Limits and Possibilities of BDDs in State Space Search[*]

Stefan Edelkamp and Peter Kissmann
Faculty of Computer Science
TU Dortmund, Germany
{stefan.edelkamp, peter.kissmann}@cs.uni-dortmund.de

Abstract

The idea of using BDDs for optimal sequential planning is to reduce the memory requirements for the state sets as problem sizes increase. State variables are encoded binary and ordered along their causal graph dependencies. 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, which determines all states reached by applying one action to the states in the input set. Iterating the process (starting with the representation of the initial state) yields a symbolic implementation of breadth-first search.

This paper studies the causes for good and bad BDD performance by providing lower and upper bounds for BDD growth in various domains. Besides general applicability to planning benchmarks, our approach covers different cost models; it applies to step-optimal propositional planning as well as planning with additive action costs.

Exponential Lower Bound

Let us consider permutation games on $(0, \ldots, N-1)$, such as the (n^2-1)-Puzzle, where $N = n^2$. 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. Hung (1997) has shown that the BDD for f_N needs more than $\lfloor \sqrt{2^N} \rfloor$ BDD nodes for any variable ordering. We validated the moderate savings for the BFS layers of the 15-Puzzle experimentally. The growth of BDD nodes is smaller than the growth of states, but still the symbolic exploration cannot be finished (within 16 GB RAM).

Polynomial Upper Bound

In other state 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.

It is not difficult to observe that 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 $\binom{n}{k} \geq 2^n/n$.

[*]Thanks to DFG for support in ED 74/3 and 74/2.

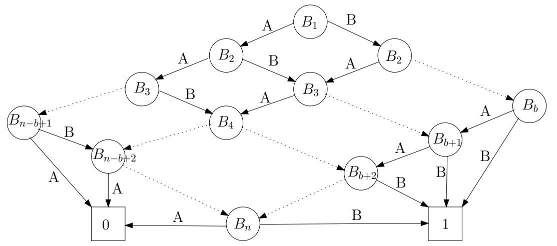

Figure 1: BDD structure for representing b balls in room B.

Helmert and Röger (2007) 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. The basic observation is that all states with an even number of balls in each room (apart from the two states with all balls in the same room and the robot in other one) 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 delivering two balls from one room to the other in any optimal plan has length six (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 1 *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.*

Proof. To encode states in Gripper, $1 + 2 \cdot \lceil \log(n+1) \rceil + 2n$ bits are required: one for the location of the robot, $\lceil \log(n+1) \rceil$ for each of the grippers to denote which ball it currently carries, and 2 for the location of each ball.

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. 1).

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 re-

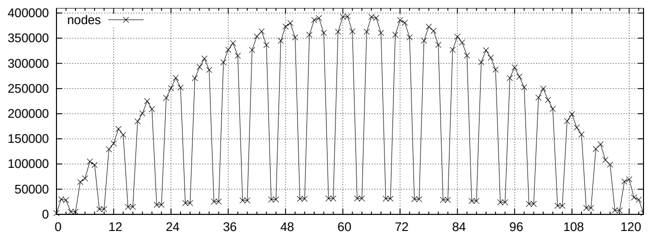

Figure 2: SBFS in Gripper-20.

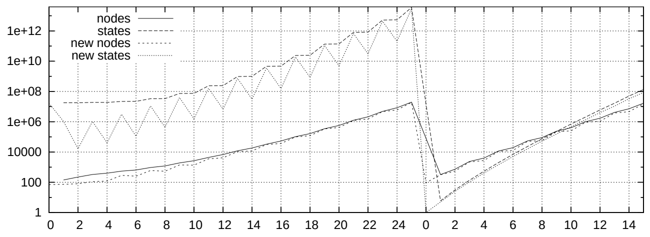

Figure 3: SBBFS in Blocksworld-15-0.

stricted 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 less than $\mathcal{O}(n^5)$ BDD nodes needed for the entire exploration. □

The Symbolic BFS (SBFS) exploration in Gripper for $n = 42$ in Fig. 2 validates the theoretical result (including the reduction/increase of BDD nodes every sixth step).

Roughly speaking, permutation games are bad for BDDs, and counting games are good.

Symbolic Bidirectional Search

Bidirectional search algorithms are distributed in the sense that two search frontiers are searched concurrently. For explicit-state search, they solve the one-pair shortest path problem. Since the seed in symbolic search can be any state set, symbolic bidirectional BFS (SBBFS) algorithms start immediately from the partial goal assignment.

In the Blocksworld planning domain, 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. For given benchmark instances, SBFS can solve problems with no more than 9 blocks step-otimally, whereas SBBFS can solve all up to one of the instances with 15 blocks. Fig. 3 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 shows that the number of states gradually exceeds the number of BDD nodes.

We next look at the Sokoban domain, a problem that is well studied in AI research (Junghanns 1999). 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.

Table 1: Results in Sokoban (runtimes in seconds).

Prob	PDB	SBBFS	SBFS	Prob	PDB	SBBFS	SBFS
7	297.2	82.6	122.5	118	209.3	99.8	94.2
35	177.1	17.31	16.27	121	547.3	414.5	361.5
54	199.5	42.6	41.2	123		1018	783.2
65	164.7	25.4	24.37	125	134.5	36.5	35.2
78	768.2	257.6	259.7	126	1014.2	1636	1611
83	392.8	41.5	40.2	129	242.9	127.7	130.3
87	124	63.93	62.5	130	261.6	38.6	37.9
95	321.6	64.2	277.4	131	65	13.8	12.5
97	286.9	115.5	109.2	133		234.6	218
99		1682	1292	134	695.4	161.2	158.9
102	149	115.2	107.2	137	1208	327.5	301.6
106	823.9	432.5	190.1	138		880	827.8
107	1272	733		140	1116.8	401.8	379
108		101.2	100.4	141	183	702.7	695
112		2169	1813	143	3607.2	5185	6355
113		213.2	206	148	721.9	94.6	93.77
114		1963	1720	150	1059.8	296.7	282
115	394.6	243.1	217.8	151	168.3	39.2	37.2
117		2824	2545				

Theorem 2 *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 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, i.e., 2 bits for each cell (stone/player/none). If we were to omit the player , we would observe the same pattern that was shown in Fig. 1, 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. □

As $\binom{n}{k} \leq (\frac{n}{k})^k$, the number of all reachable states is clearly exponential. In Table 1, we compare exploration results of symbolic uni-/bidirectional BFS with explicit-state pattern database heuristic search by Haslum et al. (2007). 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 Haslum et al.) 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).

References

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *AAAI*, 1007–1012.

Helmert, M., and Röger, G. 2007. How good is almost perfect? In *ICAPS-Workshop on Heuristics for Domain-Independent Planning*.

Hung, N. N. W. 1997. Exploiting symmetry for formal verification. Master's thesis, Faculty of the Graduate School, University of Texas at Austin.

Junghanns, A. 1999. *Pushing the Limits: New Developments in Single-Agent Search*. Ph.D. Dissertation, University of Alberta.