"Vasile Alecsandri" University of Bacău
Faculty of Sciences
Scientific Studies and Research
Series Mathematics and Informatics
Vol. 20 (2010), No. 1, 45-60

EMBRYONIC GENETIC ALGORITHM WITH RANDOM GENERATIONAL GROWING STRATEGY FOR OPTIMIZING VARIABLE ORDERING OF BDDS

OCTAV BRUDARU, IULIAN FURDU AND RÜDIGER EBENDT

Abstract. This paper addresses the problem of optimizing the variable ordering in *Binary Decision Diagrams* (BDDs). A new hybrid embryonic genetic algorithm is proposed for optimizing the variable ordering that combines a branch & bound technique with the basic genetic algorithm. It uses fitness based on a lower bound and embryos instead of full chromosomes. A novel growing technique introduces two new growing operators. The results of an experimental evaluation demonstrate the efficiency of the approach.

I. INTRODUCTION

Reduced Ordered Binary Decision Diagrams (ROBDDs, often BDDs for short) have numerous applications- e.g. in formal verification of digital circuits and other finite state systems, algorithms for symbolic model checking manipulate Boolean functions by use of BDDs or derived data structures. BDDs offer a good trade-off between efficiency of manipulation and compactness of representation of Boolean functions, and they have improved time and memory performance of the aforementioned applications.
This paper addresses the problem of optimizing the variable order in BDDs. In this problem, 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 [1]. The BDD's size is given by the number of its nonterminal nodes and strongly dependents on the chosen variable ordering. An appropriate ordering leads to a smaller number of nodes, whereas a bad

Keywords and phrases: BDD, OBDD Optimization, GA
(2000) Mathematics Subject Classification: 06E30, 94C10, 68W35

ordering can lead to an exponential growth in the size of BDD.
This sensitivity to the ordering of variables is a significant problem for some applications in logic synthesis. For example, in Pass Transistor Logic (PTL) or other multiplexor-based design styles, the BDD is mapped to a digital circuit, and therefore the BDD size directly transfers to chip area.
Therefore, many methods for optimizing the variable order [1, 2, 3] have been proposed, including both static (e.g. [4]) and dynamic techniques. The most commonly used are the dynamic techniques [5, 6, 7], which dynamically proceed in obtaining an improved variable order π' (and consequently a better size) of an already built BDD with an initial variable order π. The most popular dynamic technique is Rudell's sifting algorithm [7].

Besides sifting, simulation-based algorithms (SA) and genetic algorithms (GA) for optimizing the variable ordering have been proposed: e.g. a technique based on simulated annealing is reported in [8], and the first GA for BDD optimization has been introduced in [9]. Methods based on genetic algorithms yield better results than other heuristic techniques, as experimental studies from [9, 10] have demonstrated. The use of sifting as a genetic operation that replaces crossover techniques or advanced strategies for setting the parameters are among the key ideas of a range of approaches [11, 12, 13] that focused on the speed up of the computations.

Parallel or distributed GAs used in [14, 15] are reported to obtain better results than sequential GAs. However, the fitness evaluation remains an expensive task increasing the total cost of solution.

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 [3, 16]. This paper follows this trend and describes a new hybrid GA for optimizing the variable order in BDDs. The starting point is represented by a basic GA (BGA) that includes an information energy based mechanism [17] used for adjusting the variability level of the current population along the evolution stages [13]. A branch & bound technique is combined with the BGA by adopting and operating with embryos as subsets of orders instead of individual complete orders. This hybridization contributes to a better balancing between exploration and exploitation of the search space.

Section II formulates the problem of variable ordering in BDDs. Section III describes the main components of the double hybridized GA. Section IV presents the experimental evaluation of its performance. Last section summarizes the work.

II. THE VARIABLE ORDERING PROBLEM AND ITS IMPORTANCE FOR BDDS

In this section the reduced OBDDs and the importance of variable ordering are briefly reviewed.

Let $f\colon B^n \to B^m, B = \{0,1\}$ be a switching function and π - a total order on a fixed set of Boolean variables $x_1, x_2, ..., x_n$, $x_i \in B$.

Definition 1. An OBDD for f with respect to order π is a single rooted direct acyclic graph that satisfies [18]:

a) there are exactly two terminal nodes labeled by Boolean constants 0 and 1, respectively.

b) each non-terminal node is labeled by a variable x_i, and has two outgoing edges, called 0-edge and 1-edge. At each inner node, two subfunctions are usually computed according to Shannon decomposition [1]:

$$f = x_i f_{x_i} + \overline{x}_i f_{\overline{x}_i} \ (1 \le i \le n) \tag{1}$$

Starting at the root node that is labeled by some variable x_i, f is decomposed into two subfunctions. These are the cofactors of f with respect to x_i. The same decomposition scheme is carried out recursively at each node representing one of the subfunctions. Eventually, constant subfunctions are obtained, and the outlined process of decomposition terminates.

c) each variable appears at most once on a path from the root to a terminal node and the order in which the variables appear is consistent with the variable order π.

If reduction rules are applied [18] to obtain an irreducible form (i.e. a form where no more reduction rule applies), this form is called reduced OBDD or ROBDD. A ROBDD is a canonical representation for Boolean functions [1].

Definition 2. The size of an OBDD is given by the number of its non-terminal nodes.

The reduced OBDDs were introduced by Bryant [18], based on these rules. In the case of multi-output Boolean functions $f: B^n \to B^m$, BDDs can be defined analogously since the multi-output Boolean function can be seen as a family of single-output component functions $f = (f_i)_{1 \le i \le m}$. A BDD G_i is used for every component function f_i in the shared representation G of f. The compactness of the BDD representation can further be enhanced by the use of so-called ROBDDs. Complement Edges (CEs). They allow for the simultaneous representation of a function and its complement by the same graph. In the following, BDDs with CEs are assumed without mentioning it further (and, to obtain a more instructive presentation, also without using CEs in the illustrations). Note that all results reported here directly transfer to BDDs without CEs. Moreover, only reduced, ordered BDDs are considered, and for brevity these graphs are called BDDs.

The main problem in dealing with BDDs is the strong dependence between the order of the input variables and the BDDs' size (Fig. 1) which can vary from linear to exponential.

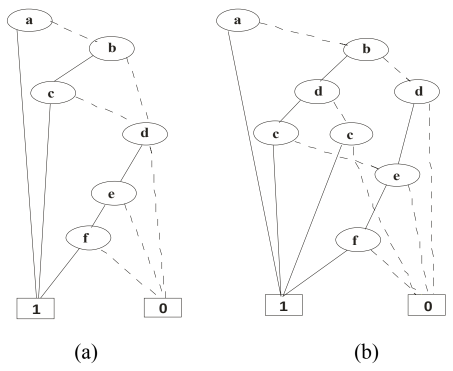


Fig. 1. *The influence of the variable ordering for f(a,b,c,d)=a+bc+def with order a, b, c, d, e, f (a) and order a, b, d, c, e, f (b), (0- edges are dashed).*

It has been proven that deciding whether the order of a given BDD can be improved is NP-complete [19]. However, by the use of heuristic approaches, we can often find an ordering of variables that is good enough to keep the size of the corresponding BDD maintainable for real-life applications like hardware verification or software model checking.

This paper describes a new technique for optimizing variable ordering in BDDs based on hybridization of GAs with a branch & bound technique. The resulting hybrid algorithm inherits the best attributes from its both constituents.

The basic components of the proposed GA are further described.

III. BASIC COMPONENTS OF THE DOUBLE HYBRIDIZED GA

Genetic algorithms [20] (GAs) for the variable ordering problem rely on a representation of the variable orderings in permutation form (chromosomes) and act on an initial population by applying genetic operators in order to obtain a new population of solutions. A GA uses a survival selection of the individuals and a fitness function that indicates the degree in which a solution fulfill the problem constraints. The chromosomes undergo repeated changes usually organized as evolution stages, by means of the chosen genetic operators until a set of best found solutions is produced.

A. Basic components of hybrid genetic algorithm

Incorporating a branch & bound technique in the mechanism of the GA implies changing the representation, the adaptation of genetic operators and an appropriate population management.

Solution representation. Each adult chromosome $x = (x_1, x_2, ..., x_n)$ represents a specific BDD variable order of f.

The combination between GA and branch & bound technique must use embryonic representations corresponding to nonterminal edges in the state space tree partially constructed during the process of this method. So, $x = (x_1, ..., x_k)$ denotes an *embryo* with $1 \le k < n$.

Initial population. Initial population P is randomly generated. It contains embryos having a random length λ_0, where $2 \le \lambda_0 \le n/3$. Table 1 describes an empiric function to choose the population size $|P|$ according to the number of input variables.

Table 1. Population size

n	<20	21-150	150-250	>250
|P|	50	75	120	200

Mutation operators. Three mutation operators are used: mutual exchange, group mutation and inversion [20]. All mutation operators are adaptive operators in the sense that the distance between the cutting points linearly decreases with the number of generations in order to ensure a small disruption rate as the algorithm converges [13]. The whole population bears mutation with a given probability p_m. The three mutation operators remain the same when dealing with embryos. A variant of AX1 which assumes that the left cut point for the bigger parent embryo is equal to the length of the smaller embryo is used as an additional crossover operator.

Crossover operators. The first crossover operator AX1 is a block variant of the alternating crossover described in [9]. A second crossover AX2 is a restricted version of AX1 and the hybridized GA makes exclusive use of it. Also crossover operators are adapted during progress of the algorithm: the lengths of the segments which the offspring inherits from its parents linearly increase with the number of generations. This ensures preserving of good constructive blocks of the solutions. The mating pool is formed by the first 40% of the best individuals and crossover operations are produced with the probability p_c. When applied to embryos, crossover operators act in the same way.

Growing operators. A growing mechanism is needed since all embryos have to reach maturity, i.e. to reach full (adult) length. Two growing operators are proposed, both of them applied with the same probability p_G.

The first growing operator extends x with a sequence of q variables, chosen by random, where $1 \le q \le min\{10, max\{1,(n-k)/10\}\}$.

The second growing operator replaces x by a number σ of its children, $1 \le \sigma \le max\{1, s/10\}$, where s is the sample size, $s = 2 + (n-2-k) \cdot r$, and $r \in [1, \ln n]$ is a parameter that determines the size of the sampling [13]. Let $S(x)$ be the set of all orders having the prefix x and let $y_1,...,y_s$ be the randomly select s members of $S(x)$. Let $c_1,...,c_s$ be random numbers between 1 and q, where q is the value defined for first growing operator. Let z_j be the prefix of length $k + c_j$, $j = 1,...s$. The best σ embryos $z_1,...,z_\sigma$ replace the parent x in the population. In this way, the branching of x uses the most promising extensions of a sample. Of note is that the original motivation for constructing the sample was the computation of a sample-based fitness function [13].

B. Fitness function

Since this genetic algorithm operates with embryos that are partial solutions, the *objective function based* fitness is replaced by a *lower bound based* fitness. This choice is a part of the heritage of the branch & bound technique. Let $x = (x_1,...,x_k)$, with $1 \le k \le n$, be a chromosome. The fitness of the prefix is $fit_{lb}(x) = \sum_{i=1}^{k} n_i + n - k$, where n_i is the number of nodes on the i-th level of the BDD built on an arbitrary ordering in $S(x)$. Clearly, if $k = n$ then $fit_{lb}(x)$ is exactly the size of the ordered BDD built on x, and the hybrid GA behaves as a classical GA.

Most of the research on BDD minimization by branch and bound algorithms use a lemma proved in [21]. The result from [21] states that n_i, $i = 1,...,k$ are the same for all possible extensions of the prefix $(x_1,...,x_k)$. The term $n-k$ corresponds to the last $n-k$ variables in the order and, in the definition of $fit_{lb}(x)$, plays the role of the predictive part, while the sum term corresponds to the first k variables and gives the contribution of the prefix itself to the size of any BDD respecting a variable order that is an extension of prefix x. In general, when the fitness function is a lower bound on 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. The *best-first* approach of [16] 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.

C. Management of population

Growing strategy. In order to obtain the final population containing adult chromosomes, the growing operators can be used in different ways. The strategy adopted in this paper for applying the growing operators is called *random generational growing* (shortly, RGG) and consists in randomly applying the growing operators with the given probability p_G to the current population. Each time, when the growing event is produced, each embryo has the same chance to be selected for growing, regardless of its length. The growing of an embryo $x = (x_1,...x_k)$ means its replacement with one or many new embryos descendants from x.

Even if the crossover is an implicit source of growing, it is not enough to ensure a feasible evolution. In RGG used in this paper, each growing operator is applied with the same probability p_G and each embryo in the current population has the same chance to suffer the growing effect.

Mechanism for controlling the variability.

A mechanism for tuning the values of p_m and p_c based on the information energy [17] is adopted in order to prevent premature stagnation and to improve the solution quality. Information energy gives a measure of variability of the fitness values in the current population. A good tradeoff between exploration and exploitation could be achieved if an appropriate level of population variability is maintained during the evolution. Therefore, the probabilities p_m and p_c both have initial value 0.5 and are modified at each evolution stage according to the rules:

$$p_m = p_m - \alpha\lambda\,,\ p_c = p_c + \beta\lambda\,,\ \ \alpha,\beta \in [0,1] \tag{2}$$

where $\lambda = (E_{ob} - E_c)/T$ is the speed wherewith a generic target variability value E_{ob} is reached by the current value of the information energy E_c, and $T, 2 \le T \le 6$ is also a parameter that controls the speed of adjustment of these probabilities. E_c is the current computed variability which is distributed along the time period T. Information energy formula in [17] gives the current level of variability E_c

$$E_c = \sqrt{\sum_{k=1}^{p} (\varphi'_k)^2} \quad , \tag{3}$$

where $\varphi_1,...,\varphi_p$ represents the relative frequencies of the values of the fitness function distributed in p sampling intervals, each such interval having the same length. A low (high) population variability is indicated by a high (low) value of E_c. As established in [13] the periodic function

$$E_{ob}(t) = (A_1 - A_2)\,|\cos \omega t\,| + A_2\,,\ 1/\sqrt{p} \le A_2 < A_1 \le 1\,,\ \omega = 2,3,... \tag{4}$$

offers good results and is adopted in this paper. Values for the other parameters that we recommend here are: $\mathrm{A1} = 1, \mathrm{A2} = 0.5$ for E_{ob} periodic, $\omega = 3$ and $T = 4$.

Selection for survival. The selection is deterministic and elitist [20]. At the end of each stage, the newly created individuals compete with the current population and those with better fitness survives for the next stage, in the limit of $|P|$.

Stop condition. The algorithm stops if the population contains only adult chromosomes and if the variation of the ratio $[\Phi_m(k) - \Phi_m(k-1)]/\Phi_m(k) < e$ is true for a given number n_{stop} of successive iterations. $\Phi_m(k)$ represents the average of the fitnesses at stage k and e is a prescribed tolerance.

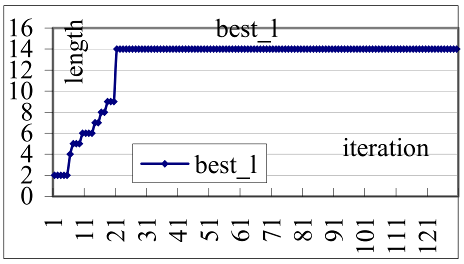
IV. EXPERIMENTAL EVALUATION

A subset of LGSynth91 benchmarks suite [22] was used in experiments that we conducted with the new hybrid approach. The algorithm was implemented using the CUDD package [23] on a Dual Core system, with 2,4 GHz processors and 2G RAM available memory. A number of 10 runs per circuit test were executed for each type of experiment.

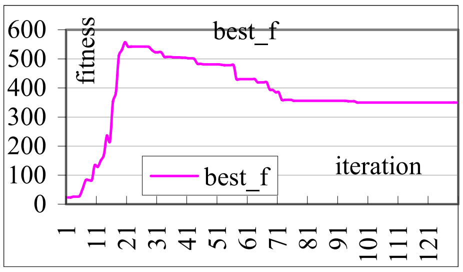
Setting of parameters. The first goal of the experiments was to find appropriate values for the parameters. The values in Table 1 are resulted from experiments. A recommended value for e involved in stop condition is 0.001 while appropriate setting for n_{stop} is 20. Adequate distribution of probability for AX1, AX2 is 0.7 and 0.3 respectively. For simple mutation, inversion and group mutation recommended probabilities are 0.3, 0.3 and 0.4, respectively. The most adequate hypermutation probability found was $p_H = 0.1$. The increment q of the extensions made by the first growing operator and the number σ of descendents of an embryo for the second growing operator was set to 2. A high value of $p_G = 0.4 \div 0.6$ is adopted in order to avoid the dominance of the short embryos over the long ones.

Qualitative estimation of the behavior. In random generational growing strategy embryos with different lengths compete between them during each evolution stage. The lengths of embryos from initial population are at most 1/3 of adult size. Typical behaviors of population when applying this strategy are illustrated below using the *alu4* benchmark (14 input, 8 output, best number of nodes 350).

Fig. 2 (a) shows typical variations of the length of the best chromosome vs. number of evolution stage. The variation of the fitness of the best chromosome is shown in Fig. 2 (b).



(a)



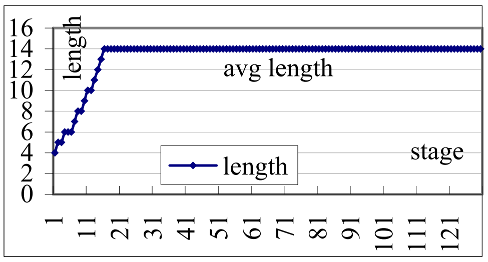
(b)

Fig. 2. *Length (a) and fitness (b) of the best chromosome vs. number of evolution stage (alu4)*

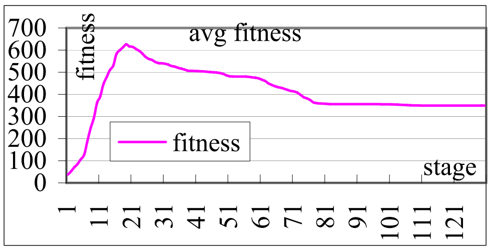
Fig. 2 (b) indicates that in the first 21 iterations the fitness increases as the accuracy of estimation by lower bound becomes higher. Fitness goes down after the phase where all embryos have become adults (starting with generation 21, all embryos have full length 14) when the GA acts as a classic GA.

Fig 3 shows the variations of the average of lengths (a) and the average of the fitness values of the chromosomes in current population during the evolution (b). Growing was applied with probability $p_G = 0.5$.

Fig. 2-3 show that a reduction to about of a half from the highest of the fitness function is obtained after all the embryos in population reach full length. To achieve this, it takes about 40-45% of the total number of evolution stages.



(a)



(b)

Fig. 3. *Average length (a) and average fitness (b) of the population during the evolution (alu4)*

As mentioned before, the adopted lower bound based fitness has a term representing the history of exploring (the sum of the nodes of the levels corresponding to $x_1,...,x_k$) and a second term ($n-k$) that is the predictive part of the fitness. Due to the fact that the predictive part completes the first one to a lower bound on the minimum size of the BDDs whose orderings respect the prefix corresponding to the current embryo x, and since this can be far enough from this minimum, the score of the shorter chromosomes could be too optimistic when compared with the actual fitness of longer chromosomes (that better reflect the sizes of BDDs resulting from extensions of the embryo $(x_1,...,x_k)$). In some cases, this could lead to a massive elimination of longer chromosomes in some stages of the evolution and to a failure in obtaining adult chromosomes. This *shorter wins* phenomenon can be compensated by a rapid growing of embryos to adult size and by using the mechanisms for controlling the population variability for providing a good variability of the embryonic population.

The *shorter wins* phenomenon is illustrated in Figure 4 that shows a case where the evolution is stuck and explores a subset of embryos of length 10. Small values for $p_G = 0.25$ and $p_c = 0.2$ leads to this behavior. Embryos can not reach full size as result of the unbalanced (and thus, unfair) competition between short and long embryos. The growing is slower and the average of fitness has a slow improvement to a value (501) far from optimum (350 for alu4).

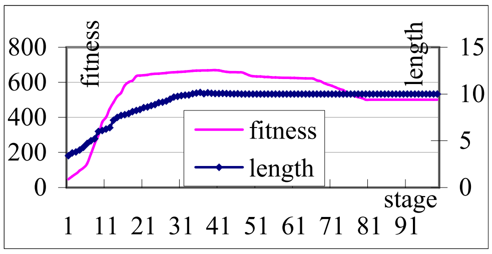
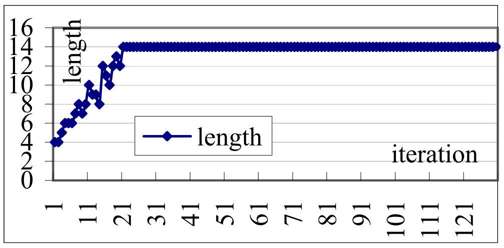
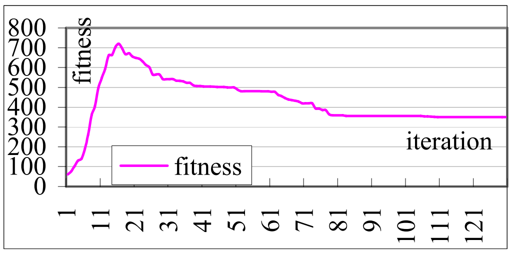


Fig. 4. *Average length and fitness vs. evolution stages in a case when embryos can not reach the adulthood (alu4)*

Figure 5 shows the phenomenon of elimination of the poorest embryos from the population. The weakest embryos are those with longer lengths which tend to loose the competition more frequently than the shorter ones.



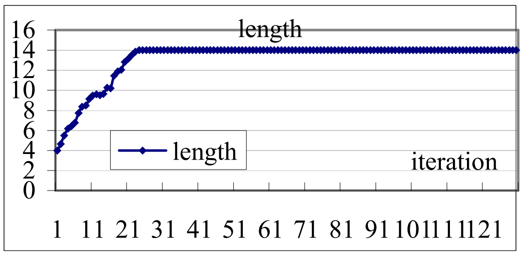
(a)



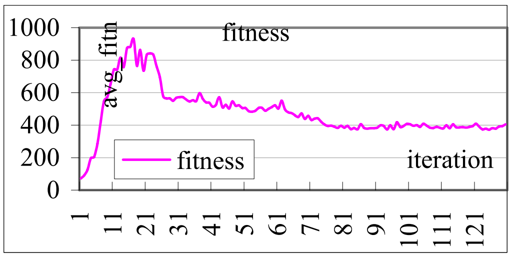
(b)

Fig. 5. *Length (a) and score (b) of the worst chromosome*

Fig. 6 describes the average length (a) and the average fitness (b) of the chromosomes that lose the competition during the evolution.



(a)



(b)

Fig. 6. *Average length (a) and average fitness (b) of the embryos removed from the curent population.*

The high variations recorded for the length of the worst chromosome (Fig. 5

a) transfer to variations of the average length of the removed individuals (Fig. 6 a).

Performance estimation. Table 2 shows best found results and costs for RGG strategy. Column #best gives the best ever reported [24] number of nodes for each benchmark. The column labeled with #nodes gives best-found results for RGG 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 [8]. Comparing with the results from [8] RGG obtained #best or best #nodes in all cases except apex7.

Table 2. Best found results and costs for random generational growing

Bench	**In**	**Out**	**#best**	**RGG**	
				#nodes	**#iter**
*cm85a	11	3	28	28	42
*cm163a	16	5	26	26	42
*cu	14	11	32	32	63
*alu4	14	8	350	350	79
*s1494	14	25	369	369	146
vda	17	39	478	479	242
misex3	14	14	478	478	90
*apex2	39	3	-	304	405
*apex7	49	37	-	243	382
dalu	75	16	689	704	610
cordic	23	2	42	42	118
ttt2	24	21	107	107	136
apex6	135	99	498	548	777
i3	132	6	133	135	696

Table 3 gives the average (m), standard deviation (σ) and unitized risk (σ / m) for absolute error abs_err = #nodes - #best. Smaller values of unitized risk indicate a more robust and stable technique as unitized risk is considered to be a measure of stability. The column labeled with *iter* contains the average number of iterations until the algorithm stops. The last column labeled with $\overline{f}$ gives the average number of fitness evaluations.

Table 3. Performance and costs for RGG

Bench	**m**	**σ**	**σ/m**	**iter**	$\overline{f}$

*cm85a	0.3	0.94	3.16	59.1	1374.71
*cm163a	1	1.05	1.05	68.7	1448.40
*cu	-	-	-	237.6	5108.40
*alu4	1.9	4.97	2.61	101.5	2395.30
*s1494	11.5	10.43	0.90	139.5	3658.48
vda	9.7	6.56	0.67	159.2	3800.10
misex3	25.5	44.38	1.74	101.7	2267.91
*apex2	360.3	63.99	0.17	387.3	7838.95
*apex7	259.7	13.11	0.05	404.8	9448.03
dalu	45.5	28.30	0.62	588.9	11813.33
cordic	1.9	3.24	1.7	140.5	3450.68
ttt2	12.4	15.79	1.27	166.3	3962.92
apex6	119.8	43.87	0.36	721.1	15049.36
i3	22.6	16.19	0.71	615.9	13617.55

The values from both tables indicate that the proposed algorithm gives good results and have a good stability. For 66% of cases #best was obtained for the benchmarks which #best is known. Combining a GA with an exact technique like branch & bound has lead to a better performance that is better than the performance of the pure GA. The search space has an initial good covering because of the high variability of the short embryos. This improvement is explained by the fact that as the algorithm converges and the embryos are growing a refinement of the search is done while the search space is narrowing.

V. CONCLUSIONS

This paper presents a new hybrid GA for the problem of finding the best variable ordering of BDDs.

The main novelty of the work is the adopting of embryonic chromosomes as subsets of variable orders instead of full-length variable orders for combining the GA with the branch & bound technique. A fitness definition based on a lower bound is used. The random generational growing strategy is used for applying the newly introduced growing operators.

The performance of the resulting hybrid method was experimentally investigated. The obtained results are reported in detail and they show that the proposed method performs very well. As further direction of investigation, new growing strategies have to be designed and evaluated.

References

[1] 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, Dordrecht/Netherlands p. 285-307, 2001.

[2] C. Meinel, T. Theobald, **Algorithms and Data Structures in VLSI Design**, Springer, 1998.

[3] R. Ebendt, G., Fey, R. Drechsler, **Advanced BDD minimization**, Springer, 2005.

[4] H. Fujii, G. Ootomo, C. Hori, **Interleaving based variable ordering methods for OBDD**, Int'l Conf. on CAD, p. 38-41, 1993.

[5] C. Meinel, A. Slobodova, **Speeding up variable reordering for OBDDs**, International Conference on Computer Design, p. 338-343, 1997.

[6] S. Panda, F. Somenzi, **Who are the variables in your neighborhood**. Int'l Conf. of CAD, p. 74-77, 1995.

[7] R. Rudell, **Dynamic variable ordering for ordered binary decision diagrams**. Int'l Conf. of CAD, p. 42-47, 1993.

[8] B. Bollig, M. Löbbing, I. Wegener, **Simulated annealing to improve variable orderings for OBDDs**, International Workshop on Logic Synth., pag. 5b:5.1-5.10, 1995.

[9] R. Drechsler, B. Becker, N. Göckel, **A Genetic Algorithm for Variable Ordering of OBDDs**, IEEE Proceedings, 143(6), p. 363–368, 1996.

[10] W. Lenders, C. Baier, **Genetic Algorithms for Variable Ordering Problem of Binary Decision Diagrams**, Lecture Notes in Computer Science, Springer, p. 1-20, vol. 3469/2005.

[11] R. Drechsler, N. Göckel, **Minimization of BDDs by Evolutionary Algorithms**, International Workshop on Logic Synthesis, 1997.

[12] M. A. Thornton, J.P. Williams, R. Drechsler, N. Drechsler, D.M. Wesels, **SBDD Variable Reordering based on Probabilistic and Evolutionary Algorithms**, IEEE Proceedings, Pacific Rim Conference, p. 381–387, 1999.

[13] I. Furdu, O. Brudaru, **New hybrid genetic algorithm with adaptive operators and variability target for optimizing variable order in OBDD**, Int’l Conf. on Mathematics and Informatics, Scientific Studies and Research Series Mathematics and Informatics vol. 19, no. 2, p.241-256, 2009.

[14] U.S. Costa, A. M. Moreira, D. Deharbe, **A cache-based parallel genetic algorithm for the bdd variable ordering problem**. Proc. of SBAC-PAD'2000, p.99-104, 2000.

[15] S. Droste, D. Heutelbeck, I. Wegener, **Distributed Hybrid Genetic programming for Learning Boolean Functions**, Parallel Problem Solving from Nature- PPSN 6th International Conference, p. 181-190, Paris, 2000.

[16] 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), p. 1515-1529, 2005.

[17] O. Onicescu, **Elements of Informational Statistics with Applications**, Technical Editing House, Bucharest, 1979 (in Romanian).

[18] R.E. Bryant, **Graph-based algorithms for Boolean function manipulation**, IEEE Trans on Computers. 35(8) pag. 667-691, 1986.

[19] B. Bollig, I. Wegener, Improving the Variable Ordering of OBDDs Is NP-Complete, IEEE, Transactions on Computers, vol. 45, p. 993–1002, 1996.

[20] D.E., Goldberg, **Genetic Algorithms in Search, Optimization and Machine Learning**, Addison Wesley, 1989.

[21] S. J. Friedman, K. J. Supowit, **Finding the Optimal Variable Ordering for Binary Decision Diagrams**, IEEE Transactions on Computers, vol. 39, p. 710-713, 1990.

[22] http://cadlab.cs.ucla.edu/~kirill/, accessed June 2010.

[23] CUDD package url: vlsi.colorado.edu/ ~fabio/ CUDD.

[24] ftp://vlsi.colorado.edu/pub/ orders.tar.gz, accessed June 2010.

Octav Brudaru
Institute of Computer Science, Romanian Academy, Iaşi Subsidiary,
"Gh. Asachi" Technical University Iaşi, Department of Management and Production Systems Engineering, Iaşi, Romania, e-mail: brudaru@tuiasi.ro

Iulian Furdu
"Vasile Alecsandri" University of Bacău,
Department of Mathematics and Informatics, Bacău, Romania, e-mail: ifurdu@ub.ro

Rüdiger Ebendt
German Aerospace Center, Institute of Transportation Systems, Berlin, Germany, e-mail: ruediger.ebendt@dlr.de