ComplementaryPDB Planner

Santiago Franco[1], Stefan Edelkamp[2], Ionut Moraru[3]

[1] Department of Computer Science, Royal Holloway University of London, Egham, UK
[2] Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, CZ
[3] School of Computer Science, University of Lincoln, Lincoln, UK
[1]Santiago.francoaixela@rhul.ac.uk, [2]stefan.edelkamp@fel.cvut.cz, [3]imoraru@lincoln.ac.uk

Abstract

ComplementaryPDB is a planner that uses heuristic search via Symbolic Pattern Databases (PDBs) and uses a greedy pattern selection algorithm – Partial Gamer – combined with pattern collections from bin-packing pattern selection algorithms. For more information on this method, we direct the reader to the main paper describing this method [Moraru et al. 2019a].

Introduction

The automated generation of search heuristics is one of the holy grails in AI, and goes back to early work of Gaschnik [Gaschnig 1979a], Pearl [Pearl 1985], and Prieditis [Prieditis 1993b]. In most cases, lower bound heuristics are problem relaxations: each plan in the original state space maps to a shorter one in some corresponding abstract one. In the worst case, searching the abstract state spaces at every given search nodes exceeds the time of blindly searching the concrete search space [Valtorta 1984a].

With pattern database heuristic (PDBs), all efforts in searching the abstract state space are spent prior to the plan search, so that these computations amortize through multiple lookups. The ComplementaryPDB planner bases its search on the PDB heuristic, combining the work from [Franco et al. 2017] and [Moraru et al. 2019a]. It is an evolution of the Complementary planners submitted at the 9th International Planner Competition.

In this planner abstract, we will briefly describe Pattern Databases. For more information on this method, we direct the reader to the main paper describing this method [Moraru et al. 2019a]. The results, evaluation and discussion sections will be added after the full results of the competition will be made public.

Pattern Databases

Initial results of Culberson and Schaeffer [Culberson and Schaeffer 1998a] in sliding-tile puzzles, where the concept of a pattern is a selection of tiles, quickly carried over to a number of combinatorial search domains, and helped to optimally solve random instances of the Rubik's cube, with non-pattern labels being removed [Korf 1997]. When shifting from breadth-first to shortest-path search, the exploration of the abstract state-space can be extended to include action costs.

The combination of several databases into one, however, is tricky [Haslum et al. 2007a]. While the maximum of two PDBs always yields a lower bound, the sum usually does not. Korf and Felner [Korf and Felner 2002] showed that with a certain selection of disjoint (or additive) patterns, the values in different PDBs can be added while preserving admissibility. Holte et al. [Holte et al. 2004a] indicated that several smaller PDBs may outperform one large PDB. The notion of a pattern has been generalized to production systems in vector notation [Holte and Hernádvölgyi 1999], while the automated pattern selection process for the construction of PDBs goes back to the work of Edelkamp [Edelkamp 2006].

Many planning problems can be translated into state spaces of finite domain variables [Helmert 2004], where a selection of variables (pattern) influences both states and operators. For disjoint patterns, an operator must distribute its original cost, if present in several abstractions [Katz and Domshlak 2008; Yang et al. 2008].

During the PDB construction process, the memory demands of the abstract state space sizes may exceed the available resources. To handle large memory requirements, symbolic PDBs succinctly represent state sets as binary decision diagrams [Edelkamp 2002a]. However, there are an exponential number of patterns, not counting alternative abstraction and cost partitioning methods. Hence, the automated construction of informative PDB heuristics remains a combinatorial challenge. Hill-climbing strategies have been proposed [Haslum et al. 2007a], as well as more general optimization schemes such as genetic algorithms [Edelkamp 2006] or culprit based predictors [Franco et al. 2017]. The biggest area of research in this area remains the quality evaluation of a PDB (in terms of the heuristic values for the concrete state space) which can only be estimated. Usually, this involves generating the PDBs and evaluating them [Edelkamp 2014; Korf 1997; Franco et al. 2017; Levi et al. 2016].

Adaptation to IPC 2023

There is work describing the participating complementary planner for the precursor IPC 2018 [Franco et al. 2018]. The work goes back to work by [Franco et al. 2017]. Based

on the results obtained in our other PDB BDD-type planners [Moraru et al. 2019b], also documented in the 2023 PhD thesis *From Pattern Databases to Plan Libraries: Utilising Memory-based Methods for Improving AI Planning Performance* of Ionut Moraru for the IPC-2023 we simply modified the variable ordering in the BDDs to what is called Gamer ordering.

Results

Different to the last IPC, where the BDD-based planners were taking all the places from 2 to 5, and the winner Delfi was a portfolio that was calling the previous BDD-based winning planner SymBA* more than 70% of its time, the results in the IPC for all BDD-based planners in the competition were unfortunate.

The winners achieved double of our score, measured in the problems being solved. There was no BDD planner in the top 10, maybe in some of the portfolios, but the domains used in this competition seemed to favor explicit-state planners.

In many cases we failed to start the planning process and were stuck either within grounding or then, by building the transition relation. For example, baseline blind, certainly not the best search technique, was better than all symbolic planners (SymBD,SymK and ours). None of the symbolic planners could solve any instance of Rubik's cube. Besides the the large size of the problems, the large number of conditional effects also negatively affected all the symbolic planners this time.

Lets have a look at the PDDL description of the Rubik's Cube. The encoding relies on a $\{1..3\}^3$ coordinate systems, with variables for the coloring of the labels, in each of the 6 colors. There are no parameters of the actions, all parameters are contained in the conditional effects.

```
(:action R
 :parameters ()
 :precondition (and)
 :effect
    (and
    (forall (?x ?y ?z)
    (when (cube5 ?x ?y ?z)
      (and (not (cube5 ?x ?y ?z)) (cube7 ?y ?x ?z))))
    (forall (?x ?y ?z) (when (cube7 ?x ?y ?z)
      (and (not (cube7 ?x ?y ?z)) (cube8 ?y ?x ?z))))
    (forall (?x ?y ?z) (when (cube8 ?x ?y ?z)
      (and (not (cube8 ?x ?y ?z)) (cube6 ?y ?x ?z))))
    (forall (?x ?y?z)
      (when (cube6 ?x ?y ?z)
        (and (not (cube6 ?x ?y ?z)) (cube5 ?y ?x ?z))))
    (forall (?x ?z) (when (edge57 ?x ?z)
      (and (not (edge57 ?x ?z)) (edge78 ?x ?z))))
    (forall (?y ?z) (when (edge78 ?y ?z)
      (and (not (edge78 ?y ?z)) (edge68 ?y ?z))))
    (forall (?x ?z) (when (edge68 ?x ?z)
      (and (not (edge68 ?x ?z)) (edge56 ?x ?z))))
    (forall (?y ?z) (when (edge56 ?y ?z)
      (and (not (edge56 ?y ?z)) (edge57 ?y ?z))))
    )
)
```

The move action, as defined, involves all the 24 cubelets instead of the actual cubelets being moved with each instantiated move. An alternative PDDL description using only the cubelets being moved would lead to a much more concise encoding. With the current action specification, most planners may not even be able to recognize that the cubelets labels are actually unique.

Even though it looks like there is only `?x?y?z` participating in the encoding, there are actually a lot of hidden parameters as the `?x?y?z` in one conditional effect might be bound to a different value than the other `?x?y?z`. Instead of only 8 cubelets with colors being moved in one twist, therefore there are in fact 24 hidden parameters variables of size 6, which corresponds to 24^6 ground operators of one action.

One first idea would be rewrite the actions into smaller subactions like

```
(:action R1
 :parameters (?x ?y ?z)
 :precondition (and (doR1)  (cube5 ?x ?y ?z))
 :effect
    (and (not doR1)
      (not cube5 ?x ?y ?z) (cube7 ?x ?y ?z) (doR2))
)
(:action R2
 :parameters (?x ?y ?z)
 :precondition (and (doR2)  (cube7 ?x ?y ?z))
 :effect
    (and (not doR2)
      (not cube7 ?x ?y ?z) (cube8 ?x ?y ?z) (doR3))
)
...
```

to break the actions into many that have to be executed one after the other. But even if this rewrite would work the subactions R1, R2 would have to be executed in parallel, not in sequence.

The PDDL for *labyrinth* has no conditional effects but each action at least 8 parameters with the labyrinth dimensions used as a variable domain, so that it will lead to very large groundings as well. There might be some split of one action to reduce the number of instantiated actions.

On the first look, the PDDL for *folding* (derived from answer set programming encoding) does not have that many parameters for the action. Here the domains of the variables can be large.

In *quantum* and *recharging* our planner results were okay, at most one problem shy from the best performers. We found one domain variable called `depth` that might raise to astronomic domain size.

Slitherlink has actions with eight parameters and will be very difficult to ground. Why we were bad at *ricochet robots*, we haven't validated yet. But both seem to have be compiled from ASP instances, which does not seem amenable to BDDs, which prefer a minimized state encoding.

Discussion

That BDDs and PDBs cannot solve a single problem in the Rubik's Cube task was astonishing, especially given that Korf's 1999 first solutions to random instances were using PDBs.

In general this assertion might not be true, there are recent pieces of work [Büchner et al. 2022] that indicate that even planning PDBs are doing best, and our conversation with David Speck indicated that BDDs perform better in a different PDDL encoding, even though we know from single-agent search research that permutation puzzles like Blocksworld, the 15-Puzzle, Rubik's Cube are not their strength.

The performance lack is due to the modeling and the excess of conditional effects. We can handle conditional effects, but the way actions are specified, they are simply too many. In a compilation the object variables in the conditions are additional parameters. The Rubik's model likely taken from [Muppasani et al. 2023] is elegant, but not good for grounding and minimizing the state encoding [Edelkamp and Helmert 1999]. The grounding might just be possible, as blind search and others can solve a few problems, but we are pretty sure that we cannot even start planning.

Large groundings are likely for many of the new PDDL IPC domains. Due to the modeling chosen, we looked at several domains in some detail, and quite often there are many parameters. Sometimes we counted 8 and more. One might split some of these operators, but to do this automatically is itself a research aspect.

What likely broke our neck were the planning models chosen. Sure this choice is up to the courtesy of the competition designers. We don't think that all the problem tasks we could not solve are difficult from a combinatorial point of view, but when we cannot start planning, we are doomed.

The fact that blind search can solve some problems we cannot is because we also have to build the transition relation, which takes time and space. These observations are likely true for all BDD based planners, which has limited their applicability in this year's IPC.

Maybe the new answer set programming grounding mechanism might help [Corrêa et al. 2023], as there certainly is a lot of redundancy in our current grounding methodology for these types of domain descriptions. We simply ground too many actions that will never be used.

Conclusion

Based on the current research interest in huge and complex groundings, our focus on improving search instead of grounding resulted in poor performance for all BDD planners. As shown in previous IPCs, BDDs can play an important role (assuming grounding the problems and building the necessary transition relations is doable), especially when it comes to combinatorial complex problems.

While lifted planning is important, we have to take care that the research does not drift, so that large models become the new starndard. Maybe, next competition there will be less stress on lifted planning, grounding size does not necessarily equate to complexity. Note that propositional planning is PSPACE-complete already in the grounded encoding.

Even with the clear-cut outcome in the current IPC that suggests that BDDs have fallen behind, we don't think that symbolic search has stopped being part of the state of the art for AI planning. As usual, the tension between domain choices and how they are modelled in PDDL makes it difficult for IPCs to be the deciders of what is the "state-of-the-art" in planning.

In general more research is needed to make BDDs efficient on these domains. Aspects like automated reformulation, lifted symbolic planning, and more efficient groundings are exciting research areas.

We suggest the organizers to run a workshop on the future of the competition in some upcoming ICAPS. After the portfolio problem, that has not been resolved but was not the core aspect here, there could be some more discussion on the conciseness of encodings. Research-wise lifted planning is interesting, but it is much slower, and heuristics do not all apply in the same way.

The winners clearly deserve the victory, and we personally prefer winners who do not over-rely on previous techniques. Even given that the current winner in fact is a portfolio, it included many new contributions, such as a module for lifted planning. Its algorithm engineering efforts resulting in a distinguished performance is impressive. The runner-up planner Scorpion-2023 was also very impressive. It is significantly different to its precursor in 2018, outperforms all BDD planners working on explicit-state abstractions.

Acknowledgments

We would like to thank all contributors to these the Fast-Downward planner, as ComplementaryPDB is built on top of it. This research was partly funded by Czech Science Foundation grant number 22-30043S

References

Büchner, C.; Ferber, P.; Seipp, J.; and Helmert, M. 2022. A Comparison of Abstraction Heuristics for Rubik's Cube. In *ICAPS 2022 Workshop on Heuristics and Search for Domain-independent Planning*.

Corrêa, A. B.; Hecher, M.; Helmert, M.; Longo, D. M.; Pommerening, F.; and Woltran, S. 2023. Grounding Planning Tasks Using Tree Decompositions and Iterated Solving. In Koenig, S.; Stern, R.; and Vallati, M., eds., *Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling, July 8-13, 2023, Prague, Czech Republic*, 100–108. AAAI Press.

Culberson, J. C.; and Schaeffer, J. 1998a. Pattern Databases. *Computational Intelligence*, 14(4): 318–334.

Culberson, J. C.; and Schaeffer, J. 1998b. Pattern Databases. *Computational Intelligence*, 14(4): 318–334.

Derval, G.; Regin, J.-C.; and Schaus, P. 2017. Improved Filtering for the Bin-Packing with Cardinality Constraint. In *CP*.

Dósa, G. 2007. The tight bound of first fit decreasing bin-packing algorithm is $FFD(I) \leq 11/9OPT(I) + 6/9$. In *Combinatorics, Algorithms, Probabilistic and Experimental Methodologies*, 1–11. Springer.

Edelkamp, S. 2001. Planning with pattern databases. In *ECP*.

Edelkamp, S. 2002a. Symbolic Pattern Databases in Heuristic Search Planning. In *AIPS*, 274–283.

Edelkamp, S. 2002b. Symbolic Pattern Databases in Heuristic Search Planning. In *AIPS*, 274–293.

Edelkamp, S. 2006. Automated creation of pattern database search heuristics. In *International Workshop on Model Checking and Artificial Intelligence*, 35–50. Springer.

Edelkamp, S. 2007. Automated Creation of Pattern Database Search Heuristics. In *MOCHART*, 36–51.

Edelkamp, S. 2014. Planning with pattern databases. In *Sixth European Conference on Planning*.

Edelkamp, S.; and Helmert, M. 1999. Exhibiting Knowledge in Planning Problems to Minimize State Encoding Length. In Biundo, S.; and Fox, M., eds., *Recent Advances in AI Planning, 5th European Conference on Planning, ECP'99, Durham, UK, September 8-10, 1999, Proceedings*, volume 1809 of *Lecture Notes in Computer Science*, 135–147. Springer.

Franco, S.; Lelis, L. H.; Barley, M.; Edelkamp, S.; Martines, M.; and Moraru, I. 2018. The Complementary2 planner in the IPC 2018. *IPC-9 planner abstracts*, 28–31.

Franco, S.; Torralba, A.; Lelis, L. H.; and Barley, M. 2017. On creating complementary pattern databases. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 4302–4309. AAAI Press.

Garey, M. R.; and Johnson, D. S. 1979. *Computers and Intractibility, A Guide to the Theory of NP-Completeness*. Freeman & Company.

Gaschnig, J. 1979a. A Problem Similarity Approach to Devising Heuristics: First Results. In *IJCAI*, 434–441.

Gaschnig, J. 1979b. A Problem Similarity Approach to Devising Heuristics: First Results. In *IJCAI*, 434–441.

Hart, N.; Nilsson, J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on System Science and Cybernetics*, 4(2): 100–107.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *AAAI*, volume 5, 9–13.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007a. Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning. In *AAAI*, 1007–1012.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007b. Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning. In *AAAI*, 1007–1012.

Helmert, M. 2004. A Planning Heuristic Based on Causal Graph Analysis. In *ICAPS*, 161–170.

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.*, 26: 191–246.

Holland, J. 1975. *Adaption in Natural and Artificial Systems*. Ph.D. thesis, University of Michigan.

Holte, R.; Newton, J.; Felner, A.; Meshulam, R.; and Furcy, D. 2004a. Multiple Pattern Databases. In *ICAPS*, 122–131.

Holte, R.; Newton, J.; Felner, A.; Meshulam, R.; and Furcy, D. 2004b. Multiple Pattern Databases. In *ICAPS*, 122–131.

Holte, R. C.; Grajkowski, J.; and Tanner, B. 2005. Hierarchical Heuristic Search Revisited. In *SARA*, 121–133.

Holte, R. C.; and Hernádvölgyi, I. T. 1999. A space-time tradeoff for memory-based heuristics. In *AAAI/IAAI*, 704–709. Citeseer.

Holte, R. C.; and Hernádvölgyi, I. T. 2001. A Space-Time Tradeoff for Memory-Based Heuristics.

I. Moraru, M. M., S. Edelkamp. 2018. Automated Pattern Selection using MiniZinc. In *CP Workshop on Constraints & AI Planning*.

Karpas, E.; Katz, M.; and Markovitch, S. 2011. When Optimal Is Just Not Good Enough: Learning Fast Informative Action Cost Partitionings. In *ICAPS*.

Katz, M.; and Domshlak, C. 2008. Optimal Additive Composition of Abstraction-based Admissible Heuristics. In *ICAPS*, 174–181.

Korf, R. E. 1997. Finding Optimal Solutions to Rubik's Cube Using Pattern Databases. In *AAAI*, 700–705.

Korf, R. E. 2002. A new algorithm for optimal bin packing. In *AAAI*, 731–736.

Korf, R. E. 2003. An improved algorithm for optimal bin packing. In *IJCAI*, 1252–1258.

Korf, R. E.; and Felner, A. 2002. *Chips Challenging Champions: Games, Computers and Artificial Intelligence*, chapter Disjoint Pattern Database Heuristics, 13–26. Elsevier.

Lelis, L. H. S.; Franco, S.; Abisrror, M.; Barley, M.; Zilles, S.; and Holte, R. C. 2016. Heuristic Subset Selection in Classical Planning. In *IJCAI*, 3185–3191.

Levi, L. H.; Franco, S.; Abisrror, M.; Barley, M.; Zilles, S.; and Holte, R. 2016. Heuristic subset selection in classical planning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, 3185–3195. AAAI Press/IJCAI.

Moraru, I.; Edelkamp, S.; Franco, S.; and Martinez, M. 2019a. Simplifying automated pattern selection for planning with symbolic pattern databases. In *KI 2019: Advances in Artificial Intelligence: 42nd German Conference on AI, Kassel, Germany, September 23–26, 2019, Proceedings 42*, 249–263. Springer.

Moraru, I.; Edelkamp, S.; Franco, S.; and Martínez, M. 2019b. Simplifying Automated Pattern Selection for Planning with Symbolic Pattern Databases. In Benzmüller, C.; and Stuckenschmidt, H., eds., *KI 2019: Advances in Artificial Intelligence - 42nd German Conference on AI, Kassel, Germany, September 23-26, 2019, Proceedings*, volume 11793 of *Lecture Notes in Computer Science*, 249–263. Springer.

Muppasani, B.; Pallagani, V.; Srivastava, B.; and Agostinelli, F. 2023. On Solving the Rubik's Cube with Domain-Independent Planners Using Standard Representations. arXiv:2307.13552.

Pearl, J. 1984. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Pearl, J. 1985. *Heuristics*. Addison-Wesley.

Pommerening, F.; Helmert, M.; and Bonet, B. 2017. Higher-Dimensional Potential Heuristics for Optimal Classical Planning.

Preditis, A. 1993a. Machine Discovery of Admissible Heurisitics. *Machine Learning*, 12: 117–142.

Preditis, A. 1993b. Machine Discovery of Admissible Heuristics. *Machine Learning*, 12: 117–142.

Valtorta, M. 1984a. A result on the computational complexity of heuristic estimates for the A* algorithm. *Information Sciences*, 34: 48–59.

Valtorta, M. 1984b. A result on the computational complexity of heuristic estimates for the A* algorithm. *Information Sciences*, 34: 48–59.

Yang, F.; Culberson, J.; Holte, R.; Zahavi, U.; and Felner, A. 2008. A general theory of additive state space abstractions. *Journal of Artificial Intelligence Research*, 32: 631–662.