Computing Plan-Length Bounds Using Lengths of Longest Paths

Mohammad Abdulaziz and Dominik Berger
Techniche Universität München, Germany

Abstract

We devise a method to exactly compute the length of the longest simple path in factored state spaces, like state spaces encountered in classical planning. Although the complexity of this problem is NEXP-hard, we show that our method can be used to compute practically useful upper-bounds on lengths of plans. We show that the computed upper-bounds are significantly (in many cases, orders of magnitude) better than bounds produced by previous bounding techniques and that they can be used to improve the SAT-based planning.

Introduction

Many techniques for solving problems defined on transition systems, like SAT-based planning (Kautz and Selman 1992) and bounded model checking (Biere et al. 1999), benefit from knowledge of *upper bounds* on the lengths of solution transition sequences, aka *completeness thresholds*. If N is such a bound, and if a solution exists, then that solution need not comprise more than N transitions.

In AI planning, upper bounds on plan lengths have two main uses related to SAT-based planning. Firstly, like for bounded model-checking, an upper bound on plan lengths can be used as a completeness threshold, i.e. to prove a planning problem has no solution. Secondly, it can be used to improve the ability of a SAT-based planner to find a solution. Typically, a SAT-based planner queries a SAT solver to search for plans of increasing lengths, aka horizons, going through many unsatisfiable formulae until the given horizon is longer than the shortest possible plan. If a horizon longer than the shortest plan were initially provided, the planner can avoid many of the costly unsatisfiable queries (Gerevini, Saetti, and Vallati 2015). Using plan length upper bounds as horizons in this way was shown to increase the coverage of SAT-based planners (Rintanen and Gretton 2013; Abdulaziz, Gretton, and Norrish 2017; Abdulaziz 2019).

Biere et al. identified the *diameter* (d) and the *recurrence diameter* (rd), which are topological properties of the state space, as completeness thresholds for bounded model-checking of safety and liveness properties, respectively. d is the longest shortest path between any two states. rd is the length of the longest simple path in the state space, i.e. the length of the longest path that does not traverse any state more than once. Both, d and rd, are upper bounds on the shortest plan's length, i.e. they are completeness thresholds for SAT-based planning. Also, d is a lower bound on rd that can be exponentially smaller.

Computing d or rd for succinctly represented transition systems, such as *factored systems*, is hindered by the worst-case complexity of all existing methods, which is exponential or doubly-exponential in the size of the given system, respectively. This complexity can be alleviated by *compositionally* computing upper bounds on d or rd instead of exactly computing them. Compositional bounding methods compute an upper bound on a factored transition system's diameter by composing together values of topological properties of state spaces of abstract subsystems (Baumgartner, Kuehlmann, and Abraham 2002; Rintanen and Gretton 2013; Abdulaziz, Gretton, and Norrish 2015, 2017; Abdulaziz 2019), and they are currently the only practically viable method to compute bounds on plan lengths or the state space diameter. Compositional approaches provide useful bounds on plan length or the diameter using potentially exponentially smaller computational effort compared to directly computing d or rd, since explicit representations of only abstract subsystems have to be constructed.

In this work we study the computation of rd for state spaces of classical planning problems, which are factored transition systems. The longest simple path and its length are fundamental graph properties. Thus, computing rd for state spaces of planning problems is inherently interesting, as it might reveal interesting properties of different planning problems. However, our goal is to devise better compositional methods to compute upper bounds on plan lengths to aid SAT-based planning. This raises an interesting question: why should we focus on computing rd instead of d? This question is reasonable since, as stated earlier, d can be computed in exponentially less time than rd, and rd is an upper bound on d that can be exponentially larger. The reason is simple: it has been shown that d cannot be bounded by diameters of *projections* (Abdulaziz 2017, Chapter 3, Theorem 1). Since projections are cornerstone abstractions for compositional bounding, a method to compute d cannot be leveraged for compositional bounding. On the other hand, previous authors showed that, theoretically, recurrence diameters of projections can be composed to bound the concrete

system's diameter (Baumgartner, Kuehlmann, and Abraham 2002; Abdulaziz, Gretton, and Norrish 2017). Here we explore the potential of this theoretical possibility: we study methods to compute rd and the use of those methods to improve existing compositional bounding algorithms.

Our first contribution concerns the relationship between rd and the *traversal diameter* (td), which is another state space topological property. The best existing compositional bounding method is due to Abdulaziz 2019, and it computes traversal diameters of abstractions and composes them into an upper bound on d and on plan-length. We show that td is an upper bound on rd, and that rd can be exponentially smaller than td. This gives an opportunity for substantial improvements in the bounds computed by compositonal bounding methods, if recurrence diameters of abstractions are used instead of their traversal diameters. However, the practical realisation of this improvement is contingent on whether there is an efficient method to compute rd.

Our second, and main, contribution is that we investigate practically useful methods to compute recurrence diameters of factored systems that come from planning problems. We implement two methods to compute recurrence diameters based on the work of Biere et al. 1999. Unlike Biere et al., who devised their method and tested it on model-checking problems, we test these methods on planning benchmarks and show that it is impractical for most planning problems. We show, however, that computing the recurrence diameter within the compositional bounding method by Abdulaziz, Gretton, and Norrish 2017 leads to bounds that are, as predicted theoretically, much tighter when rd is used instead of td.

However, a challenge is that our method to compute rd and that of Biere et al. have wort-case running times that are doubly-exponential in the size of the given factored system. This is much worse than the worst-case complexity of computing td, which is singly-exponential in the size of the factored system. This stems from the complexity of the problem of computing rd for succinct digraphs, which is NEXP-hard (Pardalos and Migdalas 2004; Papadimitriou and Yannakakis 1986). Our third contribution is that we investigate techniques to alleviate the impact of this prohibitive worst-case running time on the overall compositional bounding algorithm by combining the computation of rd and td.

Lastly, we experimentally show that the improved bounds lead to an improved problem coverage for state-of-the-art SAT-based planner MP (Rintanen 2012), when the bounds are used as horizons for it.

Background and Notation

We consider *factored transition systems* which are characterised in terms of a set of *actions*. From actions we can define a set of *valid states*, and then approach bounds by considering properties of *executions* of actions on valid states. Whereas conventional expositions in the planning and model-checking literature would also define initial conditions and goal/safety criteria, here we omit those features from discussion since the state-space topological properties we consider are independent of those features.

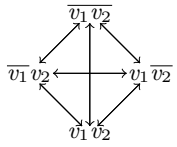

Figure 1

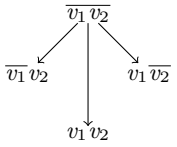

Figure 2

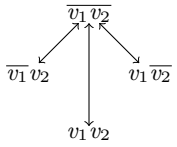

Figure 3

Figure 4: The state spaces of the systems from Examples 1, 3, and 4.

Definition 1 (States and Actions)**.** *A maplet, $v \mapsto b$, maps a variable v—i.e. a state-characterising proposition—to a Boolean b. A state, x, is a finite set of maplets. We write $\mathcal{D}(x)$ to denote $\{v \mid (v \mapsto b) \in x\}$, the domain of x. For states x_1 and x_2, the union, $x_1 \uplus x_2$, is defined as $\{v \mapsto b \mid v \in \mathcal{D}(x_1) \cup \mathcal{D}(x_2) \wedge \mathsf{if}\ v \in \mathcal{D}(x_1)\ \mathsf{then}\ b = x_1(v)\ \mathsf{else}\ b = x_2(v)\}$. Note that the state x_1 takes precedence. An action is a pair of states, (p, e), where p represents the* preconditions *and e represents the* effects*. For action $\pi = (p, e)$, the domain of that action is $\mathcal{D}(\pi) \equiv \mathcal{D}(p) \cup \mathcal{D}(e)$.*

Definition 2 (Execution)**.** *When an action $\pi\ (= (p, e))$ is executed at state x, it produces a successor state $\pi(x)$, formally defined as $\pi(x) = \mathsf{if}\ p \not\subseteq x\ \mathsf{then}\ x\ \mathsf{else}\ e \uplus x$. We lift execution to lists of actions $\vec{\pi}$, so $\vec{\pi}(x)$ denotes the state resulting from successively applying each action from $\vec{\pi}$ in turn, starting at x.*

We give examples of states and actions using sets of literals, where we denote the maplet $a \mapsto \top$ with the literal a and $a \mapsto \bot$ with the literal $\overline{a}$. For example, $(\{a, \overline{b}\}, \{c\})$ is an action that if executed in a state where a is true and b is false, it sets c to true. $\mathcal{D}((\{a, \overline{b}\}, \{c\})) = \{a, b, c\}$. We also give examples of sequences, which we denote by the square brackets, e.g. $[a, b, c]$.

Definition 3 (Factored Transition System)**.** *A set of actions δ constitutes a factored transition system. $\mathcal{D}(\delta)$ denotes the domain of δ, which is the union of the domains of all the actions in δ. Let $\mathsf{set}(\vec{\pi})$ be the set of elements in $\vec{\pi}$. The set of valid action sequences, δ^*, is $\{\vec{\pi} \mid \mathsf{set}(\vec{\pi}) \subseteq \delta\}$. The set of valid states, $\mathbb{U}(\delta)$, is $\{x \mid \mathcal{D}(x) = \mathcal{D}(\delta)\}$. $G(\delta)$ denotes the set of pairs $\{(x, \pi(x)) \mid x \in \mathbb{U}(\delta), \pi \in \delta\}$, which is all non self-looping transitions in the state space of δ.*

Example 1. *Consider the factored system $\delta = \{\pi_1 = (\emptyset, \{v_1, v_2\}), \pi_2 = (\emptyset, \{\overline{v_1}, v_2\}), \pi_3 = (\emptyset, \{v_1, \overline{v_2}\}), \pi_4 = (\emptyset, \{\overline{v_1}, \overline{v_2}\})\}$. Figure 1 shows $G(\delta)$, i.e. the state space of δ, where different states defined on the variables $\mathcal{D}(\delta) = \{v_1, v_2\}$ are shown. Since every state can be reached via one action from every other state, the state space is a clique.*

For a system δ, a bound on the length of action sequences is $\textsc{Exp}(\delta) = 2^{|\mathcal{D}(\delta)|} - 1$ (i.e. one less than the number of valid states), where $|\bullet|$ denotes the cardinality of a set or the length of a list. Other bounds employed by previous approaches are topological properties of the state space. One such topological property is the diameter, suggested by Biere et al. 1999, which is the length of the longest shortest path between any two states in the state space of a system.

Definition 4 (Diameter)**.** *The diameter, written $d(\delta)$, is the length of the longest shortest action sequence, formally*

$$d(\delta) = \max_{x \in \mathbb{U}(\delta), \vec{\pi} \in \delta^*} \min_{\vec{\pi}(x) = \vec{\pi}'(x), \vec{\pi}' \in \delta^*} |\vec{\pi}'|$$

Note that if there is a valid action sequence between any two valid states of δ, then there is a valid action sequence between them which is not longer than $d(\delta)$. Thus it is a completeness threshold for bounded model-checking and SAT-based planning. Another topological property that is an upper bound on plan lengths is the recurrence diameter, which is the length of the longest simple path in the state space of a transition system. It was proposed by Biere et al. 1999.

Definition 5 (Recurrence Diameter)**.** *Let* $\mathsf{distinct}(x, \vec{\pi})$ *denote that all states traversed by executing $\vec{\pi}$ at x are distinct states. The recurrence diameter is the length of the longest simple path in the state space, formally*

$$rd(\delta) = \max_{x \in \mathbb{U}(\delta), \vec{\pi} \in \delta^*, \mathsf{distinct}(x, \vec{\pi})} |\vec{\pi}|$$

Example 2. *For the system δ from Example 1, $d(\delta) = 1$, since every state can be reached with one action from every other state. Nonetheless, $rd(\delta) = 3$ as there are many paths with 3 actions in the state space that traverse distinct states, e.g. executing the action sequence $[\pi_1, \pi_2, \pi_3]$ at the state $\{\overline{v_1}, \overline{v_2}\}$ traverses the distinct states $[\{\overline{v_1}, \overline{v_2}\}, \{v_1, v_2\}, \{\overline{v_1}, v_2\}, \{v_1, \overline{v_2}\}]$.*

Note that in general rd is an upper bound on d, and that it can be exponentially larger than d.

Theorem 1 (Biere et al. 1999)**.** *For any system δ, we have that $d(\delta) \leq rd(\delta)$. Also, there are infinitely many systems for which the recurrence diameter is exponentially (in the number of state variables) larger than the diameter.*

Like d, rd is a completeness threshold for SAT-based planning and for safety bounded model-checking but, unlike d, rd is also a completeness threshold for bounded model-checking of liveness properties, which was the original reason for its inception (Biere et al. 1999).

Algorithms have been developed to calculate both properties for digraphs, and those algorithms can be directly applied to state spaces of explicitly represented (e.g. tabular) transition systems. Exact algorithms to compute d have worse than quadratic runtimes in the number of states (Fredman 1976; Alon, Galil, and Margalit 1997; Chan 2010; Yuster 2010), and approximation algorithms have superlinear runtimes (Aingworth et al. 1999; Roditty and Vassilevska Williams 2013; Chechik et al. 2014; Abboud, Williams, and Wang 2016). The situation is worse for rd, whose computation is NP-hard (Pardalos and Migdalas 2004) for explicitly represented systems. The impracticality of computing d and rd is exacerbated in settings where transition systems are described using factored representations, like in planning and model-checking (Fikes and Nilsson 1971; McMillan 1993). In particular, the worst-case running times are exponentially worse because, in the worst case, all known methods construct an explicit representation of the state space to compute d or rd of the state space of a succinctly represented system. This follows the general pattern of complexity exponentiation of graph problems when graphs are succinctly represented, where, for succinct digraphs, the complexity of computing d is Π_2^P-hard (Hemaspaandra et al. 2010) and the complexity of computing rd is NEXP-hard (Papadimitriou and Yannakakis 1986), instead of being in P and NP-hard, respectively, in the explicit case.

Compositional Bounding of the Diameter

The prohibitive complexity of computing d or rd suggests they can only be feasibly computed for very small factored systems, systems that are much smaller than those that arise in typical classical planning benchmarks. However, another possibility is to utilise the computation of d or rd within compositional plan length upper bounding techniques. Existing techniques compute an upper bound on d, for a given system, by computing topological properties of abstractions of the given system and then composing the abstractions' topological properties. Those abstractions are usually much smaller than the given concrete system, where their state spaces can be exponentially smaller than the given system's state space. Thus, computing topological properties of abstractions might be feasible.

Currently, the compositional bounding method by Abdulaziz, Gretton, and Norrish 2017 is the most successful in decomposing a given system into the smallest abstractions. It decomposes a given factored system using two kinds of abstraction: *projection* and *snapshotting*. Projection (Knoblock 1994; Williams and Nayak 1997) produces an over-approximation of the given system and it was used for bounding by many previous authors. Snapshotting produces an under-approximation of the given system and it was introduced by Abdulaziz, Gretton, and Norrish 2017. The compositional method devised by Abdulaziz, Gretton, and Norrish 2017 recursively interleaves the application of projection and snapshotting until the system is decomposed into subsystems that can no longer be decomposed, to which we refer here as *base case systems*. After the system is decomposed into base case systems, a topological property, which we call the *base case function*, of the state space of each of the base case systems is computed. Then the values of the base case function applied to the different base case systems are composed to bound the diameter of the concrete system.

Most authors used the base case function EXP, which is one less than the number of valid states for the given base case system (Rintanen and Gretton 2013; Abdulaziz, Gretton, and Norrish 2015, 2017). A notable exception is Abdulaziz 2019, who used the *traversal diameter*, which is a topological property of the state space, as a base case function. The traversal diameter is one less than the largest number of states that could be traversed by any path.

Definition 6 (Traversal Diameter)**.** *Let* $\mathsf{ss}(x, \vec{\pi})$ *be the set of states traversed by executing $\vec{\pi}$ at x. The traversal diameter is*

$$td(\delta) = \max_{x \in \mathbb{U}(\delta), \vec{\pi} \in \delta^*} |\mathsf{ss}(x, \vec{\pi})| - 1.$$

Example 3. *Consider the factored system $\delta = \{\pi_1 = (\{\overline{v_1}, \overline{v_2}\}, \{v_1, v_2\}), \pi_2 = (\{\overline{v_1}, \overline{v_2}\}, \{\overline{v_1}, v_2\}), \pi_3 =$*

$(\{\overline{v_1}, \overline{v_2}\}, \{v_1, \overline{v_2}\})\}$. *The digraph in Figure 2 shows the state space of* δ. *For* δ, $\text{EXP}(\delta) = 3$, *while* $td(\delta) = 1$.

Abdulaziz 2019 showed that td is an upper bound on rd and a lower bound on EXP. He also showed that td can be exponentially smaller than EXP, as shown in the above example. This is why, when Abdulaziz 2019 used td as a base case function, his method computed substantially tighter bounds than previous methods, which all used EXP.

The Recurrence Diameter Versus the Traversal Diameter

We now study the relationship between the recurrence diameter and the traversal diameter. The core insight we make here is that rd can be exponentially smaller than td.

Theorem 2. *There are infinitely many factored systems whose recurrence diameters are exponentially smaller (in the number of state variables) than their traversal diameters.*

Proof. Let, for a natural number n, $\mathcal{D}_n$ denote the indexed set of state variable $\{v_1, v_2, \ldots, v_{\lceil \log n \rceil}\}$. Let x_i^n denote the state defined by assigning all the state variables $\mathcal{D}_n$, s.t. their assignments binary encode the natural number i, where the index of each variable from $\mathcal{D}_n$ represents its endianess. Note: x_i^n is well defined for $0 \leq i \leq 2^{\lceil \log n \rceil} - 1$.

Now, for an arbitrary number $n \in \mathbb{N}$, let $\mathbb{I}_n$ denote the factored system (i.e. set of actions) $\{(x_0^{n+1}, x_i^{n+1}) \mid 1 \leq i \leq n\} \cup \{(x_i^{n+1}, x_0^{n+1}) \mid 1 \leq i \leq n\}$. The recurrence diameter of the system $\mathbb{I}_n$ is 2, regardless of n, since any action sequence that traverses more than 3 states will traverse x_0^{n+1} more than once. Now, let $\mathcal{S}$ denote the set of states in the largest connected component in the state space of $\mathbb{I}_n$, which has $n+1$ states in it. Since for any two states $x_i^{n+1}, x_j^{n+1} \in \mathcal{S}$, there is an action sequence $\overrightarrow{\pi} \in \mathbb{I}_n{}^*$ s.t. $\overrightarrow{\pi}(x_i^{n+1}) = x_j^{n+1}$, and since $|\mathcal{S}| = n+1$, then the traversal diameter of $\mathbb{I}_n$ is n. Accordingly, and since $2^{|\mathcal{D}(\mathbb{I}_n)|-2} = 2^{|\mathcal{D}_{n+1}|-2} = 2^{\lceil \log n \rceil - 1} \leq n$, we have that $2^{|\mathcal{D}(\mathbb{I}_n)|-2} \leq td(\mathbb{I}_n)$. The theorem follows from this and since $rd(\mathbb{I}_n) = 2$. □

Example 4. *The state space of* $\mathbb{I}_3$ *is depicted in Figure 3.* $rd(\mathbb{I}_3) = 2$, *and* $td(\mathbb{I}_3) = 3$.

The fact that rd can be exponentially smaller than td gives rise to the possibility of substantial improvements to the bounds computed if we use rd as a base case function for compositional bounding, instead of td.

Using the Recurrence Diameter for Compositional Bounding

Here, we investigate using rd as a base case function for the compositional algorithm by Abdulaziz, Gretton, and Norrish 2017. To do that, we firstly devise a method to compute rd. For explicitly represented digraphs, the computational complexity of finding the length of the longest path is NP-hard (Pardalos and Migdalas 2004). Thus, there is not a known method to compute it with a wort-case running time smaller than a time exponential in the size of the given digraph. Biere et al. 1999 suggested the only method to compute rd of which we are aware. They encode the question of whether a given number k is rd of a given transition system as a SAT formula. rd is found by querying a SAT-solver for different values of k, until the SAT-solver answers positively for one k. The method terminates since rd cannot be larger than one less the number of states in the given transition system. The size of their encoding grows linearly in k^2. The encoding of Biere et al. is based on the following theorem, which we restate in our notation.

Theorem 3 (Biere et al. 1999)**.** *For a factored system* δ *and a natural number* k, *we have that* $\phi_1(\delta, k)$ *is true iff* $rd(\delta) < k$, *where* $\phi_1(\delta, k)$ *denotes the conjunction of*

(i) $\forall (x, x') \in G(\delta).\ G(x, x')$,

(ii) $\forall x, x' \in \mathbb{U}(\delta).$ *if* $(x, x') \notin G(\delta)$, *then* $\neg G(x, x')$, *and*

(iii) if $\forall x_1 x_2 \ldots x_{k+1}.(\forall 1 \leq i \leq k.\ G(x_i, x_{i+1}))$ *then* $(\exists 1 \leq i < j \leq k+1.\ x_i = x_j)$.

Kroening and Strichman 2003 use sorting networks (Knuth 1998) to devise another encoding of the above question whose size grows linearly in $k \log^2(k)$. However, they report that, due to hidden constants, their encoding is only significantly smaller than the encoding by Biere et al. when $150 < k$. Since this is typically well beyond recurrence diameters that can be practically computed, we only implement the encoding of Biere et al..[1]

We use an SMT solver to reason about the encoding of rd. Thus, for decidability as well as efficiency reasons, we would like to obtain an encoding that is quantifier free, in particular, one that fits the theory of quantifier free uninterpreted functions. Since ϕ_1 is universally quantified, we reformulate it to its existentially quantified dual. For predicates Q and P of arity n, let $\bigwedge Q(T).\ P(T)$ denote the conjunction of $P(T)$, for all n-tuples T where $Q(T)$ holds. Also, let $\bigvee$ denote the analogous disjunction. Note: $\bigwedge Q(T).\ P(T)$ is only well defined if Q is true for only a finite set of n-tuples. Also, we do not explicitly bind Q or the tuple T when it is clear from context.

Encoding 1. *For* δ *and* $0 \leq k$, *let* $\phi_1'(\delta, k)$ *denote the conjunction of the formulae*

(i) $\bigwedge (x, x') \in G(\delta).\ G(x, x')$,

(ii) $\bigwedge \{x, x'\} \subseteq \mathbb{U}(\delta) \wedge (x, x') \notin G(\delta).\ \neg G(x, x')$,

(iii) $\bigwedge 1 \leq i \leq k.\ (G(y_i, y_{i+1}) \wedge \bigwedge i < j \leq k+1.\ y_i \neq y_j)$, *and*

(iv) $\bigwedge 1 \leq i \leq k+1.\ (\bigvee x \in \mathbb{U}(\delta).\ y_i = x)$.

The SMT formula above is defined over one constant x for every state in $\mathbb{U}(\delta)$, a set of uninterpreted constants $\{y_i \mid 1 \leq i \leq k+1\}$, one for every state in the simple path of length $k+1$ for which we search, and a function G that is true for a pair of constants (x, x') iff there is an edge from x to state x' in the state space of δ.

Theorem 4. $\phi_1'(\delta, k)$ *is satisfiable iff* $k \leq rd(\delta)$.

[1] The largest rd we computed in all our experiments was 93.

Proof. An SMT formula is defined over a signature Σ, which is a finite set of symbols that are either constants, uninterpreted constants, or the standard logical connectives. A model $\mathcal{M}$ for a signature is a function that maps uninterpreted constants to objects. A model $\mathcal{M}$ entails a formula ϕ, denoted $\mathcal{M} \vDash \phi$, iff ϕ evaluates to true, under the standard interpretation of logical connectives, after each uninterpreted constant v in ϕ is substituted by $\mathcal{M}(v)$. A formula ϕ is satisfiable iff $\exists \mathcal{M}. \mathcal{M} \vDash \phi$.

Lemma 1. *If $\phi'_1(\delta, k)$ is satisfiable, then there is a list of distinct states $[x_1, x_2 \dots x_{k+1}]$, such that $(x_i, x_{i+1}) \in G(\delta)$, for $1 \leq i \leq k$.*

Proof summary. Now, since the formula $\phi'_1(\delta, k)$ is satisfiable, there is a model $\mathcal{M}$, s.t. $\mathcal{M} \vDash \phi'_1(\delta, k)$. From the definition of entailment and the third conjunct of $\phi'_1(\delta, k)$, we have that $G(\mathcal{M}(y_i), \mathcal{M}(y_{i+1}))$ and $\mathcal{M}(y_i) \neq \mathcal{M}(y_j)$ hold, for all $1 \leq i \leq k$ and $i < j \leq k$. From this, the first, the second, and fourth conjuncts of $\phi'_1(\delta, k)$, we have that $(\mathcal{M}(y_i), \mathcal{M}(y_{i+1})) \in G(\delta)$. This finishes our proof. □

Lemma 2. *If there is a list of distinct states $[x_1, x_2 \dots x_{k+1}]$, such that $(x_i, x_{i+1}) \in G(\delta)$, for $1 \leq i \leq k$, then $\phi'_1(\delta, k)$ is satisfiable.*

Proof summary. Consider the model $\mathcal{M}$ defined as $\mathcal{M}(y_i) = x_i$, if $1 \leq i \leq k+1$. Note that $\mathcal{M}$ is well-defined for the set of uninterpreted constants in $\phi'_1(\delta, k)$, i.e. it is well-defined for the set $\{y_i \mid 1 \leq i \leq k+1\}$. From the assumptions of this lemma and the definition of $\phi'_1(\delta, k)$, we have that $\mathcal{M} \vDash \phi'_1(\delta, k)$. This finishes our proof. □

From the definition of rd, there is a list of actions $\vec{\pi}_k \equiv [\pi_1, \pi_2 \dots \pi_k]$ and a state $x_1 \in \mathbb{U}(\delta)$, s.t. $\vec{\pi}_k(x_1)$ traverses distinct states, iff $k < rd(\delta)$. Also, from the definition of $G(\delta)$, for any states x and x', there is an action $\pi_i \in \delta$ s.t. $x' = \pi_i(x)$ iff $(x, x') \in G(\delta)$. Accordingly, there is a list of distinct states $[x_1, x_2 \dots x_{k+1}]$, s.t. $(x_i, x_{i+1}) \in G(\delta)$, for $1 \leq i \leq k$, iff $k \leq rd(\delta)$. The theorem follows from this and Lemmas 1 and 2. □

To use the above encoding to compute rd of a given system δ, we iteratively query an SMT solver to check for the satisfiability of $\phi'_1(\delta, k)$ for different values of k, starting at 1, until the we have an unsatisfiable formula. The largest k for which the formula is satisfiable is $rd(\delta)$.

Observe that, to use Encoding 1, one has to build the entire state space as a part of building the encoding, i.e. one has to build the graph $G(\delta)$ and include it in the encoding. In fact, this is true for both methods, the one by Biere et al. and the one by Kroening and Strichman, as they are both specified in terms of explicitly represented transition systems. This means that the worst-case complexity of computing rd using either one of those encodings is doubly-exponential. Indeed, this is the best possible wort-case running time for succinct graphs generally, unless the polynomial hierarchy collapses, since computing rd is NEXP-hard.

Experimental evaluation We use Encoding 1 as a base case function for the compositional algorithm by Abdulaziz, Gretton, and Norrish 2017 instead of td, which was used as a base case by Abdulaziz 2019 and led to the tightest bounds of any existing method. We use Yices 2.6.1 (Dutertre 2014) as the SMT solver to prove the satisfiability or unsatisfiability of the resulting SMT formulae. We run the bounding algorithm by Abdulaziz, Gretton, and Norrish 2017 on standard planning benchmarks (from previous planning competitions and ones we modified), once with td as a base case and a second time with rd as a base case. We perform our experiments on a cluster of 2.3GHz Intel Xeon machines with a timeout of 20 minutes and a memory limit of 4GB. Our experiments show that Encoding 1 is not practical for planning problems when used as a base case function for the algorithm by Abdulaziz, Gretton, and Norrish 2017, where bounds are only computed within the timeout for less than 0.1% of our set of benchmarks. This is because computing rd can take time that is exponential in the size of the state space, while computing td can be computed in time that is linear in the state space (Abdulaziz 2019).

A Compact Encoding of Recurrence Diameter

We now devise a new encoding that performs better than Encoding 1. The new encoding exploits the factored representation in a way that is reminiscent to encodings used for SAT-based planning (Kautz and Selman 1992). In particular, our aim is to avoid constructing the state space in an explicit form, whenever possible. We devise a new encoding that avoids building the state space as a part of the encoding and, effectively, we let the SMT solver build as much of it during its search as needed.

Encoding 2. *For a state x, let x_i denote the formula $(\bigwedge v \in x.\ v_i) \wedge (\bigwedge \overline{v} \in x.\ \neg v_i)$. For δ and $0 \leq k$, let $\phi_2(\delta, k)$ denote the conjunction of the formulae*

(i) $\bigwedge 1 \leq i \leq k.\ \pi_i \rightarrow \mathsf{pre}(\pi)_i \wedge \mathsf{eff}(\pi)_{i+1} \wedge (\bigwedge v \in \mathcal{D}(\delta) \setminus \mathcal{D}(\mathsf{eff}(\pi)).\ v_i \leftrightarrow v_{i+1})$,

(ii) $\bigwedge 1 \leq i \leq k.\ \bigvee \pi \in \delta.\ \pi_i$, *and*

(iii) $\bigwedge 1 \leq i < j \leq k+1.\ \bigvee v \in \mathcal{D}(\delta).\ v_i \neq v_j$.

Briefly, the encoding above states that k is not rd if there is a sequence of k actions that traverses only distinct states if executed at some valid state. In more detail, the following are the intuitive meanings of uninterpreted constants in the above formulae: (i) π_i, for all $1 \leq i \leq k$ and $\pi \in \delta$, is a Boolean variable that represents whether action π is executed at state i, and (ii) v_i, for all $1 \leq i \leq k+1$ and $v \in \delta$, represents the truth value of state variable v at state i.[2]

There are three main conjuncts in the encoding. The first conjunct formalises the fact that, if an action is executed at state i, then all of its preconditions hold at state i, all of its effects hold at state $i+1$, and all the variables that are not in the effects will continue to have the same value at state $i+1$ as they did at state i (i.e. the frame axiom). The second conjunct states that at least one action must execute at state i. The third conjunct states that all states are pairwise distinct

[2]We note that this encoding can easily be formulated as a propositional formula in conjunctive normal format.

by stating that for every two states, at least one variable has a different truth value in both states.

Theorem 5. *$\phi_2(\delta, k)$ is satisfiable iff $k \leq rd(\delta)$.*

Proof. Firstly, let $\phi'_2(\delta, k)$ denote

$$\phi_2(\delta, k) \wedge \bigwedge 1 \leq i \leq k. \bigwedge \pi, \pi' \in \delta \wedge \pi \neq \pi'. \neg\pi_i \vee \neg\pi'_i.$$

Lemma 3. *$\phi_2(\delta, k)$ is satisfiable iff $\phi'_2(\delta, k)$ is satisfiable.*

Proof summary. $\Rightarrow$ Since $\phi_2(\delta, k)$ is satisfiable, then there is a model $\mathcal{M}$, s.t. $\mathcal{M} \vDash \phi_2(\delta, k)$. Note that $\mathcal{M}$ might not entail $\phi'_2(\delta, k)$ because $\phi'_2(\delta, k)$ has the extra conjunct that only one action is enabled in every step, i.e. at step i, if $\mathcal{M} \vDash \pi_i$ and $\mathcal{M} \vDash \pi'_i$, then $\pi = \pi'$. Nonetheless, conjunct (i) of Encoding 2 necessitates that in order for an action to be enabled in a step, all variables that are not in its effect are left unchanged in the next step. Accordingly, all actions enabled at a step affect the same state variables and assign all of those variables to the same value. Thus, we can construct a model that entails $\phi'_2(\delta, k)$ by leaving only one action from the set of enabled actions at every step, and disabling the rest. We formalise that as follows. For every $0 \leq i \leq k$, let $\Pi_i = \{\pi_i \mid \pi \in \delta \wedge \mathcal{M} \vDash \pi_i\}$, i.e. the set of actions enabled in step i. Let ϵ be the choice function, i.e. the function that given a set, returns an element from that set if it is not empty, and that is otherwise undefined. The model that entails $\phi'_2(\delta, k)$ is $\mathcal{M}'$, defined as follows.

$$\mathcal{M}'(c) = \begin{cases} \bot, & \text{if } p \in \Pi_i \text{ and } \epsilon(\Pi_i) \neq p \\ \mathcal{M}(c), & \text{otherwise.} \end{cases}$$

$\Leftarrow$ Since $\phi'_2(\delta, k)$ is the same as $\phi_2(\delta, k)$ conjoined with another formula, any model for $\phi'_2(\delta, k)$ is a model for $\phi_2(\delta, k)$. □

The theorem follows from Lemma 3, Theorem 4, and since $\phi'_1(\delta, k) \leftrightarrow \phi'_2(\delta, k)$. The latter fact follows by an induction on k, and from the definition of $\mathcal{D}(\delta)$ and $G(\delta)$. □

Experimental evaluation We experimentally test the new encoding as a base case function for the algorithm by Abdulaziz, Gretton, and Norrish 2017. Columns 2 and 3 of Table 1 show some data on the bounds computed with both, Encoding 2 (i.e. rd) and td, as base case functions. We note two observations. Firstly, many more planning problems are successfully bounded within the timeout when Encoding 2 is used to compute rd compared to using Encoding 1. Encoding 2 performs much better than Encoding 1 in practice since our new encoding is represented in terms of the factored representation of the system, while Encoding 1 represents the system as an explicitly represented state space. This leads to exponentially smaller formulae: Encoding 2 grows quadratically with the size of the given factored system, while Encoding 1 grows quadratically in the size of the state space, which can be exponentially larger than the given factored system. Indeed, Encoding 2 delegates the construction of the explicit state space to the SMT solver, which would effectively construct the state space during its search, but lazily. This is clearly better than constructing the state space a priori when the formula is satisfiable (i.e. when $k \leq rd$) as the SMT solver only needs to find a simple path of length $k+1$. The SMT solver does this without necessarily traversing the entire state space due to its search heuristics. When the formula is unsatisfiable, the SMT solver has to perform an exhaustive search to produce a proof of unsatisfiability, which is equivalent to constructing the entire state space explicitly. Since all queries to the SMT solver, except for the last one, are satisfiable, Encoding 2 is more practically efficient than using Encoding 1. However, it is worth noting that Encodings 1 and 2 worst-case running times grow doubly-exponentially in the size of the given factored system.

Secondly, when rd is the base case function the bounds computed are much tighter than those computed when td as a base case function. This agrees with the theoretical prediction of Theorem 2. This is shown clearly in Figure 5 and in Table 1. In particular, in the domains TPP, ParcPrinter, NoMystery, Logistics, OpenStacks, Woodworking, Satellites, Scanalyzer, Hyp and NewOpen (a compiled Qualitative Preference rovers domain), we have between two orders of magnitude and 50% smaller bounds when rd is used as a base case function compared to td. Also, the domain Visitall has twice as many problems whose bounds are less than 10^9 when rd is used instead of td. Also, specially interesting domains are Floortile and BlocksWorld, where the recurrence diameter of some of the smaller instances is successfully computed to be less than 10, but whose bounds using td are more than 10^9. In contrast, equal bounds are found using rd and td as base case functions in Zeno.

Further Experiments

Note that, although Encoding 2 is more efficient than Encoding 1, the number of problems that were successfully bounded is less when using rd as a base case function compared to td, since rd can take exponentially longer to compute than td. Figure 5 shows that running time difference for problems successfully bounded using both base case functions. Also Table 1 shows this in terms of the numbers of problems bounded within 20 minutes, when using rd vs. td.

One thing we observe during our experiments is that many of the base case systems have traversal diameters that are 1 or 2. We exploit that to improve the running time by devising a base case function that will only invoke the expensive computation of rd in case td is greater than 2.

Definition 7.

$$b_1(\delta) = \begin{cases} rd(\delta), & \textit{if } 2 < td(\delta) \\ td(\delta), & \textit{otherwise.} \end{cases}$$

Limiting the computation of rd as in the base case function b_1 significantly reduces the bound computation time. This leads to to more problems being successfully bounded as shown in column 4 of Table 1, compared to when rd is used. The substantially improved running time is shown in Figure 5. We also observe that bounds computed using rd as a base case function are exactly the same as those computed using b_1, for all problems on which they terminate. Thus, this improvement in running time does not come at the cost of looser bounds. Indeed, we conjecture the following.

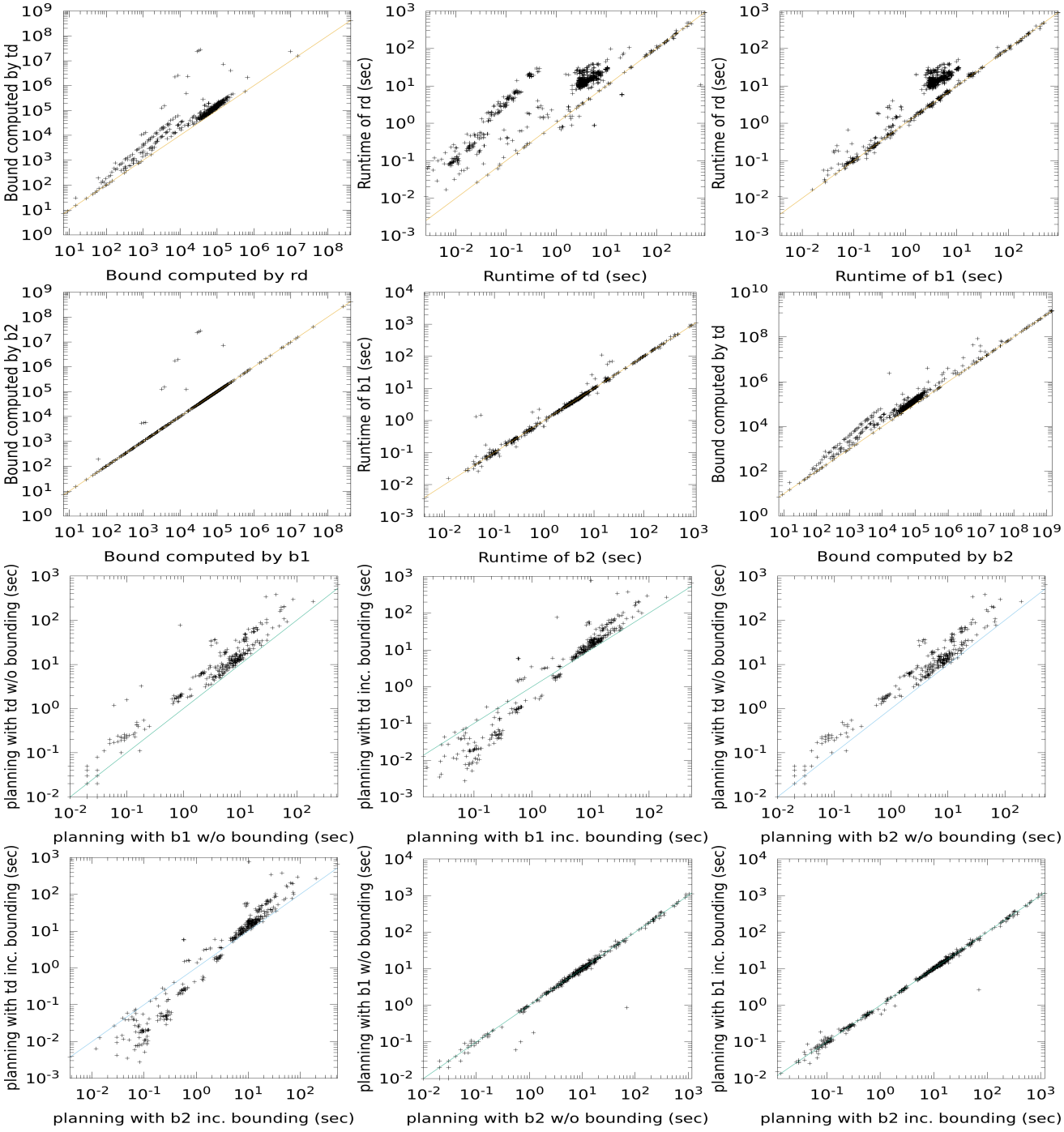

Figure 5: Different scatter-plots comparing the different bounding algorithms in terms of the quality of their bounds, the needed bound computation time, and the planning time using those bounds as a horizon for MP.

	td					rd					b_1					b_2				
Domain	Min.	Max.	Avg.	#Bnd.	#Sol.	Min.	Max.	Avg.	#Bnd.	#Sol.	Min.	Max.	Avg.	#Bnd.	#Sol.	Min.	Max.	Avg.	#Bnd.	#Sol.
newopen (1440)	3e3	7e7	4e5	848	157	2e3	1e7	9e4	592	—-	2e3	4e7	2e5	845	218	2e3	4e7	2e5	844	219
logistics (407)	7e1	4e6	4e5	406	170	6e1	1e4	3e3	194	—-	6e1	1e4	2e3	193	192	6e1	1e5	4e3	200	195
elevators (210)	3e6	1e9	3e8	14	—-	—-	—-	—-	—-	—-	—-	—-	—-	—-	—-	3e6	1e9	3e8	14	—-
rover (141)	1e2	1e6	1e5	51	42	7e1	9e4	1e4	38	—-	7e1	8e5	7e4	52	46	7e1	8e5	7e4	52	46
nomystery (124)	9e0	6e4	1e4	70	6	9e0	9e3	2e3	70	—-	9e0	9e3	2e3	70	8	9e0	9e3	2e3	70	7
zeno (50)	4e1	5e5	8e4	50	31	4e1	4e1	4e1	1	—-	4e1	4e1	4e1	1	1	4e1	2e5	1e5	16	1
hiking (40)	1e3	1e9	4e8	22	1	—-	—-	—-	—-	—-	—-	—-	—-	—-	—-	1e3	1e9	4e8	22	1
TPP (89)	2e1	6e8	4e7	16	9	2e1	4e4	8e3	15	—-	2e1	4e4	8e3	15	14	2e1	6e8	4e7	16	9
Transport (203)	4e5	2e9	3e8	14	—-	—-	—-	—-	—-	—-	—-	—-	—-	—-	—-	4e5	1e9	3e8	14	—-
GED (97)	7e6	7e8	3e8	5	—-	—-	—-	—-	—-	—-	—-	—-	—-	—-	—-	7e6	7e8	3e8	5	—-
woodworking (60)	2e2	8e7	2e7	10	—-	6e1	1e4	5e3	3	—-	6e1	1e4	5e3	3	1	2e2	1e7	3e6	10	1
visitall (70)	7e0	2e3	6e2	4	4	7e0	7e4	1e4	7	—-	7e0	1e3	4e2	6	6	7e0	1e3	4e2	6	6
openstacks (111)	3e5	3e5	3e5	6	6	4e4	4e4	4e4	6	—-	4e4	4e4	4e4	6	6	4e4	4e4	4e4	6	6
satellite (10)	6e2	6e3	3e3	10	9	4e2	2e3	9e2	6	—-	4e2	2e3	9e2	6	5	4e2	2e3	9e2	6	5
scanalyzer (60)	4e3	4e3	4e3	1	1	6e1	6e1	6e1	1	—-	6e1	6e1	6e1	1	1	4e3	4e3	4e3	1	1
storage (30)	1e2	8e7	2e7	7	4	6e0	6e0	6e0	1	—-	6e0	6e0	6e0	1	1	1e2	8e7	2e7	7	4
trucks (33)	5e2	4e8	8e7	5	2	3e2	3e2	3e2	2	—-	3e2	3e2	3e2	2	2	3e2	4e8	8e7	5	2
parcprinter (40)	8e2	4e8	7e7	6	1	8e2	4e8	8e7	9	—-	8e2	4e8	8e7	9	3	8e2	4e8	9e7	8	2
maintenance (5)	5e1	2e3	4e2	5	4	5e1	2e3	4e2	5	—-	5e1	2e3	4e2	5	4	5e1	2e3	4e2	5	4
blocksworld (10)	1e3	5e8	1e8	5	2	8e0	8e0	8e0	1	—-	8e0	8e0	8e0	1	1	1e3	5e8	1e8	5	2

Table 1: Column 1: the domain name and the number of instances in it. Column 2: when using td as a base case function: the minimum, maximum bound, the average bound computed, number of instances bounded (below 10^9), and the number of instances solved using Madagascar with the bound as the horizon. Column 3, 4, and 5: similar to column 2, but when rd, b_1, and b_2, respectively, are used as base case functions.

Conjecture 1. *For any factored transition system δ, if $td(\delta) \in \{0, 1, 2\}$, then $td(\delta) = rd(\delta)$.*

Note, however, that the number of problems successfully bounded when using b_1 as a base case function is still less than the number of problems bounded using td. This is because the large computation cost of rd on the base cases on which it is invoked is still much more than the cost of computing td. Another technique to improve the bound computation time is to limit the computation of rd to problems whose state spaces' sizes are bound by a constant. This is done with the following base case function.

Definition 8.

$$b_2(\delta) = \begin{cases} b_1(\delta), & \textit{if } 50 < \textsc{Exp}(\delta) \\ td(\delta), & \textit{otherwise.} \end{cases}$$

We set 50 as an upper limit on the state space size as more than 95% of abstractions whose rd was successfully computed had values less than 50.

As shown in column 5 of Table 1, the number of problems that are successfully bounded within 20 minutes when b_2 is used as a base case function is substantially more than those when b_1 is used, especially in the domains where rd and b_1 were less successful than td. However, the bounds computed when b_2 is used are sometimes worse than those computed when b_1 is used, like in the case of TPP. Figure 5 shows this bound degradation and bound computation time improvement for the problems on which both methods terminate. Nonetheless, the bounds computed using b_2 are still much better than td, as shown in Figure 5. This is because there are abstractions whose recurrence diameter is computable within the timeout and whose state spaces have more than 50 states. For those abstractions, td is computed instead of rd, when b_2 is used. An interesting problem is adjusting the threshold in b_2 to maximise the number of abstractions whose recurrence diameter can be computed within the timeout. We do not fully explore this problem here.

Using the bounds for SAT-based planning Table 1 shows that the coverage of MP increases if we use, as horizons, the bounds computed with base case functions involving rd, compared to when using td as a base case function. An exception is Zeno, where the better bound computation time using td is decisive. In addition to coverage, it makes sense to take a look into how the exact running times compare as many problems are solved using any of the bounds. The plots at the bottom two rows right of Figure 5 show the time needed for computing a plan when using different pairs of base case functions for problems on which both methods succeed. The plots show that the planning time (excluding bound computation) using the tighter bounds is much smaller almost always, which is to be expected. The other shows planning time including bound computation. Interestingly, although the bound computation time is more, e.g. when using b_1 than td, tighter bounds always payoff for problems that need more than 5 seconds to solve. Thus, time spent computing better bounds before planning is almost always well-spent.

Conclusion

The recurrence diameter was identified by many authors as an upper bound on transition sequence lengths in the areas of verification (Baumgartner, Kuehlmann, and Abraham 2002; Kroening and Strichman 2003; Kroening et al. 2011)

and AI planning (Abdulaziz, Gretton, and Norrish 2015, 2017; Abdulaziz 2019). However, previous authors noted that computing it is not practically useful, as it can take exponentially longer than solving the underlying planning or model checking problem. Nonetheless, we show that, indeed, computing the recurrence diameter is useful when used for compositional bounding. We do so using a SMT encoding that exploits the factored state space representation, and by combining the recurrence diameter with the traversal diameter, which is an easier to compute topological property.

Broad directions for future work include (i) devising methods to calculate base case functions that are tighter than the recurrence diameter and (ii) devising compositional methods that can compute allow better problem decompositions than those by Abdulaziz, Gretton, and Norrish 2015; 2017. A third more interesting direction is devising methods that use *radius* concepts for compositional bounding, which are the topological properties corresponding to the different diameter concepts (i.e. the diameter, the traversal diameter, and the recurrence diameter) that consider paths starting only at the initial state. Using these radius concepts for bounding has the potential to boost the efficiency of bound computation as well as bound tightness. A significant challenge is that the theory justifying existing compositional bounding methods fully relies on the fact that the base case function is a diameter concept (in particular, the *sublist diameter*) (Abdulaziz, Gretton, and Norrish 2015).

Acknowledgements

We would like to thank Dr. Charles Gretton for the very helpful discussions and comments on this paper. We also thank the anonymous reviewers whose comments helped improve this paper. We also thank the German Research Foundation for funding that facilitated this work through the DFG Koselleck Grant NI 491/16-1.

References

Abboud, A.; Williams, V. V.; and Wang, J. 2016. Approximation and Fixed Parameter Subquadratic Algorithms for Radius and Diameter in Sparse Graphs. In *SODA*.

Abdulaziz, M. 2017. *Formally Verified Compositional Algorithms for Factored Transition Systems*. The Australian National University.

Abdulaziz, M. 2019. Plan-Length Bounds: Beyond 1-way Dependency. In *AAAI*.

Abdulaziz, M.; Gretton, C.; and Norrish, M. 2015. Verified Over-Approximation of the Diameter of Propositionally Factored Transition Systems. In *ITP*.

Abdulaziz, M.; Gretton, C.; and Norrish, M. 2017. A State Space Acyclicity Property for Exponentially Tighter Plan Length Bounds. In *ICAPS*.

Aingworth, D.; Chekuri, C.; Indyk, P.; and Motwani, R. 1999. Fast Estimation of Diameter and Shortest Paths (Without Matrix Multiplication). *SICOMP* .

Alon, N.; Galil, Z.; and Margalit, O. 1997. On the exponent of the all pairs shortest path problem. *JCSS* .

Baumgartner, J.; Kuehlmann, A.; and Abraham, J. 2002. Property Checking Via Structural Analysis. In *CAV*.

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic Model Checking without BDDs. In *TACAS*.

Chan, T. M. 2010. More Algorithms for All-Pairs Shortest Paths in Weighted Graphs. *SICOMP* .

Chechik, S.; Larkin, D. H.; Roditty, L.; Schoenebeck, G.; Tarjan, R. E.; and Williams, V. V. 2014. Better Approximation Algorithms for the Graph Diameter. In *SODA*.

Dutertre, B. 2014. Yices 2.2. In *CAV*.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *AI* .

Fredman, M. L. 1976. New Bounds on the Complexity of the Shortest Path Problem. *SICOMP* .

Gerevini, A. E.; Saetti, A.; and Vallati, M. 2015. Exploiting Macro-Actions and Predicting Plan Length in Planning as Satisfiability. *AI Communications* .

Hemaspaandra, E.; Hemaspaandra, L. A.; Tantau, T.; and Watanabe, O. 2010. On the Complexity of Kings. *TCS* .

Kautz, H. A.; and Selman, B. 1992. Planning as Satisfiability. In *ECAI*.

Knoblock, C. A. 1994. Automatically Generating Abstractions for Planning. *AI* .

Knuth, D. E. 1998. *The Art of Computer Programming, Volume III, 2nd Edition*. Addison-Wesley.

Kroening, D.; Ouaknine, J.; Strichman, O.; Wahl, T.; and Worrell, J. 2011. Linear Completeness Thresholds for Bounded Model Checking. In *CAV*.

Kroening, D.; and Strichman, O. 2003. Efficient Computation of Recurrence Diameters. In *VMCAI*.

McMillan, K. L. 1993. Symbolic Model Checking. Kluwer.

Papadimitriou, C. H.; and Yannakakis, M. 1986. A Note on Succinct Representations of Graphs. *Information and Control* .

Pardalos, P. M.; and Migdalas, A. 2004. A Note on the Complexity of Longest Path Problems Related to Graph Coloring. *Applied Mathematics Letters* .

Rintanen, J. 2012. Planning as Satisfiability: Heuristics. *AI* .

Rintanen, J.; and Gretton, C. O. 2013. Computing Upper Bounds on Lengths of Transition Sequences. In *IJCAI*.

Roditty, L.; and Vassilevska Williams, V. 2013. Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs. In *STOC*.

Williams, B. C.; and Nayak, P. P. 1997. A Reactive Planner for a Model-based Executive. In *IJCAI*.

Yuster, R. 2010. Computing the diameter polynomially faster than APSP. *arXiv preprint arXiv:1011.6181* .