Genetic Algorithms Synthesis of Finite State Machines

Andrey Popov and Krasimira Filipova

andrey.popov @ gmx.net, kfilipova @ abv.bg

***Abstract*— Genetic Algorithms (GA) are stochastic, non-derivative optimization method. They use populations of acceptable solutions (genes) of the given problem, which evolve toward optimum. The genes in standard GA are Boolean vectors. When in synthesis of finite state machine are used JK and RS flip-flops there are undefined variables in the activation signals. When the finite state machine is of high order Quine-McClusky method is used, which requires exact values of the variables. At this stage the GA are used to find the optimal set of variables, in terms of simplifying the description.**

***Index Terms*— Genetic Algorithms, Multi-Objective Optimization, Finite State Machines**

I. INTRODUCTION

In the synthesis of finite state machines (FSM) and Boolean logic JK and RS flip-flops are often used, These flip-flops have a set of undefined input signals, which does not affect their action. Table I shows that necessary input signals for RS and JK flip-flops. Here $c_1, c_2, b_1, \ldots b_4$ denote input signals, which does not influence the switching and could be chosen appropriately for minimizing the automat description. When the number of inner states is small Karnaugh maps could be used for determination of those signals, where as when the number of states is bigger Quine-McCluskey method has to be used. This method requires that values of c and b to be given. In [1] is shown that one way of handling this problem is using D and T flip-flops instead of JK and RS.

TABLE I

FREE VARIABLES BY JK AND RS FLIP-FLOPS

$Q(t)$	$Q(t+1)$	R	S	J	K
0	0	c_1	0	0	b_1
0	1	0	1	1	b_2
1	0	1	0	b_3	1
1	1	0	c_2	b_4	0

In this paper we will introduce another approach based on multi-objective Genetic Algorithms (GA) optimization. The algorithm will be used to find the best set of signals c and b, comparing the different sets by the results of Quine-McCluskey minimization.

II. GENETIC ALGORITHMS

Genetic Algorithms are member of the group of Evolutionary algorithms. Evolution is a phenomenon of adapting to the environment and passing genes to next generations. In "On the Origin of Species by Means of Natural Selection", Charles Darwin did not know about the underlying genetics, but he identified three basic principles driving natural evolution: reproduction, natural selection, and diversity of individuals, maintained by variations from one to the next generation. These features of natural evolution have found entrance to a broad class of evolutionary algorithms that mimic biological evolution and natural selections.

GA owe their name to an early emphasis on representing and manipulating individuals at the level of genotype instead of phenotype representation. In Holland's original work (1975), GA were proposed to understand adaptation phenomena in both natural and artificial systems.

GA are stochastic, non-derivative optimization method. At every iteration, there is a set of possible solutions (individuals), which form the population. The individual represents a chromosome with genes. In standard GA genes are Boolean, but now there are also other types (real number, string) which are also GA.

The block scheme of GA consists of three main operations (Fig. 1): selection, recombination and mutation. In most GA the first population is randomly generated.

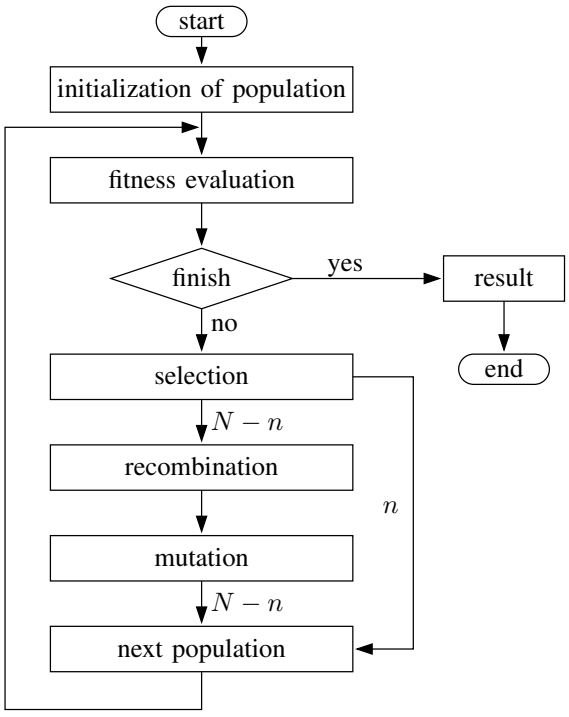

Fig. 1. Structure of GA

A. Fitness evaluation

gives quantity measurement of current set of optimization parameters (individual). This is the value of the function, which is minimized or maximized.

B. Termination

of the optimization process in GAs, most commonly is by defined number of iteration. Since GA is a stochastic algorithm, other stop criteria are dangerous, because optimization time for global search could not be feasible.

C. Selection

is a process of in which fitter individuals are selected to reproduce offspring for the next generation. In the terms of optimization, the evaluation of the fitness means evaluation of the function we are minimizing or maximizing. The purpose of the selection is to find the individuals with best fitness function from one hand, and from other to preserve the diversity of the population. As already noted GA depends on stochastic operators, which do not guarantee a monotonic improvement in objective function. To ensure a monotonic improvement, elitist strategy is used - some of the best individuals are copied into the next generation without applying any evolutionary operators.

D. Recombination

is a process in which new individuals are generated by exchanging features of the selected parents with the intent of improving the fitness of the next generation. This process is sometimes called crossover. This kind of crossover operators include one-point, two-point and multipoint. In the one-point crossover for the bitstring representation, the bits are swapped between the two parents in segments at a random point of a chromosome in between the bits. The following is an example of crossover, where P_1 and P_2 are parent individuals and C_1 and C_2 are child individuals.

$$\begin{array}{ll} P_1 = [x_1\ x_2\ x_3\ x_4\ x_5] & P_2 = [y_1\ y_2\ y_3\ y_4\ y_5] \\ C_1 = [x_1\ x_2\ x_3\ y_4\ y_5] & C_2 = [y_1\ y_2\ y_3\ x_4\ x_5] \end{array}$$

E. Mutation

is an operator, which function is to keep diversity of a population and promote the searching in the solution space that cannot be represented by the strings of the present population. In bit-string GA mutation makes a random change of a gene in random chosen chromosome of the population. Example of mutation in a single gene is

$$P = [1\ 0\ 0\ 1\ 0\ 1]; \quad C = [1\ 0\ 1\ 1\ 0\ 3]$$

F. Offspring

population is formed by gathering n elitist individuals and individuals applied to the genetic operations $(N - n)$. This new population is put again to evaluation.

III. MULTI-OBJECTIVE OPTIMIZATION PROBLEM

Real world engineering design problems are usually characterized by the presence of many conflicting objectives. Therefore, it is natural to look at the engineering design problem as a multi-objective optimization problem (MOP). A general multi-objective design problem is expressed by equation (1), where $f_i(x)$ is the i^{th} objective function, x_j is j^{th} optimization parameter and $S \in \Re^n$ is the solution or parameter space.

$$\begin{aligned} &\min_{x \in S} F(x) \\ &F(x) = \begin{bmatrix} f_1(x) & f_2(x) & \dots & f_k(x) \end{bmatrix}^T \\ &x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix} \end{aligned} \tag{1}$$

GA for MOP [2] can be categorized as plain aggregating, population-based non-Pareto and Pareto-based approaches. Our goal is to find such combination of optimization parameters that we minimize all objective functions (F^*). In most cases this is solution is utopian and we could only have individual minima of each objective function (f_i^*) (Fig. 2).

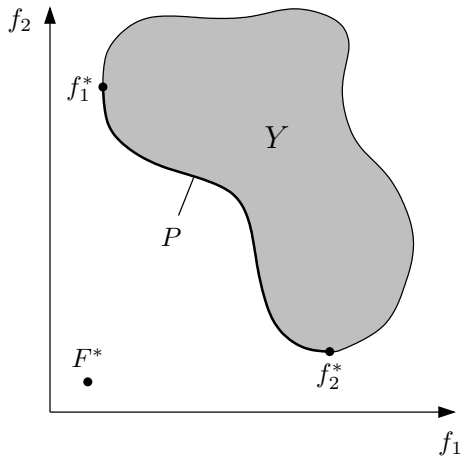

Fig. 2. Solution space and Pareto front

One property commonly considered as necessary for any candidate solution to the multi-objective problem is that the solution is not dominated. The Pareto set consists of solutions that are not dominated by any other solutions. The space in $\Re^k$ formed by the objective vectors of Pareto optimal solutions is known as the Pareto optimal frontier P. (Fig. 2).

It is clear that any final design solution should preferably be a member of the Pareto optimal set. The user should make the final choice of the optimal parameter set within all optimal solutions in Pareto set.

IV. SOFTWARE REALIZATION

As mentioned above the software is realized in MATLAB. MATLAEj is chosen because of its many toolboxes oriented toward different engineering fields, wide usage and easy

syntax. Those toolboxes allow simulation of the final state machines and control systems as well as their interconnection. The function for Quine-McCluskey minimization is the same used in [1] (the code could be downloaded from Internet).

The used GA is multi-objective, Pareto-based with standard Pareto set search and without limitations of the number of Pareto-set solutions.

V. NUMERICAL EXAMPLES

A. Example 1

Consider the automata example given in [1]. Table II shows the automat switching table. The coding of the inputs (x) and outputs (y) is given in Table III and the coding of the internal states (a) in Table IV.

TABLE II
AUTOMATA SWITCHING

	a_0	a_1	a_2	a_3	a_4	a_5	a_6
x_0	a_1	a_2	a_2	a_1	a_5	a_2	a_1
	y_0	y_0	y_0	y_0	y_0	y_0	y_1
x_1	a_0	a_0	a_3	a_4	a_0	a_6	a_0
	y_0	y_0	y_0	y_0	y_0	y_0	y_1

TABLE III
INPUT AND OUTPUT CODING

	x
x_0	0
x_1	1

	y
y_0	0
y_1	1

TABLE IV
INPUT CODING

	Q_1	Q_2	Q_3
a_0	0	0	0
a_1	0	0	1
a_2	0	1	0
a_3	0	1	1
a_4	1	0	0
a_5	1	0	1
a_6	1	1	0

The coding table when JK flip-flops are chosen is chosen in Table V and VI.

Combinations equal to 7 and 15 are not allowed (not reachable), so we could use them in the minimization.

Results achieved with Karnaugh maps are shown below.

$$J_1 = xQ_2Q_3 \qquad K_1 = Q_2 \bigvee x\bar{Q}_3 \bigvee \bar{x}Q_3$$
$$J_2 = \bar{x}Q_3 \bigvee Q_1Q_3 \qquad K_2 = Q_3 \bigvee Q_1$$
$$J_3 = \bar{x}\bar{Q}_2 \bigvee \bar{x}Q_1 \bigvee x\bar{Q}_1Q_2 \qquad K_3 = \bar{Q}_2 \bigvee x$$
$$y = Q_1Q_2$$

When applied to genetic algorithm minimization the following objective functions are considered:

- f_1 = Number of disjunctions
- f_2 = Total number of elements in disjunctions
- f_3 = Total number of inverted elements

TABLE V
REALIZATION WITH JK FLIP-FLOP

		$Q(t)$			$Q(t+1)$			
N	x	Q_1	Q_2	Q_3	Q_1	Q_2	Q_3	y
0	0	0	0	0	0	0	1	0
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	0
3	0	0	1	1	0	0	1	0
4	0	1	0	0	1	0	1	0
5	0	1	0	1	0	1	0	0
6	0	1	1	0	0	0	1	1
7	0	1	1	1	not defined combination			
8	1	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0
10	1	0	1	0	0	1	1	0
11	1	0	1	1	1	0	0	0
12	1	1	0	0	0	0	0	0
13	1	1	0	1	1	1	0	0
14	1	1	1	0	0	0	0	1
15	1	1	1	1	not defined combination			

TABLE VI
REALIZATION WITH JK FLIP-FLOP

	flip-flop 1		flip-flop 2		flip-flop 3		
N	J_1	K_1	J_2	K_2	J_3	K_3	y
0	0	d_0	0	c_0	1	b_0	0
1	0	d_1	1	c_1	b_1	1	0
2	0	d_2	c_2	0	0	b_2	0
3	0	d_3	c_3	1	b_3	0	0
4	d_4	0	0	c_4	1	b_4	0
5	d_5	1	1	c_5	b_5	1	0
6	d_6	1	c_6	1	1	b_6	1
7	d_{7J}	d_{7K}	c_{7J}	c_{7K}	b_{7J}	b_{7K}	e_7
8	0	d_8	0	c_8	0	b_8	0
9	0	d_9	0	c_9	b_9	1	0
10	0	d_{10}	c_{10}	0	1	b_{10}	0
11	1	d_{11}	c_{11}	1	b_{11}	1	0
12	d_{12}	10	c_{12}	0	b_{12}	0	0
13	d_{13}	0	1	c_{13}	b_{13}	1	0
14	d_{14}	1	c_{14}	1	0	b_{14}	1
15	d_{15J}	d_{15K}	c_{15J}	c_{15K}	b_{15J}	b_{15K}	e_{15}

The cost values for the J_3 obtained with Karnaugh will be $f_1 = 3$; $f_2 = 7$ and $f_3 = 4$.

The minimization problem is the same as in 1, where x is the set of decision variables d, c, b or e. For example when applying the design for J_3, x would be b_1, b_3, b_5, b_{7J}, b_9, b_{11}, b_{12}, b_{13} and b_{15J}

The results obtained both by minimization with and without objective function f_3 are the same and are given in Table VII.

TABLE VII
OPTIMIZATION RESULTS

input	f_1	f_2	f_3	activation function
J_1	1	3	0	$J_1 = xQ_2Q_3$
K_1	3	5	2	$K_1 = Q_2 \bigvee x\bar{Q}_3 \bigvee \bar{x}Q_3$
J_2	2	4	1	$J_2 = \bar{x}Q_3 \bigvee Q_1Q_3$
K_2	2	2	0	$K_2 = Q_3 \bigvee Q_1$
J_3	3	7	4	$J_3 = \bar{x}\bar{Q}_2 \bigvee \bar{x}Q_1 \bigvee x\bar{Q}_1Q_2$
K_3	2	2	1	$K_3 = \bar{Q}_2 \bigvee x$

It could be easily noticed that with those objective functions results obtained from Karnaugh maps and genetic algorithm minimization with Quine-McCluskey method are

exactly the same.

All minimization for J are made with 10 generations, where as for K are made with 20 generations. All runs are with 20 individuals in population. When minimizing K_3 three runs were made and the following activation logics ware achieved:

1) $K_3 = \bar{Q}_2 Q_3 \bigvee \bar{x} Q_1$
2) $K_3 = \bar{Q}_2 \bigvee x Q_3$
3) $K_3 = \bar{Q}_2 \bigvee x$

Since GA are based on stochastic operations, there is no repeatability of the results. This means that it is not guaranteed that the same results will be achieved on every run.

Assuming that we want to use OR-NOT logic and minimizing by f_3 the number of non-inverted signals we obtain exactly the same results as in Table VII. However for another problem this would in general lead to another activation functions. Therefor depending on the user's preferences different objective functions could be used and thus different results can be achieved.

B. Example 2

For the second example let us consider an automat having 2 inputs (x_0 and x_1), 11 internal states (a_0 to a_{10}) and 2 outputs (y_0 and y_1). This time we want to use RS flip-flops.

Here the coding table and the results obtained are not shown, but only a summary is made. For this example Karnaugh maps are still applicable, although using 3 dimensional maps causes some difficulties and the possibility for mistakes increases.

To illustrate the minimizing difficulties in this example we will mention that the possible combination of c variables for input RI are 33 554 432, while for S_1 there is only one unknown variable. Of course this depends on the automata description, but it shows that unbalanced "load" could appear. Because of the very large number of possible combination for R_1 and the complexity of the problem, GA did not achieve the result achieved by Karnaugh maps in reasonable time. Several optimization runs were made for R_1. The best result, among the achieved (optimization run with 1000 iterations and populations of 40 genes) did not match the one achieved with Karnaugh map. However those result was differing from the optimal one by just 1 unit in one of the thee optimization functions (same optimization functions as in Sec. V-A were considered). For prove of the capabilities of the proposed method additional run with 1O0 times increase of iteration number and 2.5 times increase of the population size were made, during which the optimal result was achieved.

VI. CONCLUSIONS AND FUTURE WORKS

In this paper a genetic algorithms are introduced as method for synthesis the activation function of flip-flops in finite state machines. Introduction of

GA and Pareto optimality is made and an example is shown. GA appear to be a powerful algorithm for Boolean string optimization. Because GA are population passed they are also suitable for multi-objective optimization.

Depending on the number of variables in the optimization, different number of iterations and genes in populations are needed to reach optimality. However, when the parameters of GA are appropriately chosen the results achieved with GA match those obtained by Karnaugh maps.

According to the user's preferences different objective functions could be used and thus different results achieved.

This paper proves that Genetic Algorithms offers correct and reliable procedure for synthesis of Finite State Machines. Since GA are designed as multi-parameter algorithms they are suited for design of finite state machines with large number of states. The method is especially appropriate when the time of optimization run is not important, but optimal description of the activation functions is needed.

For future work we intend to test two synthesis of a finite state machine at once (on one run of the GA, not input by input). We also intend to implement two levels Boolean minimization in FSM synthesis.

REFERENCES

[1] St. Mihailov, A. Popov, Kr. Filipova, N. Kasev, "Comparative Analysis of Boolean Functions Minimization in Terms of Simplifying the Synthesis", *First International Congress of Mechanical and Electrical Engineering and Technologies*, ISBN 954-20-0215-7, MARIND 2002, 6-11 Oct., Varna, pp.273-276

[2] C. Coello, "An Empirical Study of Evolutionary Techniques for Multiobjective Optimization in Engineering Design", Department of Computer Science, Tulnne University, 1996

[3] S. William, "The Role of Mutation and Recombination in Evolutionary Algorithms", dissertation for Doctor of Philosophy at George Mason University, 1998

[4] J. Andenson, "A survey of Multiobjective Optimization in Engineering Design", Department of Mechanical Engineering, Linktjping University. Sweden

[5] A. Oyama, "Wing Design Using Evolutionary Algorithms", Department of Aeronautics and Space Engineering of Tohoku University, 2000

[6] Kr. Filipova, "Automata models" (Bulgarian language), TU-Sofia, 2000

Andrey Popov received B.Sc. degree in "Systems and Control" - TU-Sofia, faculty "Automatics, Informatics and Process Control" and M.Sc. degree in "Mechatronics" - Hamburg University of Technology. Research interest in genetic algorithms, robotics, modern methods of control.
andrey.popov @ gmx.net

Krasimira Filipova received the M.Sc. degree in "Automatics and Telemechanics" from VMEI "V. I. Lenin" - Sofia in 1972, and the Ph.D. in 1985. At the moment is Assistant Professor in "Systems and Control" department in "TU-Sofia". Interests in Theory of automatic control, discrete systems, finite state machine theory, theory and application of Petri nets, distributed systems. "Systems and Control" department, TU-Sofia
kfilipova @ abv.bg