DecAbStar

Daniel Gnad[1], Silvan Sievers[2], Álvaro Torralba[3]

[1] Linköping University, Sweden
[2] University of Basel, Switzerland
[3] Aalborg University, Denmark
daniel.gnad@liu.se, silvan.sievers@unibas.ch, alto@cs.aau.dk

Abstract

DecAbStar extends Fast Downward by **Dec**oupled state space search, a technique that exploits the independence between components of a planning task to reduce the size of the state-space representation. Partitioning the state variables into components, such that the interaction between these takes the form of a **Star** topology, decoupled search only searches over action sequences affecting the center component of the topology, and enumerates reachable assignments to each leaf component separately. This can lead to an exponential reduction in the search-space representation size. It is not always easy to find a partitioning for a given planning task, though, so we extend decoupled search by a fallback option which runs explicit-state search whenever no (good) partitioning could be found. DecAbstar uses **Ab**straction heuristics to guide the search both in the decoupled and explicit search component.

General Overview

Decoupled search reduces the representation size of search spaces by exploiting the structure of the problem within the search (Gnad and Hoffmann 2015; Gnad, Hoffmann, and Domshlak 2015; Gnad and Hoffmann 2018). The size of the *decoupled state space* can be exponentially smaller than that of the explicit state space, which decoupled search achieves by partitioning the task into several components, called *factors*, trying to identify a *star topology* with a single *center factor* that interacts with multiple *leaf factors*. By enforcing this structure, and thereby restricting the dependencies between the components, decoupled search has proven to be very efficient and competitive with state-of-the-art planners.

The performance of decoupled search is highly influenced by the outcome of the *factoring* process, i. e., the process of partitioning the state variables. Just, how to find a good factoring, and what qualifies a factoring as being good? These questions have been addressed by Gnad, Poser, and Hoffmann (2017); Schmitt, Gnad, and Hoffmann (2019); Gnad, Torralba, and Fišer (2022), who devised algorithms that can detect star topologies on a wide range of planning domains. Still, the proposed algorithms can *fail* to find a factoring, or succeed, but return a factoring with undesired properties, e. g., large leaf components that incur a prohibitive runtime overhead when generating new search states. In this case, we simply run explicit-state search instead.

In decoupled search, dominance pruning instead of duplicate checking is performed to prune previously seen states during search. Dominance pruning identifies states that can be safely discarded without affecting completeness and optimality. Since decoupled states represent sets of states, decoupled search is less likely to find exact duplicates and dominance pruning can be exponentially stronger compared to duplicate checking in explicit-state search. We employ the dominance pruning techniques introduced by Gnad (2021).

We guide the search using a diverse set of abstraction heuristics, namely explicit pattern database (PDB) heuristics (Culberson and Schaeffer 1998; Edelkamp 2001), symbolic PDB heuristics (Edelkamp 2002; Kissmann and Edelkamp 2011; Torralba, Linares López, and Borrajo 2018), and merge-and-shrink heuristics (Helmert et al. 2014; Sievers and Helmert 2021). We combine multiple explicit PDBs in saturated cost partitionings (Seipp, Keller, and Helmert 2020). All these heuristics have recently been adapted to work in decoupled search (Sievers, Gnad, and Torralba 2022; Gnad, Sievers, and Torralba 2023). Variants of these heuristics are used in decoupled search as well as in the fallback case, i. e., when no good factoring could be detected.

We extend the standard preprocessor of Fast Downward with the h^2-based task simplification by Alcázar and Torralba (2015), which removes irrelevant and unreachable facts and actions from the task.

Implementation & Configurations

Decoupled Search has been implemented as an extension of the Fast Downward (FD) planning system (Helmert 2006). By changing the low-level state representation, many of FD's built-in algorithms and functionality can be used with only minor adaptations. We perform decoupled search like introduced by Gnad and Hoffmann (2018), using the factoring method called bM80s by Gnad, Torralba, and Fišer (2022). The factoring process is given a time limit of 30 seconds. After FD's translator component finishes, we perform a relevance analysis based on h^2 for 4 minutes to eliminate actions and simplify the planning task prior to the search (Alcázar and Torralba 2015).

Decoupled search is the main component of our planner. However, as outlined before, our factoring strategies are not guaranteed to find good task decompositions. Thus, in that case we run explicit-state search as fallback method. Our implementation of decoupled search does not support conditional effects, so we also fall back to explicit-state search in their presence. For the same reason, we switch off explicit PDB heuristics if there are conditional effects. More advances PDDL features such as derived predicates or axioms are not supported by the planner.

We perform a single A^* search using three heuristics, combined using maximum:

- For explicit PDBs, we combine a set of PDB heuristics using saturated cost partitioning, adapting code from the Scorpion planner for the latter (Seipp 2018). We compute patterns by running the multiple CEGAR algorithm by Rovner, Sievers, and Helmert (2019) for 70s, systematically generating all interesting patterns up to size 2 (Pommerening, Röger, and Helmert 2013) for 70s, and running hill climbing for 110s. We then compute a set of diverse (heuristic) orders for these PDBs for 110s, each optimized using dynamic greedy ordering for 2s (Seipp, Keller, and Helmert 2020). When running decoupled search, we use the single-leaf approximation to obtain an efficient heuristic computation (Sievers, Gnad, and Torralba 2022).
- For symbolic PDBs, abstractions are computed with Gamer PDBs (Kissmann and Edelkamp 2011), using the symbolic search enhancements by Torralba et al. (2017) and the CUDD 3.0.0 library (Somenzi 1997). We give 350s and 4GiB to compute the abstraction when running decoupled search, 500s and 4GiB for explicit-state search in absence of conditional effects, and 550s and 5GiB in presence of conditional effects. When running decoupled search, we use the ADD traversal lookup to obtain an efficient heuristic computation (Gnad, Sievers, and Torralba 2023).
- For merge-and-shrink heuristics, we run the sbMIASM merge strategy (Sievers, Wehrle, and Helmert 2016), using bisimulation shrinking (Nissim, Hoffmann, and Helmert 2011), and exact label reduction (Sievers, Wehrle, and Helmert 2014). We limit the size of each intermediate abstraction to 50.000 states. For decoupled search, we cluster the variables using the factoring to obtain an efficient heuristic computation by forcing the factored mapping to be compliant with the factoring. For explicit-state search, we cluster the variables using the strongly connected components of the causal graph (Sievers, Wehrle, and Helmert 2016). We impose a runtime limit of 450s to construct the heuristic for decoupled search, 600s for explicit-state search without conditional effects, and 650s in presence of conditional effects.

Post-Competition Analysis

We analyzed the competition results by investigating the log files that have been provided by the organizers. Table 1 summarizes our findings in a per-domain analysis for all competition domains. The first two columns, $\mathcal{F}$ and "oom", show the number of instances in which a factoring was detected, so decoupled search was in play, respectively in which the preprocessing of the heuristics ran out of memory. Our first observation is that there are only two domains, quantum-layout and recharging-robots, in which decoupled search can actually be performed. This is a quite low number compared to previous competitions. Hence, the results mostly show the behaviour of explicit-state search. Second, the number of out-of-memory instances during preprocessing indicates that we were a bit too generous with the memory given to the heuristics. This might have been amplified by the implementation of the external memory limit, which enforced 8GiB for the entire process, out of which the Apptainer reserves a significant chunk. With that, the planner did not even start the search in 114 out of 220 instances.

				Coverage		
Domain	#	$\mathcal{F}$	oom	Dec.	Expl.	$\sum$
folding	20	0	9	0	2	2
folding^N	20	0	10	0	2	2
labyrinth	20	0	15	0	1	1
quantum-layout	20	20	1	12	0	12
recharging-robots	20	4	2	4	0	4
recharging-robots^N	20	4	4	4	6	10
ricochet-robots	20	0	0	0	7	7
rubiks-cube	20	0	20	0	0	0
rubiks-cube^N	20	0	20	0	0	0
slitherlink	20	0	20	0	0	0
slitherlink^N	20	0	13	0	5	5
$\sum$	220	28	114	20	23	43

Table 1: Per-domain analysis of the preprocessing and coverage. $\mathcal{F}$ shows the number of instances for which a non-trivial factoring can be computed; "oom" shows the number of instances in which the planner ran out of memory when pre-computing the heuristics. The right part shows coverage for decoupled search ("Dec."), explicit search ("Expl."), and the total coverage.

The right part of the table shows detailed coverage results, distinguishing decoupled search from explicit search. We observe that from the 28 instances tackled by decoupled search, 20 were solved. Explicit search solves an additional 23 instances, summing up to a total coverage of 43.

Acknowledgments

Since we build upon Fast Downward and code from Scorpion, we would like to thank all contributors to these planners.

References

Alcázar, V.; and Torralba, Á. 2015. A Reminder about the Importance of Computing and Exploiting Invariants in Planning. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 2–6. AAAI Press.

Culberson, J. C.; and Schaeffer, J. 1998. Pattern Databases. *Computational Intelligence*, 14(3): 318–334.

Edelkamp, S. 2001. Planning with Pattern Databases. In Cesta, A.; and Borrajo, D., eds., *Proceedings of the Sixth Eu-*

ropean Conference on Planning (ECP 2001), 84–90. AAAI Press.

Edelkamp, S. 2002. Symbolic Pattern Databases in Heuristic Search Planning. In Ghallab, M.; Hertzberg, J.; and Traverso, P., eds., *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, 274–283. AAAI Press.

Gnad, D. 2021. Revisiting Dominance Pruning in Decoupled Search. In Leyton-Brown, K.; and Mausam, eds., *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2021)*, 11809–11817. AAAI Press.

Gnad, D.; and Hoffmann, J. 2015. Beating LM-Cut with h^{max} (Sometimes): Fork-Decoupled State Space Search. In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS 2015)*, 88–96. AAAI Press.

Gnad, D.; and Hoffmann, J. 2018. Star-Topology Decoupled State Space Search. *Artificial Intelligence*, 257: 24–60.

Gnad, D.; Hoffmann, J.; and Domshlak, C. 2015. From Fork Decoupling to Star-Topology Decoupling. In Lelis, L.; and Stern, R., eds., *Proceedings of the Eighth Annual Symposium on Combinatorial Search (SoCS 2015)*, 53–61. AAAI Press.

Gnad, D.; Poser, V.; and Hoffmann, J. 2017. Beyond Forks: Finding and Ranking Star Factorings for Decoupled Search. In Sierra, C., ed., *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017)*, 4310–4316. IJCAI.

Gnad, D.; Sievers, S.; and Torralba, Á. 2023. Efficient Evaluation of Large Abstractions for Decoupled Search: Merge-and-Shrink and Symbolic Pattern Databases. In Koenig, S.; Stern, R.; and Vallati, M., eds., *Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling (ICAPS 2023)*, 138–147. AAAI Press.

Gnad, D.; Torralba, Á.; and Fišer, D. 2022. Beyond Stars - Generalized Topologies for Decoupled Search. In Thiébaux, S.; and Yeoh, W., eds., *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)*, 110–118. AAAI Press.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge-and-Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *Journal of the ACM*, 61(3): 16:1–63.

Kissmann, P.; and Edelkamp, S. 2011. Improving Cost-Optimal Domain-Independent Symbolic Planning. In Burgard, W.; and Roth, D., eds., *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, 992–997. AAAI Press.

Nissim, R.; Hoffmann, J.; and Helmert, M. 2011. Computing Perfect Heuristics in Polynomial Time: On Bisimulation and Merge-and-Shrink Abstraction in Optimal Planning. In Walsh, T., ed., *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, 1983–1990. AAAI Press.

Pommerening, F.; Röger, G.; and Helmert, M. 2013. Getting the Most Out of Pattern Databases for Classical Planning. In Rossi, F., ed., *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, 2357–2364. AAAI Press.

Rovner, A.; Sievers, S.; and Helmert, M. 2019. Counterexample-Guided Abstraction Refinement for Pattern Selection in Optimal Classical Planning. In Lipovetzky, N.; Onaindia, E.; and Smith, D. E., eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 362–367. AAAI Press.

Schmitt, F.; Gnad, D.; and Hoffmann, J. 2019. Advanced Factoring Strategies for Decoupled Search Using Linear Programming. In Lipovetzky, N.; Onaindia, E.; and Smith, D. E., eds., *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)*, 377–381. AAAI Press.

Seipp, J. 2018. Fast Downward Scorpion. In *Ninth International Planning Competition (IPC-9): Planner Abstracts*, 77–79.

Seipp, J.; Keller, T.; and Helmert, M. 2020. Saturated Cost Partitioning for Optimal Classical Planning. *Journal of Artificial Intelligence Research*, 67: 129–167.

Sievers, S.; Gnad, D.; and Torralba, Á. 2022. Additive Pattern Databases for Decoupled Search. In Chrpa, L.; and Saetti, A., eds., *Proceedings of the 15th Annual Symposium on Combinatorial Search (SoCS 2022)*, 180–189. AAAI Press.

Sievers, S.; and Helmert, M. 2021. Merge-and-Shrink: A Compositional Theory of Transformations of Factored Transition Systems. *Journal of Artificial Intelligence Research*, 71: 781–883.

Sievers, S.; Wehrle, M.; and Helmert, M. 2014. Generalized Label Reduction for Merge-and-Shrink Heuristics. In Brodley, C. E.; and Stone, P., eds., *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI 2014)*, 2358–2366. AAAI Press.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An Analysis of Merge Strategies for Merge-and-Shrink Heuristics. In Coles, A.; Coles, A.; Edelkamp, S.; Magazzeni, D.; and Sanner, S., eds., *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS 2016)*, 294–298. AAAI Press.

Somenzi, F. 1997. CUDD: CU decision diagram package. Technical report, University of Colorado at Boulder.

Torralba, Á.; Alcázar, V.; Kissmann, P.; and Edelkamp, S. 2017. Efficient Symbolic Search for Cost-optimal Planning. *Artificial Intelligence*, 242: 52–79.

Torralba, Á.; Linares López, C.; and Borrajo, D. 2018. Symbolic perimeter abstraction heuristics for cost-optimal planning. *Artificial Intelligence*, 259: 1–31.