The Thirteenth International Symposium
on Combinatorial Search (SoCS 2020)

Computing Plan-Length Bounds Using Lengths of Longest Paths (Extended Abstract)

Mohammad Abdulaziz, Dominik Berger
Technical University of Munich, Munich, Germany

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 can be used as a completeness threshold, i.e. to prove a planning problem has no solution, and also it can be used to improve the ability of a SAT-based planner to find a solution.

Biere et al. (1999) 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. In this work we devise new methods to compute rd.

Background

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 $\mathcal{D}(\pi) \equiv \mathcal{D}(p) \cup \mathcal{D}(e)$. When an action $\pi\ (= (p, e))$ is executed at state x, it produces a successor state $\pi(x)$, formally defined as $\pi(x) = \mathsf{if}\ p \subseteq x\ \mathsf{then}\ e \uplus x\ \mathsf{else}\ x$. 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, which corresponds to a path in the state space. 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 it contains. Let $\mathsf{set}(\vec{\pi})$ be the set of elements from $\vec{\pi}$. 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 state space of δ, which is the set of pairs $\{(x, \pi(x)) \mid x \in \mathbb{U}(\delta), \pi \in \delta\}$, corresponding to different transitions in the state space of δ.

Definition 1. *The diameter, written $d(\delta)$, is the length of the longest shortest action sequence, formally* $\mathsf{max}\{\mathsf{min}\{|\vec{\pi}'| \mid \vec{\pi}(x) = \vec{\pi}'(x) \wedge \vec{\pi}' \in \delta^*\} \mid x \in \mathbb{U}(\delta) \wedge \vec{\pi} \in \delta^*\}$

Note that if there is a valid action sequence between any two states, then there is a valid action sequence between them which is not longer than d.

Definition 2. *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* $\mathsf{max}\{|\vec{\pi}| \mid x \in \mathbb{U}(\delta) \wedge \vec{\pi} \in \delta^* \wedge \mathsf{distinct}(x, \vec{\pi})\}$.

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

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 decompose a given factored system using two kinds of abstraction: *projection* and *snapshotting*. After the system is decomposed into abstractions, which we call *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, and then its values for base case systems are composed to bound d of the concrete system. Abdulaziz 2019 use the *traversal diameter* 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 3. *Let* $\mathsf{ss}(x, \vec{\pi})$ *be the set of states traversed by executing $\vec{\pi}$ at x. Traversal diameter, written as $td(\delta)$, is* $\mathsf{max}\{|\mathsf{ss}(x, \vec{\pi})| - 1 \mid x \in \mathbb{U}(\delta) \wedge \vec{\pi} \in \delta^*\}$.

Contributions

Our first contribution is showing that rd can be exponentially smaller than td. This 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.

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. In our experiments we use an SMT solver to reason about encoding of rd. Firstly, let for a set S of n-tuples and a predicate P of arity n, $\bigwedge S.\ P(a_1, a_2 \ldots, a_n)$ denote the conjunction of $P(a_1, a_2 \ldots, a_n)$, for all $(a_1, a_2 \ldots, a_n) \in S$. Note: $\bigwedge S.\ Q(a_1, a_2 \ldots, a_n)$ is only well defined if S is finite. Analogously, let $\bigvee$ denote a finite disjunction. We reformulate the encoding of Biere et al. as follows.

Encoding 1. *Let for δ, $\mathfrak{G}(x_1, x_2)$ denote $(x_1, x_2) \in G(\delta)$. For δ and $0 \leq k$, let $\phi'_1(\delta, k)$ denote the conjunction of the formulae (i) $\bigwedge\{(x_1, x_2) \mid \mathfrak{G}(x_1, x_2)\}.\ G(x_1, x_2)$, (ii) $\bigwedge\{(x_1, x_2) \mid x_1, x_2 \in \mathbb{U}(\delta) \wedge \neg\mathfrak{G}(x_1, x_2)\}.\ \neg G(x_1, x_2)$, and (iii) $\bigwedge\{i \mid 1 \leq i \leq k\}.\ (G(y_i, y_{i+1}) \wedge \bigwedge\{j \mid i < j \leq k\}.\ y_i \neq y_j)$.*

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 smallest k for which the formula is unsatisfiable 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. This means that the worst-case complexity of computing rd using either one of those encodings is doubly-exponential.

To experimentally test this encoding, we use it as a base case function for the compositional algorithm by Abdulaziz, Gretton, and Norrish 2017 instead of td. We use Yices 2.6.1 (Dutertre 2014) as the SMT solver to prove the satisfiability or unsatisfiability of the resulting SMT formulae. Our experiments show that Encoding 1 is not practical when used as a base case function: bounds are only computed within the timeout for less than 0.1% of our set of benchmarks.

Our second contribution is devising an encoding that performs better than Encoding 1. We observe that the encodings by Biere et al. does not exploit the compactness of factored representations of transition systems, and instead assume explicitly represented transition systems. We devise a new encoding which exploits the factored representation in a way that is reminiscent to encodings used for SAT-based planning (Kautz and Selman 1992). This avoids 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 $(\bigwedge\{v \mid v \in x\}.\ v_j) \wedge (\bigwedge\{v \mid \neg v \in x\}.\ \neg v_j)$. For δ and $0 \leq k$, let $\phi_2(\delta, k)$ denote the conjunction of the formulae*

(i) $\bigwedge\{i \mid 1 \leq i \leq k\}.\ \pi_i \rightarrow \mathsf{pre}(\pi)_i \wedge \mathsf{eff}(\pi)_{i+1} \wedge (\bigwedge\{v \mid v \in \mathcal{D}(\delta) \setminus \mathcal{D}(\pi)\}.\ v_i \leftrightarrow v_{i+1})$,
(ii) $\bigwedge\{i \mid 1 \leq i \leq k\}.\ \bigvee\{\pi \mid \pi \in \delta\}.\ \pi_i$,
(iii) $\bigwedge\{i \mid 1 \leq i \leq k\}.\ \bigwedge\{(\pi, \pi') \mid \pi, \pi' \in \delta\}.\ \neg\pi_i \vee \neg\pi'_i$
(iv) $\bigwedge\{(i, j) \mid 1 \leq i < j \leq k\}.\ \bigvee\{v \mid v \in \mathcal{D}(\delta)\}.\ v_i \neq v_j$

We experimentally test the new encoding as a base case function for the algorithm. We note two observations. 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. However, both encodings have the same worst-case doubly exponential running time.

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. 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. In contrast, equal bounds are found using rd and td as base case functions in Zeno.

Using the bounds for SAT-based planning The coverage of MP increases if we use as horizons the bounds computed when rd is the base case, compared to when using td as a base case function. Compared to td, rd increases the coverage by 187 solvable problems, overall. Those problems come from the domains: NEWOPEN (104 problems), NoMystery (23 problems), Floortile (25 problems), rover (15 problems), satellite (10 problems), TPP (8 problems), BlocksWorld (1 problem), and ParcPrinter (1 problem).

References

Abdulaziz, M.; Gretton, C.; and Norrish, M. 2017. A State Space Acyclicity Property for Exponentially Tighter Plan Length Bounds. In *International Conference on Automated Planning and Scheduling (ICAPS)*. AAAI.

Abdulaziz, M. 2019. Plan-length bounds: Beyond 1-way dependency. In *Proceedings of the Thirty-Third AAAI Conf. on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence.

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic model checking without BDDs. In *TACAS*, 193–207.

Dutertre, B. 2014. Yices 2.2. In *Computer Aided Verification - 26th International Conference, CAV 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 18-22, 2014. Proceedings*, 737–744.

Kautz, H. A., and Selman, B. 1992. Planning as satisfiability. In *ECAI*, 359–363.