Decoupled Strong Stubborn Sets (Extended Abstract)

Daniel Gnad[1], Martin Wehrle[2], and Jörg Hoffmann[1]

[1] Saarland University,
Saarbrücken, Germany
{gnad,hoffmann}@cs.uni-saarland.de
[2] University of Basel
Basel, Switzerland
martin.wehrle@unibas.ch

1 Star-Topology Decoupled State Space Search

State space search is a canonical approach to testing reachability in large transition systems, like goal reachability in classical planning which is where this work is placed. Decomposition techniques for state space search have a long tradition, most notably in the form of *Petri net unfolding* [20, 7, 14] decomposing the search over concurrent transition paths, and *factored planning* [19, 2, 8, 4] decomposing the search into local vs. global planning over separate components of state variables.

Recent work by part of the authors [9, 10] has devised *star-topology decoupling*, which can be viewed as a hybrid between Petri net unfolding and factored planning, geared at star topologies. The state variables are factored into components whose cross-component interactions form a star topology. The search is akin to a Petri net unfolding whose atomic elements are component states, exploring concurrent paths of leaf components in the star independently. Relative to both Petri net unfolding and traditional factored planning, the key advantage lies in exploiting the star topology, which gets rid of major sources of complexity: the need to reason about conflicts and reachable markings, respectively the need to resolve arbitrary cross-component interactions.

The best way to understand the star topology impact is in terms of a particular form of "conditional independence": *given a fixed path of transitions by the center component in the star, the possible center-compliant paths are independent across the leaf components*. For example, say the center is a single truck-position variable t, and each leaf is a single package-position variable p_i. Given a fixed state-transition path π^C for t, the compliant state-transition paths for any p_i, alongside π^C, are those which load/unload p_i at suitable points along π^C. Any such load/unload sequence – any π^C*-compliant path* – can be committed to for p_i, independently of what any other p_j is committed to. Star-topology decoupled search exploits this by searching over center paths π^C only. Alongside each π^C, it maintains, for each leaf separately, the leaf states reachable on π^C-compliant paths. This avoids the enumeration of combined states across leaves. In imprecise analogy to conditional independence in graphical models, star-topology decoupling "instantiates" the center to break the dependencies between the leaves.

Star-topology decoupling is exponentially separated from all previous search reduction techniques, i. e., there are example families which it handles exponentially more effectively than Petri-net unfolding, factored planning, partial-order reduction [23, 24],

symmetry reduction [22, 6], etc. While this is merely a theoretical result pointing out that star-topology decoupling is, in principle, complementary to previous methods, the potential advantage of star-topology decoupling is also very much manifested in practice. On planning problems with a pronounced star topology, the empirical impact of star-topology decoupling is dramatic. Taking the effort required to build and represent the entire state space as the most basic measure of reduction power, Table 1 gives data comparing star-topology decoupling to its closest relatives.

	# Instances		Reachable State Space. Right: Average over Instances Commonly Built									
			Success						Representation Size (in Thousands)			
Domain	All	X	Std	POR	Punf	Cunf	OPT	COM	Std	POR	OPT	COM
Solvable Benchmarks: From the International Planning Competition (IPC)												
Depots	22	22	4	4	2	2	3	**5**	30,954.8	30,954.8	35,113.1	**3,970.0**
Driverlog	20	20	5	5	3	3	8	**10**	35,632.4	35,632.4	706.1	**127.2**
Elevators	100	100	21	17	1	3	8	**41**	22,652.1	22,651.1	21,046.2	**186.7**
Floortile	80	80	**2**	**2**	0	0	0	**2**				
Logistics	63	63	12	12	7	11	23	**27**	3,793.8	3,793.8	85.5	**8.2**
Miconic	150	145	50	45	25	30	45	**145**	52,728.9	52,673.1	218.8	**2.4**
NoMystery	40	40	11	11	5	7	**40**	**40**	29,459.3	25,581.5	11.5	**10.0**
Pathways	30	30	**4**	**4**	3	3	**4**	**4**	54,635.5	**1,229.0**	11,211.9	11,211.9
PSR	50	3	3	3	3	3	3	3	39.4	33.9	**11.1**	**11.1**
Rovers	40	40	5	**6**	4	4	5	5	98,051.6	6,534.4	4,045.9	**4,032.9**
Satellite	36	36	**5**	**5**	**5**	**5**	4	4	2,864.2	582.5	2,219.1	**352.7**
TPP	30	29	5	5	4	4	**11**	**11**	340,961.5	326,124.8	.9	**.8**
Transport	140	140	28	23	11	11	18	**34**	4,958.6	4,958.5	12,486.4	**173.3**
Woodworking	100	87	11	20	16	**22**	16	16	438,638.5	**226.8**	16,624.1	9,688.9
Zenotravel	20	20	**7**	**7**	2	4	**7**	**7**	17,468.0	17,467.5	1,028.5	**99.4**
Unsolvable Benchmarks: Extended from [15]												
NoMystery	40	40	9	8	2	4	**40**	**40**	85,254.2	65,878.2	3.9	**3.8**
Rovers	40	40	**4**	**4**	0	0	**4**	**4**	697,778.9	302,608.9	22,001.8	**20,924.4**
$\sum$	1001	935	186	181	93	116	239	**398**				

Table 1. State space size data. Best results highlighted in **boldface**. "Success": reachable state space fully explored. "X": X-shape factoring identified. "Std": standard state space. Other notations see text. All planning competition benchmark domains were run. Domains on which no X-shape was identified anywhere, and domains where no approach could build any state space, are not included in the table. Multiple test suites of the same domain are accumulated into the same table row. Runtime limit 30 minutes, memory limit 4 GB.

The "OPT" variant of our technique keeps track of leaf-state costs and preserves optimality, the "COM" variant keeps track of leaf-state reachability and preserves completeness only. We compare to Petri-net unfolding using "Punf" [18], as well as contextual Petri net unfolding using "Cunf" [21] which directly supports non-consumed (prevail) preconditions. We compare to standard state space search with *strong stubborn set* pruning [24], as star-topology decoupling can also be viewed (like Petri net unfolding) as a form of partial-order reduction. We do not compare here to factored planning because, while conceptually the use of separate components is a commonality, the concrete algorithms end up being completely different when considering arbitrary cross-component interactions (as all previous method do), vs. exploiting a star topology. Representation size is the number of integer variables in our C++ implementation based on FD [12]. We do not include representation size data for Petri net unfolding as these lag far behind in terms of the number of state spaces built.

The data clearly attest to the power of our approach. There are some domains where previous techniques are stronger, but overall the picture is very clearly in our favor,

with typical improvements of orders of magnitude, up to 5 and 6 orders of magnitude in the extreme cases. Considering that partial-order reduction and unfolding are venerable techniques into which sustained research effort was invested since decades, while star-topology decoupling was only just invented, we find this remarkable.

The major weakness evident from Table 1 is the *absence* of data for all the other competition domains. We show only those domains where a simple automatic factoring strategy succeeded – taking a few milliseconds to identify what we call an *X-shape*, a simple special case of star topologies where the interaction between the center and each leaf is one-way. X-shapes do occur in planning competition domains, but not widely.

Preempting the conclusion section a bit, one major conclusion here is the need for more powerful factoring strategies. *Every* planning task has star-topology factorings. We currently do not lose any runtime on cases we cannot handle, but the number of such cases is large. Another major conclusion is that domain-independent planning may not be the prime target application for star-topology decoupling – why go search for star topologies in arbitrary input problems when there are so many important problems that come with a star topology by definition?

2 Combination with Strong Stubborn Sets Pruning

As star-topology decoupling is complementary to all previous methods, the question arises whether it can be combined with these methods to mutual benefit. The question is especially pertinent as star-topology decoupling essentially just reformulates the state space into a component-wise fashion, and should thus leave many of the technicalities of other methods intact. In the IJCAI'16 paper [11] this extended abstract is based on, we show that this is indeed so for strong stubborn set pruning, the most well-known and wide-spread partial-order reduction method.

Given a state s during search, a stubborn set for s is a subset S of actions so that, in s, to preserve optimality it suffices to branch over those actions from S applicable in s. To ensure this, S collects actions that (1) make progress to the goal, that (2) are required for this progress and are applicable in s, and that (3) interfere with applicable actions already included into S. For (1), it is enough to pick one open goal fact from the goal conjunction; for (2), one recursively includes actions achieving open preconditions of actions already included into S; given (3), all true alternatives at this point – all conflicting decisions one may take in s – are included in S and will be branched over.

As we show in detail in the paper, (1) – (3) transfer directly, almost straightforwardly, to decoupled search, when restricting to *fork* topologies where the leaf components depend on the center but not vice versa. In this setting, reachability within each leaf factor can only grow along a search path (along a transition path by the center), and one can view a decoupled search state s as the union $\overline{s}$ of all leaf states reachable at that point. Given this, in a nutshell, (1) remains as-is, (2) redefines "applicability" relative to $\overline{s}$, and (3) needs to consider only interference with applicable center actions as all applicable leaf actions (more precisely, their effects) are already incorporated into $\overline{s}$.

The only additional complication is that, to guarantee optimality, decoupled search has to proceed beyond decoupled goal states, as cheaper leaf-goal costs may become available on a longer center-component path. Standard strong stubborn sets are unde-

fined for goal states, so a new concept is required here. That can be achieved by replacing (1) with a simple notion of "making progress towards cheaper leaf-goal costs".

In theory, the combination of star-topology decoupling with strong stubborn sets dominates each of its components, and is exponentially separated from each of its components. Indeed, there are cases where the combination is exponentially stronger than *both* its components, i. e., there can be synergistic effects where, thanks to the decoupling, strong stubborn sets are able to exploit a structure they are unable to exploit in the original state space. For example, this happens in simple transportation-style domains akin to the planning competition "Logistics" benchmarks, where a decoupling over packages enables partial-order reduction over trucks.

		Blind Heuristic				LM-cut			
Domain	#	A*	SSS	DS	DSSS	A*	SSS	DS	DSSS
Driverlog	20	7	7	**11**	**11**	13	13	13	13
Logistics'00	28	10	10	22	**24**	20	20	**28**	**28**
Logistics'98	35	2	2	4	**5**	6	6	6	6
Miconic	145	50	45	35	**36**	**136**	**136**	135	135
NoMystery	20	8	7	**17**	15	14	14	**20**	19
Pathways	29	3	3	3	3	4	4	4	4
Rovers	40	6	7	7	**9**	7	9	9	**11**
Satellite	36	6	6	6	6	7	**11**	7	**11**
TPP	27	5	5	**23**	22	5	5	**18**	**18**
Woodworking'08	13	4	6	5	**7**	6	**11**	10	**11**
Woodworking'11	5	0	1	1	**2**	2	**5**	4	**5**
Zenotravel	20	8	7	**11**	**11**	13	13	13	13
$\sum$	418	109	106	145	**151**	233	247	267	**274**

Table 2. Coverage data. Best results highlighted in **boldface**. "SSS": standard search with strong stubborn sets pruning; "DS": star-topology decoupled search; "DSSS": our combination of the two. Results shown on planning competition benchmarks with a fork topology.

In practice, the proposed combination is almost as strong as in theory. It inherits the strengths of its components in almost all cases, and it outperforms both its components in some cases. Table 2 shows coverage data. Observe that the benefit of star-topology decoupling is much stronger for blind search, where the search space reduction does not compete with the reduction already provided by the heuristic function (the state-of-the-art admissible heuristic LM-cut [13]). The additional advantage brought by using strong stubborn sets on top of the decoupling is similar in both settings though.

3 Conclusion

Star-topology decoupling is a powerful new approach to state-space decomposition. The possible benefits are dramatic, the space of opportunities is wide open, the research questions are manifold.

The most obvious direct follow-up on our work here regards the extension of decoupled strong stubborn sets to general star topologies, beyond forks. We believe that this is possible and will lead to similar theoretical and practical results, but that remains to be proven. More generally, the combination with alternate search enhancements is a whole research line in its own right: symmetry reduction; heuristic functions exploiting the star topology; BDDs compactly representing leaf state spaces; adaptations to multi-core search; adaptations of bitstate hashing; etc.

Regarding domain-independent planning, the most pressing question regards more powerful factoring strategies. Much more interesting factorings than our current ones

– X-shapes and forks – definitely exist. As a simple rim case, every partition into 2 subsets of state variables is a star-topology factoring, already opening an exponentially large space of factorings to choose from. The more practically pertinent factorings, though, presumably are the ones with maximum number of leaves. These correspond to maximum independent sets in the input task's causal graph, so approximations to the latter could form the starting point for factoring strategies.

A cute thought is to generalize from the idea to fix and exploit a star-topology profile: *target-profile factoring* could, perhaps, work also for different structural profiles, like chains, trees, DAGs, etc. This suggests an entirely new way of exploiting structure in planning. Instead of *relaxing* the planning task into a (structurally defined) fragment to obtain a heuristic function, try to *factorize* the task into a fragment to obtain a plan. The huge amount of effort invested into tractability analysis (e. g. [17, 3, 5]) could then be redirected to the design of fragments suited to specialized combinatorial search algorithms. In the long term, this could lead to an entire portfolio of target profiles.

Lastly and probably most importantly, the world is full of star topologies so we should go out there and apply star-topology decoupling to those. For AI, a highly suggestive thought is that of multi-agent systems interacting via a set of shared variables – so the agents are the leaves, and the shared variables are the center? Star topology also is a classical system design paradigm, which cries out for applications in model checking. A highly relevant recent direction are concurrent programs under weak memory models (e. g. [16, 1]). Processes run on separate processors (leaves), yet a consistent view of shared memory (center) needs to be guaranteed. The objective is verification, i. e., exhausting the state space, for which star-topology decoupling is especially beneficial (compare Table 1 against Table 2). Key challenges include the adaptation to model checking languages, and the extension to properties beyond reachability.

Acknowledgments. Daniel Gnad was partially supported by the German Research Foundation (DFG), as part of project grant HO 2169/6-1, "Star-Topology Decoupled State Space Search". Martin Wehrle was supported by the Swiss National Science Foundation (SNSF) as part of the project "Automated Reformulation and Pruning in Factored State Spaces (ARAP)".

References

1. Yehia Abd Alrahman, Marina Andric, Alessandro Beggiato, and Alberto Lluch-Lafuente. Can we efficiently check concurrent programs under relaxed memory models in maude? In *Revised Selected Papers of the 10th International Workshop on Rewriting Logic and Its Applications (WRLA'14)*, pages 21–41, 2014.
2. Eyal Amir and Barbara Engelhardt. Factored planning. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 929–935, 2003.
3. Ronen Brafman and Carmel Domshlak. Structure and complexity in planning with unary operators. *Journal of Artificial Intelligence Research*, 18:315–349, 2003.
4. Ronen Brafman and Carmel Domshlak. On the complexity of planning for agent teams and its implications for single agent planning. *Artificial Intelligence*, 198:52–71, 2013.
5. Hubie Chen and Omer Giménez. Causal graphs and structurally restricted planning. *Journal of Computer and System Sciences*, 76(7):579–592, 2010.

6. Carmel Domshlak, Michael Katz, and Alexander Shleyfman. Enhanced symmetry breaking in cost-optimal planning as forward search. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, 2012.
7. Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of mcmillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.
8. Eric Fabre, Loïg Jezequel, Patrik Haslum, and Sylvie Thiébaux. Cost-optimal factored planning: Promises and pitfalls. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, pages 65–72, 2010.
9. Daniel Gnad and Jörg Hoffmann. Beating LM-cut with h^{max} (sometimes): Fork-decoupled state space search. In *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*, 2015.
10. Daniel Gnad and Jörg Hoffmann. Red-black planning: A new tractability analysis and heuristic function. In *Proceedings of the 8th Annual Symposium on Combinatorial Search (SOCS'15)*, 2015.
11. Daniel Gnad, Martin Wehrle, and Jörg Hoffmann. Decoupled strong stubborn sets. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, 2016.
12. Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.
13. Malte Helmert and Carmel Domshlak. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, pages 162–169, 2009.
14. Sarah L. Hickmott, Jussi Rintanen, Sylvie Thiébaux, and Langford B. White. Planning via petri net unfolding. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 1904–1911, 2007.
15. Jörg Hoffmann, Peter Kissmann, and Álvaro Torralba. "Distance"? Who Cares? Tailoring merge-and-shrink heuristics to detect unsolvability. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI'14)*, 2014.
16. Bengt Jonsson. State-space exploration for concurrent algorithms under weak memory orderings. *SIGARCH Computer Architecture News*, 36(5):65–71, 2008.
17. Peter Jonsson and Christer Bäckström. Incremental planning. In *European Workshop on Planning*, 1995.
18. Victor Khomenko and Maciej Koutny. Towards an efficient algorithm for unfolding petri nets. In *Proceedings of the 12th International Conference on Concurrency Theory (CONCUR'01)*, pages 366–380, 2001.
19. Craig Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68(2):243–302, 1994.
20. Kenneth L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *Proceedings of the 4th International Workshop on Computer Aided Verification (CAV'92)*, pages 164–177, 1992.
21. César Rodríguez and Stefan Schwoon. Cunf: A tool for unfolding and verifying petri nets with read arcs. In *Proceedings of the 11th International Symposium on Automated Technology for Verification and Analysis (ATVA'13)*, pages 492–495, 2013.
22. Peter Starke. Reachability analysis of petri nets using symmetries. *Journal of Mathematical Modelling and Simulation in Systems Analysis*, 8(4/5):293–304, 1991.
23. Antti Valmari. Stubborn sets for reduced state space generation. In *Proceedings of the 10th International Conference on Applications and Theory of Petri Nets*, pages 491–515, 1989.
24. Martin Wehrle and Malte Helmert. Efficient stubborn sets: Generalized algorithms and selection strategies. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*, 2014.