"Vasile Alecsandri" University of Bacău
Faculty of Sciences
Scientific Studies and Research
Series Mathematics and Informatics
Vol. 23 (2013), No. 1, 29 - 38

NEW ASPECTS IN OPTIMIZATION OF BDDS: MIXED TECHNIQUES BASED ON POPULATIONS OF SOLUTIONS AND LOWER BOUND

IULIAN FURDU AND PETRU GABRIEL PUIU

Abstract. This paper provides a comprehensive introduction on recent advances to reduced ordered binary decision diagrams (ROBDDs or BDDs) as a state-of-the-art data structure in computed-aided design. Key aspects concerning the use of techniques based on lower bounds in the context of BDD optimization are investigated. Three embryonic genetic algorithms for BDD optimization are presented (from which a new one) and their performance compared.

1. INTRODUCTION

Ordered binary decision diagram (OBDD) is the most popular data structure that offers a good trade-off between efficiency of manipulation and compactness of representation of Boolean functions. They are widely used in CAD systems, VLSI, in areas like circuit design for testability, low-power design, field programmable gate arrays etc. The BDD's size is given by the number of its nonterminal nodes and strongly depends on the chosen input variable ordering. For a given boolean function, one input variable ordering may yield an OBDD that is polynomial in the number of variables, while a different ordering may yield an exponential size OBDD.

This paper addresses the problem of optimizing the variable order in BDDs: how to determine a variable ordering for an OBDD representing a given Boolean function f such that the number of nodes in the corresponding OBDD

Keywords and phrases: Circuit optimization, Genetic algorithms, Logic design, Optimization methods, Very-large-scale integration.
(2010) Mathematics Subject Classification: 06E30, 94C10, 68W35.

for f is minimized. The problem to determine the optimal variable ordering is NP-complete [1]. First section presents OBDDs as a state-of-the-art data structure in VLSI design. Next chapters are focused on the main types of methods involved in OBDD optimization.

2. BACKGROUND

In the problem of optimizing OBDDs, a switching function is represented as a BDD, a directed acyclic graph that essentially models how the assignments of truth values to the Boolean input variables are evaluated in a fixed order π, and satisfying a set of properties [5].

Definition. Let π be a total order on the set of variables over $X_n = \{x_1, x_2, ..., x_n\}$. An *ordered binary decision diagram with respect to the variable order* π is a direct acyclic graph with exactly one root which satisfies the following properties:

1. There are exactly two nodes without outgoing edges, labeled by the constants 0 and 1, respectively, called sinks.
2. Each non-sink node is labeled by a variable x_i, and has two outgoing edges which are labeled by 0 and 1, respectively. These edges are called 0-edge (usually figured by dotted arrows) and 1-edge (line arrows), respectively (fig.1) and describe the two subfunctions for f obtained by applying Shannon's expansion in x_i [18].
3. For each edge leading from a node labeled by x_i to a node labeled by x_j it holds that $x_i <_\pi x_j$. i.e. the order in which the variables appear on a path in the graph is consistent with the variable order π.

The variable of a node v is abbreviated by $var(v)$. The computation path of an input $a=a_1, a_2, ..., a_n$ begins in the root, and in each node labeled by x_i, the path follows the edge with label $f(a_i)$.

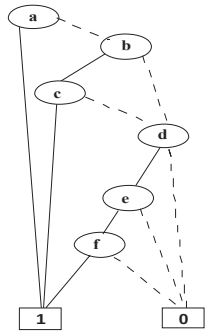

Figure 1. *OBDD for f(a,b,c,d)=a+b·c+d·e·f with order a, b, c, d, e (+ =logical OR, · =logical AND)*

If $v \in \{1,2,...n\}$ and π is an ordering on $\{1,2,...n\}$ then $cost_v(f,\pi)$ denotes the number of nodes labeled by v in $BDD(f,\pi)$ where $BDD(f,\pi)$ is the OBDD diagram of f having the variable ordering π.

Thus, the problem is to find an optimal variable ordering π that minimizes $\sum_{v=1}^{n} \text{cost}_v(f, \pi)$. If a BDD is mapped to a digital circuit, a smaller BDD size is directly transferred to a smaller circuit design (e.g. in Pass Transistor Logic).

The existing methods in finding a good variable ordering can be classified into static techniques and dynamic techniques [18] or, from other perspective in exact methods, heuristic methods and methods based genetic algorithms (GA) as a special type of heuristics. The basic operation in the standard exact minimization algorithms and also in most of the heuristic algorithms for improving variable ordering is the exchange of two adjacent variables, called swap. The most popular algorithms in BDD minimization, based on the swap are sifting algorithm [19] and its variants [2]. Main results concerning the use of lower bounds in heuristic node minimization, exact node minimization and in BDDs optimization methods based on GAs are summarized in next three chapters.

3. LOWER BOUNDS IN HEURISTIC NODE MINIMIZATION

As a BDD-based system may invoke dynamic reordering very often during the progress of BDD construction, time was still an important issue. For this reason in [17] is proposed an algorithm to partition the search space by grouping variables to improve sifting run times. A drawback of the method is that depends strongly on the initial ordering. Using sampling for space partitioning is proposed in [20], but the quality of the results varies widely depending on the choice of the sample.

Promising results have been obtained by using lower bounds to prune the search space of BDD sizes during sifting, called *lb-sifting* and *elb-sifting* [8], [10]. These lower bounds state minimum sizes for certain orderings that will be considered in the following steps of the sifting algorithm.

Further considerations are focused on the method of sifting with lower bounds. The idea is that a variable moving can be stopped if the BDD size stated by the lower bound already exceeds the smallest BDD size recorded so far. An improved lower bound is described in [11] tighter than *lb-sifting* and *elb-sifting* suggested before. The new bound behaves “orthogonally” to the old lower bounds, i.e. they are effective in situations where the old ones are not and vice versa. This leads to a better understanding of the different impact of lower bounds on the efficiency of the sifting algorithm. A combination of old and new lower bounds is introduced to prune the search space in different situations. This yields a final, tight lower bound which then is incorporated into the sifting algorithm.

The goal however is to avoid as many variable swaps as possible. The exact lower bound is weakened to a lower bound that prepares a calculation without any variable movement. This way is avoided too much computational time for bounds when used in a fast algorithm as the sifting.

Reductions in run time of almost 90% have been obtained in experimental evaluation when comparing to classical, unbounded sifting. In comparison to *lb-sifting*, i.e. sifting using the classical lower bound from [8], the improved lower bound still yields further reductions in run time of more than 10% [12]. Also, the full quality of the results was preserved.

4. EXACT NODE MINIMIZATION

The computation of the optimal variable ordering is NP-hard thus a polynomial algorithm cannot be expected. The lemma proved by [15] forms the basis of all minimization algorithms currently implemented in the available OBDD packages and is based on the observation that the size of $level(x_i)$ is independent of the order of the variables bellow or above $level(x_i)$.

Lemma. *Let $I \subseteq \{1, 2, \ldots n\}$, $k = |I|$ and $v \in I$. Then there is a constant c such that for each $\pi \in \Pi(I)$ - set of all orders on I - satisfying $\pi(k) = v$, we have:*

$$cost_v(f, \pi) = c \tag{1}$$

where $\pi(I)$ will denote the order in $\Pi(I)$ which minimizes $\sum_{v=1}^{n} \text{cost}_v(f, \pi)$ and $MinCost_I$ gives the corresponding minimal value of this sum.

More informally, lemma states that the number of nodes in a level k of an OBDD does not change if the variable ordering is changed below or above that level. [16] reformulated the above-mentioned algorithm. As a lower bound of BDD size is used lb= $MinCost_I + c - 1$, where $c = cost_{|I|}(f, \pi_I)$ because, in order to have c nodes of the |I|-th variable, there must be at least $c-1$ nodes of the variables in $N-I$ which will locate in the upper part of the BDD, $N = \{1, 2, \ldots n\}$.

In [7], a tighter lower bound for BDD size has been suggested, which drastically reduces the overall run time. By this, larger functions can be handled, e.g. exact solutions for 32-bit adders have been computed successfully. The lower bound proposed is:

$$lower\_bound = MinCost_I + max\{c + m_R, n - |I|\} + 1 \tag{2}$$

where m_R is the number of output nodes (not leaves) in levels $|I|+1, 2, \ldots n$ and $n-|I|$ is the number of variables in $X_n - I$ and c is the size of a fooling set $f(I, X_n - I)$. The constant node is always needed. The algorithm called FizZ also is the first one that applies a top-down approach.

A recent work [13] introduces an effective extension of the B&B technique that uses more than one lower bound for BDD size in parallel.

The additional lower bounds are obtained by a generalization of a lower bound known from VLSI design.

The generalized bound can be also used for bottom-up construction of a minimized BDD. So the new approach is not restricted to a top-down construction like the approach of [7]. Moreover, combining the two lower bounds yields a new lower bound that is used to exclude states earlier than in previous approaches, resulting in a further speed-up.

Early pruning of search space is achieved [12] by two techniques: one is to avoid transitions to successors that are already known not to contribute to the actualization of the smallest node number. The second is to find a means that allows testing every successor for a possible exclusion right after it was generated: in this way, unnecessary repeated movements to successor states can be avoided. Experimental results proved that the "early pruning" techniques yield a gain of up to 18.8% and a reduction in run time of 10.2%. On average, the reduction in run time is 18.5% (considering BDD reconstruction techniques, not mentioned here, [12]). Consequently, the top-down approach (JANUS↓ algorithm) is faster than FizZ, especially for larger examples achieving a reduction in run time by up to 49%. On average, the reduction in run time is 35.4% [12].

Recently, the change to another programming paradigm, ordered bestfirst search, i.e. the so-called A^*-algorithm as known from AI, has been suggested. This step has been prepared in [9] which suggest an efficient state expansion technique (NEO algorithm).

The A^*-algorithm has superior run time to all previous B&B algorithms. Also, ordered best-first search, (A^*-algorithm), can be combined with a classical B&B algorithm to save memory using a "delay state insertion" technique.

A^* is known to be optimal in the class of heuristic search algorithms that use the same evaluation function and that are guaranteed to find an optimal solution: larger parts of state spaces are pruned by A^* than for any other such algorithm. More exactly, A^* is known to minimize the number of distinct expanded states. The efficient pruning is due to a strategy to always choose the most promising state first for the next state expansion [12].

5. GENETIC ALGORITHMS FOR BDD MINIMIZATION

Genetic algorithms (GAs) for the variable ordering problem rely on a representation of the variable orderings in permutation form (chromosomes). They act on an initial population by applying genetic operators in order to obtain a new population of solutions.

First genetic algorithms in finding the best ordering for OBDDs were due to Drechsler, Becker and Gockel [6]. Their algorithm was adapted and included in

CUDD package (CUDD) as a method of optimization. This algorithm is the first that proves that GA is a feasible and practical alternative to the exact algorithm for variable ordering and is generally used (or should be) as a witness algorithm in benchmarking when dealing with GA for the variable ordering in OBDDs and especially when CUDD is also used. The objective function that measures the fitness of each element is the number of nodes in the corresponding OBDD. Obviously, the corresponding OBDD is generated in concordance to the order expressed by the chromosome. Initial population is generated and optimized by shift [19]. The better half of the population is copied in each iteration without modification. Partially Mapped Crossover (PMX) is applied to a pool of 50% individuals. Newly created individuals are then mutated by applying mutation operators with a given probability. After each iteration, population size is constant (steady state reproduction). Algorithm stops if no improvement is obtained for *50log(best_fitness)* iterations.

The use of hybrid techniques that combine GAs with other optimization method is the actual trend of the research into new methods for BDD optimization [12]. In [4], a branch & bound technique is combined with a basic GA by adopting and operating with embryos as subsets of orders instead of individual complete orders. This hybridization leads to a better balancing between exploration and exploitation of the search space. The *objective function based* fitness is replaced by a *lower bound based* fitness and a growing mechanism (random generational growing, RGG) is applied by means of two new growing operators. The same results were obtained by comparing to [6] in the same conditions (CUDD package, benchmarks from LGSynth91 [22]) for all benchmarks except one; additional more complex benchmarks were tested with very good results. In the same context of embryonic GAs, two new growing strategies were proposed in [3] named random generational growing with sampling (RGGS) and random generational growing with sampling and sifting (RGGSS). Within RGGSS, sifting is integrated as a genetic operator. Next section describes a new algorithm that operates also with embryos instead of full grown chromosomes.

6. UNITARY SYNCHRONOUS GROWING STRATEGY

As mentioned in [3] in the embryonic approach a chromosome is a prefix $x = (x_1, ..., x_k)$ called *embryo* if $1 \le k < n$, or called *adult*, for $k = n$, a fully specified variable order.

Considering embryos does not change the mutation operators but an additional crossover operator is used, a variant of AX in which the left cut point for the bigger embryo is equal to the length of the smaller embryo.

In the embryonic approach, the crossover has a collateral effect especially visible when the majority of the embryos have a small size: it could produce offspring's whose lengths are much higher than the average size in the current population. In general, when the fitness function is a lower bound of the sizes of the BDDs respecting an ordering in the set *S(x)*, the longer embryos usually lose the competition against the shorter ones because the predictive part of the lower bound could be imprecise and far enough from the bounded values, due to the scarcity of the information in the prefix. The best-first approach of [14] deals with this problem by first driving the search to the shorter ones and keeping the longer prefixes in the list of active ("open") search nodes for a later use. The risk is a list of search nodes whose length could exceed the available memory. Since the B&B approach of [13] is breadth-first, it serves the shorter prefixes strictly before the longer ones, but still completeness is ensured since all important longer prefixes are considered at the later stages of the search space traversal. In the combination of B&B and GA, such longer prefixes could become definitively lost by survival selection. The *unitary synchronous growing* described below is a remedy.

Initial population. It contains randomly generated embryos with the same prescribed length λ_0, where $2 < \lambda_0 < \max(5, n/10)$.

Fitness based on lower bound. The fitness of the prefix is $fit_{lb}(x) = \sum_{i=1}^{k} n_i + n - k$, where n_i is number of nodes on the *i*-th level of the OBDD built on an arbitrary ordering in *S(x)*. From [15] it is known that n_i, $i = 1,...,k$ are the same for all possible extensions of the prefix $(x_1,...,x_k)$. The term *n-k* in the definition of $fit_{lb}(x)$ plays the role of the predictive part, while the first part gives the contribution of the prefix itself. When comparing two embryos with equal prefix length, their intrinsic performance is reflected because their predictive parts can not discriminate between them. This suggests the way to tackle the inequity appearing when embryos with different lengths are compared. A variant to manage the growing phase is further discussed.

Unitary synchronous periodic growing. In this variant, named for short USPG, consider the parameter *T* indicating the number of successive evolution stages between two consecutive growths of the length of embryos in the current population by one. So, reaching the final length takes $it_1 = T \cdot (n - \lambda_0)$ iterations. Actually, parameter *T* adjusts the slope of the growing: the dependence of the chromosome length on the number of iterations can be expressed by a (staircase-shaped) step function with the number of iterations on the abscissa and the chromosome length on the ordinate. The ratio $T = it_1/(n - \lambda_0)$ is the step width on the abscissa with a corresponding step

height of one unit, while it_1 is the evolution stage when all chromosomes are adults. High values of T lead to small slopes of the growing. A high slope is not recommended, because this may affect the exploration phase of the GA, shortening it, with no good reason. In contrast, the main motivation for combining GAs with branch & bound exactly is to explore more important parts of the search space by the use of multiple short embryos and by guiding the search with a predictive lower bound.

The length of the individuals during stage t is given by $\lambda(t) = \lambda_0 + \lceil t / T \rceil$. So, at every T stages, each embryo $x = (x_1, ..., x_k)$ is replaced by the best $\max(1, p_b(n-k))$, $p_b \in (0,1]$ among its extensions, each of them extending its parent with exactly one variable that is not contained in the prefix. This growing is then followed by T successive stages. In this way, long chromosomes with more precise but larger score are not compared with short embryos, and consequently are not eliminated.

Performance estimation. Table 1 shows the results for these three strategies. Column *#bestknown* gives the best ever reported [21] number of nodes for each benchmark. The column labeled with #nodes gives best-found results for each strategy and the column labeled #iter gives the corresponding number of iterations. The circuits marked by an asterisk indicate the circuits for which the best-known results were obtained from [6].

Bench	In	Out	#best known	USPG		RGGS		RGGSS	
				#nodes	#iter	#nodes	#iter	#nodes	#iter
*cm85a	11	3	28	28	56	28	42	28	45
*cm163a	16	5	26	26	102	26	40	26	41
*cu	14	11	32	32	66	32	40	32	41
*alu4	14	8	350	350	109	350	89	350	52
*s1494	14	25	369	369	148	369	52	369	64
vda	17	39	478	494	155	478	96	478	196
misex3	14	14	478	478	72	478	84	478	72
*apex2	39	3	-	304	475	308	253	302	573
*apex7	49	37	-	216	437	215	305	214	1207
dalu	75	16	689	705	728	702	357	698	357
cordic	23	2	42	42	123	42	114	42	115
ttt2	24	21	107	107	269	107	71	107	59
apex6	135	99	498	530	1233	561	523	545	1061
i3	132	6	133	133	621	157	460	133	396

Table 1. Best found results for three growing strategies within embryonic GAs [3, 4]

Within RGGSS strategy - as shown in Table 1 - the fitness computation involving sampling followed by sifting offers the best results with the price of a

certain increase of run time complexity (not shown here) but in the case of apex6 benchmark, USPG strategy returned the best result. That proves that the strategy has its own potential, assumption strengthen by the fact that for smaller sized circuits *#bestknown* is obtained for almost all benchmarks.

7. CONCLUSIONS

The paper presents both classical and recent aspects, main methods and algorithms for BDDs optimization. It also proves that GAs are a feasible and practical alternative to the exact algorithms for variable ordering of BDDs by comparing the results of three embryonic GAs developed by authors. A new strategy is described and evaluated. Experimental results prove that by combining GAs with branch & bound, more important parts of the search space are explored since good results were obtained systematically. As further direction of investigation, new growing strategies have to be designed and evaluated, the use of tighter lower bounds, and working with different structures of the chromosomes for a better integration of previous results on lower bounds for BDD size.

REFERENCES

[1] B. Bollig, I. Wegener, **Improving the Variable Ordering of OBDDs Is NP-Complete**, IEEE Transactions on Computers 45 (9) (1996), 993-1002.

[2] K. S. Brace, R. L. Rudell, R. E. Bryant, **Efficient implementation of a BDD package**, Proc. of the 27th ACM/IEEE Design for Automation Conf (1990), 45-49.

[3] O. Brudaru, R. Ebendt, I. Furdu, **Optimizing Variable Ordering of BDDs with Double Hybridized Embryonic Genetic Algorithm**, 12th Int'l Symp. on Symbolic and Numeric Algorithms for Scientific Computing, Timişoara (2010), 167-173.

[4] O. Brudaru, I. Furdu, R. Ebendt, **Embryonic Genetic Algorithm with Random Generational Growing Strategy for Optimizing Variable Ordering of BDDs,** Sci. Stud. Res., Ser. Math. Inform. 20, 1 (2010), 45-60.

[5] R. E. Bryant, C. Meinel, **Ordered Binary Decision Diagrams In Electronic Design Automation: Foundations, Applications and Innovations**, Ed. S. Hassoun and T. Sasao, Kluwer Academic Publishers, 285-307, 2001.

[6] R. Drechsler, B. Becker, N. Gockel, **A genetic algorithm for variable ordering of OBDDs**, Int'l Workshop on Logic Synth. (1995), 55-64.

[7] R. Drechsler, N. Drechsler, W. Günther, **Fast exact minimization of BDDs**, IEEE Trans. on CAD 19 (3) (2000), 384-389.

[8] R. Drechsler, W. Gunther, F. Somenzi, **Using lower bounds during dynamic BDD minimization**, IEEE Trans. on CAD 20 (1) (2001), 51-57.

[9] R. Ebendt, **Reducing the number of variable movements in exact BDD minimization**, Int'l Symp. on Circuits and Systems 5 (2003), 605-608.

[10] R. Ebendt, R. Drechsler, **Lower bounds for dynamic BDD reordering**, Asia and South Pacific Design Automation Conf. (2005), 579-582.
[11] R. Ebendt, R. Drechsler, **The Effect of Improved Lower Bounds in Dynamic BDD Reordering**, IEEE Trans. on CAD of Integrated Circuits and Systems 25, 5 (2006), 902-909.
[12] R. Ebendt, R. G. Fey, R. Drechsler, **Advanced BDD Optimization**, Springer-Verlag, Berlin, 2005.
[13] R. Ebendt, W. Gunther, R. Drechsler, **An improved branch and bound algorithm for exact BDD minimization**, IEEE Trans. on CAD 22 (12) (2003),1657-1663.
[14] R. Ebendt, W. Günther, R. Drechsler, **Combining Ordered Best-First Search with Branch and Bound for Exact BDD Minimization**, IEEE Trans. on CAD of Integrated Circuits and Systems 24 (10) (2005), 1515-1529.
[15] S. J. Friedman, K. Supowitz, **Finding the optimal variable ordering for binary decision diagrams**, IEEE Trans. on Computers 39 (5) (1990), 710-713.
[16] N. Ishiura, H. Sawada, S. Yajima, **Minimization of BDD based on exchange of variables**, IEEE Proc. of the International Conference on Computer Aided Design (1991), 472-475.
[17] C. Meinel, A. Slobodova, **Speeding up variable reordering for OBDDs**, ICCD (1997), 338-343.
[18] C. Meinel, T. Theobald, **Algorithms and Data Structures in VLSI Design**, Springer-Verlag, Berlin, 1998.
[19] R. Rudell, **Dynamic variable ordering for ordered binary decision diagrams**, Int'l Conf. of CAD (1993), 42-47.
[20] A. Slobodova, C. Meinel, **Sample method for minimization of OBDD,** Int'l Workshop on Logic Synth. (1998), 311-316.
[21] CUDD package publicly available at: ftp://vlsi.colorado.edu/pub/ (accesed 07/02/2013)
[22]LGSynth91 benchmarks available from http://www.cbl.ncsu.edu:16080/benchmarks/Benchmarks-upto-1996.html

Iulian Furdu
"Vasile Alecsandri" University of Bacău
Faculty of Sciences
Department of Mathematics, Informatics and Education Sciences
157 Calea Mărăşeşti, Bacău, 600115, ROMANIA
e-mail: ifurdu@ub.ro

Petru Gabriel Puiu
"Vasile Alecsandri" University of Bacău
Faculty of Engineering
Department of Power Engineering, Mechatronics and Computer Science
157 Calea Mărăşeşti, Bacău, 600115, ROMANIA
e-mail: ppgabriel@ub.ro