What graphs can be efficiently represented by BDDs ?

C. Dong P. Molitor

Institute of Computer Science
Martin-Luther University of Halle-Wittenberg
Halle(Saale), D-06120, Germany

Abstract

We have carried out experimental research into implicit representation of large graphs using reduced ordered binary decision diagrams (OBDDs). We experimentally show that for graphs from real applications such as graphs representing the networks of the Internet or the World Wide Web and other technical and social networks the sizes of the corresponding OBDDs do not differ much from the number of edges which the graphs contain. It is noteworthy that all of these large graphs are sparse. For randomly generated dense graphs, the gain, i. e., the ratio of the number of graph edges to the OBDD size, increases with the number of vertices and the density of the graphs.

1 Introduction

Reduced ordered binary decision diagram (OBDD) is a canonical method to represent Boolean functions [3]. Because of its compact sizes and the efficient operations on it, OBDDs are used as an efficient data structure intensively in the design, verification, and test of VLSI circuits [10]. Meanwhile, many variants of OBDDs have been proposed for specific applications [4]. Recently, much efforts have been made for finding efficient OBDD based graph algorithms. Usually, adjacency matrices or adjacency lists are used to represent graphs. However, the traditional algorithms which work on adjacency matrices or lists only succeed in reasonable time if the graphs under consideration are not too large. That is the reason that researchers began to think about OBDD based symbolic graph algorithms, e. g., symbolic topological sorting [17], symbolic computation of strongly connected components [6] and so on. Of course, these OBDD based algorithms can be more efficient than the traditional ones only if the implicit representation of graphs usually is much more smaller than the representation by adjacent matrices or lists.

Recently, some theoretical results concerning the size of implicit representations of graphs have been published [12]. For instance, it has been shown that bipartite graphs and undirected graphs with n vertices can be represented by OBDDs of size $(4+o(1))\frac{n^2}{\log_2 n}$ which is (only) about factor $\log_2 n$ better than adjacency matrices. The two upper bounds are tight as corresponding worst case lower bounds are known. The problem with such theoretical bounds is that they needn't be significant for graphs occurring in real applications.[1]

[1] Confer the upper bound of $\frac{2^n}{n} + o(\frac{2^n}{n})$ for the complexity of Boolean functions proven by Lupanov

This paper aims at experimental investigations of the sizes of OBDDs representing (large) graphs. OBDDs and ZDDs (Zero-suppressed BDDs) [11], which are variants of OBDDs, of graphs from real applications and from randomly generated instances were constructed using the BDD-Packages CUDD [14] and BuDDy [8]. It turns out that implicit representation of these graphs are not much smaller than traditional graph representations (see Section 4). However, there seems to be a coherence between the gain of OBDD representations and the density of the graphs (see Section 5). Before going into details we review the basics of OBDDs representing finite directed graphs in Section 2 and 3.

2 Binary Decision Diagrams

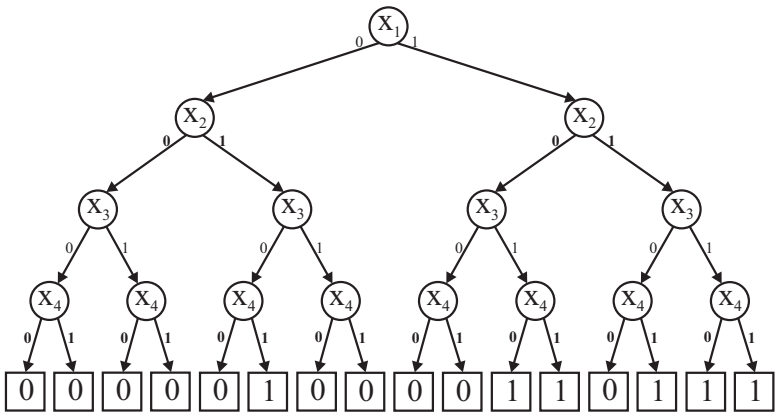


Figure 1. BDT of function g defined by $ON(g) = \{(0,1,0,1),\ (1,0,1,0),\ (1,0,1,1),\ (1,1,0,1),\ (1,1,1,0),\ (1,1,1,1)\}$, i. e., $g(\alpha) = 1 \iff \alpha \in ON(g)$. The rectangular boxes represent the terminal nodes, the circular nodes represent the non-terminal nodes. The edges which point at the low-successors are labeled with 0. The edges which point at the high-successors are labeled with 1.

A *Binary Decision Diagram* (BDD) over a set of Boolean variables $X_n = \{x_1, \ldots, x_n\}$ is a connected, directed acyclic graph $G = (V, E)$ with exactly one root node and the following properties (cf. Fig. 1 and Fig. 2)

- A node in V is either a non-terminal or a terminal node.
- Each non-terminal node v is labeled with a variable from X_n, called the index of v, and has exactly two successors in V, denoted by $low(v)$ and $high(v)$.
- Each terminal node v is labeled with either 0 or 1 and has no successor.

If the BDD has node w as root node, then it represents the Boolean function $f_w : \{0,1\}^n \to \{0,1\}$ defined as follows:

[9] which is tight for almost all Boolean functions [15]. In spite of this result Boolean functions *are* realized by hardware.

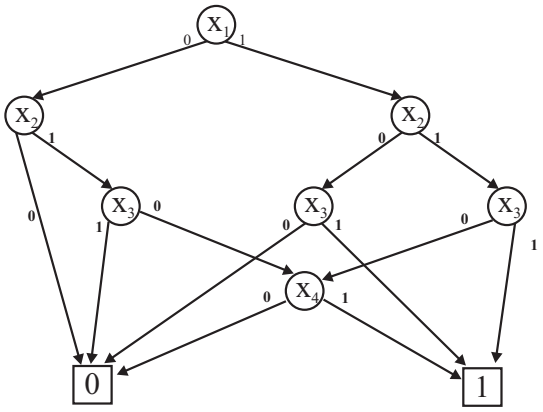


Figure 2. Reduced ordered BDD of g

- If w is a terminal node labeled with 0 (or 1), then f_w represents the constant Boolean function 0 (or 1).
- If w is a non-terminal node labeled with variable x_i, then f_w is the Boolean function defined by

$$f_w(\alpha_1, \ldots, \alpha_n) \quad = \quad \begin{cases} f_{low(w)}(\alpha_1, \ldots, \alpha_n), & \text{if } \alpha_i = 0 \\ f_{high(w)}(\alpha_1, \ldots, \alpha_n), & \text{if } \alpha_i = 1 \end{cases}$$

  for all $(\alpha_1, \ldots, \alpha_n) \in \{0,1\}^n$, i. e., f_w is by the so called Shannon decomposition given by

$$f_w = (x_i \wedge f_{high(w)}) \vee (\overline{x}_i \wedge f_{low(w)}).$$

A BDD G is *free* if each variable is encountered at most once on each path from the root to a terminal node in G. It is *complete* if it is free and each variable is encountered exactly once on each path from the root to a terminal node. A complete BDD which is a tree is called *Binary Decision Tree* (BDT). A BDD is *ordered* if it is free and the variables are encountered in the same order on each path from the root to a terminal node. Ordered BDDs needn't be complete, i. e., paths needn't contain each variable. A BDD is *reduced* if there are no two nodes $v, v' \in V$ with $v \neq v'$ which represent the same Boolean function $f : \{0,1\}^n \to \{0,1\}$, i. e.,

$$\forall v, v' \in V : f_v = f_{v'} \Longrightarrow v = v'.$$

Fig. 1 and Fig. 2 show an ordered BDT and a reduced ordered BDD, respectively, which represent the same Boolean function.

There are two reduction types, deletion reduction and merging reduction:

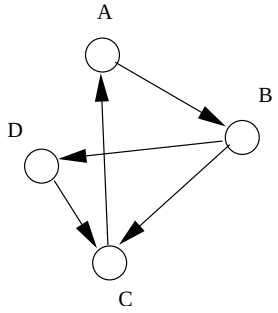
- *Deletion reduction* can be applied if there is a node v in the BDD with $low(v) = high(v) = v'$. In this case, the edges pointing to v can be redirected to node v' and node v can be deleted.

- *Merging reduction* can be applied if there are two nodes v, v' which are identically labeled such that $low(v) = low(v')$ and $high(v) = high(v')$ hold. In this case all edges pointing to v can be redirected to node v' and node v can be deleted.

In the following the abbreviation OBDD denotes *reduced ordered* BDDs. It is noteworthy that reduced ordered BDDs are a canonical representation of Boolean functions, i. e., for a given order of the variables and a Boolean function f there is exactly one reduced ordered BDD which represents f.

More details on BDDs can be found in [1].

3 The symbolic representation of graphs with OBDDs



vertex	encoding
A	00
B	01
C	10
D	11

Figure 3. A small graph $G = (V, E)$ and an encoding $c : V \to \{0, 1\}^2$ of the vertices of G

edges	encoding
(A,B)	00 01
(B,C)	01 10
(C,A)	10 00
(D,C)	11 10
(B,D)	01 11

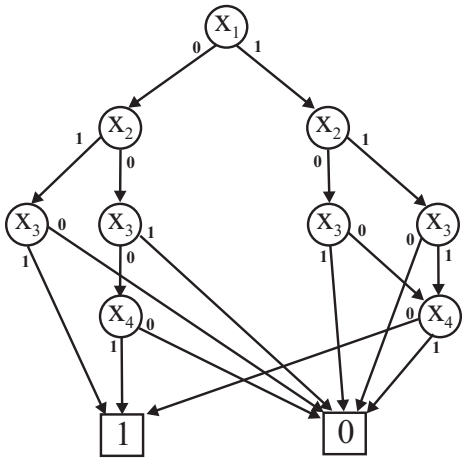


Figure 4. Resulting encoding of the edges and OBDD of the graph w.r.t. variable order $x_1 < x_2 < x_3 < x_4$.

A graph $G = (V, E)$ — V denotes the finite set of vertices; $E \subseteq V \times V$ denotes the finite set of directed edges — can be symbolically represented by BDDs through

Table 1. Graphs from real applications.

Graph	$\lvert V\rvert$	$\lvert E\rvert$	**density**
1	366131	7711904	$5.74 \cdot 10^{-5}$
2	382219	15038083	$1.02 \cdot 10^{-4}$
3	667609	27581275	$6.18 \cdot 10^{-5}$
4	49983	245300	$9.80 \cdot 10^{-5}$
5	307971	831557	$9.09 \cdot 10^{-6}$
6	394510	480259	$3.04 \cdot 10^{-6}$
7	27770	352285	$4.57 \cdot 10^{-4}$
8	124651	193620	$1.24 \cdot 10^{-5}$
9	192244	609066	$1.63 \cdot 10^{-5}$
10	520223	1470404	$5.47 \cdot 10^{-6}$
11	325729	1090108	$1.02 \cdot 10^{-5}$
12	3774768	16518947	$1.16 \cdot 10^{-6}$
13	82670	120399	$1.75 \cdot 10^{-5}$

characteristic functions[2]

$$\chi_{G,c} : \{0,1\}^k \times \{0,1\}^k \to \{0,1\}$$

with $k = \lceil \log_2 n \rceil$. Such a characteristic function $\chi_{G,c}$ does not depend on graph G only but also on the chosen encoding $c : V \to \{0,1\}^k$ of the vertices of G. It is defined by

$$(\forall \alpha,\, \beta \in \{0,1\}^k)\ \ \chi_{G,c}(\alpha,\beta) = 1 \iff (c^{-1}(\alpha), c^{-1}(\beta)) \in E.$$

To demonstrate how graphs are represented by OBDDs, we use the small graph in Fig. 3. This graph has four vertices which are labeled with the symbols A, B, C, and D, respectively. They can be encoded with binary numbers of length 2. Here we use 00, 01, 10 and 11 as the encodings of A, B, C, and D, respectively. With this encoding c of the vertices we obtain a characteristic function

$$\chi_{G,c} : \{0,1\}^4 \to \{0,1\}$$

of the graph which is defined by

$$\chi_{G,c}(\gamma) = \begin{cases} 1, & \text{if } \gamma \in \{(0,0,0,1),(0,1,1,0),(1,0,0,0),(1,1,1,0),(0,1,1,1)\} \\ 0, & \text{otherwise} \end{cases}$$

The corresponding OBDD is shown in Fig. 4. The underlying variable order is given by

$$x_1 < x_2 < x_3 < x_4.$$

[2]We only consider graphs without isolated vertices in this paper.

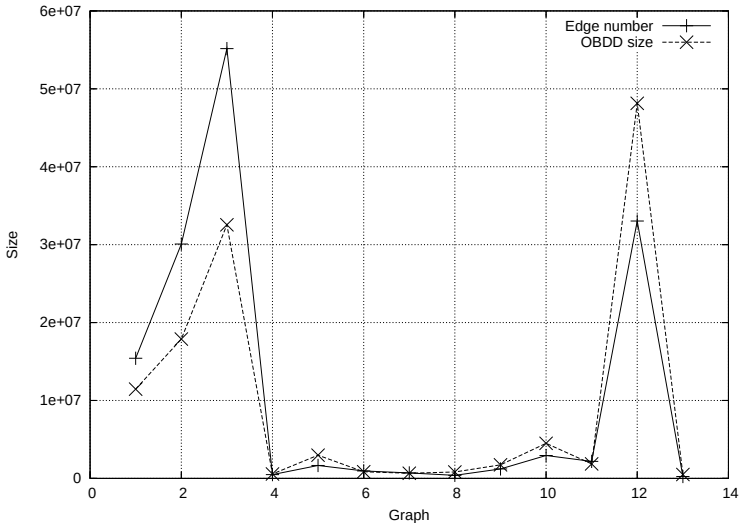


Figure 5. OBDD size vs. edge number of the real graphs

The size of this OBDD, i. e., the number of non-terminal nodes, is 9.

Now let us investigate whether OBDDs of graphs occurring in real applications are much more compact than adjacency matrices or adjacency lists. Only in this case, BDD based symbolic graph algorithms as proposed, e. g., by [6] or [17], can be more efficient than traditional graph algorithms.

4 Graphs from real applications

The graphs which we have considered in our investigations are listed in Table 1. The second and third columns give the number of vertices and the number of edges of the graphs, respectively. Note that all of the graphs listed in Table 1 are sparse, i. e., the number of the edges is much smaller than n^2 which is the number of edges in a complete graph with n vertices (if self-loops are allowed). The density of the graphs which is given by the ratio $\frac{|E|}{|V|^2}$ can be found in the fourth column. The graphs come from different domains such as the networks of the Internet or WWW and other technical and social networks. The largest graph (Graph 12) has more than 3 millions of vertices and about 17 millions of edges.

The OBDDs of these graphs were constructed with the BDD-Package BuDDy [8]. For this experiment, the vertices have been encoded consecutively in the order in which they appeared in the benchmark specification [3]. The sizes of the OBDDs are shown in Fig. 5. For comparison, the numbers of edges[4] of the graphs are also shown in this figure. Though the sizes of the OBDDs are smaller than the numbers of edges for almost all the

[3]Similar results as those presented in the following have been obtained by using other encodings of the vertices, e. g., that presented in [5].

[4]Because all the graphs of Table 1 are undirected ones, every undirected edge was substituted by two directed edges during the construction of their OBDDs. Therefore, the numbers of edges of the graphs in this figure are the doubles of that in Table 1.

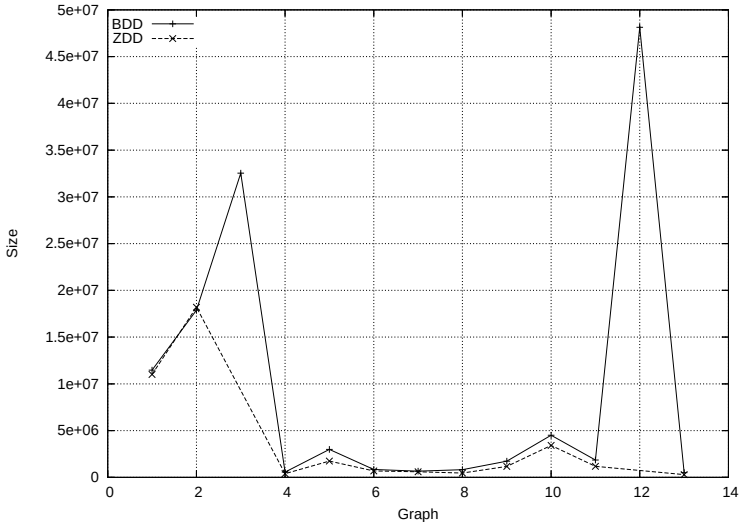


Figure 6. ZDD vs. OBDD sizes of the real graphs

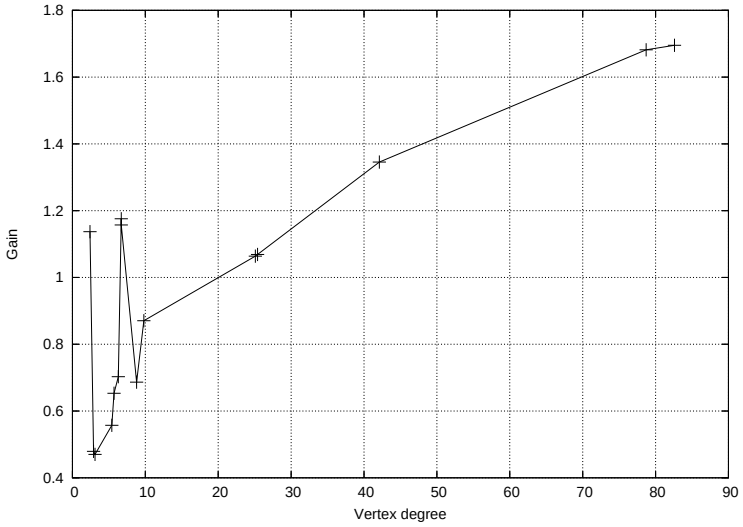


Figure 7. The dependence of the gains on the vertex degree

graphs, the gains we obtain by using OBDDs are relatively small — by "gain" we mean the ratio between the number of edges and the number of non-terminal nodes of the corresponding OBDD. Furthermore, the OBDD of Graph 12 which is the largest graph considered in our investigations has more non-terminal nodes in the OBDD than edges in the graph. The fact that all of the graphs are sparse indicates another possibility to represent them, namely by *Zero-suppressed Binary Decision Diagrams* (ZDD) which are a variant of OBDDs [11]. In ZDDs a node is removed if its *high*-edge goes to the terminal node 0 — however the deletion reduction rule must not be used any more. As there are only relative few paths from the vertex node to terminal node 1 in a OBDD of a *sparse* graph, we can hope that many BDD nodes will be removed by this reduction rule. The results of our experimental run are shown in Fig. 6. Unfortunately, it's not all it's cracked up to be. Though the ZDDs are smaller than the corresponding OBDDs, the gain is bounded by a small factor less than 2.[5]

It may be interesting to look at the gain in dependence of the average vertex degree. The tendency in Fig. 7 is obvious. The gains increase with increasing vertex degrees of the graphs. From this point of view, the situation may be different for dense graphs.

5 Randomly generated (dense) graphs

As we have not found large *dense* graphs used in real applications, we investigated the OBDD sizes of randomly generated dense graphs.

5.1 The results

To investigate the gains behavior of OBDDs, we have generated graphs with up to 10000 vertices and with density up to 0.9 and built the corresponding OBDDs. The results for the graphs with 1000 vertices are shown in Fig. 8. The upper straight line shows the number of edges, the lower curve shows the average OBDD sizes of the corresponding graphs (each taken over 10 samples). These two curves use the left-hand y-axis. The third curve in the middle is the ratio between the two curves, i. e., the average gain, and uses the right-hand y-axis. From Fig. 8 we can make the following two observations:

- Firstly, the gain increases with the density of the graphs. For the very special case of a complete graph, we have obviously the maximum gain, because the corresponding OBDD consists of the terminal node 1, only.
- Secondly, the OBDD curve is symmetric with respect to the density. This is plain because the OBDD of a graph with density d can be obtained by constructing the OBDD of the complement graph which has a density of $1 - d$ and exchanging the two terminal nodes. From the point of view of information theory, a graph with

[5]For fear of a false impression, it must be pointed out that for Graph 2 and Graph 12, which are the ones with the largest numbers of edges, we could not successfully construct the ZDDs due to the size of main memory. The BuDDy package which have used in our first experiment (see Fig. 5) does not provide ZDD operations. That is the reason for which we used the CUDD package in this experiment. For both graphs, namely Graph 2 and Graph 12, the CUDD program needs more than 12 GB main memory during the construction of the ZDDs.

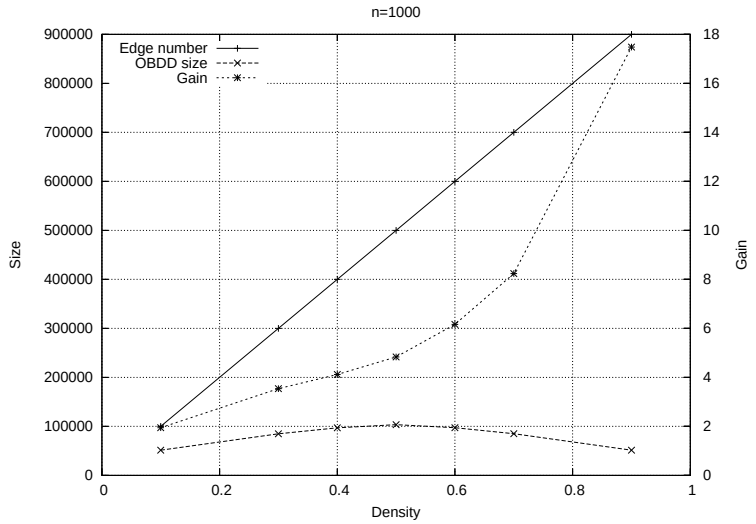


Figure 8. OBDDs of sparse and dense graphs with 1000 vertices.

density d and its complement graph, which has density $1 - d$, have same entropy though the gain for dense graphs is much higher than for sparse graphs.

We obtain similar results for random graphs with a given density and for random graphs with a given number of vertices. Fig. 9 and 10 show how the gain depends on the density if the number of vertices of the graphs is given and how the gain depends on the number of vertices if the density of the graph is given, respectively. It is noteworthy that in Fig. 10 the curve for density 0.9 has a large distance to the other curves. To sum up, the experiments show that for randomly generated graphs the gain seems to increase with density and number of vertices of the graphs.

5.2 Discussions

In this section, we try to theoretically underlay that the gain of using OBDDs instead of adjacency lists for representing graphs depends on density and number of vertices of the graphs. For this, we apply ideas from [2, 7, 16].

Let us assume that an OBDD of a Boolean function defined over s variables is generated by firstly building up the BDT which always contains $2^s - 1$ non-terminal nodes. After the generation of the BDT the reduced ordered BDD is obtained by applying the reduction rules mentioned in Section 2, namely the deletion reduction rule and the merging reduction rule.

An interesting question with respect to the approximation of the size of reduced ordered BDDs is which reduction rule is more important? Bearing this in mind, we computed BDD sizes by starting with the corresponding BDT and applying only one reduction type. The results for graphs with 32 vertices are shown in Fig. 11. It shows that by only applying the merging reduction rule we already obtain a very good approximation of the real OBDD sizes, in contrast to the case in which we only apply the deletion reduction rule. Thus we can use the ordered BDDs which we obtain by only

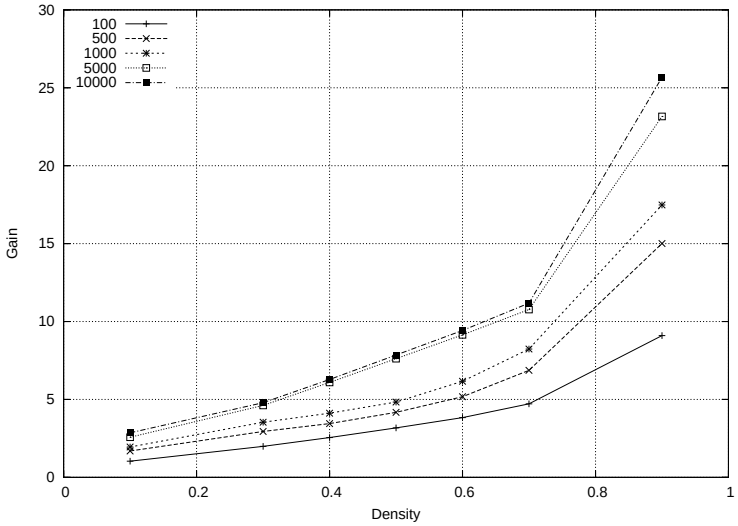


Figure 9. Density vs. gain with respect to a given number of vertices

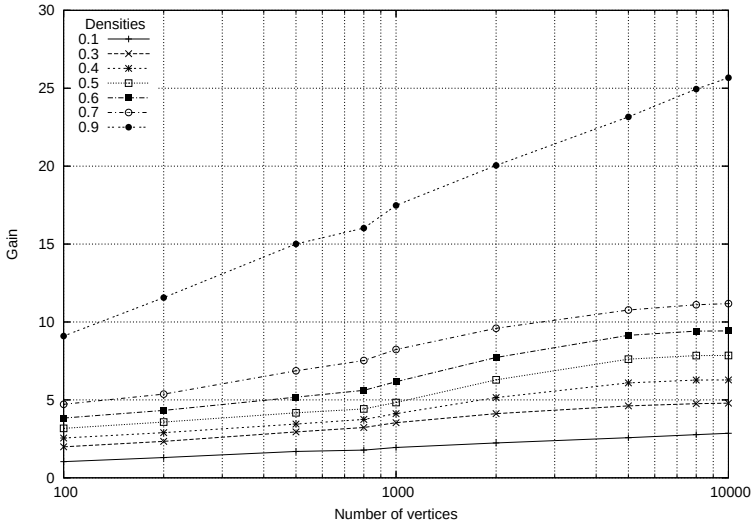


Figure 10. Number of vertices vs. gain with respect to a given density

applying the merging reduction rule for approximating the OBDD size of a graph. It is noteworthy that the distance between the curves of the real and the approximated OBDD sizes is relatively small, and not sensitive to the density of the graph.

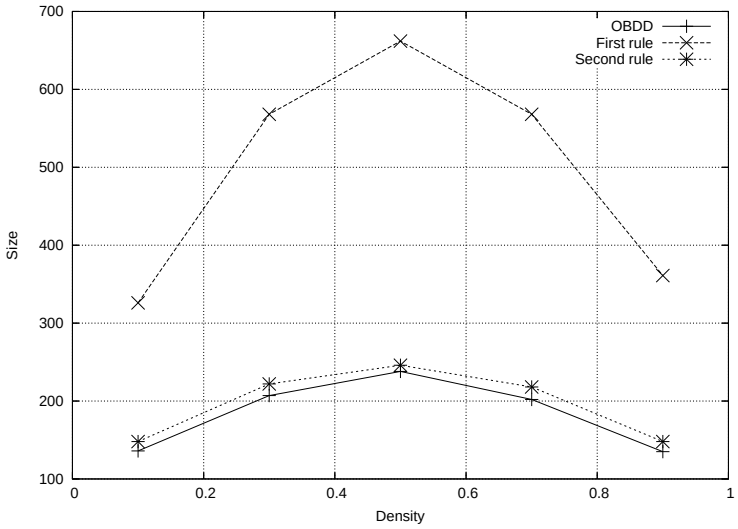


Figure 11. BDD sizes if either the deletion reduction rule, the merging reduction rule, or both reduction rules (curve labeled with "OBDD") are applied to graphs with 32 vertices.

Now let the *redundancy number* of a BDT be the number of nodes which can be removed if only the merging reduction rule is applied to the BDT. Note that, for the computation of the redundancy number of a BDT, we don't care about the deletion reduction rule. Nodes whose *low*-edge and *high*-edge point to the same node aren't removed. The redundancy numbers for graphs with 32 vertices are shown in Fig. 12. The curve of the redundancy numbers refers to the right-hand y-axis. The approximated OBDD size has been defined to be the size of the BDT minus the redundancy number. Our conclusion that the number of nodes which can be removed by applying the deletion reduction rule is small and the increment is more or less unsensitive to the density is further confirmed by graphs with different number of vertices. Fig. 13 shows the results of graphs with 128 vertices. For even larger graphs with more vertices the differences between the real and approximated OBDD sizes are neglectful.

This result justifies the determination of upper bounds on OBDD sizes by only using the merging reduction rule as follows (see [2, 16]): Start with a complete BDT of the Boolean function. If the OBDD is defined over s variables, the BDT will have $2^s - 1$ non-terminal nodes. Then for layer i ($1 \leq i \leq s$, counted from bottom up to the root) the following statements are valid:

- Layer i has 2^{s-i} non-terminal nodes.
- Every node in layer i stands for a function with i variables and therefore can be seen as a $\{0,1\}^{2^i}$-vector. There are 2^{2^i} Boolean functions over i variables.

Layer 1, e. g., contains 2^{s-1} non-terminal nodes. Every node of this level stands for a Boolean vector of length 2. For these 2^{s-1} nodes there are only four different vector-

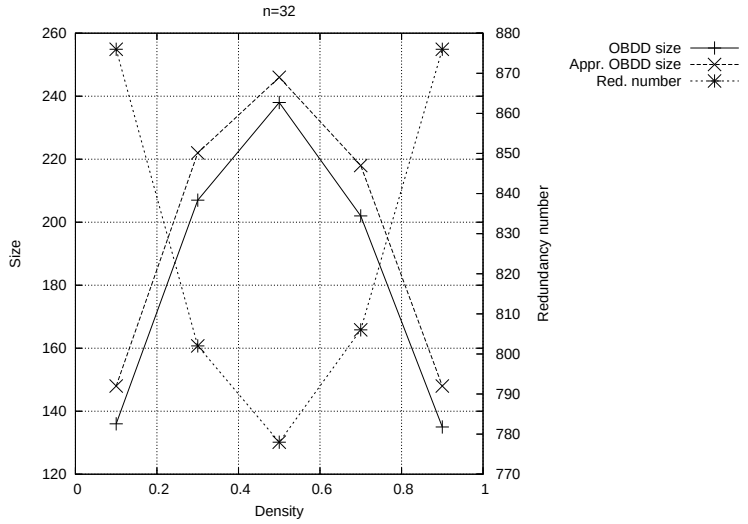


Figure 12. Redundancy numbers and (approximated) sizes of the OBDDs of graphs with 32 vertices

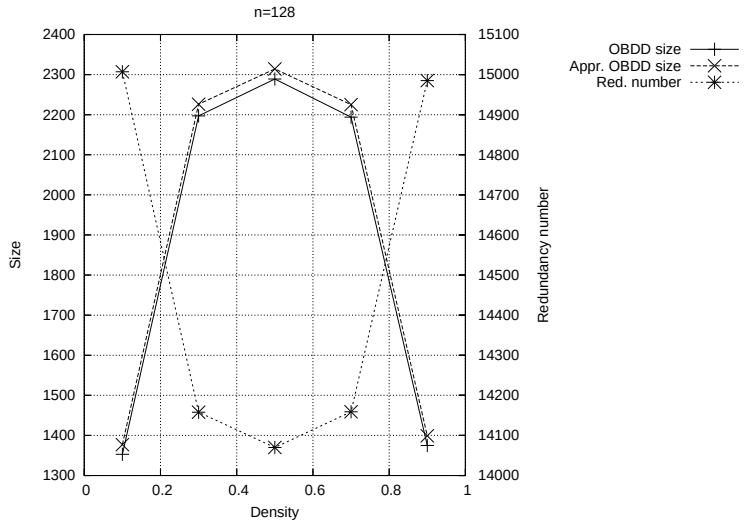


Figure 13. Redundancy numbers and (approximated) sizes of the OBDDs of graphs with 128 vertices

values, namely 00, 01, 10, and 11. Consequently for this layer there will be at most four nodes left in the OBDD.

If the number of non-terminal nodes in layer i of the BDT is smaller than the number of Boolean functions defined over i variables, then every node in this layer could remain in the OBDD. Thus, we need to find the "breakeven" point

$$2^{s-i} = 2^{2^i} \tag{1}$$

or equivalently

$$s - i = 2^i \tag{2}$$

in order to deduce a good upper bound on the size of OBDDs which takes the density of the graphs into consideration.

If i^* meets the conditions $2^{i^*} \leq s - i^*$ and $2^{i^*+1} > s - (i^* + 1)$ then we obtain the upper bound

$$S_{\max} = \underbrace{2^{2^1} + 2^{2^2} + \cdots + 2^{2^{i^*}}}_{\text{nodes below level } i^*} + \underbrace{2^{s-i^*} - 1}_{\text{nodes above level } i^*+1} \tag{3}$$

on OBDD sizes. As $2^{2^{i-t}} \ll 2^{2^i}$ for all $t > 0$ and $2^{i^*} \leq s - i^*$, we have

$$S_{\max} \approx 2^{2^{i^*}} + 2^{s-i^*} \leq 2^{s-i^*+1}. \tag{4}$$

As a graph with $n = 2^{s/2}$ vertices and density d has M edges with

$$M = d \cdot 2^s \tag{5}$$

the gain we obtain by using OBDD representations instead of adjacency lists is

$$\frac{M}{S_{\max}} \approx d \cdot 2^{i^*-1}. \tag{6}$$

Value i^* monotonically increases with the number s of variables which is proportional to the number of vertices in the graph represented by the OBDD. This explains the behavior of the gains for the randomly generated graphs. That is, the gains increase with the number of vertices and the density. Let us estimate the gains more exactly. The gains estimated by Eq. (6) are certainly smaller than the real gains as we didn't considered the deletion reduction rule. For instance, layer p with $p \leq i^*$ doesn't contain 2^{2^p} non-terminal nodes but at most $2^{2^p} - 2^{2^{p-1}}$ non-terminal nodes: For a non-terminal node u in layer p of the BDT, the subfunctions at which the *low*-edge and the *high*-edge point are represented by non-terminal nodes of layer $p - 1$ which consists of at most $2^{2^{p-1}}$ nodes. If both subfunctions coincide with each other, node u will be removed by application of the deletion reduction rule. Thus $2^{2^{p-1}}$ of all the 2^{2^p} functions are Boolean functions whose subfunctions coincide. This allows a more rigorous approximation of

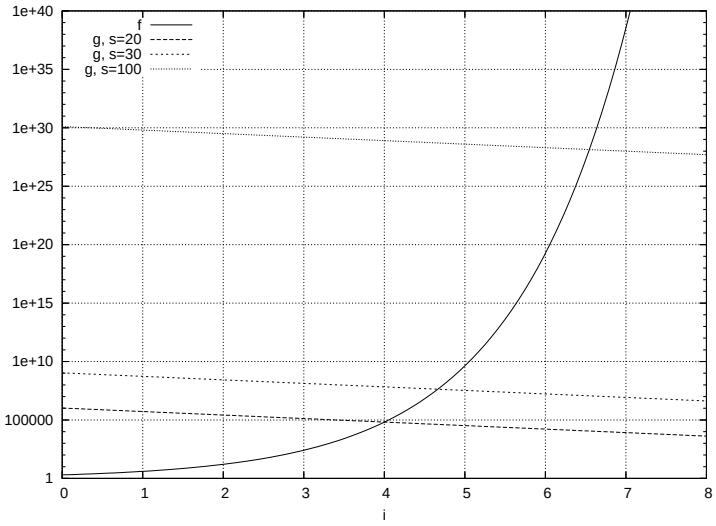


Figure 14. Determination of i^* ($f \equiv 2^{2^i}$, $g \equiv 2^{s-i}$)

$S_{\max}$. Eq. (3) can be substituted by

$$\begin{aligned} S_{\max} &= (2^{2^1} - 2^{2^0}) + (2^{2^2} - 2^{2^1}) + \cdots + (2^{2^{i^*-1}} - 2^{2^{i^*-2}}) + (2^{2^{i^*}} - 2^{2^{i^*-1}}) \\ &\quad +2^{s-i^*} - 1 \qquad (7) \\ &= 2^{2^{i^*}} - 2 + 2^{s-i^*} - 1 \\ &= 2^{2^{i^*}} + 2^{s-i^*} - 3 \\ &\leq 2^{s-i^*} + 2^{s-i^*} - 3 \\ &< 2^{s-i^*+1} \qquad (8) \end{aligned}$$

With this result the "approximate" sign ($\approx$) in Eq. (6) can be substituted by that of "larger than" ($>$):

$$\frac{M}{S_{\max}} > d \cdot 2^{i^*-1}$$

It is noteworthy to mention that the value of i^* increases very slowly with s. This can be seen in Fig. 14. For example, i^* is 4 for $s = 20$. If $s = 100$, i. e., if the number of vertices of the graphs we consider is about 2^{50}, i^* is about 6. Thus, the increase of the gain due to the increase of the number of vertices cannot be fast, as already observed by the experimental run shown in Fig. 10.

6 Conclusions

In this work we investigated the OBDD sizes of large graphs from real applications as well as randomly generated dense instances. For graphs from real applications the corresponding OBDDs mostly have as many nodes as the number of edges in the graphs. This is mainly due to the low density of the graphs. The investigation on the randomly

generated dense graphs shows the dominance of the merging deletion rule during the reduction process of BDDs and gives a good explanation of the results. Thus, research on symbolic BDD based graph algorithms seems to be worthwhile only if real applications deal with large *dense* graphs. The graphs which we encounter in practice, e. g., the graphs representing the network of the Internet, social networks, and railway connections, seem to be rather sparse (as they are more or less planar), however. This challenges the research on symbolic graph algorithms. We have to question "Where are the applications which deal with large dense graphs?"!

Acknowledgement

We would like to thank Prof. D. Wagner and her colleague T. Schank from University of Karlsruhe, Germany, for providing the large graphs from the real applications.

References

[1] R. Drechsler and B. Becker. *Binary Decision Diagrams — Theory and Implementation*. Kluwer Academic Publishers, 1998.

[2] Y. Breitbart, H. Hunt III, and D. Rosenkrantz. On the size of binary decision diagrams representing Boolean functions. *Theoretical Computer Sciences*, 145:45–69, 1995.

[3] R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. on Computers*, C35(8):677–691, 1986.

[4] R. E. Bryant. Symbolic Boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, 24(3):293–318, 1992.

[5] R. Forth and P. Molitor. An efficient heuristic for state encoding minimizing the BDD representations of the transition relations of finite state machines. In *Proc. of the IEEE/ACM Asia and South Pacific Design Automation Conference ASP-DAC*, pages 61–66, 2000.

[6] R. Gentilini, C. Piazza, and A. Policriti. Computing strongly connected components in a linear number of symbolic steps. : In *Proc. of the fourteenth Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 572–582, 2003.

[7] M. Heap and M. Mercer. Least Upper Bounds on OBDD Sizes. *IEEE Trans. on Computers*, C-43(6):764–767, 1994.

[8] J. Lind-Nielsen. *BuDDy: Binary Decision Diagram Package, Release 2.2*, 2002.

[9] Lupanov. A method of circuit synthesis. *Izv. VUZ Radiofiz* 1, pages 120–140, 1958.

[10] C. Meinel and T. Theobald. *Algorithmen und Datenstrukturen im VLSI-Design, OBDD Grundlagen und Anwendungen*. Springer, 1998.

[11] S. I. Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. In *Proc. of the Design Automation Conf.*, pages 272–277, 1993.

[12] R. Nunkesser. Zur impliziten Darstellung von Graphen durch OBDDs. Vortragsfolien, Univ. Dortmund, 2. Februar 2005.

[13] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *IEEE Int'l Conf. on CAD*, pages 42–47, 1993.

[14] F. Somenzi. *CUDD: CU Decision Diagram Package, Release 2.4.0*, 2004.

[15] I. Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner Series in Computer Science, B.G. Teubner Stuttgart, 1987.

[16] I. Wegener. *Branching Programs and Binary Decision Diagrams, Theory and Applications*. SIAM, 2000.

[17] Ph. Woelfel. Symbolic Topological Sorting with OBDDs. In *Proc. of the 28th Int'l Symp. on Mathematical Foundations of Computer Science*, Bratislava, pages 671–680, 2003.