Unranking of Reduced Ordered Binary Decision Diagrams

Julien Clément[*] **Antoine Genitrini**[†]

November 23, 2022

1 Introduction

Three decades ago a data structure emerged under the name of Binary Decision Diagrams (or BDDs) [1]. They deserve to represent Boolean functions, thus are central in computer science. Their algorithm paradigm gives great advantages: it is based on a *divide-and-conquer* approach combined with a compaction process. Their benefits compared to other Boolean representations are so obvious that several dozens of variants BDDs have been developed in recent years. In his monograph [20], Wegener presents several ones like ROBDDs [2], OKFBDDs [6], QOBDDs [19], ZBDDs [15], and others. While most of these data structures are used in the context of verification [20], they also appear, for example, in the context of cryptography [12] or knowledge compilation [5]. Also, the size of the structure, depending on the compaction of a decision tree, allows to improve classification in the context of machine learning [16]. Some specific BDDs are also relevant to strategies for the resolution of combinatorial problems, cf [11, vol. 4], like the classical satisfiability count problem.

The classical way to represent the different diagrams consists in their embedding as directed acyclic graphs (or DAGs). In the following we are interested in the original form of structures that are ROBDDs, for Reduced Ordered Binary Decision Diagrams. One of their fundamental properties relies on the single representative for each Boolean function (of a given number of Boolean variables). In his book [11] Knuth recalls and proves several combinatorial results for ROBDDs. He is, for example, interested in the profile of a typical ROBDD, or in the way to combine two structures to represent a more complex Boolean function. However, thirty years after the takeoff of BDDS, what appears somewhat unbelievable is that the results about the distribution of the Boolean functions according to their ROBDD size are still staggering. Another difficult question is about the distribution of the functions according to their DAG profile. The main problem to get improvements in these directions is that no recursive characterization was expected to describe the structure of ROBDDs, as opposed, for instance, to the recursive decomposition of binary trees which is the core approach in their combinatorial studies (profile, width, depth). Here we have no local-constraint for the decomposition of ROBDDs.

2 Related Work and disruption induced by our approach

An important step in the comprehension of the distribution of the Boolean functions according to their ROBDD size has been achieved by Wegener [19]. He proved the weak Shannon effect: almost all functions have the same ROBDD size up to a factor of $1 + o(1)$ when the number of variables k tends to infinity. Later Gröpl *et al.* [9] improved the result by exhibiting when the strong Shannon effect takes place or not, according to the

[*]Université Caen Normandie, GREYC, CNRS - UMR 6072. Julien.Clement@unicaen.fr

[†]Sorbonne Université, CNRS, LIP6 - UMR 7606. Antoine.Genitrini@lip6.fr

This work was supported by the French ANR Projects C_SyDySi [19-CE48-0007-01], and PING-ACK [18-CE40-0011].

values of k. The strong Shannon effect states that almost all functions have the same ROBDD size as the largest ROBDDs up to a factor of $1 + o(1)$ as k tends to infinity. We observe these facts in Figure 1 by observing the blue curve representing the (probability) distribution of Boolean functions with 12 variables according to their ROBDD size. In fact, as it is generally the case in the context of Boolean functions, asymptotical results can be observed for quite small values of k. A consequence of these first analyses is that picking (uniformly at random) a Boolean function whose ROBDD is small is not an easy task, although in practice ROBDDs are usually not of exponential size.

In [17], the authors study, experimentally, numerically, and theoretically, ROBDD sizes when the number k of variables is increasing. However their main approach relies on an exhaustive enumeration of the decision trees of all Boolean functions, that are in a second step compressed into ROBDDs. The doubly exponential growth of Boolean functions in k variables, equal to 2^{2^k}, gives only access to the first values for $k = 1, \dots, 4$. Then the authors extrapolate the distributions by sampling.

Later Julien Clément and I [4] obtain similar combinatorial results. Thanks to a new approach based on a partial recursive decomposition, we manage to go farther. In fact they partition the ROBDDs according to their profile yielding a much more efficient counting algorithm although using a lot of space due to memoization techniques (storing intermediate results). We obtain exact distributions of the size of ROBDDs up to $k = 9$, thus partitioning the set of 2^{512} Boolean functions into ROBDDs of sizes ranging from 3 to 143.

In this paper, we improve drastically the latter approach so that all the extrapolated results presented in [17] for Boolean functions up to 12 variables are now fully and exactly described. Using a personal computer, in a couple of hours we obtain an exhaustive enumeration of the ROBDDs representing functions up to 12 variables.

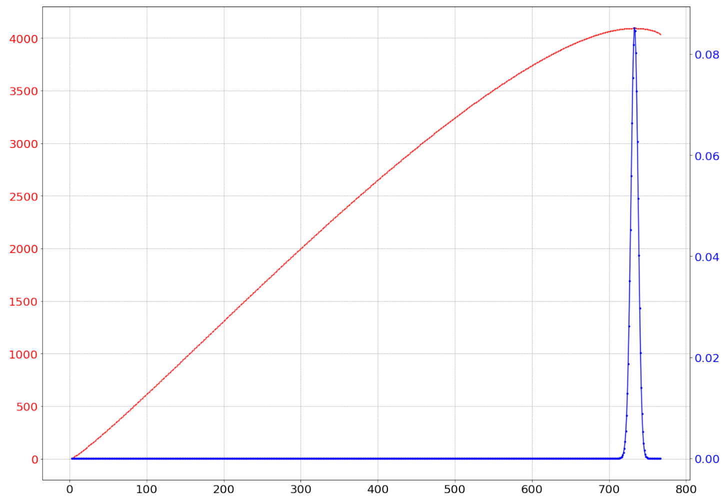

Figure 1: The distribution of functions of 12 variables

Indeed we partition the 2^{4096} Boolean functions according to their ROBDDs size. In Figure 1 the exact distribution is depicted in two ways of presentation: a red point (x, y) states that 2^y functions have a ROBDD size x, in logarithmic scale; the blue curve is the probability distribution.

Observing such curves from $k = 1$ to $k = 12$, we notice the exponential growth of the largest ROBDDs when the number k of variables increases. Indeed we define M_k to be the size of the largest ROBDDs with k variables. The sequence starts as $(M_k)_{k=1,\dots,12} = (3, 5, 7, 11, 19, 31, 47, 79, 143, 271, 511, 767)$.

Since each of these values corresponds to the length of the associated distribution support, it is natural to analyze the complexity of the algorithms computing the distribution of the size of ROBDDs with at most k variables according to M_k. We note that M_k behaves like $2^k/k$ as k tends to infinity. In this context, we obtain that essentially Newton and Verna's approach [17] is of order $\Omega(M_k^{M_k})$, i.e. that their algorithm is of near factorial complexity. Our first algorithmic approach from [4] is essentially of complexity $\Omega(M_k^{3/2 \cdot \log M_k})$. Although we manage to obtain results for bigger k, our first algorithm is not of polynomial complexity in M_k. Here we describe an algorithm that calculates the exact distribution in time complexity $O(M_k^4 \cdot \log M_k)$. To our knowledge, this is the first polynomial complexity algorithm computing the distribution of the ROBDD size of Boolean functions.

Our combinatorial approach, as by-products, gives an exhaustive generation algorithm and thus a uniform random sampler for ROBDDs of a given size. Several practical applications derive from these generation algorithms, in particular in the context of random testing of structural and algorithmic program properties. In [7] the authors execute tests for an algorithm taking as entry a ROBDD. It is based on the famous and widely used QuickCheck software [3]. This later aims at designing tests for program properties by using as an entry a random generator and building test cases for test suites. Using a uniform sampler, the goal is to derive statistical testing: since the underlying distribution of the samples is uniform, it allows to extract statistics thanks to the tests like in [10]. Another application is exhaustive testing for small structures, like the studies in [13, 14], that can be driven with QuickCheck or with other tools like Korat for Java. Finally another direction for testing is to bias the uniform complete distribution in order to focus on special corner cases. We are able to uniformly sample in polynomial time and space ROBDDs of a given size, or a given profile. Especially for small sizes of structures, while it is unlikely to sample them using the global uniform sampler, we aim, *e.g.* at constructing particular ROBDDs for instance sharing the same profile as the one of symmetric functions (known to have small ROBDDs, *cf.* [11])). Since we do not need to compute the full distribution, we can sample such structures for much larger values of k.

3 Sampling algorithms

The ranking/unranking techniques for objects of a combinatorial class $\mathcal{C}$ of size N consists in building a bijection between any $c \in \mathcal{C}$ and an integer (its *rank*) in the interval $[0\,..\,N-1]$. This leads trivially to a uniform sampling algorithm by drawing uniformly first a rank and then building the corresponding object. Our unranking algorithm consists in three phases:

Step 1. Given a rank R, the unranking algorithm will first select the profile (i.e. the node distribution in each level of the final ROBDD). This is done by iteratively computing the number of nodes in each layer, starting from layer k down to layer 1.

Step 2. Once the profile is fixed, the edge structure of the ROBDD is built in a reverse postorder process. The goal is to build the spine[3] and, along the way, to gather information (a relative rank) for the other edges.

Step 3. Finally an inorder traversal of the spine is needed, such that, for every edges that does not belong to the spine, we identify its *absolute* destination, that is a precise node. We use the fact that with an inorder traversal, we are able to identify the set of nodes, called pool, to which such edges can point knowing only a rank within this set.

Theorem 1 (Unranking algorithm for ROBDDs)**.** *The unranking algorithm for a* ROBDD *with profile* $\mathbf{p}$ *of size* n *with at most* k *variables has*

- $O(k^2\ n^5)$ *time complexity and uses* $O(n^2)$ *extra space for identifying the profile;*
- $O(k^2\ n^3)$ *time complexity to generate the* ROBDD *with fixed profile* $\mathbf{p}$*.*

The detailed preprint of these results is presented in `https://arxiv.org/abs/2211.04938`.

References

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

[2] R. E. Bryant. Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[3] K. Claessen and J. Hughes. Quickcheck: A lightweight tool for random testing of haskell programs. *SIGPLAN Not.*, 35(9):268–279, 2000.

[3]The spine of a ROBDD is the spanning tree obtained by a depth-first search of the ROBDD.

[4] J. Clément and A. Genitrini. Binary decision diagrams: From tree compaction to sampling. In *LATIN: Theoretical Informatics - 14th Latin American Symposium, Proceedings*, volume 12118 of *LNCS*, pages 571–583. Springer, 2020.

[5] A. Darwiche and P. Marquis. A knowledge compilation map. *J. Artif. Int. Res.*, 17(1):229–264, sep 2002.

[6] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M. A. Perkowski. Efficient Representation and Manipulation of Switching Functions Based on Ordered Kronecker Functional Decision Diagrams. In *DAC'94*, pages 415–419, 1994.

[7] P. Dybjer, Q. Haiyan, and M. Takeyama. Verifying Haskell Programs by Combining Testing and Proving. In *QSIC'03*, pages 272–279, 2003.

[8] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, New York, NY, USA, 1 edition, 2009.

[9] C. Gröpl, H. J. Prömel, and A. Srivastav. Ordered binary decision diagrams and the shannon effect. *Discret. Appl. Math.*, 142(1-3):67–85, 2004.

[10] R. Grosu and S. A. Smolka. Monte carlo model checking. In *11th International Conference TACAS*, pages 271–286, 2005.

[11] D. E. Knuth. *The Art of Computer Programming, Volume 4A, Combinatorial Algorithms*. Addison-Wesley Professional, 2011.

[12] L. Kruger, S. Jha, E.-J. Goh, and D. Boneh. Secure Function Evaluation with Ordered Binary Decision Diagrams. In *CCS'06*, pages 410–420. ACM, 2006.

[13] D. Marinov, A. Andoni, D. Daniliuc, S. Khurshid, and M. Rinard. An evaluation of exhaustive testing for data structures. Technical report, MIT -LCS-TR-921, 2003.

[14] A. Milicevic, S. Misailovic, D. Marinov, and S. Khurshid. Korat: A tool for generating structurally complex test inputs. In *29th International Conference on Software Engineering (ICSE 2007)*, pages 771–774. IEEE Computer Society, 2007.

[15] S.-I. Minato. Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems. *30th ACM/IEEE Design Automation Conference*, pages 272–277, 1993.

[16] C. Mues, B. Baesens, C. M. Files, and J. Vanthienen. Decision diagrams in machine learning: an empirical study on real-life credit-risk data. *Expert Systems with Applications*, 27(2):257–264, 2004.

[17] J. Newton and D. Verna. A theoretical and numerical analysis of the worst-case size of reduced ordered binary decision diagrams. *ACM TCL*, 20(1):6:1–6:36, 2019.

[18] N. J. A. Sloane. The On-Line Encyclopedia of Integer Sequences, Sep 2019. Sequence A327461.

[19] I. Wegener. The size of reduced OBDDs and optimal read-once branching programs for almost all Boolean functions. In *GTCCS'94*, pages 252–263, 1994.

[20] I. Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, 2000.