Artificial Intelligence 257 (2018) 24–60

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Star-topology decoupled state space search

Daniel Gnad [*], Jörg Hoffmann [*]

Saarland University, Saarbrücken, Germany

ARTICLE INFO

Article history:
Received 3 May 2017
Received in revised form 20 December 2017
Accepted 21 December 2017
Available online 27 December 2017

Keywords:
AI planning
Heuristic search
Problem decomposition

ABSTRACT

State space search is a basic method for analyzing reachability in discrete transition systems. To tackle large compactly described transition systems – the state space explosion – a wealth of techniques (e.g., partial-order reduction) have been developed that reduce the search space without affecting the existence of (optimal) solution paths. Focusing on classical AI planning, where the compact description is in terms of a vector of state variables, an initial state, a goal condition, and a set of actions, we add another technique, that we baptize *star-topology decoupling*, into this arsenal. A star topology partitions the state variables into components so that a single center component directly interacts with several leaf components, but the leaves interact only via the center. Many applications explicitly come with such structure; any classical planning task can be viewed in this way by selecting the center as a subset of state variables separating connected leaf components. Our key observation is that, given such a star topology, the leaves are conditionally independent given the center, in the sense that, *given a fixed path of transitions by the center, the possible center-compliant paths are independent across the leaves.* Our decoupled search hence branches over center transitions only, and maintains the center-compliant paths for each leaf separately. As we show, this method has exponential separations to all previous search reduction techniques, i.e., examples where it results in exponentially less effort. One can, in principle, prune duplicates in a way so that the decoupled state space can never be larger than the original one. Standard search algorithms remain applicable using simple transformations. Our experiments exhibit large improvements on standard AI planning benchmarks with a pronounced star topology.[1]

1. Introduction

Reachability analysis in large discrete state transition systems arises in several areas of computer science. Examples are AI planning [4] and diagnosis [5], model checking [6], and multiple sequence alignment [7]. The question is whether, starting from a given state s, the system can reach a given state t, or can reach some state satisfying a given property, like a planning goal in AI, or the negation of a safety property in model checking. Answering this question is hard due to the *state explosion* problem [8]. The transition system is compactly described, in terms of state variables and transition rules, a network of

[*] Corresponding authors.
E-mail addresses: gnad@cs.uni-saarland.de (D. Gnad), hoffmann@cs.uni-saarland.de (J. Hoffmann).

[1] Parts of the presented material were previously published [1–3]. This article introduces, analyzes, and evaluates our techniques much more comprehensively. In particular, it adds new theoretical results pertaining to the space of star-topology factorings, to pruning methods avoiding state space blow-ups, and to exponential separations from previous techniques.

https://doi.org/10.1016/j.artint.2017.12.004
0004-3702/© 2017 Elsevier B.V. All rights reserved.

synchronized automata, or a bounded Petri net. The size of the system itself – the compact description's *state space* – is exponential in the size of that description.

Forward state space search is one basic method for reachability analysis. A wealth of techniques have been developed that reduce the search space while preserving *completeness* (finding a solution if one exists) and, ideally, *optimality* (finding a solution with minimum summed-up transition cost). *Partial-order reduction* exploits permutable parts of the state space [8–16]. *Symmetry reduction* exploits symmetric parts of the state space [17–21]. In *Petri-net unfolding*, the search space is an acyclic Petri net (a DAG) over *conditions* (vertices annotated with state-variable values) and *events* (vertices annotated with transitions), where events are added based on which *markings* (combinations of state-variable values) are reachable [22–28].

Our contribution consists in a new search reduction method, *star-topology decoupling*, which is complementary to all previous methods, and can be configured to either preserve completeness and optimality, or to only preserve completeness (allowing stronger reductions). We introduce the method in AI planning, where a planning *task* is given in terms of finite-domain state variables, an *initial state*, a *goal* condition, and a set of *actions* describing the possible transitions.

1.1. Star-topology decoupling

The distinguishing feature of star-topology decoupling is the assumption of a particular structural profile, a *star topology*. Viewing disjoint subsets of state variables as *components*, in a star topology a single center component interacts with multiple leaf components, but the leaves interact with each other only via the center. Many applications explicitly come with such structure. For example, distributed systems are often synchronized via a central component (client–server architectures, shared-memory computing systems), and cooperative agents are synchronized via their shared (commonly affected/queried) state variables. Arbitrary AI planning tasks can be viewed in this way by selecting the center as some subset of state variables breaking the dependencies between connected (leaf) components of the remaining state variables.

The key to star-topology decoupling is a particular form of "conditional independence": *given a fixed path of transitions by the center, the possible center-compliant paths are independent across the leaves*. For example, say the center C is a state variable encoding the position of a vehicle v, and each leaf L is a state variable encoding the position of a transportable object o. Given a fixed path π^C of vehicle moves, the compliant moves for any object o, alongside π^C, are those which load/unload o at those points on π^C where v is currently at the required location. Any sequence of such load/unload actions for o – any *π^C-compliant path* for o – can be committed to for o, independently of what any other transportable object o' is committed to. *Decoupled search* exploits this property by searching over center paths π^C only; it maintains, alongside each π^C, the leaf states reachable on π^C-compliant paths. This way, the component (local) state spaces are searched separately, avoiding the enumeration of combined (global) states across leaves. In catchy (though imprecise) analogy to conditional independence in graphical models, star-topology decoupling "instantiates" the center to break the dependencies between the leaves.

The search is least-commitment in the sense that leaf moves are committed to only at the end, when the goal is reached. During the search, the leaf states reachable on π^C-compliant paths are maintained exhaustively. Namely, the end point of each center path π^C in the search is associated with a *decoupled state* s, which for every leaf component L stores those leaf states s^L of L reached by π^C-compliant paths. One can view s as a compact representation of a set of global states, its *hypercube* $[s]$: all states formed from the center state reached by π^C, and any combination of π^C-compliant s^L. These are exactly those global states that can be reached on a transition path whose center sub-path is π^C.

Optionally, each s^L is annotated with the cost of a cheapest π^C-compliant path achieving s^L. We refer to that cost as s^L's *price*: it is not a cost we have already committed to paying, but a cost we *will* pay if, at the end, we commit to using s^L. Maintaining leaf state prices allows to preserve optimality; maintaining only leaf state reachability suffices to preserve completeness.

In the example above, for any transportable object o, the object's initial location l_0 is reachable on a π^C-compliant path, namely the empty path, at cost 0. Every other location l for o is reachable on a π^C-compliant path of length 2 – load at l_0, unload at l – iff π^C visits l_0 and afterwards visits l. Once the goal location of o is reachable, we can commit to a compliant path, i.e., a suitable load/unload pair, moving o to its goal location. In case different loads/unloads have different cost, distinguishing leaf state prices allows to select a cheapest such pair for each o. Searching over center paths enabling different-cost pairs allows to guarantee global optimality.

Star-topology decoupling has been inspired by *factored planning* methods, which also partition the state variables into components [29–41]. *Hierarchical* factored planning is remotely related; the search for a plan proceeds top-down in a hierarchy of increasingly more detailed levels identified by the factors. *Localized* factored planning is more closely related; the search for a plan proceeds by first planning locally on individual components, followed by global cross-component constraint resolution. In comparison to both, the key feature of star-topology decoupling is the focus on star topologies, which limits the possible interactions across factors, facilitating specialized search algorithms (as outlined above).

Star-topology decoupling also relates to Petri net unfolding, specifically to *contextual unfolding* [27] as planning actions typically have non-consumed preconditions (typically called *prevail conditions*). For example, a load action requires, but does not consume, a particular vehicle position. One can view star-topology decoupling as a new form of unfolding where the "conditions" are component states, and the star topology is exploited (a) to avoid the enumeration of exponentially many event "histories", keeping track of which prevail conditions may have been consumed in the past; as well as (b) to get rid of the **NP**-hard problem of testing the reachability of a marking in an unfolding prefix (for decoupled search, this is testable in linear time). As we shall see, these theoretical advantages often translate into empirical ones.