On Performance Guarantees for Symbolic Search in Classical Planning (Extended Abstract)*

David Speck, Malte Helmert

University of Basel, Switzerland
⟨davidjakob.speck, malte.helmert⟩@unibas.ch

Abstract

Traditional symbolic search algorithms for classical planning can incur exponential overhead compared to explicit blind search in the presence of complex conditions and effects. To address this problem, we explore conjunctive partitioning and propose fully automated, domain-independent methods for representing actions and goal conditions in a partitioned form. One of our methods, based on the Tseitin transformation, yields a symbolic search algorithm that in the worst case incurs only a polynomial overhead and in the best case can be exponentially more efficient than its explicit counterpart. An empirical evaluation shows that our theoretical findings carry over into practice: our algorithms solve planning problems previously intractable for symbolic search and perform favorably overall compared to traditional symbolic search, explicit blind search, and other state-of-the-art planners.

Introduction

Symbolic search is an established method for classical cost-optimal planning. It is known to have complementary strengths to explicit heuristic search and is often assumed to be superior to explicit blind search. While this is often true for planning problems with simple conditions and effects, such as STRIPS (Fikes and Nilsson 1971) or SAS$^+$ (Bäckström and Nebel 1995), symbolic search algorithms face severe performance problems in the presence of complex conditions and effects (Speck, Seipp, and Torralba 2025). This can be seen, for example, in the 2023 edition of the International Planning Competition (Taitler et al. 2024). In domains like Rubik's Cube, all planners based on symbolic search were not able to solve a single task, although some of them were solvable with only a few actions, and explicit blind search easily solved those. The reason for this is that it is infeasible to represent the actions as logical formulas, as so-called transition relations, in the form of monolithic binary decision diagrams (BDDs) (Bryant 1986).

We propose a conjunctive partitioned representation of complex formulas to address the representation challenges encountered by modern planners based on symbolic search. While the conjunctive partitioning of transition relations has long been standard in model checking (Burch, Clarke, and Long 1991), the planning community has largely neglected it, considering disjunctive partitioning to be a more appropriate approach for representing actions. On the theoretical side, we describe a method that can effectively derive a variable-based conjunctive form (Burch, Clarke, and Long 1991) of complex actions. Furthermore, we propose an approach based on the Tseitin transformation (Tseitin 1968) which has a polynomial performance guarantee over explicit blind search. On the practical side, symbolic search with the proposed partitioned representations can for the first time effectively solve problems from domains with complex effects such as Rubik's Cube. This establishes a novel symbolic search approach that compares favorably to traditional symbolic search and other explicit search approaches.

*Abridged version of a paper at the European Conference on Artificial Intelligence (ECAI) 2025 (Speck and Helmert 2025).

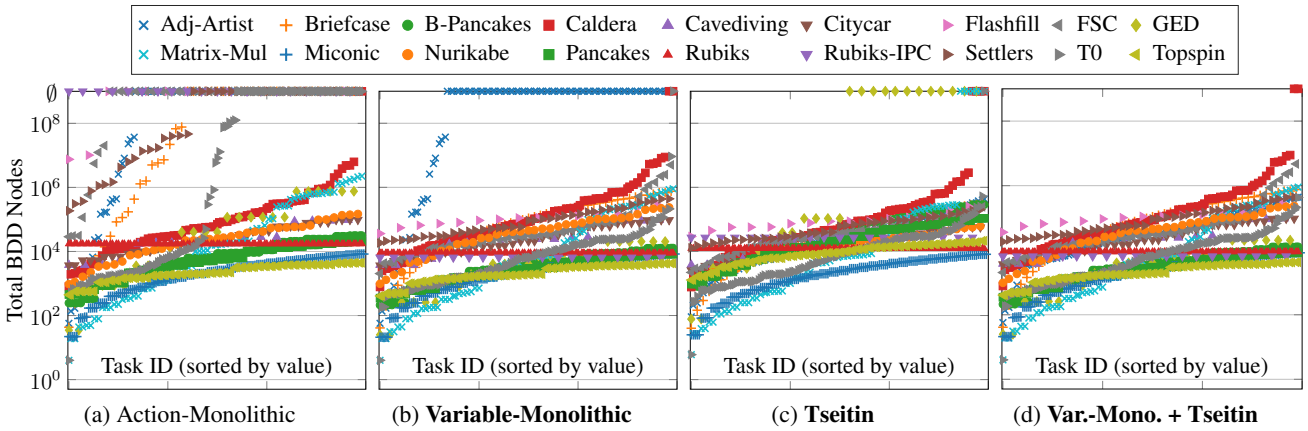
Conjunctive Representations

In symbolic search, the formulas of a planning task must be represented as BDDs. The critical components are the actions (as transition relations) and the goal formula. In traditional symbolic search for planning, each action and the goal is represented by a single, monolithic BDD (e.g., Edelkamp and Helmert 2001; Torralba et al. 2017). This is not to be confused with having a single monolithic transition relation for all actions by disjunctive merging. We refer to this per-action representation as the **action-monolithic** form.

We propose to *conjunctively partition* these formulas: we use a tuple of formulas $\langle \phi_1, \ldots, \phi_n \rangle$ to represent the formula $\phi = \phi_1 \wedge \cdots \wedge \phi_n$. More precisely, the transition relation τ_a of an action a or the goal formula γ are represented as tuples of BDDs, whose conjunction represents the function τ_a or γ.

In the **variable-monolithic** form, we represent the transition relation of each action as a tuple with $1 + |V|$ BDDs, one for the precondition and one for each variable. More precisely, we create a formula that encodes for each variable v the condition under which v is true after applying action a. This representation was originally proposed by Burch, Clarke, and Long (1991) in the context of model checking.

Our final approach to representing formulas of a planning task is what we call the **Tseitin** form. It is based on the Tseitin transformation (Tseitin 1968), which converts a formula into conjunctive normal form in linear time and space by introducing auxiliary variables. In this form, we apply the Tseitin transformation to each transition relation and the goal formula, representing every resulting clause as an individual BDD. Moreover, we impose an order on the clauses,



(a) Action-Monolithic (b) **Variable-Monolithic** (c) **Tseitin** (d) **Var.-Mono. + Tseitin**

Figure 1: Sizes of BDDs required to represent planning tasks with complex effects along the y-axis. Tasks that cannot be represented as BDDs within the resource limits are marked with ∅. Newly introduced representations are highlighted in bold.

induced by the partial order of the subformulas, to sequentially determine the values of the auxiliary variables.

Consider the successor computation in symbolic search. We compute all successor states *succ* of a given set of states *states* for a transition relation τ_a representing action a. In the action-monolithic case, we compute the standard *image* operation on BDDs, $(\exists V\,(\mathit{states} \wedge \tau_a))[V' \rightarrow V]$ (Burch, Clarke, and Long 1991; Torralba et al. 2017). For the variable-monolithic and Tseitin forms, we exploit the partitioned form of the transition relation τ_a. Consider a transition relation $\tau_a = t_1 \wedge t_2 \wedge t_3$. In the variable-monolithic and Tseitin cases we compute $(((\mathit{states} \wedge t_1) \wedge t_2) \wedge t_3)$, while in the action-monolithic case we have the computation $(\mathit{states} \wedge \tau_a) = (\mathit{states} \wedge (t_1 \wedge t_2 \wedge t_3))$. Thus, instead of computing the complete predecessor-successor relation of an action as a monolithic BDD, we reorder the conjunction to place the state BDD *states* first. This allows us to compute only those successors that are relevant with respect to *states*. Note that in Tseitin form, a quasi-equivalent formula is computed, which is equivalent after forgetting auxiliary variables (Tseitin 1968; Kuiter et al. 2022). Similarly, we can exploit the partitioned representation of the goal formula for checking if a goal state is found.

All such representations can be exponentially more efficient than explicit blind search, i.e., Dijkstra's algorithm (Dijkstra 1959). For the Tseitin form, we can derive a performance guarantee relative to its explicit counterpart: in the worst case, it requires only polynomially more time and memory, whereas all previous approaches may incur exponential overhead.

Experiments

We implemented symbolic search with conjunctive partitioning in the SymK planner (Speck, Mattmüller, and Nebel 2020), which is based on Fast Downward 23.06 (Helmert 2006). We additionally consider a variant that first tries for five seconds to construct the formula in variable-monolithic form, and then switches to the Tseitin form.

Fig. 1 shows the cumulative size of BDDs required to represent the initial state, goal, and transition relations for each task of 18 domains with complex conditional effects. We see that the action-monolithic form fails to create these data structures in multiple domains, among them the IPC 2023 Rubik's Cube domain. The variable-monolithic and Tseitin forms both perform significantly better, although each outperforms the other in different domains. Our hybrid variant combines the best of all approaches for these domains.

Explicit		Forward				Bidirectional + M			
Blind	Scorp.	AM	**VM**	**T**	**VM+T**	AM	**VM**	**T**	**VM+T**
379	513	432	448	431	474	557	585	645	**653**

Table 1: Number of tasks solved out of 1227 tasks from 18 domains for explicit and symbolic forward/bidirectional search with or without transition relation merging (M).

Tab. 1 shows the overall coverage of symbolic search compared to explicit blind search and explicit heuristic search in the form of Scorpion (Seipp, Keller, and Helmert 2020; Seipp 2023). In addition to "vanilla" forward symbolic search, we report results for bidirectional search that merges the conjunctively or disjunctively partitioned transition relations (M), up to 100k BDD nodes (Torralba et al. 2017). The new representations perform domainwise favorably compared to the action-monolithic form and can increase the overall number of solved tasks (see Speck and Helmert (2025) for details). By combining the variable-monolithic and Tseitin forms with transition-relation merging and bidirectional search, we introduce a novel symbolic search approach that outperforms the state of the art in both symbolic and explicit search, solving 140 more tasks than Scorpion, the top non-portfolio planner at IPC 2023.

Conclusions

We presented new methods to advance symbolic search for classical planning using conjunctive partitioning. As future work, we aim to conduct similar investigations for symbolic search with heuristics (Edelkamp and Reffel 1998; Fišer, Torralba, and Hoffmann 2024), with the goal of establishing performance guarantees relative to explicit heuristic search.

Acknowledgments

This work was funded by the Swiss National Science Foundation (SNSF) as part of the project "Unifying the Theory and Algorithms of Factored State-Space Search" (UTA).

References

Bäckström, C.; and Nebel, B. 1995. Complexity Results for SAS^+ Planning. *Computational Intelligence*, 11(4): 625–655.

Bryant, R. E. 1986. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, 35(8): 677–691.

Burch, J. R.; Clarke, E. M.; and Long, D. E. 1991. Symbolic Model Checking with Partitioned Transition Relations. In *Proc. VLSI 1991*, 49–58.

Dijkstra, E. W. 1959. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1: 269–271.

Edelkamp, S.; and Helmert, M. 2001. The Model Checking Integrated Planning System (MIPS). *AI Magazine*, 22(3): 67–71.

Edelkamp, S.; and Reffel, F. 1998. OBDDs in Heuristic Search. In *Proc. KI 1998*, 81–92.

Fikes, R. E.; and Nilsson, N. J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *AIJ*, 2: 189–208.

Fišer, D.; Torralba, Á.; and Hoffmann, J. 2024. Boosting optimal symbolic planning: Operator-potential heuristics. *AIJ*, 334: 104174.

Helmert, M. 2006. The Fast Downward Planning System. *JAIR*, 26: 191–246.

Kuiter, E.; Krieter, S.; Sundermann, C.; Thüm, T.; and Saake, G. 2022. Tseitin or not Tseitin? The Impact of CNF Transformations on Feature-Model Analyses. In *Proc. ASE 2022*, 110:1–110:13.

Seipp, J. 2023. Scorpion 2023. In *IPC-10 Planner Abstracts*.

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

Speck, D.; and Helmert, M. 2025. On Performance Guarantees for Symbolic Search in Classical Planning. In *Proc. ECAI 2025*.

Speck, D.; Mattmüller, R.; and Nebel, B. 2020. Symbolic Top-k Planning. In *Proc. AAAI 2020*, 9967–9974.

Speck, D.; Seipp, J.; and Torralba, Á. 2025. Symbolic Search for Cost-Optimal Planning with Expressive Model Extensions. *JAIR*, 82: 1349–1405.

Taitler, A.; Alford, R.; Espasa, J.; Behnke, G.; Fišer, D.; Gimelfarb, M.; Pommerening, F.; Sanner, S.; Scala, E.; Schreiber, D.; Segovia-Aguas, J.; and Seipp, J. 2024. The 2023 International Planning Competition. *AI Magazine*, 45(2): 280–296.

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

Tseitin, G. 1968. On the Complexity of Derivation in the Propositional Calculus. In *Studies in Constructive Mathematics and Mathematical Logic, Part II*, 115–125. Consultants Bureau, New York. English Translation.