planning. In *Proc. SoCS 2012*, pages 105–111, 2012.

[Valmari, 1989] Antti Valmari. Stubborn sets for reduced state space generation. In *Proc. APN 1989*, pages 491–515, 1989.

[van den Briel *et al.*, 2007] Menkes van den Briel, Subbarao Kambhampati, and Thomas Vossen. Fluent merging: A general technique to improve reachability heuristics and factored planning. In *ICAPS 2007 Workshop on Heuristics for Domain-Independent Planning*, 2007.

[Wehrle and Helmert, 2012] Martin Wehrle and Malte Helmert. About partial order reduction in planning and computer aided verification. In *Proc. ICAPS 2012*, 2012.

[Wehrle and Helmert, 2014] Martin Wehrle and Malte Helmert. Efficient stubborn sets: Generalized algorithms and selection strategies. In *Proc. ICAPS 2014*, pages 323–331, 2014.

[Winkler, 1987] Peter Winkler. Factoring a graph in polynomial time. *European Journal of Combinatorics*, 8:209–212, 1987.