IDGBDD:The novel use of ID3 to improve Genetic algorithm in BDD reordering

M. Takapoo, M. B. Ghaznavi-Ghoushchi

School of Engineering, Shahed University

misagh_takapoo@yahoo.com, ghaznavi AT shahed.ac.ir

***Abstract*— ID3 is a classical decision algorithm that has been widely used in decision making problem. Ordered Binary Decision Diagram (OBDD) is a data structure for switching functions that has been used in many areas of Boolean function manipulation. The major problem with BDD-based calculations is the variable ordering which addresses the question of finding an ordering of the input variables which minimizes the size of the BDD-representation. We used ID3 as a heuristic method to improve the genetic algorithm method that has used for reordering BDD (IDGBDD) [1]. In this approach, we used ID3 as a best preorder predictor. Then we applied this order set as a first order level in BDDs. At the end, we used the genetic algorithm as a process stage to perform the final reordering to reduce the number of nodes. The result of using ID3 algorithm was reduction the number of nodes in 57% of the benchmark experiment examples. however, in 9.5% of the benchmark experiment examples, we have encountered increase in the number of nodes. It also decreased the number of iteration in 71% of the examples, but we also have encountered increase of the iteration in 4.7% of ran examples. The average improvement amount that obtained in decreased node number was equal 11.183%. Also the average improvement amount that obtained in decreased iteration was equal 25.79%.**

***Keywords* — ID3, Binary Decision Diagrams (BDDs), Variable Order, Genetic Algorithm.**

I. INTRODUCTION

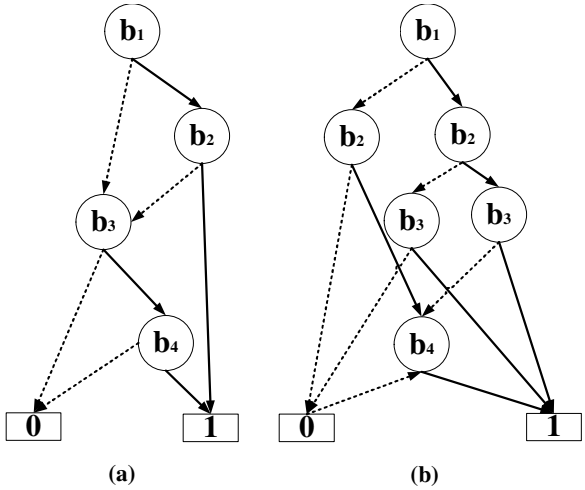
In this paper, to show our work we must introduce three different subjects that each of them is more useful on different branches of knowledge. These subjects are BDDs, ID3 algorithm and Genetic algorithm. After introducing these topics, we show the common point of these topics and what that guide us to use ID3 and Genetic algorithm as algorithms to reduce the size of BDDs.

A. Binary Decision Diagram

A binary decision diagram (BDD), is a data structure that is used to represent a Boolean function. BDDs are extensively used in CAD software to synthesize circuits (logic synthesis) and in formal verification. The basic concepts of BDDs graph are based on Shannon expansion theorem. This theory says that we can separate a binary function to two binary functions [2]:

$$F(X_1, X_2, ..., X_n) = X_1.F(1, X_2, ..., X_n) + X_1^{'}.F(0, X_2, ..., X_n) \quad (1)$$

Where, X_1 is one of the inputs of function F. As Shannon expansion shows, Boolean function can be represented as a rooted, directed, acyclic graph. A BDD consists of decision nodes and two terminal nodes called 0-terminal and 1-terminal. Each decision node is labeled by a Boolean variable and has two child nodes called low child and high child. The edge from a node to a low/high child represents an assignment of the variable to 0/1 [3]. A BDD is called 'ordered' if different variables appear in the same order on all paths from the root [4]. The root



node represents the Boolean expression describing f. An example of a BDD implementing the function b1b2 + b3b4 is shown in Fig.1 (a). Each non-terminal node t is labeled with a decision variable, and a solid line indicates a high edge while a dashed line indicates a low edge.

Figure 1. BDD of function b1b2+b3b4 (a) with variable order b1, b2, b3, b4 (b) with variable order b1, b3, b2,b4

In an "ordered" BDD, the sequence of variables evaluated along any path in the DAG is guaranteed to respect a given total decision variable order. The choice of decision variable order can significantly affect the number of nodes required in a BDD. The BDD in Fig. 1 (a) uses variable order b1, b2, b3, b4, while the BDD in Fig.1(b) represents the same function, with variable order

b1, b3, b2, b4[4]. Both figures have same function but different number of nodes. Binary decision diagrams were originally invented for hardware verification to efficiently store a large number of states that share many commonalities [4]. Concerning to this fact and all other things that we said about effect of order in BDDs size, finding a method to decrease the size of BDD graphs is a big research area in this branch of knowledge[5-6-7].

B. ID3

ID3, as a well-known decision tree induction algorithm, was proposed by Quinlan J R in 1977 [8-9]. It builds decision tree using a top-down procedure, based on information gain measure coming from Shannon's information theory, which heuristically leads to small trees. The algorithm computes the information gain of each attribute in training set and chooses the attribute with the highest information gain as the test attribute for the given set. A root node is created and labeled with the attribute, branches are created for each value of the attribute and the samples are partitioned accordingly. Then the algorithm uses the same process recursively to form a decision tree for the samples at each partition until all samples for a given node belong to the same class or that the case meet other end conditions.

ID3 is a useful concept learning algorithm because it can construct a decision tree efficiently that generalizes well. However, it also has some shortages: first, the information gain measure tends to prefer attributes with many values; second, each node is labeled by just one attribute which ignores the relationship exiting between some attributes; third, the algorithm requires all attributes to be discrete and cannot deal with the continuous value attributes; fourth, for incremental learning tasks, it would be far preferable to accept instances incrementally without needing to build a new decision tree each time and finally, the problem of data quality like noise, missing value, data redundancy or incompetence may result in ineffectiveness of classification [10]. The actual method that Quinlan offered for ID3 is as the following: Set up S is a set that contained s data samples, classification attribute could take m different valuations, corresponding "m" different categories, $C_i \in \{1,2,...,m\}$. Set up is sample's numerate in category C_i and then classifying the given data objects, the required information content is:

$$I(s_1, s_2, ..., s_m) = -\sum_{i=1}^{m} p_i \log_2 p_i \qquad (2)$$

Set up an attribute A get v different valuations Set up an attribute A get v different valuations $\{a_1,a_2,...,a_v\}$ using attribute A could divide set s into v subsets $\{S_1,S_2,...,S_v\}$. And amount it, S contained the data sample that the attribute A getting valuation a_j in set S_j. If attribute A was chosen the testing attribute (for dividing the current sample multitude). Set up S_{ij} is the sample multitude in subset.

S_j that belonged to C_i. Using attribute A dividing information entropy which was required by current sample sets.

$$E(A) = \sum_{j=1}^{v} \sum_{i=1}^{m} \frac{S_{1j} + S_{2j} + ...S_{mj}}{s} p_{ij} \log_2 p_{ij}. \qquad (3)$$

Here, $p_{ij} = S_{ij} / |S_j|$ is the probability of any data sample in subset S_j belonging to the category C_i.

In this way, using attributes A dividing the corresponding sample set for the current branch node, it's getting the information gain is [16]:

$$Gain(A) = I(S_1, S_2, ..., S_m) - E(A). \qquad (4)$$

In digital relations, we define three values as possible input and output variables. These values include zero as "0", one as "1" and don't care as" x". Also, Concerning to what we defined in above, we encounter with tow concept, classification attributes and attribute values, in ID3 algorithm. To apply ID3 in digital area, we set the amount of classification attribute in this algorithm as three, because in outputs digital table we encounter maximum three different values as possible classification attribute that can occur in this table. Therefore we set the amount of m equal to three in relation (2). As way as described the amount of classification attribute in digital area, now we describe the amount of attribute values as three. Because in inputs digital table we encounter maximum three different values as possible attribute values that can occur in inputs table. Therefore we set the amount of v equal to three in relation (3).

C. Genetic and BDD

Genetic algorithms (GAs) are often used in optimization and machine learning [12]. In many applications they are superior to the classical optimization techniques (e.g. gradient descent). Recently, GAs has successfully been applied to several complicated problems in computer aided design (CAD), like routing, placement, test pattern generation and logic synthesis [13-14]. In this paper to apply GA on OBDD and SBDD, we used the method that introduced in [1] to simulate GA on BDDs. To reach this goal we used the BeDD (V1.01) [15]. This tool supports reordering in BDDs with GA method [15]. To run GA on the BDDs we must set two variables in GA operator. These operators are population size and mutation rate. The population size must be three times larger than maximum inputs number in all examples [1], but In experimental result, we found out to have rational time of iteration, we must set the population size about 1.5 times larger than the average amount of inputs number in all examples. Also, for mutation rate, we used original amount that set in BeDD tool. The mean of original ordering is referring order to input variables according the use of each input on the BDD function sentences. Concerning to what we said above, we set population size parameter as 16 and mutation rate as 0.25 for all examples. When the algorithm run, it produce 16 set of inputs variable in each iteration for BDD. The algorithm save the best result of inputs variable set as first generate in each iteration. The algorithm stops if the best BDD has the same size for a number of consecutive

generations. If we want to reach early stopping, we have to use another operator's code that stop iteration process without satisfying stop condition. The algorithm is really slow, with order $O(2^n)$. This is due to that, a new population of BDD is building up from the scratch for each new generation [1-15].

II. MUTIVATION OF IDGBDD

Concerning to what we described about order problem in BDD, and also importance role of GA to decrease the number of nodes in BDDs, now we can understand that improving GA and make it so useful as a BDD reordering method, is very important for us. To achieve this goal, we use ID3 as a preprocess stage algorithm because first, this algorithm is so simple and powerful, second it take very short time to calculate against the time that consume to perform genetic algorithm stages on each examples. By using ID3, we decrease the node number of GA method, and make it as a convenient method to appliqué in CAD tools. We named this method as IDGBDD. In next section we describe detail's method of performing ID3 on GA and then we present our experimental result that was all carried out on same BDD package.

III. IDGBDD METHOD

In ID3 structure, to build Decision Tree, we use arbitrary and examples table to calculate information entropy and information gain. As we described in 1.2, to carry out ID3 in digital platform, we must use digitals tables. This goal achieves by use the LGSynth91 Benchmark circuits table. To perform ID3 on these tables, we calculate ID3 information algorithm (2) on all of the output functions in each instance circuit. To show detail of this work, we by chose SEX example from the LGSynth91 Benchmark circuits and carry out all process on it. The result of calculation (2) on this example shows in Table I. In this table we show all output information content of the SEX example. After this stage, we chose the output function with the highest information content as a reference output and then perform other steps on those reference output. For example, in Table I, we chose output number 7 as information content. After choosing reference output, we do entropy algorithm (3) for each input variables, careful of this output. Then perform gain algorithm (4) on all of the input variables. The result of performing above stages on SEX example shows on Table I.

After calculating all inputs gain, by using the sorting algorithm we sort all input variables in the order table to use them as a first input variable order. If we have inputs with equal information gain, we sort these inputs consecutive in the order table without considering to primary place of them in inputs table.

To do all these sections on ID3 algorithms, we wrote several C++ programs that read all branch circuits and calculate all ID3 information contents, information gain on those circuits and produce ".BSH" file format for the BeDD as a standard input file.

A sketch of all algorithms is given in psudocode structure at Table II. OutputSelector() ,Id3Algorithm() and BeDD() are the three main procedure of the algorithm which are used for finding the major output, perform ID3 algorithm on inputs and finally feed the result and suggested order to the genetic algorithm as first order. INPUT() procedure is used to fetch the standard input file and make it as standard input file for other stage of the program. Also MakeFile() procedure is used to make the standard input file for BeDD program

TABLE II.
INFORMATION CONTENT (SEX OF LGSYNTH91)

Output	Information Content	Input	Information Gaintalic
1	0.276214	1	0.173483
2	0.453747	2	0.230761
3	0.453747	3	0.206316
4	0.453747	4	0.0055125
5	0.591713	5	0.0919489
6	0.591713	6	0.0919489
7	0.918358	7	0.0976392
8	0.591713	8	0.0553018
9	0.702514	9	0.0204518
10	0.702514	-------	----------
11	0.453747	-------	----------
12	0.276214	-------	----------
13	0.276214	-------	----------
14	0.276214	-------	----------

TABLE I.
SKETCH OF ALL ALGORITHMS

```
INPUT (.PLA file )
    OutNumber=GetOutputNumber();
    InNumber=GetInputNumber();
Examples[InNumber];
While(OutNumber !=Null){
    GetOutputSets[OutNumber];//Set of Outputs
  OutNumber--;
}
 OutputSelector(Examples[],GetOutputSets[],
Attributes);
 Id3Algorithm(Examples[],OutSets[0],
GetOutputSets[0],Attributes);
 MakeFile(); Make bsh file
 BeDD(MakeFile, ID3Order);//Run BeDD
 Procedure           OutputSelector(Examples[],
GetOutputSets[], Attributes)
    Begin
     While(OutNumber!=Null){
OutSets[OutNumber]=InfoGain(GetOutputSets[OutN
umber]);
               OutNumber --;
                }
               Sorter(OutSets[]);#sort
information gain as high to low

   Return   OutSets[0];//highest   information
gain
         END
 Procedure InfoGain(GetOutputSet[OutNumber]);
){
  Begin
 Subs=subset(GetOutputSet[],Attributes);

#divide GetOutputSet[] on basis of Attributes
       Div=(number  in  sub)/(total  number  of
examples);
Gain -=Div*Entropy(Div);#reverse sum of each
sub
        Return Gain;
```

that perform the genetic algorithm. InfoGain() is a procedure that used to calculate the information gain for each stage. This procedure used as a sub function for several stage of psudocode attentive of the pure ID3 algorithm. Entropy() is the sub procedure that perform the basic definition of entropy concept. ReArrange() is the most important sub processor of Id3Algorithm. This sub processor is used for rearrange the input examples according to the attributes and respectively rearrange the output set. Other function like Sorter() and RunBeDD() do what their names explain them. For example Sorter()

```
    }
    END
Procedure Entropy(Div)
    Begin
      {
       log2(Div) = log(Div)/log(2);
       result=log2(Div);
       return result;
     }
    END
Procedure   Id3Algorithm(Examples[],OutSets[0],
GetOutputSets[0],Attributes);
     Begin
      {
While(InNumber !=Null){
[ExamplesDiv[InNumber][
Attributes],GetOutputSetDiv[0][   Attributes]]=
ReArrange(Examples[InNumber],
GetOutputSets[0]);
While(Attributes !=Null){
InfoGain(GetOutputSetDiv[0][ Attributes];
Gain-=Gain;
}
Gain[InNumber]=OutSets[0]-Gain;
}
    Sorter(Gain[]);#sort  gain  as  high  to  low
value
ID3Order(sorter);#  get  place  of  inputs  from
sorter base on their gain
       Return ID3Order;
      }
Procedure BeDD(MakeFile, ID3Order)
    Begin
      {

RunBeDD(MakeFile,ID3Order,ID3NumberOfNodes );
        RunBeDD(MakeFile,
OrdinaryOrder,OrdinaryNumberOfNodes);
         Return
Comprator(OrdinaryNumberOfNodes,
ID3NumberOfNodes);
       }
    END
```

function, sorts the input set as high to low value. Attributes value in all stage of psudocode is equal the "m" value that defined at formula (4).

IV. SIMULATION RESULT

For comparing the effect of using ID3 as a first order motion, and the original order motion, we used BeDD on each instance in both original ordering and ID3 first order offering method. The result of both simulations is shown in Table III. In this table we show iteration value with IT tag. As it appears on table III, use the ID3 as a first order variable motion is effective on the number of nodes and decreases them in many of these examples. We have shown these effects on Fig.2 . However as what we show on Fig. 2, this method is more effective on examples with more nodes with original orders. Table III also have shown that use ID3 method also have most effect on the number of iterations in each instance and occasion to early stopping in the GA algorithm. This event causes the decrease of the GA processing time. These effect also have shown on Fig. 3. Considering the Table III, using ID3 as a pre process method and first order motion, decreases the number of nodes in 57% of the examples. In addition, it causes the decrease of the number of iterations on GA in 71% of the examples. As we can observe, the advantage of using
ID3 as pre process, is that we can decrease the number of iterations without using early stop operators in GA and also by using ID3 we can affect the number of nodes and decrease them. To examine this idea, we applied ID3 on only the short examples of LGSynth91 Benchmark circuits because when we ran the large circuit of this benchmark on BeDD, we encounter large time of simulation as it caused in hanging the system.

This problem happened because of using java for programming in BeDD that uses all of the CPU power and system encounter infinite time.

TABLE III.
COMPARISON OF NODES NUMBER AND ITERATIONS

Name	In	Out	G.A with Original Order		IDGBDD		Improvement Percent(%)	
			Size	IT	Size	IT	Size	IT
Add4m	9	8	208	15	195	12	6.25	20
Alu2	10	8	131	8	111	8	15.3	0
Alu1	12	8	202	13	195	10	3.5	23
Alu2b	10	8	70	12	72	9	-2.9	25
Alu3	10	8	81	9	77	7	4.9	22.2
Alpa	10	12	102	13	102	12	0	7.7
Cm162a	14	5	40	7	34	15	15	-114
Dc2	8	7	66	12	66	11	0	8.3
Dist	8	5	154	17	154	9	0	47
Dk17	10	11	70	9	70	6	0	33.3
Dk27	9	9	27	6	27	6	0	0
F51m	8	8	69	12	69	8	0	33.3
Inc	7	9	82	7	75	7	8.5	0
Log8mod	8	5	67	13	65	9	3	30.8
Misex1	8	7	43	7	39	6	9.3	14
Pcle	19	9	92	8	67	8	27	0
Risc	8	31	80	8	75	6	6.25	25
Sao2	10	4	18	13	18	8	0	38.5
Sex	9	14	51	13	51	6	0	53.8
Squar5	5	8	41	9	39	7	22.2	4.9
X2	10	7	46	6	40	10	13	-66

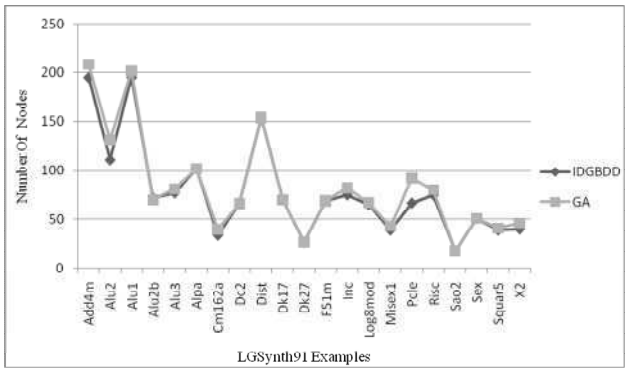


Figure 2. Number of Nodes for G.A with Original Order and G.A With ID3 Order

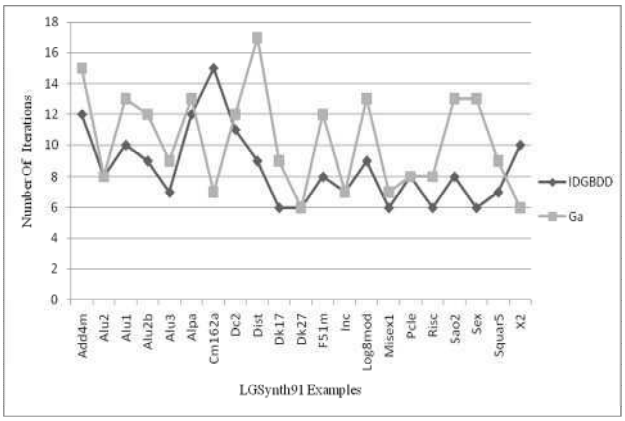


Figure 3. Number of Iterations for G.A with Original Order and G.A With ID3 Order

V. CONCLUSION

In this paper we showed IDGBDD as a heuristic method for decreasing the number of nodes and number of iteration on GA method [1] by applying the ID3 Algorithm as a pre process method. By using the first order variable set that ID3 suggests as first arrangement of input variables in SBDDs, we can improve GA algorithm as one of the most powerful algorithm that can be used as reordering BDDs. The power of using ID3 will show off when we encounter the huge number of iterations in some examples that run with GA. As we indicated, this process decreases both the number of nodes and the number of iteration on this algorithm that make GA as a good method for using in CAD tools.

REFERENCES

[1] Drechsler, R; Becker, B.; Gockel, N.; "Genetic algorithm for variable ordering of OBDDs**".** *IEE Proceedings-Computers and Digital Techniques*. Volume 143, Issue 6, Nov 1996 Page(s):364 – 368.

[2] Shannon, C. E; "The Synthesis of Two-Terminal Switching Circuits". *Bell System Technical Journal*, vol.28, pp.59-98, original text

[3] C. Y. Lee. "Representation of Switching Circuits by Binary-Decision Programs". *Bell Systems Technical Journal*, 38:985–999, 1959

[4] Randal E. Bryant. "Graph-based algorithms for Boolean function manipulation". *IEEE Transactions on Computers*, C-35(8):677–691, August 1986

[5] Hung, W.N.N.; Xiaoyu Song; Aboulhamid, E.M.; Driscoll, M.A. "BDD minimization by scatter search**".** *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. Volume 21, Issue 8, Aug 2002 Page(s): 974 – 979

[6] Hung, W.N.N.; Xiaoyu Song." BDD variable ordering by scatter search*". International Conference on Computer Design, 2001. ICCD 2001*. Volume , Issue , 2001 Page(s):368– 373

[7] Drechsler, R.; Drechsler, N.; Gunther, W." Fast exact minimization of BDDapos" *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume 19, Issue 3, Mar 2000 Page(s):384 - 389

[8] Quinlan, J.R."Introduction of decision trees. Machine Learning", 1986.1(1):81-106

[9] Quinlan, J. R. " Learning efficient classification pocedures and their application to chess and games". In Michalski, Carbonell and Mitchell Eds. *Machine Learning: An Aritificial Intelligence Approach, Morgan Kaufinann*, 1983.463-482

[10] Wu Sen, Wu Ling-yu, Long Yu, Gao Xue-dong "Improved Classification Algorithm by Minsup and Minconf Based on ID3". *International Conference on Management Science and Engineering, 2006. ICMSE* apos; 06. Volume , Issue , 5-7 Oct. 2006 Page(s):135 – 139

[11] Yan Li. Fa-Chao Li. "A Kind of Inductive Learning Classification Algorithm Based on Statistics Rule". In *International Conference on Machine Learning and Cybernetics*. Aug. 2006 . Page(s):1308 – 1313

[12] DAVIS, L. "Handbook of genetic algorithms" (van NostrandReinhold, New York, 1991)

[13] Ho, M.C; Leung, S.; Kurokawa, H.; Choy, O.C; "Digital logic synthesis using genetic algorithms" *Second International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications***.** 2-4 Sep1997On page(s): 296-301

[14] RUDNICK, E.M., PATEL, J.H., GREENSTEIN, G.S., NIERMANN, T.M. "Sequential circuit test pattern generation in a genetic algorithm framework". *Design automation conference,* 1994, pp. 698-704

[15] http://www.mitzanu.ro/java/index.html ,23/9/2009.