ICCC 2004

2004 International Conference on Computational Cybernetics

Proceedings

Vienna, Austria - August 30 - September 1, 2004

Hungarian Fuzzy Association

Budapest Tech, Hungary

Vienna University of Technology, Austria

IEEE SMC Chapter, Hungary

IEEE NN Chapter, Hungary

IEEE Joint Chapter of IES and RAS, Hungary

IEEE R8

IEEE Systems, Man, and Cybernetics Society

EUROFUSE

Japan Society for Fuzzy Theory and Intelligent Informatics

John von Neumann Computer Society, Budapest, Hungary

IEEE

Population Size and Processing Time in a Genetic Algorithm

Cecília Reis [1], J. A. Tenreiro Machado[1]
[1]Institute of Engineering of Porto
Polytechnic Institute of Porto
Rua Dr. António Bernardino de Almeida, 4200-072 Porto
Portugal
{cecilia, jtm}@dee.isep.ipp.pt

J. Boaventura Cunha[2]
[2]University of Trás-os-Montes and Alto Douro
Engineering Department
Apt. 1013, 5000-911 Vila Real
Portugal
jboavent@utad.pt

***Abstract* – This paper presents several experiments with a genetic algorithm (GA) for designing combinational logic circuits. The study addresses the population size and the processing time for achieving a solution in order to establish a compromise between the two parameters. Furthermore, it is also investigated the use of different gate sets for designing the circuits namely RISC and CISC like gate sets.**

I. INTRODUCTION

It is well known that GAs are widely used as search and optimization techniques that have many areas of application. The success of GAs depends on the right parameter setting, such as the population size, and on the interaction of the different operators, namely the crossover and the mutation operators, which create new individuals from existing ones [1]. This paper concentrates on studying population sizing for obtaining the minimum processing time of GAs applied to the design of combinational logic circuits [18].

Bearing this fact in mind, this paper is organized as follows. Section 2 deals with population sizing background. Section 3 presents the GA used in terms of the circuit encoding as a chromosome, the genetic operators and the fitness function and section 4 develops several simulations and analyzes the results. Finally, section 5 outlines the main conclusions.

II. POPULATION SIZING

Increasing the optimization complexity leads to the need for larger population size. Therefore, it is important to understand the role of the population in GA processes. Goldberg, Deb and Clark (1992) proposed population sizing models for different selection schemes. Their model is based on deciding correctly between the best and the second best building blocks (BBs) in the same partition. They incorporate noise arising from other partitions into their model. However, they assume that if wrong BBs are chosen in the first generation, then the GA is unable to recover from the error [3]. Harik, Cantu-Paz, Goldberg and Miller (1997) refined the previous model by incorporating cumulative effects of decision making over time rather than in the first generation only. These authors modelled the decision making between the best and the second best BBs in a partition as a gambler's ruin problem [4]. This scheme is based on the assumption that the selection process consists on a tournament selection without replacement. Miller (1997) extended this model to predict population sizing in the presence of external noise [7].

Bearing these ideas in mind, we observe that the major work done in the population sizing of GAs can be divided into two main groups, namely the population sizing based on initial supply of BBs and the population sizing based on good decision making between competing BBs [1]. Both issues are combined together in the Gambler's Ruin model [4].

A. Building Blocks supply

The question of how to choose an adequate population size for a particular domain is complex. If the population is too small, it is not likely that the GA will find a good solution for the problem running only on a few generations. On the other hand, if the population size is increased, in order to find high quality solutions, the GA will waste time processing unnecessary individuals, and this may result in an unacceptably slow performance. The problem consists of finding a population size that is large enough to permit a correct exploration of the search space without wasting computational resources.

There are two approaches to the BBs supply question, one spatial and one temporal. The spatial approach seeks to adjust the population size adequately to ensure, probabilistically, that sufficient diversity and numbers of BBs are present initially, thereby supplying the genetic algorithm with material for subsequent search. The temporal approach assumes the existence of a mutation or another diversity generator to return sufficient BB diversity on an appropriate time scale [5].

The issue of BBs supply was raised by Holland (1973) that was followed by several others investigators such as De Jong (1975), Grefenstette (1986), Goldberg (1985, 1989), Alander (1992) and Reeves (1993), all of them studying and improving population sizing theory. Recently, minimum population size that has all raw of BBs has been estimated and experimentally verified by Goldberg, Sastry and Latoza (2001).

B. Building Blocks decision-making

The building block decision problem is the recognition that the GA can make mistakes when deciding between a BB and its competitors. Over the years, theoretical models have addressed this topic beginning with Holland (1973, 1975) followed by De Jong (1975) and Goldberg (1989). Holland's work was further analysed by recent studies [16] where the population size was calculated in the presence of collateral noise leading to an equation that has proved to be very conservative.

The above model was improved by, combining initial supply of BBs, and correct selection of the best BB over its competitors. The result is an equation that relates the size of the population with the desired quality of the solution, as well as the problem size and difficulty.

C. Building Blocks mixing

The above models have ignored the effect of genetic operators on the population sizing. Yet, some work has been done in analysing recombination operators in the view point of a diversity preservation tool. Only a few studies have been performed in developing facetwise models to analyze recombination as an innovation operator [17]. Therefore, there is an immediate need for a better facetwise scheme that addresses the BB mixing problem and analyzes fixed recombination to answer the mixing question.

Most of the above models, although accurate, are difficult to apply in practice because they rely on several assumptions that may not hold for real-world problems.

III. THE ADOPTED GENETIC ALGORITHM

A. Problem definition

In this article a GA strategy is adopted to design combinational logic circuits. The circuits are specified by a truth table and the goal is to implement a functional circuit with the least possible complexity. Several sets of logic gates have been defined, as shown in Table 1, being Gset 1a and 1b the simplest ones (*i.e.*, a RISC-like sets) and Gset 6 the most complex (*i.e.*, a CISC-like set).

Table 1 Gate sets

Gate Set	Logic gates
Gset 1a	{NAND,WIRE}
Gset 1b	{XOR,WIRE}
Gset 2	{AND,XOR,WIRE}
Gset 3	{AND,OR,XOR,WIRE}
Gset 4	{AND,OR,XOR,NOT,WIRE}
Gset 5	{AND,OR,XOR,NOT,NAND,WIRE}
Gset 6	{AND,OR,XOR,NOT,NAND,NOR,WIRE}

For each gate set the GA searches the solution space, based on a simulated evolution aiming the survival of the fittest strategy. In general, the best individuals of any population tend to reproduce and survive, thus improving successive generations. However, inferior individuals can, by chance, survive and also reproduce [2]. In our case, the individuals are digital circuits, which can evolve until the solution is reached (in terms of functionality and complexity).

B. Circuit enconding

In the GA scheme the circuits are encoded as a rectangular matrix **A** (row × column = $r \times c$) of logic cells as represented in figure 1.

Each cell is represented by three genes: <*input*1><*input*2><*gate type*>, where *input*1 and *input*2 are one of the circuit inputs, if they are in the first column, or one of the previous outputs, if they are in other columns. The *gate type* is one of the elements adopted in the gate set. The chromosome is formed by as many triplets of this kind as the matrix size demands. For example, the chromosome that represents a 3 × 3 matrix is depicted in figure 2.

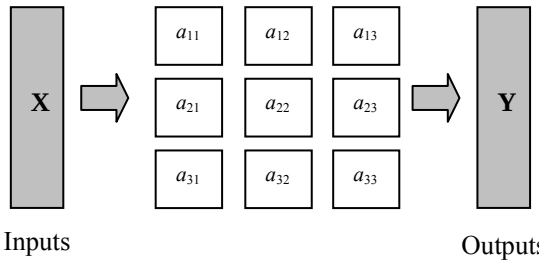

Fig. 1. A 3 × 3 matrix **A** representing a circuit with input **X** and output **Y**.

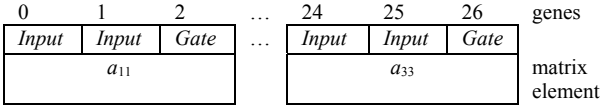

Fig. 2. Chromosome for the 3 × 3 matrix of figure 1.

C. The genetic operators

The initial population of circuits (strings) is generated at random. The search is then carried out among this population. The three different operators used are reproduction, crossover and mutation, as described in the sequel.

In what concern the reproduction operator, the successive generations of new strings are reproduced on the basis of their fitness function. In this case, it is used a tournament selection [2] to select the strings from the old population, up to the new population.

For the crossover operator, the strings in the new population are grouped together into pairs at random. Single point crossover is then performed among pairs. The crossover point is only allowed between cells to maintain the chromosome integrity.

The mutation operator changes the characteristics of a given cell in the matrix. Therefore, it modifies the gate type and the two inputs, meaning that a completely new cell can appear in the chromosome. Moreover, it is applied an elitist algorithm and, consequently, the best solutions are always kept for the next generation.

To run the GA we have to define the number of individuals to create the initial population *P*. This population is always the same size across the generations, until the solution is reached.

The crossover rate *CR* represents the percentage of the population *P* that reproduces in each generation. Likewise *MR* is the percentage of the population *P* that mutates in each generation.

D. The fitness function

The calculation of the fitness function *F* is divided in two parts, f_1 and f_2, which measure the functionality and the simplicity, respectively. In a first phase, we compare the output **Y** produced by the GA-generated circuit with the required values $\mathbf{Y_R}$, according with the truth table, on a bit-per-bit basis. By other words, f_1 is incremented by *one* for each correct bit of the output until f_1 reaches the maximum value f_{10}, that is, when we have a functional circuit. Once the circuit is functional, in a second phase, the GA tries to generate circuits with the least number of

gates. This means that the resulting circuit must have as much genes <*gate type*> ≡ <*wire*> as possible. Therefore, the index f_2, that measures the simplicity (the number of null operations), is increased by *one* (*zero*) for each *wire* (*gate*) of the generated circuit, yielding:

$$f_{10} = 2^{ni} \times no \quad (1a)$$

$$f_1 = f_1 + 1 \text{ if bit } i \text{ of } \mathbf{Y} = \text{bit } i \text{ of } \mathbf{Y_R},\ i = 1, \ldots, f_{10} \quad (1b)$$

$$f_2 = f_2 + 1 \text{ if } gate\ type = wire \quad (1c)$$

$$F = \begin{cases} f_1, & F < f_{10} \\ f_1 + f_2, & F \geq f_{10} \end{cases} \quad (1d)$$

where *ni* and *no* represent the number of inputs and outputs of the circuit.

IV. EXPERIMENTS AND SIMULATION RESULTS

Reliable execution and analysis of a GA usually requires a large number of simulations to provide a reasonable assurance that stochastic effects have been properly considered [8]. Therefore, in this study are developed $n = 40$ simulations for each case under analysis.

The experiments consist on running the GA to generate a typical combinational logic circuit, namely a 2-to-1 multiplexer. The circuit is generated with gate sets presented in Table 1 for $CR = 95\%$, $5\% \leq MR \leq 90\%$ and $6 \leq P \leq 10^4$.

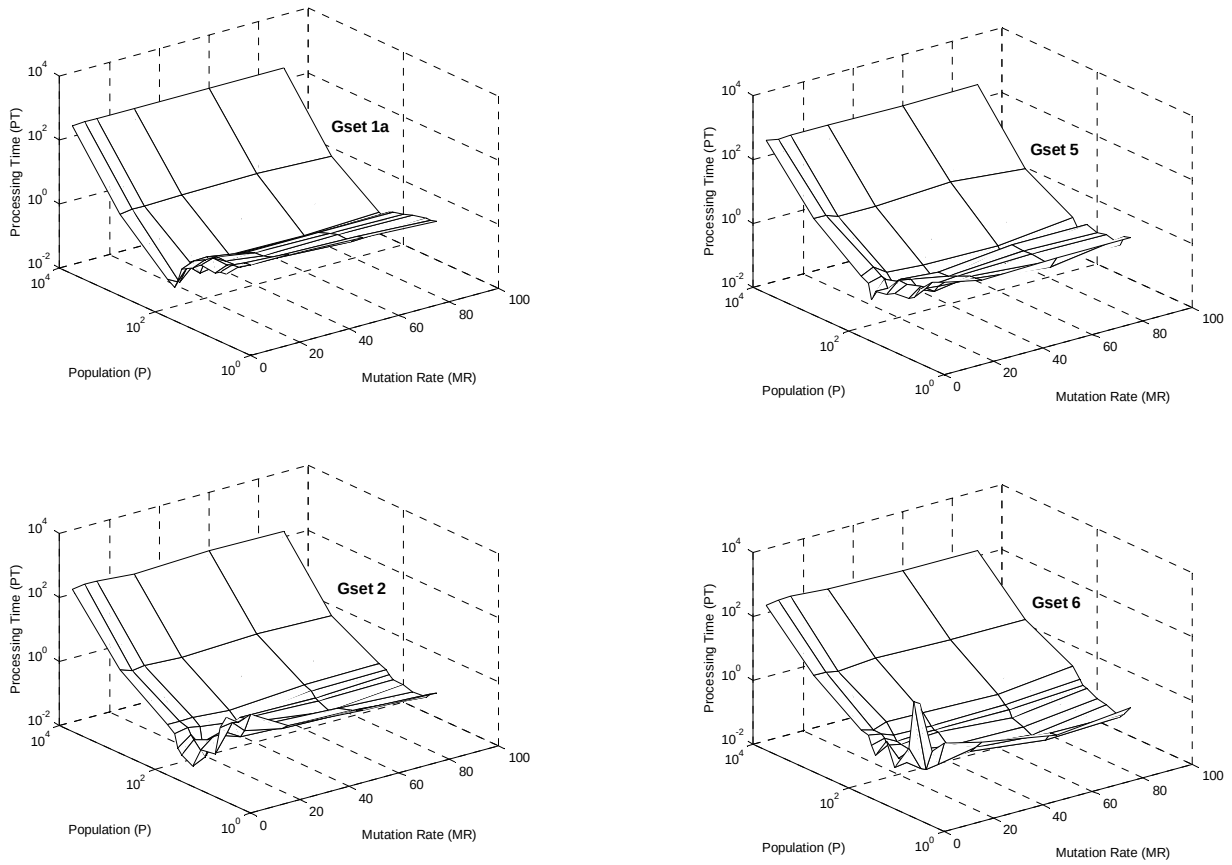

Fig. 3. *PT versus* (*P*, *MR*) with *CR* = 95% using Gset 1a up to Gset 6 for the 2-to-1 Multiplexer

In order to obtain the processing time PT to achieve the solution, the following steps are implemented:

i. We calculate the median $m_e(N_s)$ of the number of generations to achieve the solution N_s;

ii. We estimate the processing time for one generation PT_1 using the average when running five simulations;

iii. The processing time to achieve the solution PT is:

$$PT = m_e(N_s) \times PT_1 \quad (2)$$

The 2-to-1 multiplexer circuit has a truth table with 3 inputs $\mathbf{X} = \{S_0, I_0, I_1\}$ and 1 output $\mathbf{Y} = \{O\}$. In this case, the matrix $\mathbf{A}$ has a size of $r \times c = 3 \times 3$ and the length of each string representing a circuit (*i.e.*, the chromosome length) is $CL = 27$.

Figure 3 shows the processing time PT for gate sets 1a to 6, respectively. With Gset 1b it was impossible to achieve a solution.

It is clear that PT decreases with P. However, for very small populations the GA has an extreme difficulty in obtaining a solution, due to the lack of BBs, which leads to an increase of the PT. For $P < 50$, in 5% of the simulations, the GA ends without giving a solution. Therefore, for the 2-to-1 multiplexer circuit it is better to use population sizes $50 \leq P \leq 100$. Table 2 presents the best PT for each gate set.

Table 2

Gate Set	***P***	***MR***	**Minimum *PT* (seconds)**
Gset 1a	70	5%	0.05742
Gset 2	70	10%	0.11656
Gset 3	60	5%	0.11109
Gset 4	70	10%	0.13533
Gset 5	70	10%	0.19287
Gset 6	50	5%	0.13400

The best performance goes to Gset 1a with $PT = 57.42$ ms for $P = 70$ and $MR = 5\%$.

In what concerns MR the conclusion is that its influence upon the PT seems to be of minor importance. Nevertheless, the best results occur with $5\% \leq MR \leq 10\%$, with exception of Gset 6 as can be confirmed by figure 4 which presents PT for solution *versus* MR for all the gate sets under study.

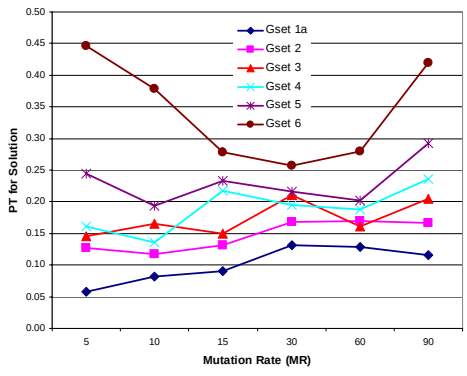

Fig. 4. PT for solution *versus* MR, with $CR = 95\%$ and $P = 70$, for the 2-to-1 Multiplexer

We verify that PT_1 increases significantly with P but is almost independent of MR. Figures 5 and 6 show PT_1 *versus* P and MR, respectively, for $CR = 95\%$ and $MR = 5\%$.

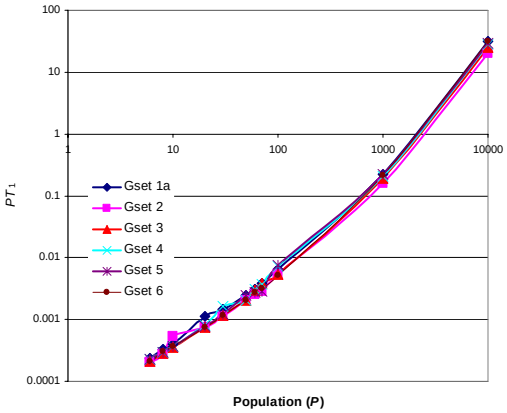

Fig. 5. PT_1 *versus* P with $CR = 95\%$ and MR=5% for the 2-to-1 Multiplexer

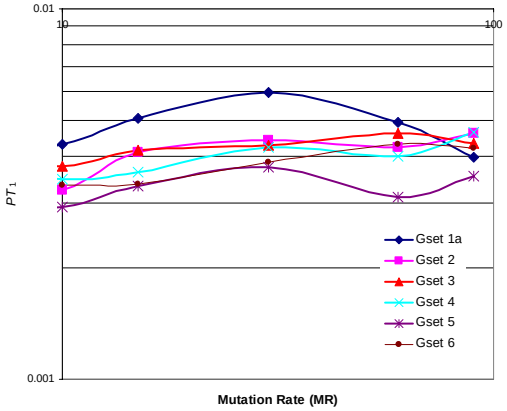

Fig. 6. PT_1 *versus* MR with $CR = 95\%$ and $P = 70$ for the 2-to-1 Multiplexer

The above experiments were based on the number of generations to achieve the solution N_s, for several combinations of the parameters CR, MR, P and gate sets. The results reveal that the number of generations to achieve the solution N_s has a Pareto distribution, that is, has a probability density function:

$$f(n_s) = ca^c / n_s^{c+1},\ n_s \geq a \quad (3)$$

The density function qualitatively resembles that of the exponential distribution, but decays far more slowly, *i.e.*, has a longer "tail". It has been much used in economics (where Pareto first introduced it as describing the distribution of incomes) but is of possible utility as a model in any situation demanding a long-tailed one-sided distribution. Thus, the probability that a random income, in some defined population, exceeds a minimum, a, is ~Pareto.

The parameters (a, c), scale and shape parameters, respectively, can be estimated through the maximum likelihood method:

$$a = \min x_i \tag{4a}$$

$$1/c = (1/n) \sum_{\min x_i}^{n} \ln(x_i / a) \tag{4b}$$

Unfortunately, since the estimation of the Pareto parameters are based on a minimum we can get some error even after analysing a large volume of data.

The median of the Pareto distribution is calculated by:

$$m_e(N_S) = a\, 2^{1/c} \tag{5}$$

In order to evaluate the statistics of N_S new experiments were developed. In order to compare results we consider not only the 2-to-1 multiplexer but also the 4-bit parity (even) checker. The 4-bit parity checker has a truth table with four inputs $\mathbf{X} = \{A_3, A_2, A_1, A_0\}$ and one output $\mathbf{Y} = \{P\}$. The size of the matrix is $r \times c = 4 \times 4$ and the chromosome length is $CL = 48$.

This experiments consider the gate sets of Table 1 for $n = 1000$ simulations.. Figure 7 presents the results obtained comparing the Pareto median with the experimental median, for the two circuits. The other gate sets lead to similar results.

For the 4-bit parity checker it is used the Gset 1b (with Gset 1a it was not possible to achieve a solution), and results are not presented for $P = 10^4$ because a solution always appears on the initial population.

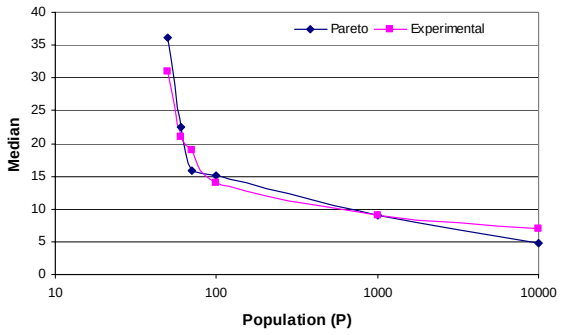

Fig. 7. Medians *versus* P with $CR = 95\%$ and $MR = 5\%$ for the 2-to-1 Multiplexer, using Gset 1a and $n = 1000$.

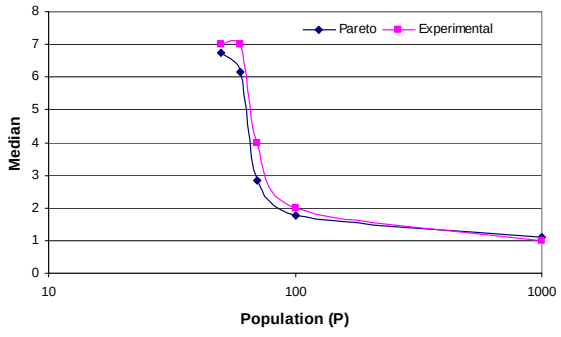

Fig. 8. Medians *versus* P with $CR = 95\%$ and $MR = 5\%$ for the 4-bit Parity Checker, using Gset 1b and $n = 1000$.

We observe a close agreement but the need of a large number of experiments for estimating (a, c).

Bearing this statistical requirements in mind, figure 9 depict the parameters (a, c) for $CR = 95\%$, $MR = 5\%$, $P = 70$, $n = 1000$ and the six gate sets.

We have inverse monotonic variation of (a, c), with the gate set complexity.

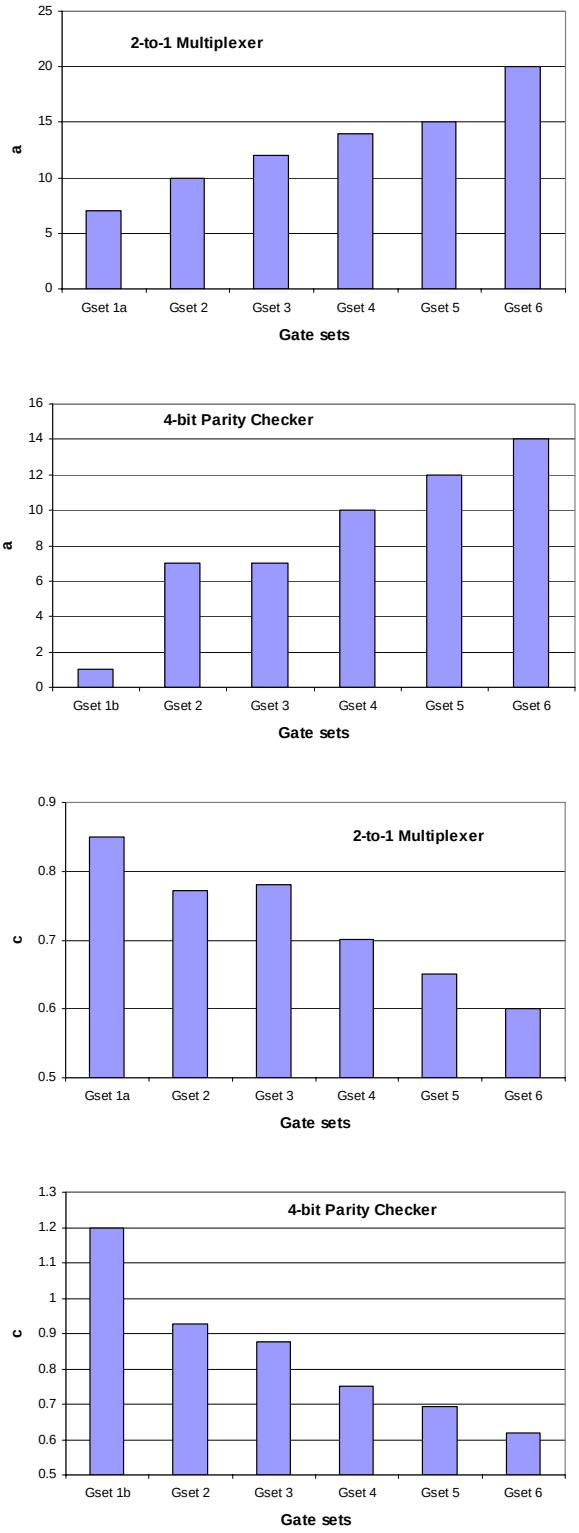

Fig. 9. Parameters (a, c) *versus* Gsets with $CR = 95\%$, $MR = 5\%$, $P = 70$ and $n = 1000$ for the 2-to-1 Multiplexer and the 4-bit Parity Checker.

Another important aspect consists on the variability of the GA to attain the solution. In this line of thought the inter-decile interval I_D are also calculated in figure 10 for both circuits.

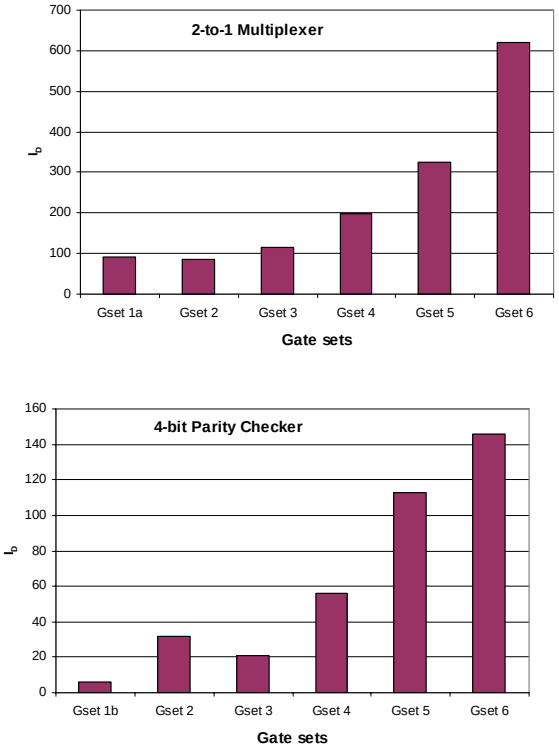

Fig. 10. I_D *versus* Gsets with *CR* = 95%, *MR* = 5% and *P* = 70 for the 2-to-1 Multiplexer and the 4-bit Parity Checker.

We verify again a monotonic-like variation leading to a large variability, particularly for Gsets 4, 5 and 6.

VI. SUMMARY AND CONCLUSIONS

This paper studied the influence of the population size on the processing time to obtain a solution, in this particular case consists on a digital circuit. The results demonstrate that, even the GA required number of generations decreases as the population size increases, the best processing time to achieve a solution occur for small population sizes.

It was also analyzed the performance of several different gate sets. Following the RISC *vs* CISC processor dilemma we conclude that the RISC gate sets have superior performances to the CISC gate sets. Nevertheless, we have to be careful when using gate sets with only one logic gate, because it depends of the digital circuit that we are generating. Consequently, the extreme case of RISC sets can be not robust to distinct circuit demands.

An important theoretical conclusion is that the number of generations to achieve a solution has a Pareto distribution. Therefore, we have a mathematical basis to evaluate the statistical performance of the GA schemes.

V. REFERENCES

[1] Nazan, K., "Population Sizing in Genetic and Evolutionary Algorithms" in *Proceedings of the Genetic and Evolutionary Computation Conference*.

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

[3] Goldberg, D., Deb, K. and Clark, J., "Genetic Algorithms, Noise, and the Sizing of Populations". *Complex Systems*, Vol. 6, 1992, pp 333-362.

[4] Harik, G., Cantu-Paz, E., Goldberg, D. And Miller, B., "The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations" in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1997, pp 7-12.

[5] Goldberg, D. E., *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Kluwer Academic Publishers, 2002.

[6] De Jong, K. A, "An Analysis of the Behavior of a Class of Genetic adaptative Systems", Doctoral dissertation, University of Michigan, 1975.

[7] Miller, B. L., "Noise, Sampling, and efficient genetic algorithms", Doctoral dissertation, University of Illinois at Urbana-Champaing, Urbana, IL, 1997.

[8] Morrison, R. W., "Dispersion-Based Population Initialization" in *Proceedings of the Genetic and Evolutionary Computation Conference – GECCO 2003,* pp 1210-1221.

[9] Holland, J., "Genetic Algorithms and the Optimal Allocations of Trails" in *SIAM Journal of Computing,* Vol 2(2), 1973, pp 88-105.

[10] Grefenstette, J. J., "Optimization of Control Parameters for Genetic Algorithms", *in IEEE-SMC-16*, 1986, pp122-128.

[11] Goldberg, D. E., "Optimal Initial Population Size for Binary-Coded Genetic Algorithm" in TCGA Report No 85001, Tuscaloosa: University of Alabama, 1985.

[12] Alander, J. T., "On Optimal Population Size of Genetic Algorithms"n Size for Binary-Coded Genetic Algorithm" in *Proceedings of CompEuro 92,* IEEE Computer Society Press, 1992, pp 65-70.

[13] Reeves, C. R., "Using Genetic Algorithms with Small Populations" in *Proceedings of the Fifth International Conference On Genetic Algorithms,* 1993, pp 92-99.

[14] Goldberg, D. E., Sastry, K. and Latoza, T., "On the Supply of Building Blocks", in *Proceedings of GECCO-2002*, 2002, pp 333-362.

[15] Holland, J., "Adaptation in Natural and Artificial Systems" Ann Arbor, MI:University of Michigan Press.

[16] Goldberg, D. E. and Rudnick, M., "Genetic Algorithms and the Variance of Fitness", in *Complex Systems*, Vol 5, 1991, pp 265-278.

[17] Sastry, K., "Analysis of Mixing in Genetic Algorithms: A Survey", in *IlliGAL Report No. 2002012*, 2002, University of Illinois at Urbana Champaign.

[18] Reis, C. and Machado, J. A. T., "An Evolutionary Approach to the Synthesis of Combinational Circuits", in *Proceedings of the IEEE International Conference on Computational Cybernetics,* 2003.