BAYES-DIRICHLET BDD AS A PROBABILISTIC MODEL FOR LOGIC FUNCTION AND EVOLUTIONARY CIRCUIT DECOMPOSER

Josef Schwarz
Jiří Očenášek

Brno University of Technology
Faculty of Information Technology
Department of Computer Systems
CZ - 61266 Brno, Božetěchova 2

Tel.: 420 5 41141210, fax: 41141270, e-mail: schwarz@dcse.fee.vutbr.cz,
Tel.: 420 5 41141206, fax: 41141270, e-mail: ocenasek@dcse.fee.vutbr.cz

Abstract: *This paper deals with the utilizing of Binary Decision Diagrams built on the Bayes-Dirichlet metric for representation of logic functions. In the first phase the Binary Decision Tree is built followed by a reduction process resulted in binary decision diagram (BDD). The BDDs are also used as a probabilistic model for advanced Bayesian decomposer for digital circuit partitioning. It is shown that this approach is more efficient than the utilizing of Bayesian networks (BN) and in addition the concept of BDDs parallelization is simple to implement.*

Key Words:*. Logic function, binary decision diagrams, Bayes-Dirichlet metric, digital circuit partitioning.*

1 Introduction

Many "decision" procedures use a branching process which consists of testing some property with the branch depending on the test outcome (typical examples are identification of objects or classification). Concrete application include also simulation or modeling of digital circuits [1]. This branching process can be represented usually by graph structure called decision diagram (DD). A decision diagram is a directed acyclic graph in which each decision node is labelled by a variable/attribute tested in this node (control variable/attribute). The edges coming out from the decision node leading to the nodes in subsequent levels correspond to the values of the control variable. The number of the edges relates to the values of the control variable. Besides decision nodes there are leaves (terminal nodes) labelled by the value of the function that is being evaluated by the diagram. Important parameter of a given DD is the size of the DD (the number of decision nodes). The construction of minimum-sized DDs belongs to NP-hard problems. The survey of optimization algorithm for binary DD (BDD) based on top-down as well as bottom-up techniques is published in [2] and [3]. Let it be noted that DD can be seen also as a reduction of decision tree built by the known Shannon's expansion. The key problem is the finding a proper ordering of the DD variables. In this paper we applied a new approach for DD construction based on the Bayes- Dirichlet metric.

2 Bayes- Dirichlet metric for BDD building

This technique is based on the construction of binary decision tree (BDT) using Bayes-Dirichlet metric[4] followed by a fast heuristic for reducing of BDT to BDD. We specify its main feature on the case of classification and prediction. BDT is expressed by:

- a leaf (terminal) node - indicates the value of the target attribute (class) of examples
- a decision node (split) - specifies a test to be carried out on a single attribute-value, with one branch and sub-tree for each possible outcome of the test.

A binary decision tree can be used to classify an example by starting at the root of the tree and moving through it towards a leaf node, which provides the classification of the instance. Most algorithms that have been developed for learning decision trees employ a top-down, greedy search through the space of possible decision trees. Our algorithm searches through the attributes of the training instances and extracts the attribute that best separates the given examples. If the attribute perfectly classifies the training sets then it stops, otherwise it recursively operates on the two partitioned subsets to get their "best" attribute. The algorithm uses a greedy search, that is, it picks the best attribute and never looks back to reconsider earlier choices.

The central focus of the binary decision tree growing algorithm is selecting which attribute to test at each decision node in the tree. From the Bayes-Dirichlet metrics (BD) we derived the incremental equation for adding one new binary split. This measure is used to select among the candidate attributes at each step while growing the tree:

$$Gain(X_i, X_j) = \frac{\Gamma\left(\sum_{r\in\{0,1\}}\sum_{s\in\{0,1\}}(m_{r,s}+1)\right) \cdot \prod_{r\in\{0,1\}}\prod_{s\in\{0,1\}}\Gamma\left(m_{r,s}+1\right)}{\left(\prod_{s\in\{0,1\}}\Gamma\left(\sum_{r\in\{0,1\}}(m_{r,s}+1)\right)\right) \cdot \left(\prod_{r\in\{0,1\}}\Gamma\left(\sum_{s\in\{0,1\}}(m_{r,s}+1)\right)\right)}, \quad (1)$$

where X_i is the target attribute, X_j is the investigated split attribute, and $m_{r,s}$ is the number of individuals having $X_j=r$ and $X_i=s$. For practical purposes we use logarithm of this metric, which avoids multiplication operations. The splitting is performed recursively, so $m_{r,s}$ is determined only from the subpopulation being split. To avoid over-fitting the data, the gain of each split operation can be penalized by the model complexity (e.g. the depth of the tree).

	$\mathbf{X_i = 0}$	$\mathbf{X_i = 1}$
$\mathbf{X_j = 0}$	$m_{0,0}$	$m_{0,1}$
$\mathbf{X_j = 1}$	$m_{1,0}$	$m_{1,1}$

Table 1. Notation of symbols $m_{r,s}$

Note that we also deal with continuous and categorial (integer) variables. If the investigated split attribute X_j is continuous, we use fast heuristics to find the optimal split point E_j. The training cases are split according to condition "$X_j<E_j$" into two parts, where the value of target attribute is more determined.

2.1 BDD for representation of Boolean function

Binary Decision Diagrams are commonly used for representation of Boolean functions because of their efficiency in terms of time and space. Besides Boolean function, BDDs can be also used for representation of other types of discrete functions, such as multi-valued functions, cube sets and arithmetic formulas [2]. BDD is a rooted acyclic graph G=(V,E) with node set V containing nonterminal/decision and terminal nodes. A decision node x represents the Shannon's expansion of the Boolean function:

$$f = (x_i f_1 + \bar{x}_i f_0) \quad (2)$$

where i is the index of the decision node, f_0 and f_1 are the functions of the nodes pointed to by *0*- and *1*- edges, respectively. The terminal nodes represent the logic values *(1/0)*.
The previous approaches suffer from the necessity of a specification of the node ordering. It is often provided by genetic algorithms [3]. In another approaches the ordering is found dynamically during the process of BDD building.

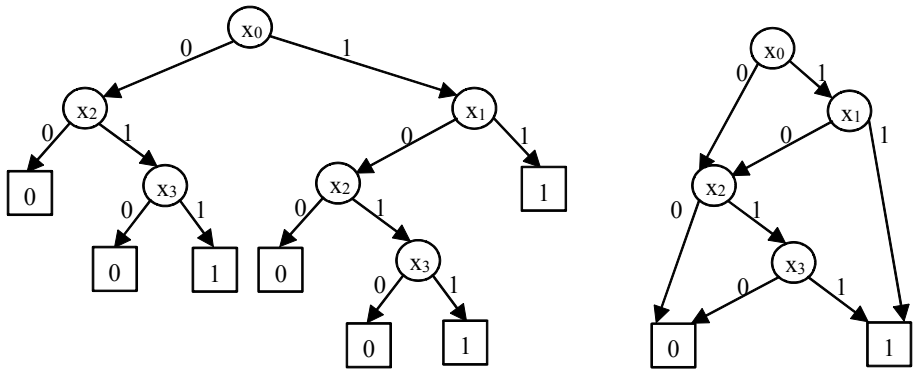

Fig. 1 Representation of multiplexor function $F=x_1x_2+x_3x_4$ by a) BDT b) BDD/OBDD

As mentioned above our algorithm constructs decision tree (BDT) from a set of training cases. The logic function truth table is used as the training set – the input variables are candidates for splits/decision nodes and the function value is the target attribute. Then, BDD is constructed from BDT using two basic reduction rules: 1) reduction of nodes with unique ancestor nodes and 2) sharing all equivalent sub-graphs. In Fig. 1 an example of BDT and the final OBDD for multiplexor function is shown. Our approach for building the binary decision tree from truth table is illustrated in the Fig.2 for logic function $F = x_1x_2+x_3$.

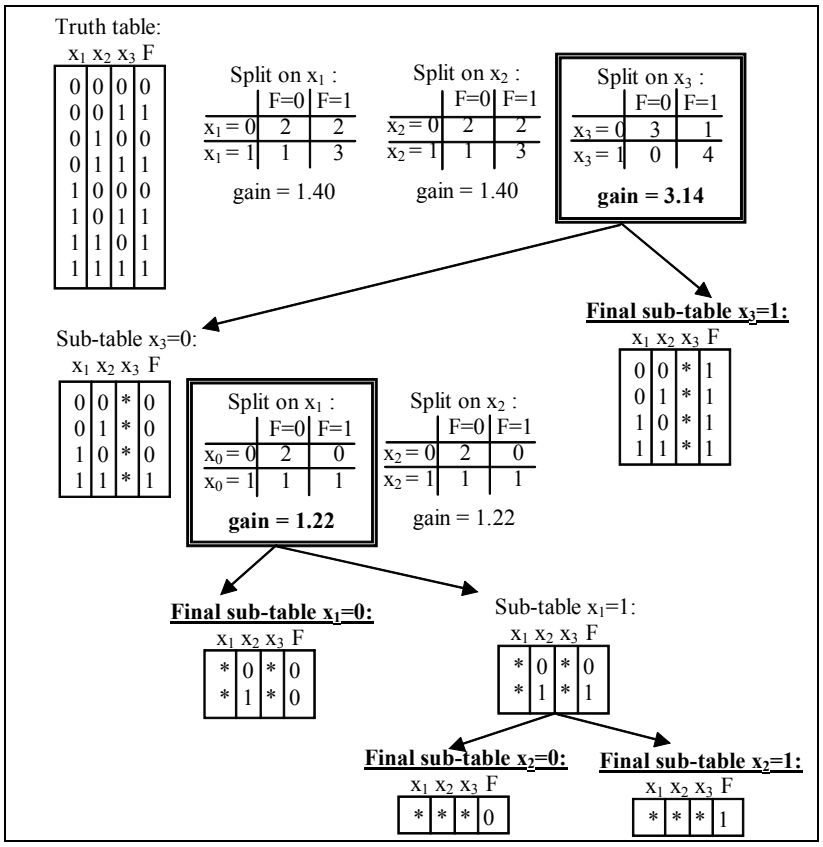

Fig. 2: Building binary decision tree for $F=x_1x_2+x_3$

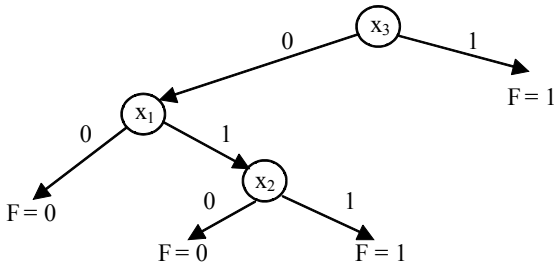

Fig. 3a: Binary decision tree for $F = x_1x_2+x_3$

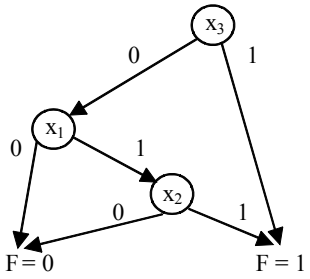

Fig. 3b: Binary decision diagram for $F =x_1x_2+x_3$

The function must be represented by truth table. The advantage of suggested approach lies in the implicit ordering of BDD nodes and lower BDT complexity. The worst case time complexity of our BDT construction is $O(n^2 2^n)$. Time complexity of classical simple greedy BDD construction is $O(n!2^n)$ and time complexity of advanced classical algorithm based on dynamic programming is $O(n^2 3^n)$. In [1] the suboptimal bottom-up approach for every ordering of variable with time complexity $O(n^2 2^n)$ is described.

The maximal structure complexity of the BDD (the size of BDD) is given by the number of nonterminal nodes. As mentioned before the structure complexity of BDD is determined by variable ordering. In case of simple Boolean function $F_m = x_1x_2+x_3x_{4+,\ldots,+}x_{2n-1}x_{2n}$ the upper bound of structure complexity is $O(2^n-1)$.

We have applied our BDT constructor on the function F_m for increasing number of variables from 4 to 20 with folowing results:

n - number of variables	4	6	8	10	12	14	20
m - number of BDT decision nodes	6	14	30	62	126	254	2046
Number of reduced BDD decision nodes	4	6	8	10	12	14	20

Table 2. Resulting size of BDT/BDD representing the function F_m

From this experiment the size complexity of BDT was derived as $O(2^{n/2+1})$. The BDD (OBDD) was derived from BDT by reduction heuristic with time complexity $O(m^2 \log m)$ where m is the size of BDT. The BDD size complexity is O(n).

2.2 BDD for multi-valued Boolean function

We extended the idea of decision diagram to integer domains for multi-valued logic function (see Fig. 4). If the target variable X_i is categorial, we use in (1) symbols m_{ij} from the following contingency table with multiple columns (each of them representing the possible value of X_i) to evaluate the gain of possible split/decision nodes:

	$\mathbf{X_i = 0}$	$\mathbf{X_i = 1}$	$\mathbf{X_i = 2}$	$\mathbf{X_i = 3}$
$\mathbf{X_j = 0}$	$m_{0,0}$	$m_{0,1}$	$m_{0,2}$	$m_{0,3}$
$\mathbf{X_j = 1}$	$m_{1,0}$	$m_{1,1}$	$m_{1,2}$	$m_{1,3}$

Table 3. An example of m_{ij} coefficients for target variable $X_i \in \{0,1,2,3\}$

x_0	x_1	x_2	F
0	0	0	3
0	0	1	5
0	1	0	3
0	1	1	4
1	0	0	4
1	0	1	4
1	1	0	4
1	1	1	5

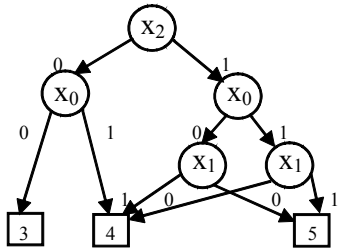

Fig. 4 An example of multi-valued Boolean function a) truth table, b) BDD

3 BDD based Bayesian decomposer

BDD based Bayesian decomposer was developed for general problem of system decomposition. We demonstrate its performance on the partitioning of digital circuits, see chapter 4. The decomposer is based on the Bayesian Optimization Algorithm (BOA)[5] and [6] that belongs to the probabilistic model building genetic algorithms where crossover and mutation operators are replaced by probability estimation and sampling techniques. The skeleton of common EDA algorithm (Estimation Distribution Algorithm) can be specified as follows:

Generate initial population of individuals of size M (randomly);
While *termination criteria is false* **do**
begin
Select parent population of N individuals according to a selection method;
Estimate the probability distribution of the selected parents;
Generate new offspring according to the estimated probabilistic model;
Replace some individuals in current population with generated offspring;
end

Individuals in the population are treated as vectors of instantiations of n random variables X_i, each random variable (gene) represents one parameter of a solution

$$\boldsymbol{X} = (X_0, X_1, \ldots, X_{n-1})$$

Most methods for automated learning of parameter dependencies in EDAs have been adopted from the area of data mining. We proposed the mBOA algorithm based on our BDT construction. The idea of the utilizing DBT (resp. BDD) in BOA algorithm was mentioned for the first time in [7]. We extended the idea of BDT (BDD) decision trees to continuous and integer domains. This way we were allowed to extend the standard BOA algorithm to Mixed Bayesian Optimization Algorithm (mBOA). Our mBOA is the only one EDA which is able to solve problems with mixed real-discrete parameters (alleles) without conversion to binary representation.
The estimated probability model encodes the probability of whole chromosome X as the product of local conditional probabilities. In mBOA the probability model is composed of n decision trees. An ordering permutation of variables $(o_0,o_1,\ldots,o_{n-1})$ exists such that the variables $\{Xo_0,Xo_1,\ldots,Xo_{i-1}\}$ can serve as splits in the binary decision tree of target variable Xo_i. Each leaf determines $p(Xo_i)$ among the individuals fulfilling the split conditions on the path from the root.

As a result of probabilistic model construction we obtain a set of decision trees, one tree for each variable. This set of trees is used for generation of the offspring (new population) during the Probabilistic Logic Sampling (PLS). During this PLS process the variables of each offspring are generated by traversing decision trees in (o_0,o_1,…,o_{n-1}) order.

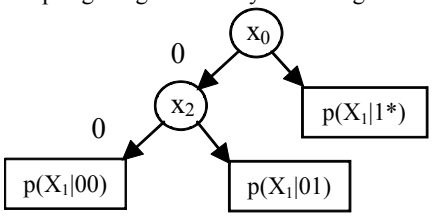

Fig. 5 Example of binary decision tree for the determination of X_1

In the example in Fig. 5 the concrete value of variable X_1 is generated according to concrete values of X_0 and X_2 using $p(X_1)$ value from corresponding leaf of decision tree.

3.1 Parallel construction of BDD

The probabilistic model construction is the most time consuming task in mBOA. We are currently working on the distributed mBOA using Message Passing Interface (MPI). The goal is to utilize more processors when searching for a good model. Our consolation is that the BD metric is separable and can be written as a product of n factors, where i-th factor expresses the quality of decision tree for variable X_i. It is possible to use up to n processors, each processor has its own local copy of parent population and it builds tree for different variable. The addition of splits to the trees is parallel, so we need an additional mechanism to keep the mutual dependencies acyclic. In [8] we proposed the concept of restricted set of parents in Bayesian network. We are going to implement this concept for BDD in mBOA. In each generation, variables will be ordered in advance, according to a random permutation vector $(o_0, o_1,..,o_{n-1})$. Each decision tree of variable X_{oi} may contain only such parental splits X_{oj} having $i < j$. This approach ensures linear scalability because no communication overhead is required.
The following algorithm specifies the parallel construction of BDTs and their sampling:

Gererate random permutation vector (o_0 ,o_1,..,o_{n-1});
For *i:=0* **to** *problem_size–1* **do in parallel**
begin
Model estimation: Build BDT for target variable X_{oi} (in each processor independently);
Model sampling: Receive from other processors values of predecessor variables X_{o0}, X_{o1}, …, X_{oi-1} ;
Generate X_{oi} according to BDT and received variables;
Broadcast X_{oi} ;
end

The time for BDT construction increases with i (as more variables are allowed to be used as splits). In addition, scalable methods for overlapping the communication latency during generation, evaluation and broadcasting of new population among the processes will be implemented using the farmer-workers architecture. For example the communication latency during receiving X_{o0}, X_{o1}, …, X_{oi-1} from the other processors can be overlapped by starting the next construction of BDT from the queue of tasks.

4 Decomposition of large digital circuits

Decomposition/partitioning a large circuit into k subcircuits/modules is frequently solved problem that has several applications in VLSI circuit design ranging from circuit layout to logic minimization, simulation and testing. The k-way partitioning can be defined as follows: Let us assume a circuits (E,S) with a set of circuit elements $E=\{e_1,....e_n\}$ that are connected by the set of nets $S=\{s_1, ...,s_m\}$, where a net is a subset of elements to be interconnected. The goal is to find such a partition of the set of elements into k modules $\{M_1, M_2,..., M_k\}$ so as to minimize the amount of interaction among modules under the size constraint

$$(1-\beta)\frac{|E|}{k} \le M_i \le (1+\beta)\frac{|E|}{k}, \quad i = 1,2,...,k \tag{3}$$

β is user-specified parameter to express the measure of the size imbalance of individual modules in equi-partitioning process.

To minimize the number of interactions among modules various criteria/cost functions have been used depending on the specific application context in which the partitioning is performed. Three main cost functions can be used:

- net-cut value *Ce* (cut-value, cut-size) – number of external nets
- wire-cut W_e - number of external wires
- pin-count P_c – number of external pins

A relation among these cost criteria exists: $P_c = C_e + W_e$. In case of two point nets it holds: $C_e = W_e$ and $P_c = 2C_e$.

The choice of a proper criterion is provided by the application context (average delay of net, via-count, test generation etc.) In Fig. 6 there are two cases of *5*-way partitioning with constant pin-cut value $P_c=15$. In the first case cut-value is very small, C_e =3 but $W_e=12$ unlike the second case with C_e =7 and $W_e=8$.

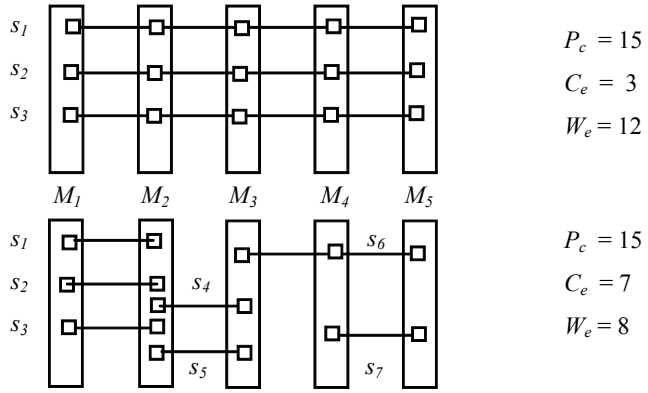

Fig. 6. Comparison of two cases of *5*-way partitioning in the layout context.

In the layout context, W_e represents the number of printed wires so the first partitioning is more adequate.
In the context of the net average delay $D_a = W_e/C_e$ we get for the first partitioning $D_a=12/3=4$ and for the next one $D_a =8/7=1,14$ so the first partition has to be preferred too.

4.1 Ratio-cut partitioning

To realize the multi-way partitioning the recursive minimum-cut bisection is mostly used producing modules with the evenly size or with a small size imbalance according to constraint (3). But this approach in common is not capable to identify a natural cluster in logic design which is crucial for the successful hierarchical design of the large digital circuits. Therefore a new approach was developed in [9] and [10] based on the ratio-cut metric. It is a variant of minimum–cut metric where the constraint (3) has been moved into the cost function. The ratio-cut cost function is given by

$$Rc= C/[size(M_1)*size(M_2)*...*size(M_k)] \quad (4)$$

where *C* represents one of the three cost functions C_e, W_e and P_c.
We have compared the standard recursive minimum-cut bisection technique (with zero imbalance of module size) with the recursive ratio-metric for each of the three cost functions.

5 Experimental results

5.1 Test benchmarks

The complexity of the partitioning problem is determined by the type and complexity of the instances – benchmark graphs. We used two benchmarks consisting of two real circuits (random logic) from benchmark package [11] with following parameters: circuit IC_67 consists of 67 elements and 138 nets, the IC_116 consists of 116 elements and 329 nets.

5.2 Summary of experimental results

We have arranged *8*-way partitioning experiments (ten runs for each one) for three type of cost criteria (see Fig. 7-9):

- net-cut C_e (cut-value, cut-size) – number of external nets
- wire-cut W_e - number of external wires
- pin-count P_c – number of external pins

We have implemented and compared two algorithms – AMB based on standard recursive minimum-cut bisection technique (with zero size imbalance) and ARC based on the recursive minimum ratio-cut partitioning.

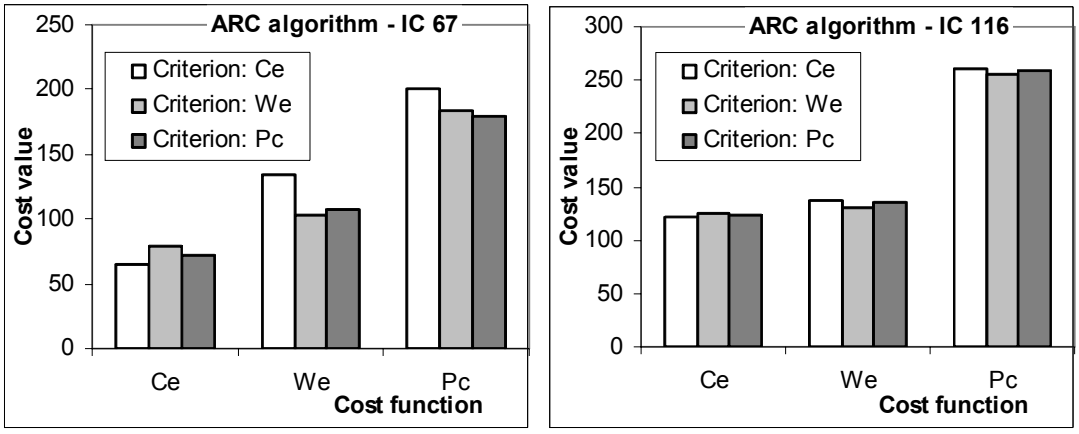

Fig. 7. Efficiency of each criteria expressed by values of other cost functions: a) for IC 67, b) for IC 116

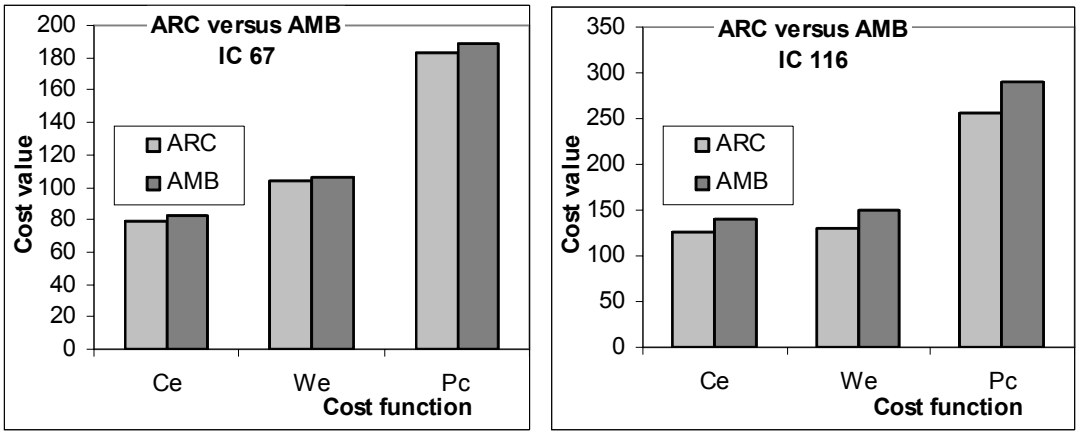

Fig. 8 Comparison of performance of algorithm ARC and AMB for W_e criterion: a) for IC 67, b) for IC 116

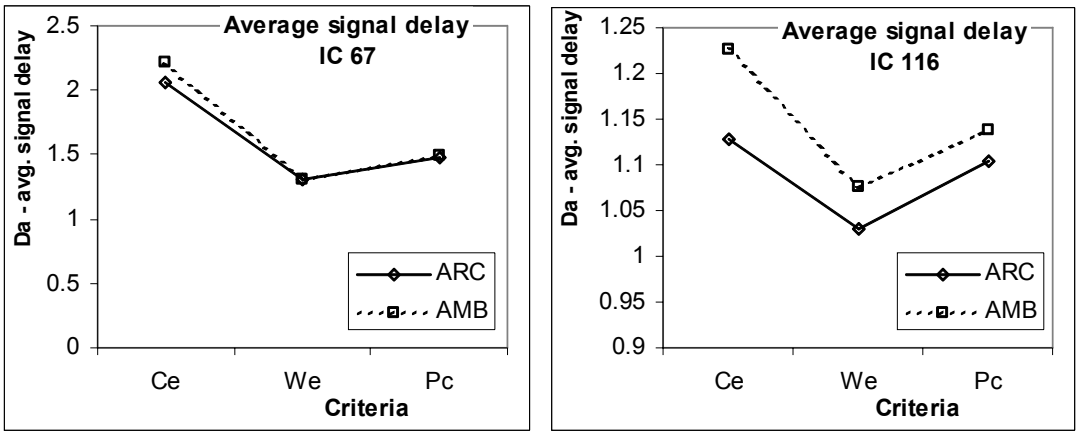

Fig. 9 Dependence of average signal/net delay on the type of optimization criteria: a) for IC 67, b) for IC 116

6 Conclusions

We have developed and implemented a new two phase algorithm for BDD construction which removes the problem of variable ordering. In the first phase the minimized BDT was found using Bayes-Dirichlet metric. In the second phase fast heuristics was used based on the reduction of decision nodes with equal ancestor nodes and sharing all equivalent sub-graphs. We have tested our approach on the well known benchmark representing multiplexor- Boolean function F_m up to 20 variables - the known global optimum is found in all cases. But it is desirable to test the BDD constructor on the larger number of benchmarks. We also have implemented a circuit decomposer based on BDD mBOA algorithm for multi-way partitioning of digital circuits. Its main advantage against the mostly used move-based heuristic methods lies in the ability to discover and determine the amount of epistasis in a given problem instance and to find the optimal solution. The Bayesian statistics and BDD diagrams used in probabilistic model cause its high performance. In Fig. 7 the efficiency of each criterion is shown. It is evident that criteria W_e and P_c performed quite well and can be used alternatively. From Fig. 8 it follows that the recursive ratio-cut algorithm ARC produces better results than the recursive min-cut bisection algorithm AMB. The difference is not too high, it is caused by the nature of applied benchmarks that probably do not include strong clusters. In Fig. 9 the average delay of nets D_a is represented as a function of optimization criterion – it is evident that the criterion C_e provides the best value of D_a.
The future activity will be focused on the sophisticated ARC algorithm including Rent's rule and the implementation of parallelization of the BDD mBOA algorithm. Another task to be solved is the deeper exploitation of the concept of OBDD construction using the Bayes-Dirichlets metric.

Acknowledgement

This research has been carried out under the financial support of the Research intention no. CEZ: J22/98: 262200012- "Research in information and control systems" (Ministry of Education, CZ) and the research grant GA 102/02/0503 "Parallel system performance prediction and tuning" (Grant Agency of Czech Republic).

References

[1] Dvořák V.: Bounds on Size of Decision Diagram. JUCS, Vol.3, 1997, pp. 2-22.

[2] Sasao T., Fujita, M., editors: Representations of Discrete Functions. Kluwer Academic Publisher, London 1996.

[3] Drechsler R.: Evolutionary algorithms for VLSI CAD. Kluwer Academic Publishers, London 1998, ISBN 0-7923-8168-8, pp. 1-183.

[4] Heckerman, D., Geiger, D., & Chickering, M. (1994). Learning Bayesian Networks: The combination of Knowledge and Statistical Data (Technical Report MSR-TR-94-09), Redmont, WA: Microsoft Research, 1995, pp. 1-53.

[5] Pelikan, M.: A Simple Implementation of Bayesian Optimization Algorithm in C++ (Version 1.0). Illigal Report 99011, February 1999, pp. 1-16.

[6] Pelikan, M., Goldberg, D. E., & Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Model, Illigal Report 99018, September 1999, pp. 1-12.

[7] Pelikan, M., Goldberg, E., Sastry, K.: Bayesian Optimization Algorithm, Decision Graphs, and Occams Razor. Illigal Report No. 2000020, May 2000, pp.1-24.

[8] Očenášek, J., Schwarz, J.: The Distributed Bayesian Optimization Algorithm, Proceeedings of the Eurogen 2001 - Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, The National Technical University of Athens, Greece, 19-21 September 2001, in print.

[9] Wei, Y.- C., Cheng, C.-K: Ratio Cut Partitioning for Hierarchical design, IEEE Trans. Computer Aided Design Integrated Circuit & System, Vol.10 (No.7), July 1991, pp. 911-921.

[10] Stroobandt, D.: Pin Count Prediction in Ratio Cut Partitioning for VLSI and ULSI. In M. A. Bayoumi, Editor, Proceedings of the IEEE International Symposium on Circuits and Systems, May 1999, Pages VI/262-VI/265.

[11] A Benchmark Set, University of California, Los Angeles, VLSI CAD Laboratory, http://vlsicad.cs.ucla.edu/~cheese/benchmarks.html.