Act Local, Think Global: Width Notions for Tractable Planning

Hubie Chen
Dept. of Information and Communication Technologies
Universitat Pompeu Fabra
Passeig de Circumval·lació, 8
08003 Barcelona, Spain
hubie.chen@upf.edu

Omer Giménez
Dept. of Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Jordi Girona, 1-3
08034 Barcelona, Spain
omer.gimenez@upc.edu

Abstract

Many of the benchmark domains in AI planning are tractable on an individual basis. In this paper, we seek a theoretical, domain-independent explanation for their tractability. We present a family of structural conditions that both imply tractability and capture some of the established benchmark domains. These structural conditions are, roughly speaking, based on measures of how many variables need to be changed in order to move a state closer to a goal state.

Introduction

Background and motivations. Many of the benchmark domains in AI planning such as *Blocksworld*, *Gripper*, and *Logistics* are structurally simple when looked at individually. For these domains, the problem of plan generation–generate a (non-optimal) plan if one exists–is not only polynomial-time tractable, but can be solved by extremely simple and efficient algorithms. Interestingly, in addition to being *individually* tractable with respect to polynomial-time computation, several benchmark domains can be effectively handled simultaneously by *domain-independent planners* (Hoffmann & Nebel 2001). As a result strongly evidencing this latter claim, we name the recent work of (Vidal & Geffner 2005) giving a planning algorithm which, by use of inference, solves instances of a number of benchmark domains in a backtrack-free manner.

The *empirically observed* domain-independent tractability of many common benchmark domains naturally calls for a *theoretical explanation*: are there tractable classes of planning problems that simultaneously capture a number of the benchmark domains? Here, by a theoretical explanation, we mean that one that is mathematical and rooted in the theory of computational complexity: by a tractable class, we mean a set of problem instances which can be solved in polynomial time and is delimited by a *formal, mathematical definition*, and we are interested in demonstrations that a tractable class contains a domain, that are given by *mathematical proof*. There is certainly a literature that has studied theoretically tractable classes in planning. However, to the best of our knowledge this literature has limited coverage of benchmark domains, and has instead focused, for example, on syntactic restrictions on the operator set, as in (Bylander 1994; Bäckström & Nebel 1995; Erol, Nau, & Subrahmanian 1995), or restrictions on causal graph structure, as in (Brafman & Domshlak 2003; 2006; Helmert 2006). There is related work in (Helmert 2004; 2006; Haslum 2007) which we contrast with our work later in this section.

Now, what would constitute a satisfactory theoretical account for the domain-independent tractability of benchmark domains? As we have already mentioned, the benchmark domains of interest are already known to be tractable on an *individual* basis. What we propose is worth searching for, then, is a *uniform* explanation for the tractability of benchmark domains. Since we are searching for tractable classes that are given by formal definitions, from the viewpoint of favoring uniformity, we are interested in definitions that are as simple as possible, and are preferrably based on a single concept (or very few concepts). In other words, we believe that one would like to identify (via a formal definition) a generic structural property that accounts for the tractability for the benchmark domains.

A related property that we believe is desirable for the definition of tractable classes is mathematical tangibility, by which we mean that it should be relatively easy to prove both inclusion results ("domain D is contained in tractable class C") as well as non-inclusion results. Clearly, tangibility is closely tied to simplicity, and may be viewed as evidence for simplicity. In addition, mathematical tangibility of a tractable class yields human understanding of the class and its boundaries; understanding why a problem does not belong to a class–that is, understanding a proof of non-inclusion–can yield human insight into domain structure in the same way that understanding an inclusion result can. Put differently, tangibility may allow us to *classify* domains according to their structural properties.

Contributions. Guided by these questions and motivations, in this paper we identify a family of simple structural conditions that both imply tractability and apply to multiple benchmark domains–including *Gripper* and *Logistics*. Not only are our introduced conditions simple to understand and handle theoretically, but the corresponding algorithms which demonstrate tractability are also conceptually simple.

More precisely, we identify four related measures of complexity for planning problems. The most basic one we call *width*; the others are extensions of this measure. Roughly speaking, a planning instance has *width* k if for every state s, we can bring any variable u to its goal via a plan that changes no more than k variables (assuming that a goal state is reachable). The plan may change any variables, but upon termination must restore all variables that were previously in the goal state. We show that a set of planning instances having bounded width (width bounded above by a constant) is polynomial-time tractable, by the following algorithm, which we simply call the *width* k *algorithm*. This algorithm, given a planning instance, arbitrarily chooses an ordering of the variables and, according to this ordering, tries to bring each of them into their goal states (via plans of the mentioned form). Note that we prove that this algorithm, given an instance of bounded width, solves the instance regardless of the variable ordering chosen; that is, the algorithm behaves correctly on all variable orderings. As an example, we show that the *Gripper* domain has bounded width.

After defining this notion of width, we define and study two orthogonal extensions of this notion. The first is *persistent width*; the definition of *persistent width* k is similar to the definition of width k, except instead of requiring that for every state s, *every* variable u can be brought to its goal, it is merely required that for every state s, there *exists* a variable u that can be brought to its goal. The second extension of width is *Hamming width*; the definition of *Hamming width* k is similar to the definition of width k, except when trying to bring a variable into its goal state, we do not only consider plans that change no more than k variables, but the more general class of plans that stay within Hamming distance k from the start state. In fact, each of these two extensions is motivated by giving a domain that does not have bounded width, but does have bounded persistent width or bounded Hamming width.

In total, we obtain four measures of complexity for planning instances: width, persistent width, Hamming width, and persistent Hamming width, the latter of which is the natural unification of persistent width and Hamming width. Again, we show that if some set of planning instances is bounded with respect to one of these measures, then the set is polynomial-time tractable. We remark that persistent Hamming width is the "most powerful" of these measures in the sense that the conditions of bounded width, bounded persistent width, and bounded Hamming width each imply the condition of bounded persistent Hamming width.

After introducing these measures, their corresponding algorithms, and examples, we investigate the robustness of these measures. First, we establish that our width measures exhibit *action monotonicity*, which roughly means that adding actions to a planning instance does not increase the width. Put differently, for each of our tractable classes, the addition of actions renders an instance more likely to fall into the tractable class. The underlying intuition here is that our width measures are concerned with the existence of certain types of plans, and adding actions to an instance certainly preserves such existence. We would like to emphasize that tractable classes based on the "sparseness" of the *causal graph* do *not* exhibit action monotonicity. For example, (Brafman & Domshlak 2006) present a tractability result that applies when the treewidth of the causal graph is bounded; however, adding actions to a planning problem can only enlarge the edge set of the causal graph, which in turn increases the treewidth. Indeed, tractable classes of this form exhibit *action non-monotonicity*: the *removal* of actions renders an instance more likely to fall into the class.

We also define and study a notion of *reformulation*. We prove–roughly speaking–that if a set of planning instances $\mathcal{C}$ has bounded persistent Hamming (PH) width, then any *reformulation* of this set also has bounded PH width. That is, the concept of bounded PH width is robust with respect to reformulation. We give a natural definition of the *binary formulation* $\mathcal{B}(\Pi)$ of a multi-valued instance Π and show that for any instance Π, the binary formulation $\mathcal{B}(\Pi)$ and the instance Π itself are reformulations of each other, and hence the robustness result applies.

Finally, we study two types of composition operations, showing that, for various width measures, the set of instances having width k is closed under these composition operations.

Related work. Our notions of width can be viewed as elaborations of the notion of *subgoal serializability* (Korf 1987), and we believe that it may be didactic to contrast our notions with this one. A set of subgoals is defined to be *serializable* if there exists an ordering of the subgoals such that they can be achieved sequentially without violating previous subgoals in the ordering. In contrast, if a planning instance has width k, this implies that for *any* ordering of its variables, each variable may be brought into its goal states via a plan of a "limited form". Here, after the plan for each variable is executed, it must be the case that all previous variables are also in their goal state; however, in contrast to subgoal serializability, the plan for a variable is permitted to change the state of the previous variables so long as their goal state is restored after the plan. Also, while we show that having bounded width (for any of our width notions) implies tractability, subgoal serializability is known not to guarantee tractability. In this vein, one might also mention related work by Barrett and Weld (Barrett & Weld 1993) which refines Korf's work. The present paper could be viewed as a *quantitative* elaboration of the ideas in these works which allows one to assign a complexity measure to a given planning domain.

We now describe related work on tractability. Many of the early tractability results on planning focused on syntactic restrictions on the set of operators; see for example (Bylander 1994; Bäckström & Nebel 1995; Erol, Nau, & Subrahmanian 1995). Jonsson and Bäckström (1994b; 1994a) presented complexity results based on restrictions on the domain-transition graph for each variable.

Planning with unary operators was considered by (Brafman & Domshlak 2003), who gave a number of tractability and complexity results based on the structure of the causal graph. As already mentioned, (Brafman & Domshlak 2006) gave a tractability result based on the treewidth of a causal

graph; their work followed (Amir & Engelhardt 2003) which investigated a notion of factoring.

(Helmert 2004; 2006) also presents a tractability result based on the causal graph structure. (Haslum 2007) (implicitly) defines a tractable class using a set of multiple simplification rules. In contrast, here we aim to define tractable classes via single structural conditions. Evaluation in these papers is focused on empirical results.

We want to emphasize that we believe that the tractable classes of the present paper offer a new perspective on the benchmark domains that is different from that given in the mentioned papers, and reveals a shared structural property among the studied domains that has not previously been identified in its form here. We would also like to add that, to the best of our knowledge, previously presented tractable classes have not been demonstrated to be closed under the transformations considered here, such as reformulation.

Note: due to space restrictions, some of the proofs have been omitted.

Preliminaries

An instance of the planning problem is a tuple $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ whose components are described as follows.

- V is a finite set of variables, where each variable $v \in V$ has an associated finite domain $D(v)$. Note that variables are not necessarily propositional, that is, $D(v)$ may have any finite size. A *state* is a mapping s defined on the variables V such that $s(v) \in D(v)$ for all $v \in V$. A *partial state* is a mapping p defined on a subset $\mathsf{vars}(p)$ of the variables V such that for all $v \in \mathsf{vars}(p)$, it holds that $p(v) \in D(v)$.
- init is a state called the *initial state*.
- goal is a partial state.
- A is a set of *operators*; each operator $a \in A$ consists of a *precondition* $\mathsf{pre}(a)$, which is a partial state, as well as a *postcondition* $\mathsf{post}(a)$, also a partial state. We sometimes denote an operator a by $\langle \mathsf{pre}(a); \mathsf{post}(a) \rangle$.

Note that when s is a state or partial state, and W is a subset of the variable set V, we will use $(s \restriction W)$ to denote the partial state resulting from restricting s to W. We say that a state s is a *goal state* if $(s \restriction \mathsf{vars}(\mathsf{goal})) = \mathsf{goal}$.

We define a *plan* (for an instance Π) to be a sequence of operators $P = a_1, \ldots, a_n$. Starting from a state s, we define the state resulting from s by applying a plan P, denoted by $s[P]$, inductively as follows. For the empty plan $P = \epsilon$, we define $s[\epsilon] = s$. For non-empty plans P, denoting $P = P', a$, we define $s[P', a]$ as follows.

- If $(s[P'] \restriction \mathsf{vars}(\mathsf{pre}(a))) \neq \mathsf{pre}(a)$ (that is, the precondition of a does not hold in $s[P']$) then $s[P', a] = s[P']$.
- Otherwise, $s[P', a]$ is the state equal to $\mathsf{post}(a)$ on variables $v \in \mathsf{vars}(\mathsf{post}(a))$, and equal to $s[P']$ on variables $v \in V \setminus \mathsf{vars}(\mathsf{post}(a))$.

We say that a state s is *reachable* (in an instance Π) if there exists a plan P such that $s = \mathsf{init}[P]$. We are concerned with the problem of *plan generation*: given an instance $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ obtain a plan P that *solves* it, that is, a plan P such that $\mathsf{init}[P]$ is a goal state.

Width

In this section, we define the notion of width. First, some definitions are required. Let $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ be an instance. We say that a plan P *improves* variable u in state s if:

- for all $v \in V$, if $v \in \mathsf{vars}(\mathsf{goal})$ and $s(v) = \mathsf{goal}(v)$, then $(s[P])(v) = \mathsf{goal}(v)$; and,
- if $u \in \mathsf{vars}(\mathsf{goal})$, then $(s[P])(u) = \mathsf{goal}(u)$.

That is, after P is executed, all variables that had values as in the goal state still have values as in the goal state, and in addition, the variable u is in the goal state.

Now, let W be a subset of V. We say that a plan P *uses only the variables* W if for every operator a in P, it holds that $\mathsf{vars}(\mathsf{post}(a)) \subseteq W$. We say that a variable u is *k-improvable* in state s if there exists a plan P and a subset $W \subseteq V$ of size $|W| \leq k$ such that P uses only the variables W and improves u in s. In general, we will use k to denote an integer greater than or equal to 1.

For any state s, we define $\mathsf{wrong}(s) = \{v \in \mathsf{vars}(\mathsf{goal}) : s(v) \neq \mathsf{goal}(v)\}$. The set $\mathsf{wrong}(s)$ is the set of variables on which s differs from the goal.

Definition 1 *A planning instance has* width k *if no plan solving it exists, or for every reachable state s that is not a goal state, every variable $u \in \mathsf{wrong}(s)$ is k-improvable in s.*[1]

We define the width k algorithm as follows. The algorithm is given an instance Π.

- Pick an arbitrary ordering $v_1, \ldots, v_n$ of the variables.
- Set s to be the initial state init.
- Set Q to be the empty plan ϵ.
- Loop from $i = 1, \ldots, n$:
  if $v_i \in \mathsf{wrong}(s)$, try to find a plan P using at most k variables that improves v_i in s. If such a plan P is found, replace s with $s[P]$, and append P to Q. If no plan is found, output "?" and halt.
- Output Q.

Theorem 2 *Let $\mathcal{C}$ be a set of planning instances having width k. The plan generation problem for $\mathcal{C}$ is solvable in polynomial time via the width k algorithm, in time $O(n^{k+1}d^k a)$. Here, n denotes the number of variables, d denotes the maximum size of a domain, and a denotes the number of actions.*

Note that we say that an algorithm solves the plan generation problem for $\mathcal{C}$ if on every instance Π from $\mathcal{C}$, a plan solving Π is output by the algorithm if and only if a goal state is reachable.

[1] In this definition, one can drop the requirement that s not be a goal state, since in the case that s is a goal state, $\mathsf{wrong}(s)$ is empty. We elected the given definition because it reflects our usage of the defined concept and for symmetry with the definitions of the other width notions.

Proof. Clearly, if the algorithm outputs a plan, it is correct. Assume that a plan exists for an instance Π of $\mathcal{C}$.

We prove by induction that, before the ith iteration of the loop, none of the variables $v_1, \ldots, v_{i-1}$ are in $\mathsf{wrong}(s_i)$. Here, s_i denotes the state s before the ith iteration. The base case $i = 1$ is trivial. For the induction, we observe that since Π has width k and a plan exists, the variable v_i is either not in $\mathsf{wrong}(s_i)$ or k-improvable. In the first case, the algorithm does nothing in the ith iteration and thus $v_i \notin \mathsf{wrong}(s_{i+1})$, and in the second case, the algorithm finds a plan improving v_i in s_i.

To search for a plan improving v_i, we do the following. For every choice of k variables W, we create a graph that has as vertices the at most d^k states that differ from s_i at only variables in W. There is an edge from vertex s to vertex s' if there exists an action a such that $s[a] = s'$. We perform a search to check if some vertex s such that $\mathsf{wrong}(s) \cap \{v_1, \ldots, v_i\} = \emptyset$ is reachable from s_i. This search can be performed in time $O(|X| + |E|)$, where $|X|$ is the number of vertices and $|E|$ is the number of edges. We have $|X| \leq d^k$ and $|E| = d^k a$, so this search can be performed in time $O(d^k a)$. Note that creating the graph can also be performed in time $O(d^k a)$.

Since there are $\binom{n}{k}$ choices for W, and we potentially improve n variables, the total running time is bounded by $O(n^{k+1} d^k a)$. □

As an example, we show that all instances of the Gripper domain have bounded width. For simplicity, we consider a variation of the Gripper domain where there is only one hand. It is to easy that the same proof with the same width bounds applies no matter the number of holding devices the robot has.

Domain 3 (Gripper domain) In the Gripper domain, we have a robot with a hand that can pick up and drop balls, and move them from one location to another. The hand can hold one ball at a time. Formally, in an instance $(V, \mathsf{init}, \mathsf{goal}, A)$ of the Gripper domain, there is a set of balls B, and a set of locations L. The variable set V is defined as $B \cup \{\mathsf{pos}, \mathsf{hand}\}$ where $D(\mathsf{pos}) = L$, $D(\mathsf{hand}) = B \cup \{\mathsf{empty}\}$, and for all $b \in B$, $D(b) = L \cup \{\mathsf{hand}\}$.

There are three kinds of actions.

- $\forall l, l' \in L$, $\mathsf{move}_{l,l'} = \langle \mathsf{pos} = l; \mathsf{pos} = l' \rangle$
- $\forall l \in L, b \in B$, $\mathsf{drop}_{l,b} = \langle \mathsf{pos} = l, \mathsf{hand} = b; \mathsf{hand} = \mathsf{empty}, b = l \rangle$
- $\forall l \in L, b \in B$, $\mathsf{pick}_{l,b} = \langle \mathsf{pos} = l, \mathsf{hand} = \mathsf{empty}, b = l; \mathsf{hand} = b, b = \mathsf{hand} \rangle$

We remark that we consider only instances of the Gripper domain that have a *consistent* initial state, by which we mean that $(\mathsf{init}(\mathsf{hand}) = b \Leftrightarrow \mathsf{init}(b) = \mathsf{hand})$ holds. □

Theorem 4 *All instances of Gripper have width* 4.

Proof (Sketch). The proof is not hard, but there are several details that need to be addressed in a case by case basis (after all, the proof must depend on the precise definition of the actions of the Gripper domain). We briefly explain the reason why Gripper has bounded, small width.

Let s be a reachable, non-goal state of a Gripper instance, and let u be a variable to improve in s. To improve it, start by dropping whatever ball the robot may hold, so that the arm becomes free, and then bring the variable u to its goal value, by moving the robot and using the arm if necessary. Finally, to guarantee that no variable that had in state s the same value as in the goal state has now a wrong value, we make the robot hold the ball $\mathsf{goal}(\mathsf{hand})$, if not empty, and move to $\mathsf{goal}(\mathsf{pos})$, if it is not already there. Clearly this plan only changes the values of variables pos, hand and 3 balls, so that Gripper has width 5. With a bit more care we see that, in fact, we do not need to change the values of more than 2 balls, so that Gripper has width 4. □

Corollary 5 *The plan generation problem for all instances of Gripper is solvable in polynomial time via the width* 4 *algorithm.*

Proof. Immediate from Theorems 2 and 4. □

Persistent width

In this section, we present the notion of *persistent width*. As a motivating example, we consider the *Unlock* domain, which we show does not have bounded width, but does have bounded persistent width.

Domain 6 (Unlock domain) This domain is based on the benchmark *Grid* domain; the differences are that here we permit movement in an arbitrary graph as opposed to a grid graph, and we require that all locations are unlocked in the goal state. For simplicity we consider only domains where each key opens a single location, although the same result applies for any arbitrary relation between keys and locations they open.

In the Unlock domain, we have a robot that moves among a set of locations. Each location is either locked or unlocked. To unlock a location, the robot has to pick up a key for that location and unlock the location from an adjacent location. Formally, in an instance of this domain, there is a set of locations L, a set of keys K, a function $f : K \to L$, and an undirected graph G with vertex set L. The variable set V is defined as $L \cup K \cup \{\mathsf{pos}, \mathsf{hand}\}$. For each $l \in L$, $D(l) = \{\mathsf{locked}, \mathsf{unlocked}\}$. For each $t \in K$, $D(t) = L \cup \{\mathsf{hand}\}$. In addition, $D(\mathsf{pos}) = L$ and $D(\mathsf{hand}) = K \cup \{\mathsf{empty}\}$. The actions are as follows.

- $\forall \{l, l'\} \in E(G)$ $\mathsf{move}_{l,l'} = \langle l = \mathsf{unlocked}, l' = \mathsf{unlocked}, \mathsf{pos} = l; \mathsf{pos} = l' \rangle$
- $\forall l \in L$, $\forall t \in K$, $\mathsf{drop}_{l,t} = \langle \mathsf{pos} = l, t = \mathsf{hand}; t = l, \mathsf{hand} = \mathsf{empty} \rangle$
- $\forall l \in L$, $\forall t \in K$, $\mathsf{pick}_{l,t} = \langle \mathsf{pos} = l, \mathsf{hand} = \mathsf{empty}, t = l; t = \mathsf{hand}, \mathsf{hand} = t \rangle$
- $\forall t \in K$, $\forall l \in L$ such that $\{f(t), l\} \in E(G)$, $\mathsf{unlock}_{l,t} = \langle \mathsf{pos} = l, \mathsf{hand} = t; f(t) = \mathsf{unlocked} \rangle$

Note that we consider only initial states where $(\mathsf{init}(\mathsf{hand}) = t \Leftrightarrow \mathsf{init}(t) = \mathsf{hand})$, and we assume that the goal state specifies that all locations are unlocked. □

Theorem 7 *For each $k \geq 1$, there exists an instance Π_k of the Unlock domain such that Π_k does not have width k.*

Proof. We show such an instance Π_k. Let $L = K = \{1, \ldots, k\}$, let f be $f(t) = t$ for all $t \in K$, and let G be the graph with edges $\{i, i+1\}$ for all i in $[1, k-1]$. Initially, the instance Π_k has all keys and the robot at location 1, and all locations are locked. The instance does not have width k because, starting at the initial state, there exists a variable u (the one corresponding to the k-th location) that cannot be improved without unlocking all intermediate locations. □

Definition 8 *A planning instance has* persistent width k *if no plan exists, or for every reachable state s that is not a goal state, there exists a variable $u \in \mathsf{wrong}(s)$ such that u is k-improvable in s.*

Notice that if an instance Π has width k, it also has persistent width k.

The corresponding persistent width k algorithm closely resembles the width k algorithm, except that the algorithm does not initially create a pre-established ordering $v_1, \ldots, v_n$ of the variables, but an ordering is created during the execution of the algorithm. In the ith iteration variables $v_1, \ldots, v_{i-1}$ have already been defined, and the algorithm tries to find a k-improvable variable among the remaining ones. If one is found, that one becomes v_i; otherwise, the algorithm outputs "?" and halts. To find such a k-improvable variable v, we perform the following subroutine that can be carried out in time $O(n^k d^k a)$. For every choice of k variables W, we create a graph as in the width k algorithm, but our search looks for a vertex s such that $\mathsf{wrong}(s) \cap \{v_1, \ldots, v_{i-1}, v\} = \emptyset$ for some variable v outside of $v_1, \ldots, v_{i-1}$.

Theorem 9 *Let $\mathcal{C}$ be a set of planning instances having persistent width k. The plan generation problem for $\mathcal{C}$ is solvable in polynomial time via the persistent width k algorithm in time $O(n^{k+1} d^k a)$.*

Theorem 10 *All instances of the Unlock domain have persistent width* 5.

Proof. Let Π be an instance of the Unlock domain. We assume Π is solvable (otherwise it has persistent width 5 by definition). Any plan $\tilde{P}$ that solves Π induces an ordering on the locations of Π: say $l < l'$ if location l was unlocked *before* location l' during the course of the plan $\tilde{P}$.

Let s be a reachable state of Π that is not a goal state. Let l be the smallest locations in l such that $s(l) = \mathsf{locked}$. We show that it is possible to 5-improve the variable $u = l$ by means of the following plan P.

- Drop the key $s(\mathsf{hand})$ (if the hand is not empty) in location $l_1 = s(\mathsf{pos})$.
- Move from l_1 to the location l_2 where the key to unlock l is.
- Pick it up.
- Move to a neighbour location l_3 of l.
- Unlock l with the key.
- Move back to l_2.
- Drop the key.
- Move back to l_1.
- Pick up the key $s(\mathsf{hand})$ (if the hand was not empty at the beginning).

Clearly the plan P 5-improves l ($(s[P])(l) = \mathsf{goal}(l)$ and $(s[P])(v) = s(v)$ for all remaining variables). All it remains to show is that the movements described in the plan are feasible, that is, there is a path between l_1 and l_2, and a path between l_2 and l_3, that is not blocked by unlocked locations. Let U be a set of locations, and let $G[U]$ be the graph that describes valid movements when the only unlocked locations are those in U, namely, $G[U]$ is the restriction of G to the set of vertices U. Let U be the set of locations smaller than the location l we are trying to improve, and let U' be the set of locations unlocked in state s. We know that l_1 is connected to $\mathsf{init}(\mathsf{pos})$ in $G[U']$ (otherwise the state s would not be reachable). The plan P would be feasible if both l_2 and l_3 are also connected to $\mathsf{init}(\mathsf{pos})$ in $G[U']$. The way l was chosen implies that $U \subset U'$, thus $G[U] \subset G[U']$. The fact that plan $\tilde{\Pi}$ managed to unlock location l implies that location l_2 and at least a neighbour location l_3 of l are connected to $\mathsf{init}(\mathsf{pos})$ in $G[U]$. It follows from $G[U] \subset G[U']$ that l_2 and l_3 are also connected to $\mathsf{init}(\mathsf{pos})$ in $G[U']$

It remains to consider the case when all locations in s are unlocked. Then the problem is the same as Gripper, except for the restrictions the graph G imposes on the movements. The fact that Π is solvable means that we can restrict to the connected component the robot starts in in order to reach a goal state. Hence any action $\mathsf{move}_{l,l'}$ appearing in a plan P of Π when seen as an instance of Gripper can be safely translated to a sequence of move actions in Π when seen as an instance of Unlock (namely, a path in G from l to l'). By means of this translation it follows that we can 4-improve any variable u in s, due to the Theorem 4. □

Hamming width

This section presents the notion of Hamming width. As in the previous section, we consider a domain to motivate this extension of width: we show that an extension of the *Logistics* domain does not have bounded width, but does have bounded Hamming width.

Domain 11 (Logistics domain) In our formulation of the Logistics domain, we have trucks and airplanes that can be used to move packages between locations. Trucks and airplanes cannot in general move arbitrarily between pairs of locations, but rather, graphs specify permitted movements. Note that, in contrast to a typical formulation of this domain, we permit arbitrary graphs to specify movements.

In an instance of the Logistics domain, there is a set of locations L, a set of trucks T, a set of airplanes A, and a set of packages Q, and two undirected graphs G_T and G_A, both having L as vertex set. The variable set is the union $T \cup A \cup Q$, where $D(t) = L$ for all $t \in T$, $D(a) = L$ for all $a \in A$, and $D(p) = L \cup A \cup T$ for all $p \in Q$.

- $\forall t \in T$ and $\forall \{l, l'\} \in E(G_T)$ $\mathsf{move}_{t,l,l'} = \langle t = l; t = l' \rangle$
- $\forall a \in A$ and $\forall \{l, l'\} \in E(G_A)$ $\mathsf{move}_{a,l,l'} = \langle a = l; a = l' \rangle$
- $\forall v \in T \cup A$, $\forall l \in L$, $\forall p \in Q$, $\mathsf{drop}_{v,l,p} = \langle v = l, p = v; p = l \rangle$

- $\forall v \in T \cup A$, $\forall l \in L$, $\forall p \in Q$, $\mathsf{pick}_{v,l,p} = \langle v = l, p = l; p = v \rangle$

□

Theorem 12 *For each $k \geq 1$, there exists an instance Π_k of the logistics domain such that Π_k does not have width k.*

Proof. We show such an instance Π_k. Let $L = \{1, \ldots, k+1\}$, and let A and T be such that for any $i \in \{1, \ldots, k\}$ we have a truck t_i if i is odd, and an airplane a_i if i is even. Let G_T be the graph on L with edges $\{i, i+1\}$ for i odd, and G_A be the graph with edges $\{i, i+1\}$ for i even. Let Q contain a single packet p such that $\mathsf{init}(p) = 1$ and $\mathsf{goal}(p) = k+1$. Let this be the only goal. It follows easily that init is not k-improvable, because the only variable $p \in \mathsf{wrong}(\mathsf{init})$ cannot be brought to its destination $\mathsf{goal}(p)$ without using the k trucks and airplanes. □

We say that a variable u is *k-Hamming improvable* in state s if there exists a plan $P = a_1, \ldots, a_n$ improving u in s such that for all $i = 1, \ldots, n$, $\mathsf{d_h}(s, s[a_1, \ldots, a_i]) \leq k$. Here, $\mathsf{d_h}(\cdot, \cdot)$ denotes the Hamming metric, that is, the number of variables at which the two arguments differ.

Definition 13 *A planning instance has* Hamming width k *if no plan exists, or for every reachable state s that is not a goal state, every variable $u \in \mathsf{wrong}(s)$ is k-Hamming improvable in s.*

To obtain the corresponding Hamming width k algorithm we just need to make a minor change to the width k algorithm. The inner loop now looks for plans P that improve the variable v_i without differing from state s more than k variables at any time, irrespective of how many variables P uses in total. Note that to improve one variable in a state s, we only need to search a single graph whose vertices are all states within Hamming distance k from s.

Theorem 14 *Let $\mathcal{C}$ be a set of planning instances having Hamming width k. The plan generation problem for $\mathcal{C}$ is solvable in polynomial time via the Hamming width k algorithm, in time $O(n^{k+1}d^k a)$.*

Theorem 15 *All instances of the Logistics domain have Hamming width* 2.

Proof (Sketch). Let s be reachable state in a solvable instance Π, and let u be a variable to Hamming improve in s. When u is a vehicle it is enough to consider a sequence of move actions; when u is a package, we may need to use several vehicles to bring u to its goal location. The Hamming width is 2 because this can be done in such a way that, at any time, at most one vehicle v needs to stay away from its initial location $s(v)$ to move the package: when v has done its share, it returns to $s(v)$. □

Persistent Hamming width

Definition 16 *A planning instance has* persistent Hamming width k *if no plan exists, or for every reachable state s that is not a goal state, there exists a variable $u \in \mathsf{wrong}(s)$ such that u is k-Hamming improvable in s.*

The corresponding persistent Hamming width k algorithm is just the combination of the improvements of the persistent width k and the Hamming width k algorithms.

Theorem 17 *Let $\mathcal{C}$ be a set of planning instances having persistent Hamming width k. The plan generation problem for $\mathcal{C}$ is solvable in polynomial time via the persistent Hamming width k algorithm, in time $O(n^{k+1}d^k a)$.*

The persistent Hamming width k algorithm gives a uniform explanation of the tractability of all of the domains considered so far.

Theorem 18 *Let $\mathcal{C}$ be the set of instances from the Gripper, Unlock, and Logistics domains. The plan generation problem for $\mathcal{C}$ is solvable in polynomial time via the persistent Hamming width* 5 *algorithm.*

Action monotonicity

In this section, we establish that the introduced width measures exhibit *action monotonicity*: adding actions that do not enlarge the set of reachable states does not increase width. (Note that we use the abbreviations P, H, and PH for persistent, Hamming, and persistent Hamming, respectively.)

Theorem 19 *Let $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ be a planning instance having width k (respectively, P width k, H width k, PH width k). Let A' be a set of actions that is a superset of A such that every reachable state in $\Pi' = (V, \mathsf{init}, \mathsf{goal}, A')$ is reachable in Π. Then, the instance Π' has width k (respectively, P width k, H width k, PH width k).*

Proof. We prove this in the case of width k; the proof is similar for the other notions of width. Assume that Π has width k. If there is no plan solving Π', then by definition it has width k. Otherwise, in Π', let s be a reachable state that is not a goal state, and let u be a variable in $\mathsf{wrong}(s)$. By assumption s is reachable in Π. Since Π and Π' have the same goal state, s is not a goal state in Π, and $u \in \mathsf{wrong}(s)$ with respect to Π. Since Π has width k, there exists a plan P that k-improves the variable u in state s, in the instance Π. The same plan P also k-improves u in state s in the instance Π'. □

Reformulation theorem

This section defines a notion of reformulation and demonstrates that, relative to this notion, if an instance Π' is a reformulation of an instance Π, then bounded PH width of Π implies the bounded PH width of Π'. We then define the binary formulation $\mathcal{B}(\Pi)$ of an instance Π and show that, with respect to our definition of reformulation, the instances Π and $\mathcal{B}(\Pi)$ are always reformulations of each other. We can consequently observe that any class of instances $\mathcal{C}$ has bounded PH width if and only if the binary formulation $\mathcal{B}(\mathcal{C})$ does. This may be viewed as a robustness result for the concept of bounded PH width.

Definition 20 *Let $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ and $\Pi' = (V', \mathsf{init}', \mathsf{goal}', A')$ be instances, and let S and S' denote the set of states of Π and Π', respectively. The instance Π' is a* reformulation *of the instance Π with non-decreasing*

blowup function $b : \mathbb{N} \to \mathbb{N}$ *if there exists a relation* $R \subseteq S \times S'$ *such that the following conditions hold.*

1. *For every reachable* Π'*-state* $s' \in S'$*, there exists a reachable* Π*-state* $s \in S$ *such that* $(s, s') \in R$
2. *there exists a function* $t : A \to A'$ *such that for all* $(s, s') \in R$*, and all actions* $a \in A$*, it holds that* $(a(s), (t(a))(s')) \in R$
3. *for all* $(s, s') \in R$*,* s *is a goal state in* Π *if and only if* s' *is a goal state in* Π'
4. *if* $(s_1, s'_1), (s_2, s'_2) \in R$ *and* $\mathsf{wrong}(s_1) \subsetneq \mathsf{wrong}(s_2)$*, then* $\mathsf{wrong}(s'_1) \subsetneq \mathsf{wrong}(s'_2)$
5. *for all* $(s_1, s'_1), (s_2, s'_2) \in R$*, it holds that* $\mathsf{d_h}(s'_1, s'_2) \leq b(\mathsf{d_h}(s_1, s_2))$

Theorem 21 *If instance* $\Pi' = (V', \mathsf{init}', \mathsf{goal}', A')$ *is a* reformulation *of instance* $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ *with blowup function* $b : \mathbb{N} \to \mathbb{N}$*, and* Π *has persistent Hamming width* k*, then* Π' *has persistent Hamming width* $b(k)$*.*

Proof. We first make a general observation. Let $P = a_1, \ldots, a_n$ be any plan in Π and let $(s, s') \in R$ such that for all $i = 1, \ldots, n$, $\mathsf{d_h}(s, s[a_1, \ldots, a_i]) \leq k$. By successive application of condition 2 it follows that for all $i = 1, \ldots, n$,

$$(s[a_1, \ldots, a_i], s'[t(a_1), \ldots, t(a_i)]) \in R$$

and by condition 5 it follows that for all $i = 1, \ldots, n$, $\mathsf{d_h}(s', s'[t(a_1), \ldots, t(a_i)]) \leq b(\mathsf{d_h}(s, s[a_1, \ldots, a_k])) \leq b(k)$.

Now we prove the theorem. Let s' be a reachable Π'-state that is not a goal state. By condition 1 in Definition 20 there exists a reachable Π-state s such that $(s, s') \in R$; the state s is not a goal state by condition 3. Thus, there exists a plan $P = a_1, \ldots, a_n$ that k-Hamming improves a variable u in state s.

Let P' denote the plan $t(a_1), \ldots, t(a_n)$. By the observation, we have $(s[P], s'[P']) \in R$. Note that $\mathsf{wrong}(s[P]) \subsetneq \mathsf{wrong}(s)$. By condition 4, $\mathsf{wrong}(s'[P']) \subsetneq \mathsf{wrong}(s')$. Let u' be any variable in $\mathsf{wrong}(s') \setminus \mathsf{wrong}(s'[P'])$. Thus, the plan P' improves u' in s'. Moreover, the plan P' k-Hamming improves u' in s', since by the observation, for all $i = 1, \ldots, n$, $\mathsf{d_h}(s', s'[t(a_1), \ldots, t(a_i)]) \leq b(k)$. □

We remark that this theorem does not hold for width nor persistent width.

The binary formulation $\mathcal{B}(\Pi)$ of a planning instance $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ is described as follows. Consider a new variable v_d for any pair $v \in V$ and $d \in D(v)$, and define $\mathcal{B}V$ as the set of all such variables, namely $\mathcal{B}(V) = \bigcup_{v \in V, d \in D(v)} v_d$. For each variable $w \in \mathcal{B}(V)$, we define $D(w) = \{0, 1\}$. That is, for every variable $v \in V$, one binary variable is introduced for each domain element in $D(v)$. For a partial state p of Π, we define $\mathcal{B}(p)$ to be the partial state defined on the variables $\bigcup_{v \in \mathsf{vars}(p), d \in D(v)} v_d$ such that $(\mathcal{B}(p))(v_d)$ is 1 if $p(v) = d$, and 0 otherwise. For an action $a \in A$ of Π, we define $\mathcal{B}(a)$ to be the action with precondition $\mathcal{B}(\mathsf{pre}(a))$ and with postcondition $\mathcal{B}(\mathsf{post}(a))$. Also, we define $\mathcal{B}(A) = \bigcup_{a \in A} \mathcal{B}(a)$. We define $\mathcal{B}(\Pi)$ to be $(\mathcal{B}(V), \mathcal{B}(\mathsf{init}), \mathcal{B}(\mathsf{goal}), \mathcal{B}(A))$.

We show that for any instance Π, the instances Π and $\mathcal{B}(\Pi)$ are reformulations of each other.

Proposition 22 *For any instance* Π*, the instance* $\mathcal{B}(\Pi)$ *is a reformulation of the instance* Π *with blowup function* $b(n) = 2n$*.*

Proof. Letting S denote the set of states of Π, we define $R = \{(s, \mathcal{B}(s)) : s \in S\}$. We verify that the five conditions hold. Condition (1) holds by the fact that a state s is reachable from init by plan $a_1, \ldots, a_n$ in Π if and only if the state $\mathcal{B}(s)$ is reachable by $\mathcal{B}(a_1), \ldots, \mathcal{B}(a_n)$ in $\mathcal{B}(\Pi)$. Condition (2) follows immediately from the definition of the actions $\mathcal{B}(a)$. For condition (3), we observe that s is a Π-goal state if and only if $\forall v \in \mathsf{vars}(\mathsf{goal}), s(v) = \mathsf{goal}(v)$. This holds if and only if $\forall v \in \mathsf{vars}(\mathsf{goal}), \forall d \in D(v)$ we have $(\mathcal{B}(s))(v_d) = (\mathcal{B}(\mathsf{goal}))(v_d)$ which in turn holds if and only if $\mathcal{B}(s)$ is a $\mathcal{B}(\Pi)$-goal state. Conditions (4) and (5) follow from the observation that if two (partial) Π-states p, p' differ on variable v, then $\mathcal{B}(p)$ and $\mathcal{B}(p')$ differ on $v_{p(v)}$ and $v_{p'(v)}$ in Π'; and, if p, p' do not differ on variable v, then for all $d \in D(v)$ we have that $\mathcal{B}(p)$ and $\mathcal{B}(p')$ are equal on v_d. □

Proposition 23 *Any instance* Π *is a reformulation of the instance* $\mathcal{B}(\Pi)$ *with blowup function* $b(n) = \lfloor \frac{n}{2} \rfloor$*.*

Proof. Letting S denote the set of states of Π, we define $R = \{(\mathcal{B}(s), s) : s \in S\}$. Conditions (1)-(4) are verified as in the proof of Proposition 22. From the discussion of Conditions (4) and (5) in that proof, we also have that if $(s_1, s_2), (s'_1, s'_2) \in R$, then $\mathsf{d_h}(s_1, s_2) = 2 \cdot \mathsf{d_h}(s'_1, s'_2)$. It follows that $\mathsf{d_h}(s'_1, s'_2) \leq \lfloor \frac{\mathsf{d_h}(s_1, s_2)}{2} \rfloor$, yielding condition (5). □

Theorem 24 *Let* $\mathcal{C}$ *be a set of planning instances. If* $\mathcal{C}$ *has PH width* k*, then* $\mathcal{B}(\mathcal{C}) = \{\mathcal{B}(\Pi) : \Pi \in \mathcal{C}\}$ *has PH width* $2k$*. If* $\mathcal{B}(\mathcal{C})$ *has PH width* k*, then* $\mathcal{C}$ *has PH width* $\lfloor \frac{k}{2} \rfloor$*. Thus, the set* $\mathcal{C}$ *has bounded PH width if and only if* $\mathcal{B}(\mathcal{C})$ *has bounded PH width.*

Proof. Immediate from Propositions 22 and 23, and Theorem 21. □

Composition operations

In this section, we further investigate the robustness of our width notions by considering two binary composition operations on instances. We give results showing that our width notions are closed under these operations.

Disjoint union

Definition 25 *Let* $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ *and* $\Pi' = (V', \mathsf{init}', \mathsf{goal}', A')$ *be instances. The* disjoint union $\Pi \oplus \Pi'$ *is defined to be the instance* $(V \cup V', \mathsf{init} \cup \mathsf{init}', \mathsf{goal} \cup \mathsf{goal}', A \cup A')$*. Here, we assume that the variable sets* V *and* V' *are disjoint; if they are not, we rename them accordingly.*

Note that, by the union of two partial states $p \cup p'$, we mean the function defined as p on all variables in $\mathsf{vars}(p)$ and as p' on all variables in $\mathsf{vars}(p')$. Note also that this notation is used only when $\mathsf{vars}(p)$ and $\mathsf{vars}(p')$ are disjoint.

Theorem 26 *Let* $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ *and* $\Pi' = (V', \mathsf{init}', \mathsf{goal}', A')$ *be instances having width* k *(respectively, P width* k*, H width* k*, PH width* k*). The instance*

$\Pi \oplus \Pi'$ *has width k (respectively, P width k, H width k, PH width k).*

Proof. We prove the theorem for persistent width k. The proof for the other width notions is similar. Let s be a reachable state that is not a goal state, in $(\Pi \oplus \Pi')$. Then either $(s \restriction \mathsf{vars}(\mathsf{goal})) \neq \mathsf{goal}$ or $(s \restriction \mathsf{vars}(\mathsf{goal}')) \neq \mathsf{goal}'$. If $(s \restriction \mathsf{vars}(\mathsf{goal})) \neq \mathsf{goal}$, then in Π the state $(s \restriction V)$ is reachable but not a goal state. By the assumption that Π has persistent width k, there exists a variable $u \in \mathsf{wrong}(s \restriction V)$ that is k-improvable in $(s \restriction V)$. The plan P performing this k-improvement also k-improves the variable u in state s, in the instance $\Pi \oplus \Pi'$. In the case that $(s \restriction \mathsf{vars}(\mathsf{goal}')) \neq \mathsf{goal}'$, the proof is similar. □

Sequencing

Definition 27 *Let $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ and $\Pi' = (V', \mathsf{init}', \mathsf{goal}', A')$ be instances. Define A'_{goal} to contain an action a'_{goal} for every action $a' \in A'$ such that $\mathsf{pre}(a'_{\mathsf{goal}}) = \mathsf{pre}(a') \cup \mathsf{goal}$ and $\mathsf{post}(a'_{\mathsf{goal}}) = \mathsf{post}(a')$. The* sequence *$\Pi \odot \Pi'$ is defined to be the instance $(V \cup V', \mathsf{init} \cup \mathsf{init}', \mathsf{goal} \cup \mathsf{goal}', A \cup A'_{\mathsf{goal}})$. Here, we assume that the variable sets V and V' are disjoint; if they are not, we rename them accordingly.*

Theorem 28 *Let $\Pi = (V, \mathsf{init}, \mathsf{goal}, A)$ and $\Pi' = (V', \mathsf{init}', \mathsf{goal}', A')$ be instances having P width k (respectively, PH width k). The instance $\Pi \odot \Pi'$ has P width k (respectively, PH width k).*

Proof. We prove the theorem for P width k; the case of PH width k is similar. Let s be a reachable state that is not a goal state. We consider two cases. The first case is when $(s \restriction \mathsf{vars}(\mathsf{goal})) \neq \mathsf{goal}$. In this case, $(s \restriction V)$ is, in Π, a reachable state that is not a goal state. Since Π has P width k, there exists a variable $u \in V$ where goal and s differ that, in Π, can be k-improved in state s. A plan k-improving u in s with respect to the instance Π also performs the k-improvement in the instance $\Pi \odot \Pi'$. In the second case, where $(s \restriction \mathsf{vars}(\mathsf{goal})) = \mathsf{goal}$, since s is by assumption not a goal state, we have $(s \restriction \mathsf{vars}(\mathsf{goal}')) \neq \mathsf{goal}'$. Then, $(s \restriction V')$ is, in Π', a reachable state that is not a goal state. Since Π' has P width k, there exists a variable $u \in V'$ where goal' and s differ that, in Π', can be k-improved in state s. Let $P = a'_1, \ldots, a'_n$ be a plan that k-improves u in s with respect to the instance Π'. Then, the plan $P_{\mathsf{goal}} = a'_{1\mathsf{goal}}, \ldots, a'_{n\mathsf{goal}}$ k-improves u in s in the instance $\Pi \odot \Pi'$. Note that in state s it holds that $(s \restriction \mathsf{vars}(\mathsf{goal})) = \mathsf{goal}$, and moreover that this property is preserved by the application of actions in A'_{goal}. Thus, during the execution of P_{goal} the "added precondition" goal in the actions of A'_{goal} is always satisfied. □

Acknowledgements

This work was partially supported by grant TIN2004-07925-C03-01 (GRAMMARS).

References

Amir, E., and Engelhardt, B. 2003. Factored planning. In *IJCAI 2003*, 929–935.

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS+ planning. *Computational Intelligence* 11(4):625–655.

Barrett, A., and Weld, D. 1993. Characterizing subgoal interactions for planning. In *Proceedings of IJCAI-93*, 1388–1393.

Brafman, R., and Domshlak, C. 2003. Structure and complexity of planning with unary operators. *JAIR* 18:315–349.

Brafman, R., and Domshlak, C. 2006. Factored planning: How, when, and when not. In *AAAI 2006*.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69:165–204.

Erol, K.; Nau, D. S.; and Subrahmanian, V. S. 1995. Complexity, decidability and undecidability results for domain-independent planning. *Artificial intelligence* 76:625–655.

Haslum, P. 2007. Reducing accidental complexity in planning problems. In *Proc. 20th International Joint Conference on Artificial Intelligence*.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, 161–170.

Helmert, M. 2006. The fast downward planning system. *Journal of Artifical Intelligence Research* 26:191–246.

Hoffmann, J., and Nebel, B. 2001. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Jonsson, P., and Bäckström, C. 1994a. Complexity results for state-variable planning under mixed syntactical and structural restrictions. Technical Report R-95-17, Department of Computer and Information Science, Linköping University.

Jonsson, P., and Bäckström, C. 1994b. Tractable planning with state variables by exploiting structural restrictions. Technical Report R-95-16, Department of Computer and Information Science, Linköping University.

Korf, R. E. 1987. Planning as search: A quantitative approach. *Artificial Intelligence* 33:65–88.

Vidal, V., and Geffner, H. 2005. Solving simple planning problems with more inference and no search. In *Proc. of the 11th Int. Conf. on Principles and Practice of Constraint Programming (CP-05)*.