Multiple Hybridation in Genetic Algorithms

Rustem Popa, Dorel Aiordăchioaie, Viorel Nicolau
Department of Electrical Engineering
University of Galati
Domnească Street 111, 6200 Galati
Romania
email: Rustem.Popa@ugal.ro, Dorel.Aiordachioaie@ugal.ro, Viorel.Nicolau@ugal.ro

Abstract

This paper develops a multiple hybrid optimisation algorithm, by combining the genetic algorithms approach and two other optimisation techniques: inductive search and simulated annealing. Hybridation of a genetic algorithm with each of these techniques one by one has been tried before, but our approach facilitates the reduction of the search space with the introduction of more diversity in population to prevent the problem of premature convergence. This idea of double hybridation has been used to solve a NP-complete critical problem, as Automatic Test Pattern Generation for digital circuits. The experiments have shown a better global performance in comparison with other simple hybrid genetic algorithms.

1 Introduction

Genetic Algorithms are an adaptive searching technique for solving optimisation problems based on the mechanics of natural genetics and natural selection. The success of the application of Genetic Algorithms to an optimisation problem depends on the representation of chromosomes, fitness function, method of crossover, mutation operation, and on the diverse information held in the chromosomes. When the diversity is lost well before the global optimum solution is found, the performance of Genetic Algorithms deteriorates and their solution processes converge prematurely. Moreover, the mutation operation is important. While the mutation operation adds new information to a chromosome, it can also destroy useful information originally held in the chromosome. The performance of Genetic Algorithms can be improved if a mechanism can be incorporated in them so that the positive effects of mutation will be retained but the adverse effects will be eliminated [Wong, and Wong, 1994].

The paper develops a Genetic Algorithm hybridated twice: first with an Induction Search Algorithm, and then with a Simulated Annealing technique. We have compared the performances of this algorithm, with two Genetic Algorithms hybridated with each of these techniques one by one. The performances of these hybridated algorithms are demonstrated through an example of Automatic Test Pattern Generation in a digital circuit.

The rest of the paper is structured as follows. The Standard Genetic Algorihm is introduced in Section 2. A simple hybridation of the Genetic Algorithm with the Inductive Search is described in Section 3, and the combination between Genetic Algorithm and the Simulated Annealing technique is presented in Section 4. Section 5 describes the Multiple Hybridated Genetic Algorithm. Section 6 points on the Automatic Test Pattern Generation problem and the last two sections, 7 and respectively 8, presents our experiments and conclusions.

2 The Standard Genetic Algorithm

In Genetic Algorithms the search is conducted using information of a population of candidate solutions, called chromosomes, so that the chance of the search being settled in a local optimum can be significantly reduced. Four essential components need to be designed in applying a Genetic Algorithm for an optimisation problem: chromosomes representation, crossover operator, mutation operator and fitness function.

Each chromosome is represented by a binary string of finite length. The function of the crossover operator is to generate new child chromosomes from two selected parents chromosomes. We have used uniform crossover with a high rate, even if the computation effort in exploring unpromising regions of the solution space may be higher. Mutation operator changes the value of one bit in the chromosome, with some probability. If the frequency of the mutation operation is too high, valuable old information may be destroyed in the chromosomes. On the other hand, if the frequency is low, the chromosomes remain virtually unchanged and little information will be added to assist the search in Genetic Algorithm. After all, the fitness function can be defined in terms of the objective function of the optimisation problem. The search for the global optimum solution is then equivalent to finding the chromosomes having the maximum fitness.

The structure of a Standard Genetic Algorithm is as follows:

begin
Generate randomly the initial population of chromosomes;
repeat
- calculate the fitness of chromosomes in current generation;
repeat
- select proportional with fitness, in a stochastic manner, 2 chromosomes as parents;
- apply crossover to the selected parents to obtain 2 child chromosomes;
- calculate the fitness of chromosomes;
until end of the number of new chromosomes
- apply mutation to the new chromosomes;
- update the population, in accordance with the fitness of each chromosome;
until end of the number of generations
end

Like mutation, crossover may be applied with some probability to reduce the computation effort, and the number of generations can be conveniently expressed as number of iterations.

3 The Inductive Genetic Algorithm

The search space reduction methodology developed by Bilchev, and Parmee [1996] was called the inductive search. The problem of global optimisation is partitioned into a sequence of subproblems, which are solved by searching of partial solutions in subspaces with smaller dimensions.

The structure of the Inductive Genetic Algorithm is as follows:

begin
Initialize a partial solution for $N = 1$;
for $k = 2$ to N **,**
Generate randomly the initial population of chromosomes;
repeat
- append each chromosome to the partial solution, evaluate it and assign fitness;
repeat
- select proportional with fitness, in a stochastic manner, 2 chromosomes as parents;
-apply crossover to the selected parents to obtain 2 child chromosomes;
- calculate the fitness of chromosomes;
until end of the number of chromosomes
- apply mutation to the new chromosomes;
- update the population, in accordance with the fitness of each chromosome;
until end of the number of generations
Update the partial solution;
end
end

The solution is generated step by step, beginning from the base of the induction and at each step following an induction rule to update the solution. First induction rule asserts that a set of local optima of dimension $n+1$ may be derived from the set of local optima of dimension n. The second induction rule asserts that global optima of dimension $n+1$ may be derived from the global optima of dimension n.

The overall algorithm could also be viewed as a genetic algorithm with dynamic fitness function, i.e. the fitness function changes at each generation. The content of "for" cycle constitute a genetic algorithm, the rest implement the inductive search.

4 Combining a Genetic Algorithm and Simulated Annealing Technique

The optimisation process in Simulated Annealing is essentially a simulation of the annealing process of a molten particle. Starting from a high temperature, a molten particle is cooled slowly. As the temperature reduces, the energy level of the particle also reduces. When the temperature is sufficiently low, the molten particle becomes solidified. Analogous to the temperature level in the physical annealing process is the iteration number in Simulated Annealing. In each iteration, a candidate solution is generated. If this solution is a better one, it will be accepted and used to generate yet another candidate solution. If it is a deteriorated solution, the solution will be accepted with some probability.

The structure of a Genetic Algorithm hybridated by Simulated Annealing is as follows:

begin
Generate randomly the initial population of chromosomes and establish the temperature T_0;
repeat
- calculate the fitness of chromosomes in current iteration;
repeat
- select proportional with fitness, in a stochastic manner, 2 chromosomes as parents;
- apply crossover to the selected parents to obtain 2 child chromosomes;
- apply mutation to the new chromosomes;
- calculate the fitness of chromosomes;
- the new chromosomes are accepted or not accepted in the new population;
until end of the number of new chromosomes
- update the population;
- the temperature is decreased;
until end of the number of iterations
end

The probability of acceptance can be:

$$P(\Delta) = 1/(1+\exp(\Delta/T)), \qquad (1)$$

where Δ is the amount of deterioration between the new and the old solutions and T is the temperature level at which the new solution is generated.

The temperature is a parameter depending by the number of iterations. We have considered in our examples the following function of temperature:

$$T(i) = T_0/(1+\ln i), \qquad (2)$$

where T_0 is the initial high temperature and i is the number of iterations.

The probability of acceptance will be low when the temperature is low. Some valuable chromosomes will be replaced during the entire period of evolution, but this chance is greately reduced towards the end of the process. In this way, sufficient diversity of chromosomes can be maintained and premature convergence can be eliminated.

5 The Multiple Hybridated Genetic Algorithm

Each of the two distinct methods of hybridation discussed above have some advantages. The inductive search effort at each inductive step controls the trade-off between the computational complexity and the expected quality of results, while Simulated Annealing avoids the premature convergence and reduces the adverse effects of the mutation operation.

Our idea was to cumulate all these advantages in a single algorithm, through a double hybridation of the Genetic Algorithm: with Inductive Search on the one hand, and with Simulated Annealing technique on the other hand.

The structure of the Multiple Hybridated Genetic Algorithm is as follows:

begin
Initialize a partial solution for $N = 1$ and establish the initial temperature T_0 ;
for $k = 2$ to N **,**
Generate randomly the initial population of chromosomes;
repeat
- append each chromosome to the partial solution, evaluate it and assign fitness;
repeat
- select proportional with fitness, in a stochastic manner, 2 chromosomes as parents;
-apply crossover to the selected parents to obtain 2 child chromosomes;
- apply mutation to the new chromo-somes;
- calculate the fitness of chromosomes;
- the new chromosomes are accepted or not accepted in the new population;
until end of the number of chromosomes
- update the population, in accordance with the fitness of each chromosome;
- the temperature is decreased;
until end of the number of generations
Update the partial solution;
end
end

The inductive formulation also gives meaning to intermediate solutions. The number k of partial solutions which gives a satisfactory global solution can serve as an efficient stopping condition. In this algorithm, appears like a new parameter the number of generations per inductive step. The search effort at each inductive step controls the trade-off between the computational complexity and the expected quality of results.

6 The Automatic Test Pattern Generation Problem

Modern VLSI circuits usually contain in their structures test and monitoring systems, known as built-in self-test circuits (BIST). These circuits generate a sequence of input test vectors and expect a sequence of output vectors. These test patterns are generated automatic by BIST circuits.

Fault analysis is the process used to determine the fault detection coverage of a particular design. The fault analysis process for a design involves the optimisation of the input stimulus to fully exercise all components to increase the testability, while logic simulation involves the optimisation of the functionality of the design. These tasks are very different processes, and both tasks are necessary within the design development process. The fault analysis process fits after the initial functional verification of the design and before the physical hardware testing of the product.

The amount of fault coverage within a design depends on the following two factors: comprehensiveness of the test code, and inherent testability of the logic design. In this paper we concentrate on the first factor and formulate the problem of finding an effective set of input test vectors as a search problem.

The digital structures of PLA type, like the one presented in figure 1, are fully testable for "stuck-at 0" faults. The binary function F depends of all 12 inputs, in order to eliminate possible constraints imposed on the test codes. We have injected in this circuit a number of 5 random "stuck-at 0" faults. These faults are detected with the first 16 test vectors generated by the standard genetic algorithm. The waveforms achieved in PSPICE simulation are represented in figure 2. The arrows from the bottom of this representation show the instant of each fault detection.

Hwang and Shen [1996] have established a relation to calculate the maximum number of possible "stuck-at 0" faults in a PLA structure.

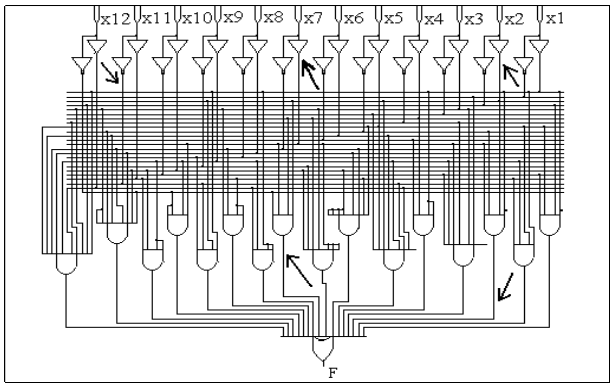

Figure 1. The PLA structure for implementation of the binary function F and 5 "stuck-at 0" faults.

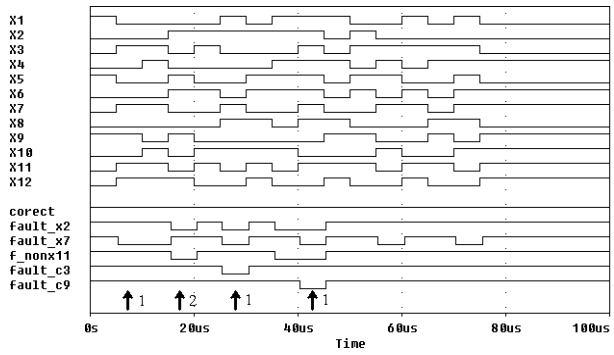

Figure 2. The test vectors and the coverage of the 5 faults indicated in the foregoing figure.

They have shown that this number is:

$$N_{\max} = 2 \cdot N_{in} + N_{inv} + N_g , \qquad (3)$$

where N_{in} is the number of inputs, N_{inv} is the number of invertors and N_g is the number of gates. In our case, $N_{\max} = 2 \cdot 12 + 12 + 16 = 52$ faults.

7 Experiments

We have conducted the experiments with all algorithms described above, in the purpose to find the maximum fault coverage with only 6 test vectors. We have tested the PLA structure from the figure 1 for about 50 potential "stuck-at 0" faults.

If n is the number of covered faults and N is the number of all faults in the fault population, the associated fitness function is $f = \frac{n}{N} \cdot 100\%$. There may also be a number of constraints concerning the possible combinations of input signals. The designers of the circuit define the set of legal combinations in terms of the legal states of a number of channels. The set of all legal templates defines the feasible region.

The main genetic parameters used in these algorithms are: a population size of 20 chromosomes, uniform crossover with 100% rate, uniform mutation with 1% rate. Figure 3 shows the evolution of the standard genetic algorithm with changed parameters. These results represent the average values of 10 succesive runs. We can see that algorithm is very sensitive to its basic parameters, and therefore the requirement of a careful setting of these parameters is absolute necessary. Popa, *et al*. (1998) has shown a much better performance of the Inductive Genetic Algorithm from this point of view. The hybridation of the standard genetic algorithm by induction seems to be more robust, since the deviation of fault coverage is only 3%, comparatively with almost 15% in the case of standard genetic algorithm.

The inductive search transforms the problem of finding a sequence of N test vectors that maximizes the fault coverage of a circuit, into the problem of finding a sequence of k test vectors that maximizes the fault

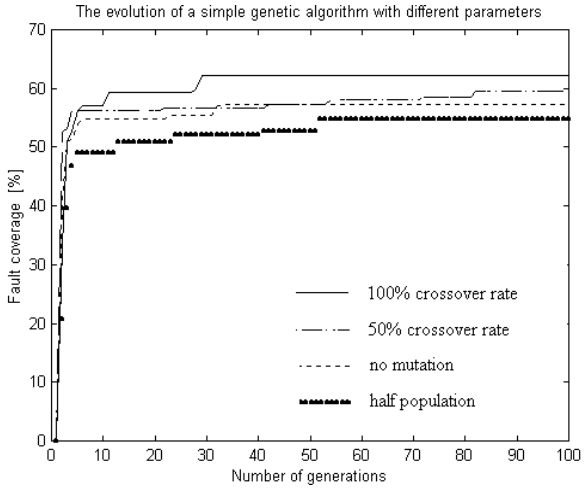

Figure 3. Standard genetic algorithm with different parameters

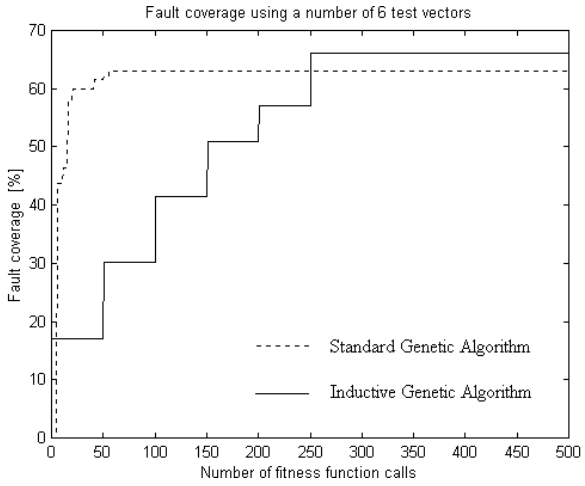

Figure 4. Runs of the two genetic algorithms on a fault coverage problem of 50 possible faults

coverage, for each k = 1 to N.

Figure 4 shows the comparative performances of the two genetic algorithms on a fault coverage problem consisting of 50 possible faults. The maximum fault coverage achieved with 6 input test vectors was about 63% for the Standard Genetic Algorithm, and about 66% for the Induction Genetic Algorithm. These results represent the average values of 10 succesive runs. The horizontal levels in hybrid algorithm show the maximum fault coverage with 1, 2, 3, 4, 5 and 6 test vectors one-by-one.

Figure 5 shows the performance of the Inductive Genetic Algorithm as a function of the number of input test vectors. We can see that for a number of vectors $N > 5$ the Inductive Genetic Algorithm is more efficient, and a full 100% fault coverage is achieved with only 19 test vectors in this case, and with 26 test vectors in the case of the Standard Genetic Algorithm. Thus the full coverage of the faults in this PLA structure is possible.

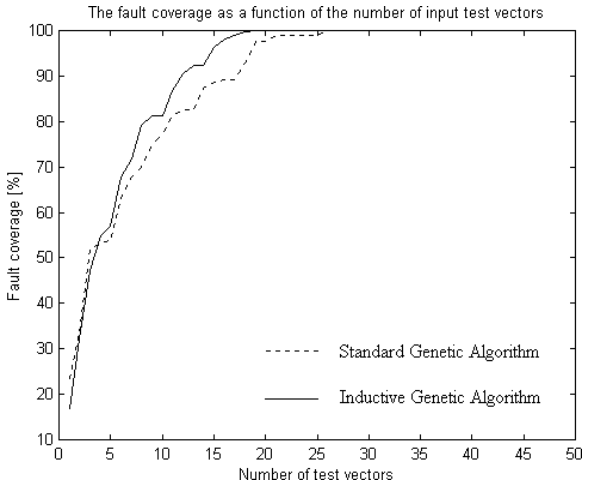

Figure 5. The fault coverage as a function of the number of input test vectors

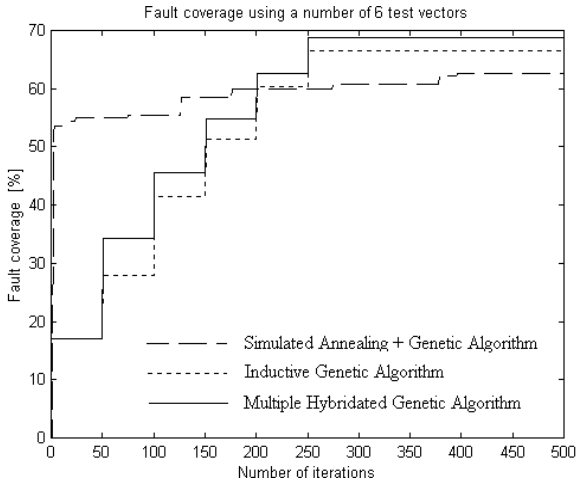

Figure 7. Runs of the three hybridated algorithms on a fault coverage problem of 50 possible faults

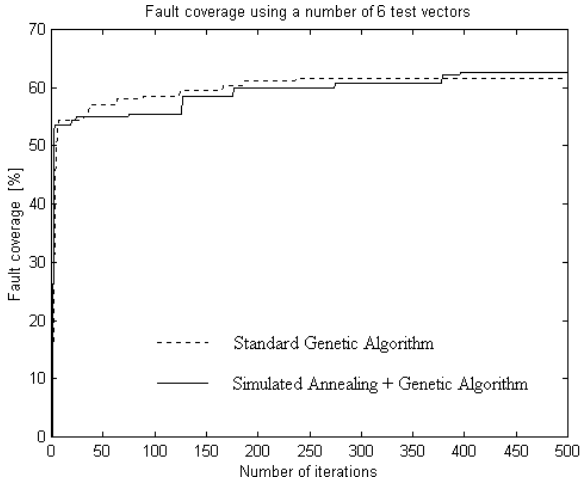

Figure 6. Runs of the two genetic algorithms on a fault coverage problem of 50 possible faults

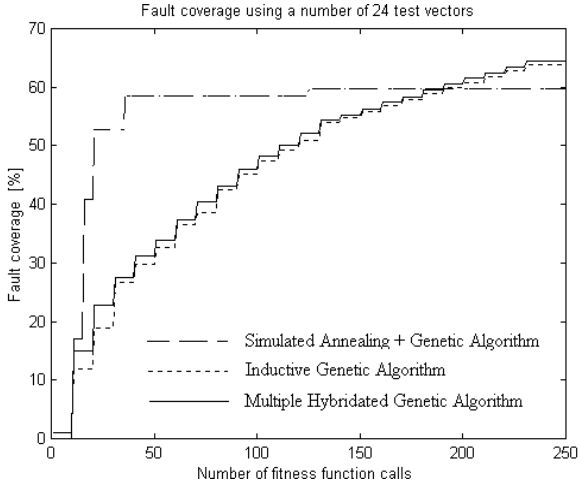

Figure 8. Runs of the three hybridated algorithms on a fault coverage problem of 200 possible faults

Figure 6 represents the comparative performances of the two genetic algorithms on the same problem. These results represent the average values of 5 succesive runs. The maximum fault coverage is better in the case of the hybridation between Simulated Annealing and Genetic Algorithm with about 1,3%.

Simulations from the figure 7 point towards better performances of the Multiple Hybridated Genetic Algorithm by comparing with other simple hybridated algorithms discussed above.

The maximum fault coverage achieved with the Multiple Hybridated Genetic Algorithm after 500 iterations was about 69%, while the maximum fault coverage achieved with the Inductive Genetic Algorithm, the best of the two single hybridated genetic algorithms, was about 66%. These results represent the average values of 5 succesive runs. We have tried even with 10 or more number of runs, but the results are basically the same.

Another set of experiments were made on a more complex digital structure with 200 faults in the fault population. Figure 8 shows the comparative performances of the three hybridated genetic algorithms on this new fault coverage problem. The number of input test vectors is 24. After 250 fitness function calls, that is 25 iterations, each with 10 generations per inductive step, the fault coverage of the Multiple Hybridated Genetic Algorithm is with about 1% better than the fault coverage of the Inductive Genetic Algorithm.

Previous experiments with a standard genetic algorithm in which the chromosome coded all the 24 input test vectors, each of length 12 bits, produced the best fault coverage of 57% [Bilchev and Parmee, 1996]. We have achieved about 59% with the combination between Simulated Annealing and Genetic Algorithm and almost 64% with the Multiple Hybridated Genetic Algorithm, which is the best solution of all hybridated algorithms discussed in this paper.

8 Conclusions

In this paper we have described a new algorithm based on a double hybridation of a classical genetic algorithm. As compared to previous experiments, our algorithm has improved the performance of fault coverage from 57% to almost 64%.

This comparison is certainly thereabouts, because we didn't have any idea about the circuit used by Bilchev and Parmee [1996] in their experiments. All available information consists in a digital structure with 200 faults tested with a number of 24 input vectors, each of length 12 bits. We have constructed a similar complex digital PLA structure, which has been tested in the same way.

A more comprehensive result is the comparison of the Multiple Hybridated Genetic Algorithm with two other hybridated algorithms: the Inductive Genetic Algorithm and the Genetic Algorithm hybridated by Simulated Annealing. We have proved on two different examples, with different complexities, that the performance of the Multiple Hybridated Genetic Algorithm is better than the performance of any other algorithm obtained through simple hybridation of a standard genetic algorithm.

The only drawback of this algorithm is most likely the requirement of an appreciable number of iterations until the straight result is delivered. This implies a sufficient high CPU time requirement.

References

[Bilchev and Parmee, 1996] George Bilchev and Ian Parmee. Constraint Handling for the Fault Coverage Code Generation Problem: An Inductive Evolutionary Approach. In *Proc. of 4-th Conference on Parallel Problem Solving from Nature (PPSN IV),* pages 880-889, Berlin, September 1996.

[Hwang and Shen, 1996] G. H. Hwang and W. Z. Shen. Fault analysis and automatic test pattern generation for break faults in programmable logic arrays. In *IEE Proc. – Circuits Devices Syst., Vol. 143, No. 3,* pages 157-166, June 1996.

[Popa *et al*.,1998] Rustem Popa, Constantin Cruceru, and Mircea Iliev. A Hybrid Genetic Algorithm for Automatic Test Pattern Generation. In *The 10-th Symposium on Modelling, Simulation and Identification Systems (SIMSIS 10'98),*pages 74-78, Galati, Romania, October 1998.

[Wong and Wong, 1994] Kit Po Wong and Yin Wa Wong. Development of Hybrid Optimisation Techniques Based on Genetic Algorithms and Simulated Annealing. In *Workshop on Evolutionary Computation (AI'94),* pages 127-154, Armidale, Australia, November 1994.