Limits and Possibilities of BDDs in State Space Search*

Stefan Edelkamp and Peter Kissmann

Faculty of Computer Science
TU Dortmund, Germany

Abstract. This paper investigates the impact of symbolic search for solving domain-independent action planning problems with binary decision diagrams (BDDs). Polynomial upper and exponential lower bounds on the number of BDD nodes for characteristic benchmark problems are derived and validated. In order to optimize the variable ordering, causal graph dependencies are exploited.

1 Introduction

Optimal planning is a natural requirement for many applications, and posed in form of challenging benchmarks at international planning competitions. Due to the unfolding of the planning graph, parallel-optimal planners [2, 20, 21, 24, 25] appear to be less effective in large search depth, even though more efficient encodings have been found [23].

The state-of-the-art in optimal sequential planning includes heuristic search planning approaches with admissible search heuristics, such as the flexible abstraction heuristic [14] and explicit-state pattern databases [12]. The corresponding planners refer to a fully-instantiated planning problem together with a minimized state encoding [13, 6] and have been reported to compare well with other techniques [26, 11, 16].

Binary decision diagrams (BDDs) contribute to various successful planning approaches, e.g. [4, 18]. The idea of using BDDs is to reduce the memory requirements for planning state sets as problem sizes increase. For several optimal sequential planning benchmark domains, symbolic breadth-first search with BDDs is competitive with the state-of-the-art [5]. Recent experimental findings additionally show large compression ratios for symbolic pattern databases in many planning benchmark domains [1].

This paper studies the causes for good and bad BDD performance by providing lower and upper bounds for BDD growth in characteristic benchmark domains. The approach applies to step-optimal propositional planning and transfers to planning with additive action costs.

The paper is structured as follows. First, we introduce BDD-based search. In the core of the paper, we then study the impact of BDDs for selected characteristic benchmark problems, for which we provide state space encodings, lower and upper bounds. We consider improvements obtained in bidirectional search. Finally, we give some concluding remarks.

* Extended version of a poster at the National Conference of Artificial Intelligence (AAAI-08) having the same title [8]. Thanks to DFG for support in ED 74/3 and 74/2.

2 Symbolic Planning

Throughout the paper, we consider optimal planning using the SAS$^+$ representation of planning tasks [13]. More formally, a SAS$^+$ planning task with costs is defined as $\mathcal{P} = (\mathcal{S}, \mathcal{V}, \mathcal{A}, \mathcal{I}, \mathcal{G}, \mathcal{C})$ with $\mathcal{S}$ being a set of states, $\mathcal{V} = \{v_1, \ldots, v_n\}$ being a set of state variables (each $v \in \mathcal{V}$ has finite domain D_v), $\mathcal{A}$ being a set of actions given by a pair (P, E) of preconditions and effects, $\mathcal{I}$ being the initial state, and $\mathcal{G}$ being the goal state set in form of a partial assignment. Such partial assignment for $\mathcal{V}$ is a function s over $\mathcal{V}$ with $s(v) \in D_v$, if $s(v)$ is defined. Moreover, cost function $\mathcal{C}$ maps $\mathcal{A}$ to $I\!N$. The task is to find a sequence of actions $a_1 \ldots, a_k \in \mathcal{A}$ from $\mathcal{I}$ to $\mathcal{G}$ with minimal $\sum_{i=1}^{k} \mathcal{C}(a_i)$. For step-optimal planning we have $\mathcal{C}(a) = 1$ for all $a \in \mathcal{A}$.

Symbolic planning sometimes refers to analyzing planning graphs [2], or to checking the satisfiability of formulas [20]. In contrast, we refer to the exploration in the context of using BDDs [3]. For constructing the BDDs for the initial and goal states as well as the individual actions, variables are encoded in binary.

Sets of planning states are represented in form of Boolean functions, and actions are formalized as (transition) relations. This allows to compute the successor state set (the image) as a conjunction of the given state set and the transition relation, existentially quantified over the set of the current state variables. This way, all states reached by applying one action to a state in the input set are determined. Iterating the process (starting with the representation of the initial state) yields a symbolic implementation of breadth-first search. Results in international planning competitions show that symbolic bidirectional breadth-first search performs well for many benchmarks[1].

The variable ordering has an exponential impact to the size of the BDDs. In our setting, state variables are encoded in binary and ordered along their causal graph dependencies [13] by greedily minimizing $\sum_{1 \leq i \neq j \leq n, v_i \textit{ depends on } v_j} |i - j|$. Roughly speaking, the more related two variables are, the more likely they will be located next to each other.

In the following we study the impact of BDDs for space improvement in characteristic problems, for which we provide state space encodings, lower and upper bounds.

3 Exponential Lower Bound

Let us consider permutation games on $(0, \ldots, N-1)$, such as the (n^2-1)-Puzzle, where $N = n^2$. It is well known that for the (n^2-1)-Puzzle, half of all $n^2!$ possible permutations are reachable from the initial state. With explicit-state breadth-first search, Korf and Schultze [22] have generated the entire state space of the 15-Puzzle on disk.

The characteristic function f_N of all permutations on $(0, \ldots, N-1)$ has $N\lceil \log N \rceil$ binary state variables and evaluates to *true*, if every block of $\lceil \log N \rceil$ variables corresponds to the binary representation of an integer and every satisfying path of N integers is a permutation. In his master's thesis, Hung [17] has shown the following result.

[1] The fact that BFS performs well wrt. heuristic search is an observation also encountered in explicit-state space search [15]. It is related to the exponential increase in the number of states below the optimum cost for many benchmark domains, even if only small constant errors in the heuristic are assumed.

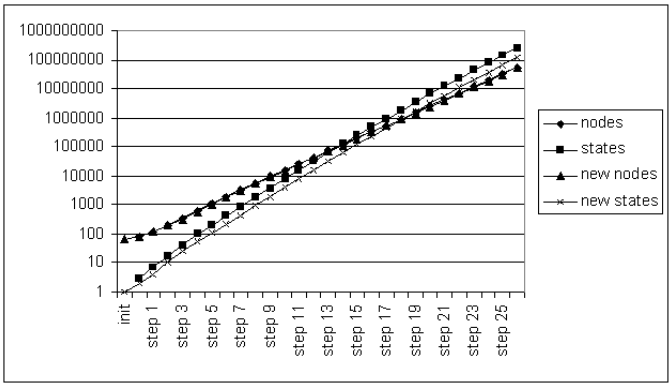

Fig. 1. Symbolic BFS in the 15-Puzzle.

Theorem 1. *The BDD for f_N has more than $\lfloor\sqrt{2^N}\rfloor$ nodes for any variable ordering.*

Given that BDD nodes also consume space, the memory savings of BDDs for permutation games like the (n^2-1)-Puzzle are limited. Fig. 1 validates this hypothesis for the BFS layers of the 15-Puzzle. The growth of BDD nodes is smaller than the growth of states (numbers match the ones by Korf & Schultze), but still the symbolic exploration cannot be finished (within 16 GB RAM). Note that a 15-Puzzle instance can be encoded in 60 bit (4 bits per tile), while a BDD node consumes about 20 bytes. In related research [9] the overall compression ratio for a 6-tile symbolic pattern database of the 35-Puzzle was about factor 2.65.

4 Polynomial Upper Bound

In other search spaces, we obtain an exponential gain using BDDs. In Gripper, there is one robot to transport $2k = n$ balls from one room A to another room B. The robot has two grippers to pick up and put down a ball.

The state space grows exponentially. Since we have $2^n = \sum_{i=0}^{n} \binom{n}{i} \leq n\binom{n}{k}$, the number of all states with k balls in one room is $\binom{n}{k} \geq 2^n/n$. Helmert and Röger [15] have shown that the precise number of all reachable states is $S_n = 2^{n+1} + n2^{n+1} + n(n-1)2^{n-1}$, where $S_n^0 := 2^{n+1}$ corresponds to the number of all states with no ball in a gripper. All states, where the number of balls in each room is even (apart from the two states with all balls in the same room and the robot in the other room), are part of an optimal plan. For larger values of n, therefore, heuristic search planners even with a constant error of only 1 are doomed to failure.

The robot's cycle for the best delivery of two balls has length 6 (picking up the two balls, moving from one room to the other, putting down the two balls, and moving back), such that every sixth BFS layer contains the states on an optimal plan with no ball in a gripper. Yet there are still exponentially many of these states, namely $S_n^0 - 2$.

Theorem 2. *There is a binary state encoding and an associated variable ordering, in which the BDD size for the characteristic function of the states on any optimal path in the breadth-first exploration of Gripper is polynomial in n.*

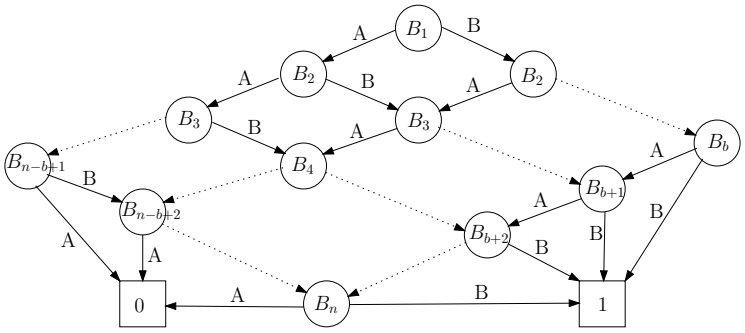

Fig. 2. BDD structure for representing b balls in room B.

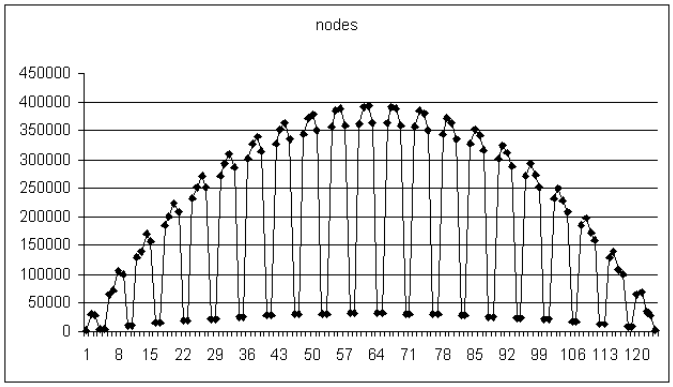

Fig. 3. Sybombolic BFS in Gripper-20.

Proof. To encode states in Gripper[2] , $1+2\cdot\lceil\log(n+1)\rceil+2n$ bits are required: one for the location of the robot (A/B) , $\lceil\log(n+1)\rceil$ for each of the grippers to denote which ball it currently carries, and 2 for the location of each ball (A/B/none) .

According to BFS, we divide the set of states on an optimal path into layers l, $0 \leq l \leq 6k-1$. If both grippers are empty, we are in level $l = 6d$ and all possible states with $b = 2d$ balls in the right room have to be represented, which is available using $\mathcal{O}(bn)$ BDD nodes (see schema in Fig. 2).

The number of choices with 1 or 2 balls in the gripper that are addressed in the $2\lceil\log(n+1)\rceil$ variables is bounded by $\mathcal{O}(n^2)$, such that intermediate layers with $l \neq 6d$ lead to an at most quadratic growth. Hence, each layer restricted to the states on the optimal plan contains less than $\mathcal{O}(n^2 \cdot dn) = \mathcal{O}(dn^3) = \mathcal{O}(n^4)$ BDD nodes in total. Accumulating the numbers along the path, whose size is linear in n, we arrive at a polynomial number of BDD nodes needed for the entire exploration.

The forward BFS exploration in Gripper for $n = 42$ in Fig. 3 validates the theoretical result (including the reduction/increase of BDD nodes every sixth step). Roughly

[2] This encoding is found using a static analyzer.

speaking, *permutation* benchmarks are bad for BDDs, and *counting* benchmarks are good.

5 Symbolic Bidirectional Search

Bidirectional search algorithms are distributed in the sense that two search frontiers are searched concurrently. For explicit-state space search, they solve the one-pair shortest path problem. Since the seed in symbolic search can be any state set and symbolic predecessors are the straight inverse of symbolic successors, symbolic bidirectional search algorithms start immediately from the partial goal assignment. Moreover, bidirectional BFS comes at a low price as it does not rely on any heuristic evaluation function.

In Blocksworld, towers of labeled blocks have to be built. As a full tower can be casted as a permutation, we do not expect polynomially sized BDDs. If we look at benchmark instances (see Fig. 4) bidirectional search improves symbolic BFS from stacking 9 blocks step-optimally to 15 blocks[3].

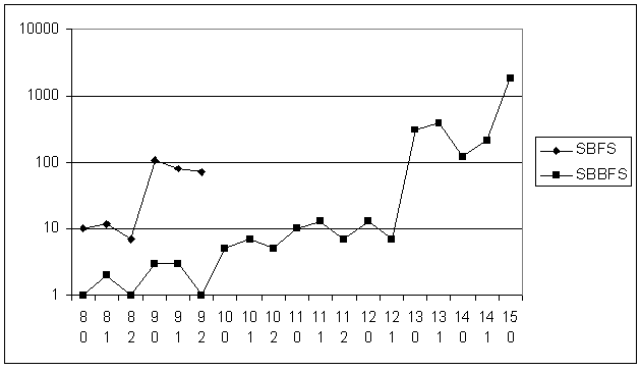

Fig. 4. Time Performance (in seconds) of Symbolic Uni- and Bidirectional BFS in Blocksworld.

Fig. 5 gives insights to the exploration for problem 15-0. Backward search (sorted to the left of the plot to avoid an interleaved order) shows that it contains a large number of illegal states, and forward search (to the right of the plot) shows that the number of states exceeds the number of BDD nodes.

We next look at the Sokoban domain, a problem that is well studied in AI research [19]. A level represents a maze of cells, where stones appear to be randomly placed. The player, also located in one of the cells, pushes the stones around the maze so that, at the end, all stones are on a target location. We observe another exponential gap between explicit-state and symbolic representation.

Theorem 3. *If all* $\binom{n}{k} \cdot (n-k)$ *configurations with* k *balls in a maze of* n *cells in Sokoban are reachable, there is a binary state encoding and an associated variable*

[3] All runtimes wrt. Linux PC with 2.6 GHz CPU

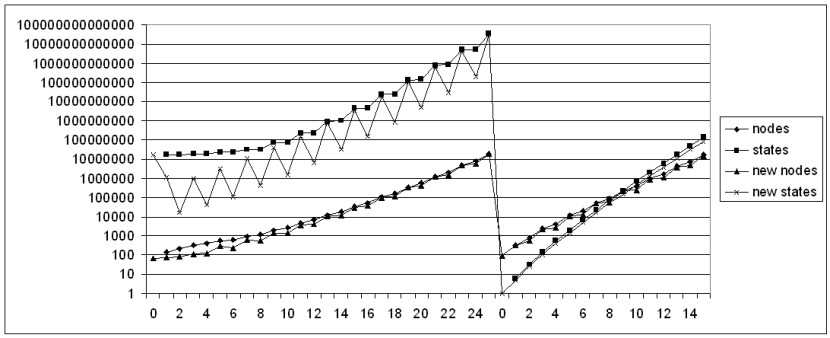

Fig. 5. Symbolic Bidirectional BFS in Blocksworld-15-0.

ordering, in which the BDD size for the characteristic function of all reachable states in Sokoban is polynomial in n.

Proof. To encode states in Sokoban, $2n$ bits are required[4], i.e., 2 bits for each cell (stone/player/none). If we were to omit the player and branch on binary state variables (stone/none), we would observe the same pattern that was shown in Fig. 2, where the left branch would denote an empty cell and the right branch a stone, leaving a BDD of $\mathcal{O}(nk)$ nodes. Integrating the player gives us a second BDD of size $\mathcal{O}(nk)$ with links from the first to the second. Therefore, the complexity for representing all reachable Sokoban positions requires a polynomial number of BDD nodes.

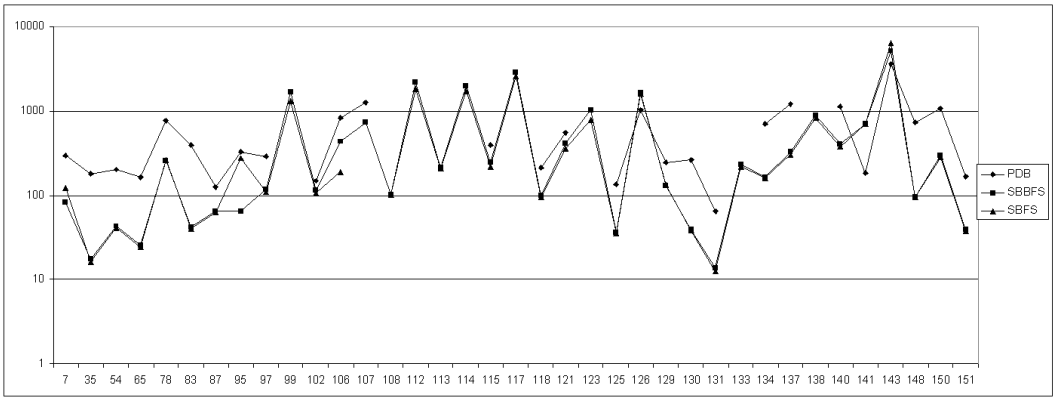

Fig. 6. Results in Sokoban; Unsolved Instances for Symbolic Bidirectional BFS are 93, 105, and 145; Symbolic BFS additionally cannot solve Instance 107.

As $\binom{n}{k} \leq (\frac{n}{k})^k$, the number of all reachable states is clearly exponential. In Fig. 6, we compare exploration results of symbolic uni-/bidirectional BFS (SBFS/SBBFS)

[4] This encoding is found using our static analyzer.

with explicit-state pattern database heuristic search (PDB) [12]. In all but instances 126, 141 and 143, SBBFS is faster. Moreover, it solves 9 of the 12 unsolved instances (out of the 40 problem instances of [12]) in reasonable time. The discrepancy between uni- and bidirectional search is often small; the former is often slightly better in finding long plans (due to saturation of the reachable set) and sometimes worse on small-sized plans (due to non-saturation).

6 Conclusion and Future Work

The advantage of BDD-based compared to SAT-based and constraint-based planning is that the number of variables does not increase with the search depth. Moreover, heuristics are often of limited help in many planning benchmarks.

We gave insights on when and why BDDs work well for compressing the planning state space and its exploration. State spaces that have permutation character are less appropriate than state spaces with selection of elements in a set (balls in a room, locations in a maze). In difference to other posterior state space compressing techniques, see e.g. [10], the compression is on-the-fly, during the exploration of the state space. We obtained promising results for a selection of planning benchmarks, improving the best known results for Sokoban problems.

Optimal planning problems with discrete action costs and soft constraints are the main challenges in the deterministic part of the international planning competition in 2008. The net-benefit to optimize is the total benefit of satisfying the goals, minus the total action cost to achieve them. This results in a plan metric that is a linear expression over indicator variables for the violation of the constraints added to the action cost total. For computing net-benefits with BDDs, we contributed a symbolic breadth-first branch-and-bound search algorithm together with several search refinements [7], to which the above studies apply.

References

1. M. Ball and R. C. Holte. The compression power of symbolic pattern databases. In *ICAPS*, 2008. To appear.
2. A. Blum and M. L. Furst. Fast planning through planning graph analysis. In *IJCAI*, pages 1636–1642, 1995.
3. R. E. Bryant. Symbolic manipulation of boolean functions using a graphical representation. In *DAC*, pages 688–694, 1985.
4. A. Cimatti, M. Roveri, and P. Traverso. Automatic OBDD-based generation of universal plans in non-deterministic domains. In *AAAI*, pages 875–881, 1998.
5. S. Edelkamp. Symbolic shortest path planning. In *ICAPS Workshop on Heuristics for Domain-independent Planning: Progress, Ideas, Limitations, Challenges*, 2007.
6. S. Edelkamp and M. Helmert. Exhibiting knowledge in planning problems to minimize state encoding length. In *ECP*, pages 135–147, 1999.
7. S. Edelkamp and P. Kissmann. GAMER: Bridging planning and general game playing with symbolic search. In *Proceedings of the* 6^{th} *International Planning Competition*, 2008. To Appear.

8. S. Edelkamp and P. Kissmann. Limits and possibilities of BDDs in state space search. In *AAAI*, pages 1452–1453, 2008.
9. S. Edelkamp, P. Kissmann, and S. Jabbar. Scaling search with symbolic pattern databases. In *MOCHART*, 2008. To appear.
10. A. Felner, R. Meshulam, R. C. Holte, and R. E. Korf. Compressing pattern databases. In *AAAI*, pages 638–643, 2004.
11. S. Grandcolas and C. Pain-Barre. Filtering, decomposition and search-space reduction in optimal sequential planning. In *AAAI*, pages 993–998, 2007.
12. P. Haslum, A. Botea, M. Helmert, B. Bonet, and S. Koenig. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *AAAI*, pages 1007–1012, 2007.
13. M. Helmert. A planning heuristic based on causal graph analysis. In *ICAPS*, pages 161–170, 2004.
14. M. Helmert, P. Haslum, and J. Hoffmann. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, pages 176–183, 2007.
15. M. Helmert and G. Röger. How good is almost perfect? In *AAAI*, pages 944–949, 2008.
16. S. L. Hickmott, J. Rintanen, S. Thiébaux, and L. B. White. Planning via petri net unfolding. In *IJCAI*, pages 1904–1911, 2007.
17. N. N. W. Hung. Exploiting symmetry for formal verification. Master's thesis, Faculty of the Graduate School, University of Texas at Austin, 1997.
18. R. Jensen, E. Hansen, S. Richards, and R. Zhou. Memory-efficient symbolic heuristic search. In *ICAPS*, pages 304–313, 2006.
19. A. Junghanns. *Pushing the Limits: New Developments in Single-Agent Search*. PhD thesis, University of Alberta, 1999.
20. H. Kautz and B. Selman. Pushing the envelope: Planning propositional logic, and stochastic search. In *ECAI*, pages 1194–1201, 1996.
21. H. Kautz, B. Selman, and J. Hoffmann. Satplan: Planning as satisfiability. In *Proceedings of the* 5^{th} *International Planning Competition*, 2006.
22. R. E. Korf and T. Schultze. Large-scale parallel breadth-first search. In *AAAI*, pages 1380–1385, 2005.
23. N. Robinson, C. Gretton, D. N. Pham, and A. Sattar. A compact and efficient SAT encoding for planning. In *ICAPS*, 2008. To Appear.
24. V. Vidal and H. Geffner. Branching and pruning: An optimal temporal POCL planner based on constraint programming. *Artificial Intelligence*, 170(3):298–335, 2006.
25. Z. Xing, Y. Chen, and W. Zhang. Maxplan: Optimal planning by decomposed satisfiability and backward reduction. In *Proceedings of the* 5^{th} *International Planning Competition*, 2006.
26. R. Zhou and E. Hansen. Breadth-first heuristic search. In *ICAPS*, pages 92–100, 2004.