Proceedings of the 3rd Nordic Workshop on GA (3NWGA)

18-22 August 1997, Helsinki, Finland

DRAFT version of Jan. the 22nd 1997

November 22, 1999

Chapter 1

Decomposition of r-valued functions and GA

Tatiana Kalganova, **Natalia Strechen**,
Belarussian State University of Informatics and Radioelectronics
Laboratory of Image Processing and Pattern Recognition,
P. Brovky St., 6, Minsk, Republic of Belarus
phone +375-172-491 981, fax +375-172-495 106
E-mail: `pottosina@risq.belcaf.minsk.by` or
`jack@expert.belpak.minsk.by`,

Abstract. This paper presents a variable partition algorithm which combines the quasi-reduced ordered multiple-terminal multiple-valued decision diagrams and genetic algorithms (GAs).

The algorithm is better than the previous techniques which find a good functional decomposition by non-exhaustive search and expands the range of searching for the best decomposition, providing the optimal subtable multiplicity.

The possible solutions are evaluated using the gain of decomposition for a multiple-output multiple-valued logic function.

The distinct feature of GA is the possible solutions being coded by real numbers. Here, the simplex-based crossover is proposed to use for the recombination stage of GA. It permits to increase the GA coverage.

Keywords: disjoint decomposition, multiple-valued decision diagrams, genetic algorithm, simplex-like crossover

---

The complete title of this article: Genetic algorithm approach to find the best input variable partitioning.

1.1 Introduction

Functional decomposition of switching functions has applications in binary and multiple-valued (MV) circuit design, machine learning, knowledge discovery from data bases and testing the logic circuits. The many variable functions are decomposed into sub-functions with fewer variables, expressed by two-level or multi-level logic structures. In recent years, increasing attention has been paid to decomposition methods for single-output and multiple-output multiple-valued functions. Thus, Abugharieh and Lee [7], [8] had been generalized the decomposition algorithm for binary functions of Shen' s, Mc.Kellar' s and Weiner' s to r-valued functions with $r > 2$. Their method is of not much use in machine learning or circuit implementation of MV systems, because only simple disjunctive decompositions of single-output functions have been discussed. Some decomposition algorithms for r-valued functions have been investigated in [12], [13], [5].

A classical decomposition method contains the two following main phases: 1) Finding the partition of X that provides the minimum number of sub-functions, where X is the set of input variables. 2) Encoding the sub-functions using the minimum number of r-valued variables. The optimization of these phases leads to decreasing the size of MV combinational circuit. In this paper we search the best partition of X. Note that the success of functional decomposition depends on the partition of X. Therefore the problem to find the best partition of X is actual and requires the further researches especially for the single-input and multiple-output r-valued functions.

The search for a fast algorithm to determine the best decomposition of switching (binary) functions is studied by T. Sasao [14] . He presented an algorithm based on testing the necessary condition of function decomposability, using reduced ordered binary decision diagrams. By performing of this algorithm we can intend the best partition of X with miminum column multiplicity. Generalization of this algorithm to a multiple valued logic had been done in [9]. Note that this algorithm is applied to a single-output binary function and does not examine the dependence of the variable permutations on the column multiplicity for given partition of X. This drawback can be removed by combined using the multiple-terminal multiple-valued decision diagrams (MTMDD) and GA.

The objective of this research is to use such optimization technique as GA combined with MTMDD to search the best partition of X for a multiple-output r-valued function. The fitness function determines the best partition for specific variable's permutation.

Thus, this paper addressed the problem to find the best decomposition of multiple-output functions without considering all decomposition variants. For this purpose the GA is applied. The distinct feature of this algorithm is that the possible solutions are coded by real numbers. Here, the simplex-based crossover is proposed to use it on the recombination stage of GA. It permits to increase the GA convergence.

The rest of the paper is organized as follows. Section 2 introduces multiple-valued decision diagrams and disjoint disjunctive decomposition. We then discuss the GA and its properties for input variable partition problem (Section 3). The experimental results are considered in Section 4. Section 5 concludes the paper.

1.2 Multiple-valued decision diagrams and disjoint disjunctive decomposition

We discuss Ashenhurst-like decomposition $F(X) = G(h_1(X_1), \ldots, h_k(X_1), X_2)$ for multiple-output r-valued function $F(X) = \{f_1(X), f_2(X), \ldots, f_m(X)\}$ (Fig. 1.1) [7], [12].

The set X of input variables is partitioned into two sets: *free variables* X_1 are direct inputs to the circuit H and bound variables X_2 are inputs to the circuit G of the disjoint disjunctive decomposition. The process of finding sets X_1 and X_2 is called *input variable partitioning.* A multiple-output r-valued function *F(X)* is said to have a *disjoint generalized disjunctive decomposition* regarding X_1 if the one-output functions $h_1, h_2, \ldots, h_k$ and m-output function $G = (g_1, g_2, \ldots, g_m)$ such that $F(X) = G(h_1(X_1), h_2(X_1), \ldots, h_k(X_1), X_2)$ exist. The projection of *F(X)* over $X_1 = \tau$, $F(\tau, X_2)$ is the value of *F(X)* evaluated with $X_1 = \tau$.

For disjoint disjunctive decomposition a multiple-output r-valued function $F(X) = \{f_1(X), f_2(X), \ldots, f_m(X)\}$ is represented as Karnaugh map with X_1 variables as columns and X_2 variables as rows. Columns corresponding to the specific input combination of X_1 and different m r-valued functions generate *subtable multiplicity.* The *subtable multiplicity index* μ is the number of different types of subtable patterns. Note that the subtable multiplicity defines the type of disjunctive decomposition. An r-valued function *F(X)* is said to have a simple disjunctive decomposition if μ is less or equal to r [7], [6], [10], [11] and a generalized disjunctive decomposition if μ is less or equal to r^{n_1}, where n_1 is the number of elements in free set X_1 [10], [11]. The gain of the decomposition for a given partition is determined by *gain index* γ calculated as $\gamma = min(r^{n_1}, r^{r^{n_2}})/\mu$, where n_1 and n_2 are the number of elements of the free and bound variables respectively.

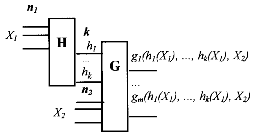

Figure 1.1: Circuit diagram of $F(X) = G(h_1(X_1), \ldots, h_k(X_1), X_2)$

x_3 \ x_1x_2	00	01	02	10	11	12	20	21	22
0	1 2	0 1	1 2	0 0	0 1	0 0	1 2	0 1	0 1
1	0 0	1 1	0 0	2 1	1 1	2 1	0 0	1 1	1 1
2	0 2	2 0	0 2	1 1	2 0	1 1	0 2	2 0	2 0
h_1	0	1	0	2	1	2	0	1	1

(a) $f_1(x_1, x_2, x_3); f_2(x_1, x_2, x_3)$

x_2 \ x_1	0	1	2
0	0	2	0
1	1	1	1
2	0	2	1

(b) $h_1(x_1, x_2)$

x_3 \ h_1	0	1	2
0	1 2	0 1	0 0
1	0 0	1 1	2 1
2	0 2	2 0	1 1

(c) $g_1(h_1, x_3); g_2(h_1, x_3)$

Table 1.1: Decomposition of $F(X) = G(h_1(x_1, x_2), x_3)$

Example. Consider the 2-output 3-valued 3-variable function $F(X) = \{f_1, f_2\}$ described in Table 1.1(a). Since the subtable multiplicity is 3, which equals to r, the function is decomposable with a bound set $\{x_1, x_2\}$ and a free set $\{x_3\}$. The function *F(X)* have a simple disjunctive decomposition. Let the mapping of the three destinct subtables be as shown in Table 1.1(a). The truth table of function $h_1(x_1, x_2)$ is illustrated in Table 1.1(b). The functions $g_i(h_1, x_3), i = 1, 2$ are constructed such as shown in Table 1.1(c).

Any n-input m-output r-valued function can be represented by *quasi-reduced ordered multi-terminal multiple-valued decision diagram* (OMTMDD) [15], [4]. Each internal node of this diagram can have not more than r children, and input variables appear in a fixed order in all paths of the graph and no variable does more than once in a path. The terminals are m-bit r-valued vectors. An assignment of values to the variables corresponds to a path that concludes on a vector of m-bits corresponding to m output r-valued functions. An OMTMDD is *a quasi reduced OMTMDD* if every path from the root to the terminal nodes involves all variables, and has no isomorphic sub-graphs in the same level.

The *path function* of a node in a MTMDD is an r-valued input 2-valued output function that represents the conditions for existance of a path from root to the node. The *sub-path function* of a node in a MTMDD is an 2-valued input r-valued output function and represents the conditions for existance of the path from a node to the terminal node. A sub-path function of a node is the projection of *F(X)* over $X_1 = \tau$, $F(\tau, X_2)$.

The decomposition of an n-input m-output r-valued function can be estimated by OMTMDD. Fig. 1.2 shows the general structure of disjoint decomposition.

First level represents the projections of $F(X)$ over $X_1 = \tau$, $F(\tau, X_2)$, where $\tau \in \{0, 1, \ldots, r^{n_1 - 1}\}$. The values of $F(\tau, X_2)$ are formed on the output of first level. Each function $F(\tau, \eta)$, $\eta \in \{0, 1, \ldots, r^{n_2 - 1}\}$ is formed on the output of second level. It is described by the sub-graph of second level. These MTMDDs share sub-graphs. There are not more than r^{n_2} different MTMDDs. Existence of the MV Ashenhurst-like decomposition for a given bound set X_2 and free set X_1 can be checked by using the following Theorem.

Theorem. Let X_1, X_2 be a partition of X. Suppose that the quasi-reduced ordered MTMDD of $F(X)$ is partitioned into two disjoint blocks. Let t be the number of nodes in the lower block that are adjacent to the boundary of the two blocks, and μ be the subtable multiplicity. Then, $t = \mu$.

Example. Fig. 1.3 shows three different partitions:

1. $X_1 = \{x_1, x_2\}, X_2 = \{x_3, x_4, x_5\}$.
2. $X_1 = \{x_1, x_2, x_3\}, X_2 = \{x_4, x_5\}$.
3. $X_1 = \{x_1, x_2, x_3, x_4\}, X_2 = \{x_5\}$.

By Theorem, the subtable multiplicities for these decompositions are five, four and three, respectively. The first partition gives the generalized disjunctive decomposition, the second and third partitions are the simple disjunctive decomposition. The decomposition gain index γ for the first partition is calculated as follows: $\gamma_1 = min\left(3^2, 3^{3^3}\right)\backslash 5 = 1.8$. γ_2 andγ_3 for second and third partitions of input variables are 2.25 and 9 respectively.

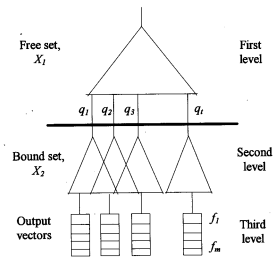

Figure 1.2: General structure of disjoint decomposition using a quasi-reduced OMTMDD

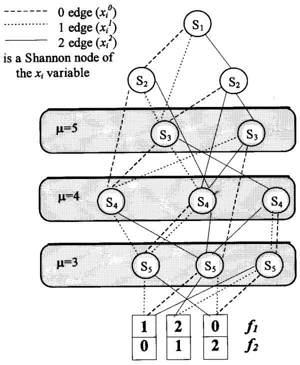

Figure 1.3: Determing the column multiplicity using OMTMDD for 5-variable 2-output 3-valued function.

1.3 Genetic algorithm for input variable partitioning

Analogies with natural processes help in understanding complex systems and suggest new methods for solving problems [3]. The natural genetic and evolution process are the prototypes of GAs. These algorithms maintain populations of individuals that are called chromosomes and represent potential solutions of optimization problem. A survival of the fittest strategy is implemented by a selection mechanism. Here, the fittest chromosomes are chosen as the potential solutions for the next population. Genetic recombination of the selected high-fittest chromosomes in evolution produce offsprings (new chromosomes) for the new generation. The recombination is effected through the two genetic operators of crossover and mutation. First we map a solution to a string. Then we discuss the basic operators applied in searching for an optimum partition of X.

1.3.1 String encoding

The variable partition problem lends itself to a straightforward representation by means of genetic algorithm techniques. Indeed, as required for a classical genetic algorithm approach, a partition of X is easily encoded by a string of integer numbers of length n, where n is the number of variables on which the r-valued function depends. But we solve the coding problem by nonrtaditional way. Let $\mathbf{b}=<b_1 b_2 \cdots b_n>$ be the chromosome containing n genes. Each gene b_i of chromosome $\mathbf{b}$ takes a real value from the domain $[0 \cdots 1]$. The less value of b_i corresponds to the variable with the less index. For an n variable r-valued function, the search space (the set of all possible partitions of X) has $2^n - n - 2$ elements.

Example. For 3-valued 5-variable 2-output function a possible solution can be represented as $\{x_2 \ x_1 \ x_3 \ x_5 \ x_4\}$. The corresponding string is the sequence $\{0.25 \ 0.01 \ 0.43 \ 0.89 \ 0.56\}$.

1.3.2 Fitness function

The fitness function depends on the decomposition gain index (which defines the type of decomposition and estimates the decomposition success. The specific properties of the gain are shown in Table 1.2. As we can see the partition with maximal gain is useful. In our genetic algorithm the gain of the decomposition determines the evaluation function. For example, for an 3-valued 2-output 5-variable function shown in Fig. 1.3 the fitness function is $max(1.8, \ 2.25, \ 9) = 9$.

1.3.3 Crossover

The classical crossover operator is based on the processing of the two parents (chromosomes), their cutting and paired different part of their chunk. Then these offsprings are replaced replace the parents in the breeding pool after all crossover operations have been completed. We use the simplex-like crossover which had been proposed and investigated in [1]. It operates with three randomly selected chromosomes (parents) and generates an offspring. The sense of crossover operator is preserved: the obtained chromosomes have better values then parents. This distinctive feature is essential convention of GA convergence. In case when a crossover operator does not examine in algorithm, the mutation become the main operator on which depends the search of the optimal solution. It means that the genetic algorithm is degenerated into random search.

The crossover operator is built by analogy with the simplex-method [1]. One application of the this crossover operator takes three randomly selected chromosomes and creates an offspring in the following way:

1. Three chromosomes **x, y, z** are randomly selected from current population ($\mathbf{x}<=x_1 x_2 \cdots x_n>$, $\mathbf{y}=<y_1 y_2 \cdots y_n>$, $\mathbf{z}=<z_1 z_2 \cdots z_n>$), where n is the number of variables on which function depends;

Value of γ	*Condition of decomposition existence*	*Notes*
$r < \gamma$	$k < n_1$	This condition determines the existence of generalized disjunctive decomposition.
$1 < \gamma \leq r$	$k = 1$	This condition defines the existance of simple disjunctive decomposition.
$\gamma = r^{n_1}$	k is absent	The value of function *f(X)* does not depends on the variables of X_1. In this case we can design circuit implemented only r-valued function $f(X_1)$. The number of inputs in MV circuit is reduced.
$\gamma = r^{n_1 - 1}$	$k = 1$	The "don't cares" inputs for the design of G are absent. This is a simple disjunctive decomposition.
$1 < \gamma < min(r^{n_1}, r^{r^{n_2}})$ but ratio $\frac{\gamma}{r}$ is real	$k < n_1$	The number of the outputs of H cannot be reduced. However, we can design H such that it does not produce particular output combinations. Such combinations can be used as the "don't cares" for the design of G.
$1 < \gamma < min(r^{n_1}, r^{r^{n_2}})$ but ratio $\frac{\gamma}{r}$ is integer	$k < n_1$	The same like in previous row, but output combinations of H applied as the "don't cares" for the design of G are absent.
$\gamma = 1$	$k = n_1$	Because the purpose of the functional decomposition is to obtain the subfunctions with fewer variables, this decomposition is not applied in practice.
$0 < \gamma \leq 1$	$k = n_1$	The number of outputs of H can be smaller than n_1. Because the goal of the functional decomposition is to get the subfunctions with fewer variables, this decomposition is not used in practice.

Table 1.2: The specific properties of gain index γ

2. The genes of offspring are calculated by following equation:

$$p_i = \frac{x_i + y_i}{2} + \sigma\left(\frac{x_i + y_i}{2} - z_i\right) \tag{1.1}$$

   where x_i, y_i, z_i are the i-th genes of three initial chromosomes arranged in decreasing order of fitness-function, $i = \overline{1, n}$, σ is a coefficient called as crossover force;

3. New chromosome $\mathbf{p}=< p_1 p_2 \cdots p_n >$ is formed.

The number of parents is determined by α coefficient called as crossover probability. Note that a triad of chromosomes produces only one child.

The analysis of local extreme is envisaged in given algorithm. The sense of this procedure is as follows. If the fitness-function from new generation does not increase, than a decision about reaching the global extreme or about falling into a "trap" of a local extreme is made. The additional procedures, for example "shaking" of population, are used in order to go out from a local extreme [16]. However, such an approach requires high temporal expenditures (it needs to modeling the evolution for hundred populations). Instead of "shaking" of population two procedures are used. One of them is to use "intense" mutation of current population. In given case, parents as well as offsprings mutate. The other one is based on changing the selection operator. The new population is formed from the fittest chromosomes as well as from chromosomes changing by random way. Experimental investigations show that the suggested procedure allows to go out from the local extreme and to continue the search of the global extreme.

1.3.4 Mutation

The mutation operator transforms a chromosome by means of randomly changing of one or some genes' values in the range [0..1]. The mutation is used for a part of chromosomes from new population. The number of mutated chromosomes is defined by β coefficient called as mutation probability.

The size of population is increased at the crossover and mutation execution. Current population contains both parents and offsptings. Decreasing the size of population is executed by selection operator. It forms new population by the choice of the fittest chromosomes.

1.4 Experimental results

The GA success depend on the correct choice of the size of populations, crossover probability and the number of populations. These GA parameters are selected as in [1], taking into consideration the computer characteristics such as the size of main memory and CPU time. The results obtained by applying GA to randomly

MV PLA circuit	n	m	*Initial size of MV PLA*	*Final size of combinational circuit*	
				without GA	*with GA*
1	12	2	1824	1168	948
2	12	3	2301	1560	1206
3	12	4	3160	2726	2069
4	10	2	768	662	524
5	10	3	2176	1562	1104
6	8	2	1274	678	576
7	8	3	1512	952	844

Table 1.3: Table comparing the results of circuit design with and without GA implementing the 3-valued functions

generated 3-valued circuits are tabulated in Table 1.3. The NOR-TSUM PLA proposed by Pelayo and etc. is used to synthesize the combinational multiple-valued circuit [2]. In all cases, the results obtained are better than those obtained without using the GA. q The dependance of the time needed to find the optimal solution on the crossover probability is shown in Fig. 1.4. Note the execution time of GA is increased at the decreasing the crossover probability.

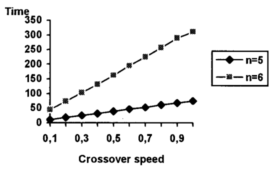

Figure 1.4: Dependence of crossover probability on time execution

1.5 Conclusion

A new technique to optimize the partition of X which leads to reducing the size of synthesized MV combinational circuit is presented. This technique is that the GA and the MTMDDs are combined. Each chromosome is evaluated using a decomposition gain index γ. Two techniques are incorporated that allows to obtain the global optimum. A permutation of input variables is executed in GA. The result of similar algorithms without permutation (without GA) does not gives the optimum.

1.6 Bibliography

See chapter bibliography at the end of this proceedings.

Contents

List of Figures

List of Tables

Bibliography

[1] Kochergov E. Optimization of decision acceptance in person identification system due to handwritten. In *Proceedings 2nd International Conference. on the New Information Technologies in Education*, volume 1, pages 371–380. Minsk, Belarus, 1996.

[2] Pelayo F.J., Prieto A., lloris A., and Ortega J. Cmos current-mode multiple-valued pla' s. In *IEEE Transactions*, pages 434–441, 1991.

[3] Maini H., Mehrotra K., Mohan C., and Ranka S. Knowledge-based nonuniform crossover. In *Comlex Systems*, volume 8, pages 479–488, 1994.

[4] Shin ichi Minato. A graph-based representations of discrete functions. In *Proceedings of the Reed-Muller Workshop*, pages 1–10, 1995.

[5] Perkowski M. A new representation of strongly unspecified functions and its application to multi-level and/or/exor synthesis. In *Proceedings of the Reed-Muller Workshop*, pages 143–151, 1995.

[6] Rakov M.A., editor. *Specialized multiple-valued analyzers.* Naukova dumka, Kiev, 1977.

[7] Abugharieh S.B. and S.C. Lee. A fast algorithm for the disjunctive decomposition of m-valued functions. part 1. the decomposition algorithm. In *IEEE Transactions on Computers*, pages 118–125. IEEE Computer Society Press, Los Alamitos, CA, 1995.

[8] Abugharieh S.B. and S.C. Lee. A fast algorithm for the disjunctive decomposition of m-valued functions. part 2. time complexity analysis. In *IEEE Transactions on Computers*, pages 126–131. IEEE Computer Society Press, Los Alamitos, CA, 1995.

[9] Kalganova T. Combinational multiple-valued circuit design by generalized disjunctive decomposition. In *Proceedings of the European Conference on Circuit Theory and Design (ECCTD' 97).* Budapest, Hungary, 1996.

[10] Kalganova T. Functional decomposition methods for multiple-valued logic functions and its system. In *Proceedings of the 3rd International Conference on Application of Computer Systems*, pages 75–82. Szczecin, Poland, 1996.

[11] Kalganova T. The studding of the functional decomposition methods for r-valued logic functions in the logic design courses. In *Proceedings of the 2nd International Conference. on the New Information Technologies in Education*, volume 2, pages 150–158. Minsk, Belarus, 1996.

[12] Luba T. Decomposition of multiple-valued functions. In *Proceedings of the 25th international symposium on the multiple-valued logic*, pages 256–261. IEEE Computer Society Press, Los Alamitos, CA, 1995.

[13] Luba T., Mochocki M., and Rybnik J. An implementation of decomposition algorithm ans its application in information systems analysis and logic synthesis. In *Proceedings of the international Workshop on Rough Sets and Knowledge Descovery*, pages 487–498, 1993.

[14] Sasao T. Fpga design by generalized functional decomposition. In *Logic Synthesis and Optimization*, pages 231–258. Kluweer Academic Publishers, 1993.

[15] Sasao T. and Butler J.T. A method to represent multiple-output switching functions by using multi-valued decision diagrams. In *Proceedings of the 26th international symposium on the multiple-valued logic*, pages 248–254. IEEE Computer Society Press, Spain, 1996.

[16] Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlar, Berlin, Heidelberg, 1992.

COPYRIGHT STATEMENT

As the first author of the paper titled: Genetic algorithm approach to find the best input variable partitioning.