The Combinatorics of Binary Decision Diagrams

Antoine Genitrini* Antoine.Genitrini@lip6.fr.

March 14, 2022

Our project consists of an algorithmic and quantitative study of classical data structures from computer science under the prism of combinatorics. In the last decade many improvements have been achieved in order to characterize the associated directed acyclic graphs (or DAGs) combinatorially, paving the way for the analysis of objects induced by compaction procedures. In particular, we will focus on decision diagrams, representing Boolean functions.

A line of studies is based on a constructive way to handle the DAGs. The term constructive here means that no use of the inclusion-exclusion principle is necessary. The first paper by De Felice and Nicaud [FN13] aims at deriving an efficient algorithm to sample objects in the special class of deterministic acyclic automata. Another paper by Genitrini *et al.* [GPV21] presents another constructive enumeration and an effective uniform sampling for DAGs.

Using all these notions together, we are now ready to analyze properties of large random DAGs used at different places in computer science. Our expertise lies in analysis of algorithms using analytic and enumerative combinatorics in order to structurally describe combinatorial objects, to find new specifications and to exhibit universal properties of data structures.

In this PhD project we aim at studying DAGs structures obtained through the compaction of trees using common substructures sharing. We plan to extend the general methodology to classical families of DAGs, that can still be seen as compacted structures. Our main focus is on studying typical structure of DAGs obtained by compaction from a combinatorial point of view. Thus we are interested in enumeration, random sampling and probabilistic analysis with the help of tools from combinatorics and probability theory.

1 Objectives and research hypothesis

We aim at deriving new quantitative results and algorithmic tools in the context of fundamental syntactic data structures in computer science. The focus is on structures that are deeply related to a compaction process. So we are dealing with structures at the frontier between trees and DAGs. To describe our studies we highlight first the data structures and then the methodology.

Our focus lies on the classical data structures of binary decision diagrams.

General approach. Our approach relies on *analytic combinatorics*. Thus the classical round trip between combinatorial models, asymptotic analysis and applications guides our research.

Let us recall Philippe Flajolet's mantra: "*If you can specify it, you can analyze it*". The now classical global approach, underlying Flajolet's idea is the following:

(1) Combinatorial problem
→ (2) Specification of some fundamental combinatorial objects
→ (3) Embedding in the complex plane
→ (4) Analysis of the associated complex functions

*Sorbonne Université, UMR 7606, LIP6, F-75005, Paris, France.

→ (5) Asymptotic behavior of the original objects
→ (6) Resolution of the initial problem.

Whatever the kind of structures under consideration (labeled or not during step (2)), the embedding is possible either in the ordinary generating functions algebra or in the exponential one for the steps (3) and (4).

Static point of view. A first approach was designed by Flajolet and Odlyzko, in particular for the case of binary trees, but it does not directly apply to our DAG structures.

Dynamic point of view. The second idea, that allows us to exhibit such kind of iterated specification relies on an approach, often used in the study of dynamic networks, but that is still uncommon in analytic combinatorics, because the models of structures that encode our problems are fixed and do not evolve. However they can be seen as the result of a dynamical model up to a given time. Thus the dual actions of cuts and grafts (see [BG15]) or the mutations can directly interact in an evolution process leading to a final structure (of a given size for example).

2 Binary decision diagrams

The representation of a Boolean function as a binary decision tree has been used for decades. Its main benefit, compared to other representations like a truth table or a Boolean circuit, comes from the underlying *divide-and-conquer* paradigm. Thirty years ago a new data structure emerged, based on the compaction a of binary decision tree [Bry86]. Its takeoff has been so spectacular that many variants of compacted structures have been developed, named by many different acronyms like ROBDDs [Bry92], OKFBDDs [DST+94], QOBDDs [Weg94], ZBDDs [Min93], etc. While most of these data structures are central in the context of verification [Weg00], they also appear, for example, in the context of cryptography [KJGB06]. Some specific classes are also relevant to strategies for the resolution of combinatorial problems, see e.g. [Knu11, vol. 4], like the classical satisfiability count problem.

More precisely a Boolean function can be represented as a rooted, directed, acyclic graph, which consists of decision nodes and terminal nodes. There are two types of terminal nodes $\top$ and $\bot$ corresponding to truth values of the Boolean domain. Each decision node ν is labeled by a Boolean variable x_ν and has two child nodes (called *low child* and *high child*). The edge from node ν to a low (or high) child represents an assignment of x_ν to FALSE (or TRUE) and is represented as a dotted (or solid) line .

A full binary decision tree is an instance of such a representation where no subtree is shared. A typical operation consists in evaluating a function for a given assignment of the variables. An efficient way regarding time complexity consists in using a full decision binary tree, however this solution is not space efficient. In the left-hand side of Figure 2 we illustrate a decision tree of a Boolean function f on 4 variables. For the assignment $(x_4, x_3, x_2, x_1) =$ (TRUE, FALSE, FALSE, TRUE), we follow the path starting at the root x_4, going to the right using the solid edge (meaning that $x_4 =$ TRUE) then leaving the node x_3 through the dotted edge (meaning $x_3 =$ FALSE) and so on. Thus, for the latter assignment the function reaches the leaf $\bot$, mapped to the value FALSE. Furthermore, the leaf $\top$ is mapped to TRUE. The sequence of labels of the leaves of the decision tree corresponds to the truth table of the function f.

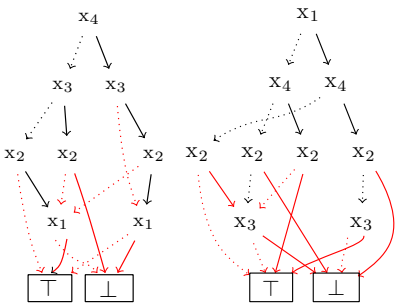

Figure 1: *Two Ordered Reduced Binary Decision Diagrams associated to the same Boolean function*

In contrast to full binary decision trees we may minimize the number of decision nodes by factoring and sharing common substructures according to some rules. Given the binary decision

tree of a k-variable Boolean function, its *reduced ordered binary decision diagram* (or ROBDD) is obtained in linear time with respect to the size of the decision tree. Here, representing occurrences of repeated subtrees only once, pointers will point to representations of shared subtrees, so that the original tree becomes a DAG. The compaction algorithm solves the classical *common subexpression recognition* problem, relying on a postorder traversal of the decision tree.

For instance the decision tree of Figure 2 yields the ROBDD presented on the left-hand side of Figure 1. Note that for a given Boolean function, using two distinct variable orderings can lead to two reduced ordered binary decision diagrams of different sizes (see Figure 1 for such a situation). Nonetheless, an ordering of the variables being fixed, each Boolean function is represented by exactly one single ROBDD obtained through the compaction of its decision tree for this order.

In his book [Knu11] Knuth proves and recalls combinatorial results, like properties for the profile of a ROBDD, or the way to combine two structures to represent a more complex function. However, one notes an unseemly fact: there are no results about the distribution of the Boolean functions according to their ROBDD size. In fact in contrast to (e.g.) binary trees where there is a recursive characterization that allows to well specify the trees, we have no local-constraint here for ROBDDs and thus a similar recurrence is unexpected. Very recently, two papers have been published in this direction [NV19] and [CG20].

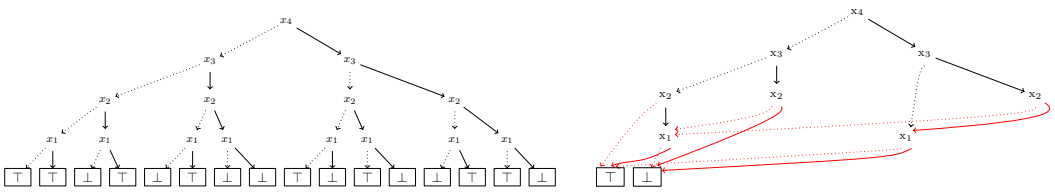

Figure 2: *A decision tree and its compaction*

While the latter two papers rely essentially on an algorithmic approach of the problem, the first question we are interested in is the definition of a generating function approach of the BDDs, the second one is related on a biased generator for BDDs. Note we aim at obtaining results for various families of decision diagrams. The last one is related to some typical properties of BDDs.

Questions:

- In fact, as it has been revealed in [CG20], the complete profile of the DAG representation (i.e. the sequence of the number of nodes at each level of the DAG) is necessary to obtain a specification for the recursive construction of the ROBDD. The generating function approach should rely on multivariate generating functions in k variables, each one attached to a level of the ROBDD, i.e. to a variable x_i of the Boolean function. Once a specification for the BDDs is set, could we obtain some structural information about a typical BDD?

- The approach presented in [CG20] to construct randomly uniform ROBDDs of a given size relies on an exhaustive partition of the Boolean functions in k variables according to a complex equivalence relation. This induces a hard and long precomputation step to obtain then an efficient sampler. Another point of view here would be to propose an algorithmic approach to sample BDDs with a given profile, uniformly at random. The fact that the profile is fixed at the beginning drastically reduces the complexity of the combinatorics relying on the shapes of the BDDs, and then the only remaining combinatorial explosion is due to the possible pointers inside the structure. Thus such an approach could be a possible answer for the effective BDDs sampling.

- Natural and classical questions concern the typical measures induced by the uniform distribution on BDDs (depth and width of the DAG structure for example, but one could also be interested in cryptographic properties of the Boolean functions obtained in this way).

References

[BG15] O. Bodini and A. Genitrini. Cuts in increasing trees. In *2015 Proceedings of the Twelfth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 66–77, 2015.

[Bry86] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Trans. Computers*, 35(8):677–691, 1986.

[Bry92] R. E. Bryant. Symbolic Boolean Manipulation with Ordered Binary-Decision Diagrams. *ACM Comput. Surv.*, 24(3):293–318, 1992.

[CG20] J. Clément and A. Genitrini. Binary decision diagrams: From tree compaction to sampling. In *14th Latin American Symposium, Proceedings*, volume 12118 of *LNCS*, pages 571–583. Springer, 2020.

[DST+94] 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*, pages 415–419, 1994.

[FN13] S. De Felice and C. Nicaud. Random generation of deterministic acyclic automata using the recursive method. In *8th International Computer Science Symposium in Russia, CSR*, volume 7913 of *LNCS*, pages 88–99. Springer, 2013.

[GPV21] A. Genitrini, M. Pépin, and A. Viola. Unlabelled ordered dags and labelled dags: constructive enumeration and uniform random sampling. In *To appear in the proceedings of LAGOS'21*, 2021.

[KJGB06] 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.

[Knu11] D. E. Knuth. *The Art of Computer Programming, Volume 4A, Combinatorial Algorithms.* Addison-Wesley Professional, 2011.

[Min93] S.-I. Minato. Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems. *30th ACM/IEEE Design Automation Conference*, pages 272–277, 1993.

[NV19] 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.

[VB04] J. Vuillemin and Fr. Béal. On the BDD of a Random Boolean Function. In *ASIAN'04*, pages 483–493, 2004.

[Weg94] 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.

[Weg00] I. Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM, 2000.