Sample Method for Minimization of OBDDs

Anna Slobodová* Christoph Meinel*†

Abstract

This paper contributes to the solution of the minimization problem of Ordered Binary Decision Diagrams by means of variable reordering. We suggest a new heuristic that is based on sampling. A small OBDD sample is chosen from the OBDDs that are considered for minimization. Solving the problem for this small sample, we obtain a variable order that is adapt for the entire OBDDs. We present first experimental results with the Sample Reordering targeted at combinatorial verification. The suggested heuristic is substantially faster than Sifting.

1 Introduction

The problem of highest priority for all practical applications that use Ordered Binary Decision Diagrams (OBDDs, for definitions see [Bry86]) is their conciseness. Since the size of the OBDD representation for a function may vary exponentially for different orders of variables, the problem of finding an optimal variable ordering that realizes the minimal size is of special interest. This problem is known to be NP-complete [THY93, BW96]. The exact algorithm works well for small numbers of variables only, and cannot be used for a general purpose. More relevant to the practice are variable reordering heuristics (e.g., [Rud93, PSP94, PS95]).

The variable reordering problem of OBDDs is a typical combinatorial problem with a huge search space of feasible solutions. In addition, there is no known efficient method for exact evaluation of a variable ordering with respect to the size of corresponding OBDDs. More precisely, having an OBDD for a function f and a variable ordering π, we have no efficient procedure to compute the size estimation of πOBDD(f). The only way is to construct the OBDD, which can be performed efficiently merely in the case when the size of the resulting OBDD is polynomially related to the size of the initial OBDD [MS94]. This complicates the problem even more, since heuristics that choose several candidates for a "good" ordering cannot avoid the construction of the OBDDs for their evaluation. This is the reason why methods like simulated annealing [BLW95] or genetic reordering algorithm [DBG96] take too much time.

When a problem appears unsolvable in its full dimension, a natural approach is to reduce it to problems of lower dimensions. This idea can be found in several reordering heuristics: e.g., Sifting [Rud93] looks for a good position for one variable, thus obtaining a feasible solution that improves the size of the OBDD. This step is then repeated for all variables. Another example is Block-restricted Sifting [MS97] where OBDDs are partitioned horizontally into blocks that are minimized independently.

In this paper, we follow the idea mentioned above combining it with the idea of sampling: A part of the considered OBDDs is taken as a representative sample and the variable ordering problem

This work was partially supported by the German research society (DFG) via the project Me 1077/12-1

*Institute of Telematics, Bahnhofstr. 30-32, 54292 Trier, Germany, e-mail: slobodova/meinel@ti.fhg.de

†University of Trier, FB IV - Informatik, 54286 Trier, Germany. e-mail: meinel@uni-trier.de

is solved for it. This subproblem has a substantially lower dimension, making it possible to find quickly a feasible solution that is then used as an approximation of a "good" variable ordering for the entire OBDDs. The goal is to obtain not the best, but an acceptable order in a short time. Our experimental evaluation shows that the suggested reordering method is substantially faster than Sifting, and there is no clear advantage of any of both methods with respect to the final size of the OBDDs.

There are several problems that appear in the realization of this idea. They are discussed in the next section. Section 3 contains experiments with the method used dynamically during symbolic simulation of benchmark circuits. Evaluation of the method is performed by comparison with the most stable and widely used Sifting Algorithm as proposed by Rudell [Rud93].

2 Sampling Method in Variable Reordering

In this section, we describe our application of the sampling to variable reordering of OBDDs. The main idea is to find a good variable ordering for a small sample of given OBDDs and to adapt it for the entire OBDDs. There are three questions to be discussed:

1. How to find the sample?
2. How to minimize it?
3. How to adapt the variable ordering of a minimized sample to the entire OBDDs?

Each of these points may have an essential influence on the final solution. From our experience, simple methods are often more effective in practical applications than more sophisticated and complex ones. Hence, we will start with the simplest variant and then discuss possible improvements.

A more appropriate phrasing of the first question would be: "How to find a *good* sample?", where *good* means that any variable ordering that is optimal for the sample is efficiently transformable into an ordering that is optimal for the entire OBDDs. Obviously, because of the NP-hardness of the optimal variable ordering problem, there is no hope to find an efficient algorithm for finding a small (e.g., less than half the size of the considered OBDDs) sample with the desired property mentioned above. Moreover, the same holds if we look for a small sample whose reduction yields an ordering that is efficiently transformable at least to a better ordering for the entire OBDDs. An interesting question is: How to find a sample that assures in most cases an ordering that is better than the initial ordering of the OBDDs? In the very first step, we choose a sample in a random manner, in order to see whether there is any potential in the sampling idea. On one hand, this makes the method independent from an application, which can be seen as a positive property. On the other hand, this approach clearly does not exploit the whole potential of the method that could make use of application specific information. In Section 3, we describe the choice of the sample suitable for an application during symbolic simulation of a circuit.

Another important question is the choice of the size of the sample. A small sample can be reordered fast, but it gives less information about a good variable ordering for the entire OBDDs than a bigger one. The overhead for copying and reordering of the sample must be in balance with the quality of the ordering found, i.e., the smaller the sample, the easier the handling, but there will also be less information about the OBDDs. An appropriate value of the sample size parameter can be derived experimentally.

Besides the size of the sample, it is also important what portion of the variables from the support of the entire OBDDs it contains. We do not explicitly require a fixed portion of variables, but the choice of the sample is aimed to cover most of the variables.

The second question concerning the minimization of the sample partially depends on the OBDD package used and the optimization technique for the minimization of the sample. Considering the implementation details of the CUDD package that we use for our experiments, we chose the following approach: The sample is copied and reordered by Sifting.

The last, but nonetheless important question to be discussed is how to adapt the variable order achieved by minimization of the sample to entire OBDDs. This question actually consists of two parts: how to derive the new variable order for the entire OBDDs from the obtained variable order in the sample, and how to rebuild the OBDDs with respect to this new order. The trivial solution for the new order is to fix the positions of the variables that do not appear in the sample and reorder the rest of variables according to their positions in the reordered sample. There may be also some restrictions about the positions of variables that are implied by the meaning of the functions represented by the OBDDs, e.g., that the present and next state variables of a finite state machine should stay together. These must be met in the new ordering, too.

In the very first experiments, we have tried to sift the variables that do not appear in the sample, but only in a restricted manner. Each such variable was sifted between two closest positions occupied by the variables that appeared in the sample. This additional reordering did not yield an essential reduction of the size, and we omitted it from the next experiments.

We can also try to estimate the quality of a new ordering and to avoid rebuilding to poor orderings. Our conservative estimation is based on the following assumption: If the reordering of a sample did not bring substantial reduction, we do not expect that the order obtained substantially reduces the entire OBDDs.

For reordering to a new order, we use a shuffling procedure that sifts the variables upwards to their new positions, starting with the variable that is positioned on the topmost level in the new order, and proceeding subsequently to lower positions. We have observed that if the size starts to grow, the target ordering is usually not good. Hence, the rebuilding process is stopped whenever the OBDD size increases beyond a given factor. The rebuilding approach used has an advantage in that we get information about the OBDD size for some other orderings that appear on the way to the target one. If one of these intermediate orderings happens to be better than the found one, we sift the corresponding variables back to their best found position. If the first attempt fails, we decide whether we try to reorder again, based on another sample.

3 Experiments

The method described in the previous section has been implemented in the Colorado University Decision Diagrams package (CUDD-2.1.2) and used as dynamic reordering method for symbolic simulation of the LGSynth91 benchmark circuits and some circuits contained in the CUDD package. The parameters in CUDD were left at their default values in all experiments. The results are compared to Sifting Algorithm with respect to final size and time. All experiments ran on Pentium Pro 200's with 64MB memory. In this section, we describe some of the experiments and, based on their evaluation, we propose an appropriate parameter setting. The section is closed by the particular application of the sample reordering method to combinatorial verification.

A chosen sample is copied and then reordered by means of Sifting as implemented in CUDD. The variables in the copy are created in the order of their appearance in the copy process. Since we keep the correspondence of the variables in the sample and in the entire OBDDs, we can easily determine a new variable ordering for the OBDDs from the ordering in the sample using the strategy described in the previous section. Afterwards, the copy can be discarded.

In order to see whether the idea of sampling works for reordering of OBDDs, we ran several experiments. The measures of interest are the time and OBDD size. The method is compared to Sifting (as implemented in CUDD-2.1.2) with respect to these values. Time is considered as being more important, as long as the size remains acceptable. After initial experiments with one-time reordering, we continued the experiments with dynamic reordering during combinatorial simulation of the circuits. To give an impression of the usability of the method, we present the results for dynamic reordering applied on a sample of larger circuits. Since the number of reorderings done depends on the quality of the variable order found by the heuristic, this type of experiments gives more information than a single application of the method.

The first series of experiments was aimed to help in the choice of the sample parameters: sample size α, the growth factor that determines the size bound for shuffling to the new ordering β, and the number of reordering attempts allowed for one dynamic invocation of the reordering γ. We have observed that 10% to 20% size growth during the shuffling to a new variable order, and 2 to 3 attempts per invocation of the reordering are satisfying parameter values. Then we held these two parameters constant and ran the experiments with varying sample size. The method was always used for dynamic reordering and for a final reordering of the OBDDs created by the symbolic simulation of circuits. The variable occurence in circuit description implied the initial order in all experiments. The results clearly showed that while a sample of 10% did not yield enough information, the overhead for a 60% sample was too large. There was a clear time saving for sample size of 15% to 30%. Besides that, smaller final size of OBDDs reordered by Sample Reordering was frequent, too.

Sample Reordering Strategy for Combinatorial Verification

In a particular application of the Sampling, we can use additional information for the choice of a sample. We propose a simple reordering strategy with an appropriate parameter setting, targeted to symbolic simulation of circuits:

A small sample is chosen from the newly created roots and copied to be reordered by Sifting (note that any reordering method can be used at this place). The ordering obtained is applied to the entire OBDDs as described above. If this first attempt fails to reach an acceptable improvement, the same process is repeated for a new sample.

Let us go into details. The new roots obtained as results from the Boolean operations applied during the symbolic simulation, i.e., OBDDs of some internal gates, are pushed onto a stack. Any garbage collection of the unreferenced nodes is completed by cleaning the stack. The size of the stack is bounded. Its capacity can be set according to the considered application and examples (in the presented experiments, we worked with a stack size of 500). The push operation into a full stack discards the bottom item. When the sample reordering is invoked, the sample is preferably built from the roots in the stack. If the OBDDs whose roots are in the stack do not suffice to cover the requirements on the size of the sample, we choose additional roots randomly. Then we proceed as described above: The sample is reordered using Sifting and the resulting order is used for reordering of the entire OBDDs. If the gain of this reordering amounts to at least 30% of the initial size, we stop the reordering and continue in the application. Otherwise, i.e., if the reduction does not reach the expected value, we try to choose another sample.

The suggested Sampling strategy is evaluated in three series of experiments. The results are summarized in Table 1. Parameter β is set to 1.2, i.e., the allowed size growth during the shuffling of variables to a new order is 20% like the default size growth during the Sifting. The size of the sample is set to 25% in all experiments. The number of attempts is at most 2. In order to avoid using the same sample repeatedly, which may happen if we work with a constant sample size parameter, half of the sample is chosen at random in the second attempt. The first experiment (the column labelled by $2 \times 25\%$) ran as described above. The second experiment (the column labelled by $25\% + 25\%$) differs from the first in that, in the second attempt, we do a conservative pre-estimation of the reduction reachable by reordering to the new order obtained by the reordering of the sample. If the reduction of the sample does not reach the expected value of 30% , then the entire OBDDs are not reordered to the new ordering. This decreases the number of shuffle attempts and leads to a further decrease of time. In all but the third experiment, the same method is used for dynamic and final reordering.

If the resulting OBDDs are further processed, e.g., in the case of the sequential circuits, if the application continues by reachability analysis, then it makes sense to spend more time by reordering at the end of symbolic simulation. The third experiment differs from the second in that Sifting is used as the final reordering.

The number of reorderings during the symbolic simulation of a circuit varied from 4 to 22 (10 in average) in our experiments.

The values of time and size for the Sample Reordering are the average values from 10 runs. In comparison with Sifting, we have a remarkable saving of time without incurring a penalty with respect to the total value of the final OBDD size.

References

[BLW95] B. Bollig, M. Löbbig, and I. Wegener. Simulated Annealing to Improve Variable Orderings for OBDDs. *Proc. IWLS*, 5b:5.1–5.10, 1995.

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

[BW96] B. Bollig and I. Wegener. Improving the Variable Ordering of OBDDs is NP–complete. *IEEE Transactions on Computers*, (45(9)):993–1002, 1996.

[DBG96] R. Drechsler, B. Becker, and N. Göckel. A Genetic Algorithm for Variable Ordering of OBDDs. *IEEE Proc. Computers and Digital Techniques*, 143(6):364–368, 1996.

[MS94] Ch. Meinel and A. Slobodová. On the Complexity of Constructing Optimal Ordered Binary Decision Diagrams. *Proc. MFCS*, (LNCS 841):515–525, 1994.

[MS97] Ch. Meinel and A. Slobodová. Speeding up Variable Reordering of OBDDs. *IEEE Proc. of ICCD*, 338–343, 1997.

[PS95] S. Panda and F. Somenzi. Who are the Variables in Your Neighbourhood. *Proc. ACM/IEEE ICCAD*, 74–77, 1995.

[PSP94] S. Panda, F. Somenzi, and B.F. Plessier. Symmetry Detection and Dynamic Variable Ordering of Decision Diagrams. *Proc. ICCAD*, 628–631, 1994.

[Rud93] R. Rudell. Dynamic Variable Ordering for Ordered Binary Decision Diagrams. *Proc. ACM/IEEE ICCAD*, 42–47, 1993.

[THY93] S. Tani, K. Hamaguchi, and S. Yajima. The Complexity of the Optimal Variable Ordering Problem of Shared Binary Decision Diagrams. *Proc. ISAAC*, (LNCS 762):389–398, 1993.

Circuit		Sampling 2×25%	Sampling 25%+25%	Sampling 25%+25% final Sifting	Sifting
bw11x11	time	721.04	**308.66**	388.81	**1033.86**
	size	150,842	182,269	**136,543**	**285,137**
bw8x8	time	5.01	**4.22**	4.64	**6.11**
	size	9,641	9,719	**8,190**	**9,050**
C499	time	**12.87**	17.62	25.51	**20.14**
	size	**32,911**	44,238	41,900	**26,624**
C880	time	7.01	**4.99**	6.90	**11.24**
	size	13,495	18,665	**10,920**	**10,440**
C1355	time	23.76	**21.32**	24.17	**76.01**
	size	**27,063**	29,681	29,192	**29,562**
C3540	time	91.82	**41.87**	49.57	**46.74**
	size	34,286	34,060	**31,858**	**23,950**
C7552	time	91.31	**47.74**	52.92	**30.99**
	size	69,452	28,440	**15,683**	**8,241**
i10	time	26.43	**21.62**	35.39	**174.83**
	size	33,351	34,154	**32,605**	**67,971**
mm30a	time	18.02	**16.42**	18.79	**137.34**
	size	21,548	18,433	**17,659**	**100,591**
s13207.1	time	19.94	**15.21**	16.52	**42.02**
	size	5,003	6,514	**3,158**	**3,008**
s15850.1	time	**67.36**	68.94	77.19	**75.66**
	size	27,409	32,241	**19,812**	**12,539**
s35932	time	37.97	**34.27**	42.12	**50.84**
	size	5,866	5,842	**4,987**	**5,010**
s38584.1	time	59.38	**54.81**	63.37	**121.54**
	size	28,344	30,860	**16,680**	**15,121**
s4863	time	93.54	**83.53**	131.97	**254.03**
	size	80,691	82,612	**69,476**	**64,245**
s6669	time	52.31	48.74	**48.46**	**111.29**
	size	25,626	27,299	**22,351**	**22,109**
Total	time	1,328	**790**	986	**2,193**
	size	565,528	585,027	**461,014**	**683,598**

Table 1.