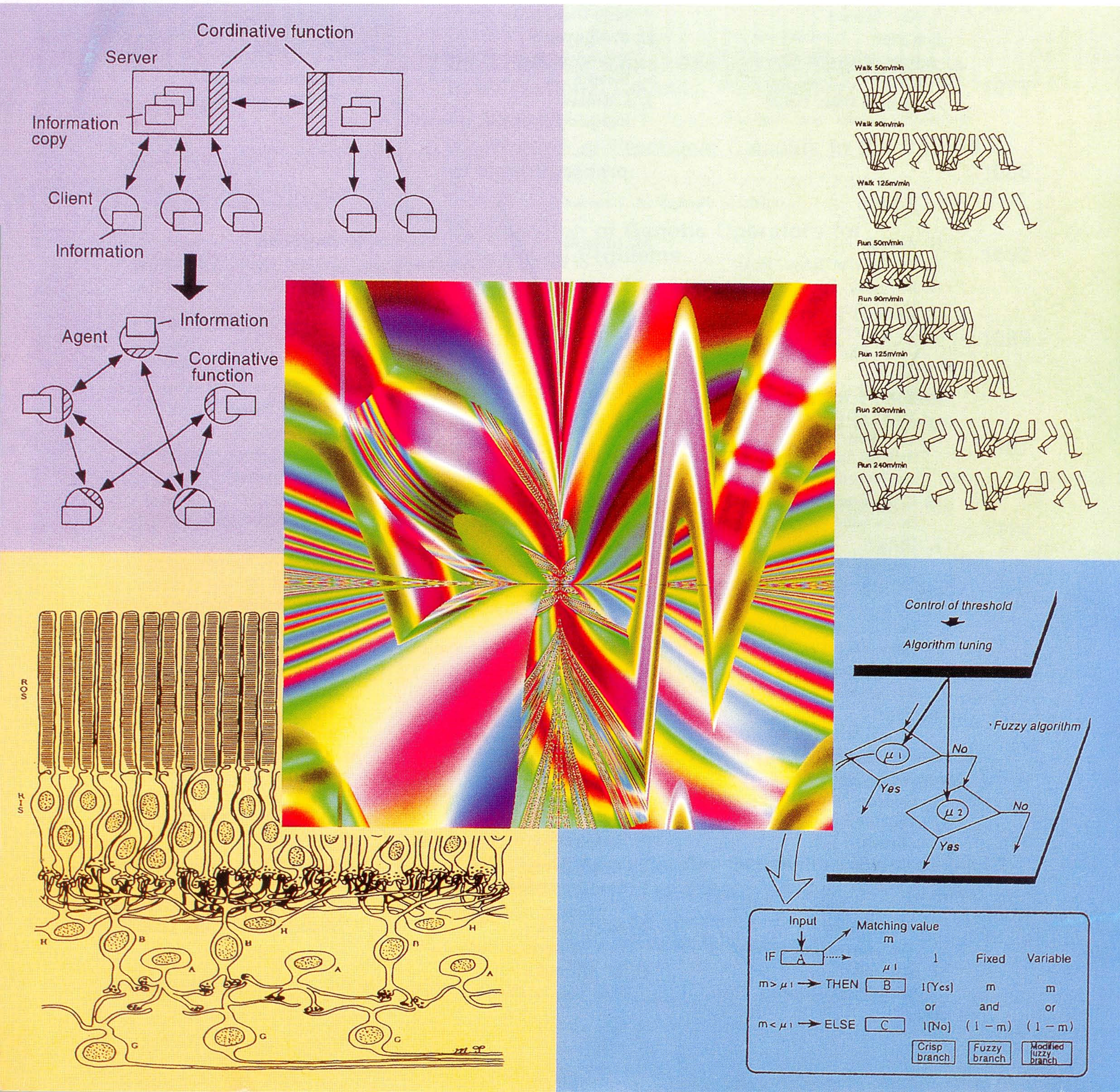
Journal of Advanced Computational Intelligence and Intelligent Informatics 第１１集 第９号 通巻第５９号 平成１９年１１月２０日発行 ISSN 1343-0130

Journal of Advanced Computational Intelligence and Intelligent Informatics

Vol.11 No.9
Nov. 2007

In Cooperation with
International Fuzzy Systems Association (IFSA),
Japan Society for Fuzzy Theory and intelligent informatics (SOFT),
Brazilian Society of Automatics (SBA),
Society of Instrument and Control Engineers (SICE, Japan),
John von Neumann Computer Society (NJSZT),
Vietnamese Fuzzy Systems Society (VFSS),
Fuzzy Systems and Intelligent Technologies Research Society of Thailand (FIRST),
The Korean Fuzzy Logic and Intelligent System Society (KFIS), and
The Taiwanese Association for Artificial Intelligence (TAAI)



Paper:

Computational Intelligence in Circuit Synthesis

Cecília Reis and J. A. Tenreiro Machado

Institute of Engineering of Porto / GECAD, Rua Dr. António Bernardino de Almeida, Porto, Portugal
E-mail: {cmr, jtm}@isep.ipp.pt

[Received March 13, 2007; accepted June 14, 2007]

This paper is devoted to the synthesis of combinational logic circuits through computational intelligence or, more precisely, using evolutionary computation techniques. Are studied two evolutionary algorithms, the Genetic and the Memetic Algorithm (GAs, MAs) and one swarm intelligence algorithm, the Particle Swarm Optimization (PSO). GAs are optimization and search techniques based on the principles of genetics and natural selection. MAs are evolutionary algorithms that include a stage of individual optimization as part of its search strategy, being the individual optimization in the form of a local search. The PSO is a population-based search algorithm that starts with a population of random solutions called particles. This paper presents the results for digital circuits design using the three above algorithms. The results show the statistical characteristics of this algorithms with respect to the number of generations required to achieve the solutions. The article analyzes also a new fitness function that includes an error discontinuity measure, which demonstrated to improve significantly the performance of the algorithm.

Keywords: computational intelligence, evolutionary algorithms, swarm intelligence, logic circuits design

1. Introduction

In recent decades computational intelligence by means of evolutionary computation techniques have been applied to the design of electronic circuits and systems, leading to a novel area of research called Evolutionary Electronics (EE) or Evolvable Hardware (EH) [3]. EE considers the concept for automatic design of electronic systems. Instead of using human conceived models, abstractions and techniques, EE employs search algorithms to develop implementations not achievable with the traditional design schemes, such as the Karnaugh or the Quine-McCluskey Boolean methods [1,4,5].

This paper proposes three evolutionary techniques for the design of combinational logic circuits, namely a Genetic Algorithm (GA), a Memetic Algorithm (MA) and a Particle Swarm Optimization (PSO) scheme.

Bearing these ideas in mind, the organization of this article is as follows. Section 2 presents the GA, the MA is described in section 3 and the PSO is detailed in section 4. Section 5 exhibits the computational results. Finally, section 6 outlines the main conclusions.

Table 1. Gate sets.

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

2. The Genetic Algorithm

Genetic Algorithms are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and genetic. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem [2].

First pioneered by John Holland in the 60s, GAs has been widely studied, experimented and applied in many fields in engineering worlds. Not only does GAs provide an alternative methods to solving problem, it consistently outperforms other traditional methods in most of the problems.

GAs were introduced as a computational analogy of adaptive systems. They are modelled loosely on the principles of the evolution via natural selection, employing a population of individuals that undergo selection in the presence of operators such as mutation and recombination (crossover). A fitness function is used to evaluate individuals, and reproductive success varies with fitness.

In this section we present the adopted GA, in terms of the circuit encoding, the genetic operators and the fitness function.

2.1. Problem Definition and Circuit Encoding

A truth table specifies the circuits and the goal is to implement a functional circuit with the least possible complexity. Four sets of logic gates have been defined, as shown in **Table 1**, being Gset 2 the simplest one and Gset 6 the most complex gate set. Logic gate named WIRE means a logical no-operation.

In the presented scheme the circuits are encoded [6] as a rectangular matrix **A** (*row*×*column* = *r*×*c*) of logic cells.

Three genes represent each cell: <*input1*> <*input2*> <*gate type*>, where <*input1*> and <*input2*> 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 with as many triplets as the matrix size demands (e.g. triplets $= 3 \times r \times c$).

2.2. 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 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, the mutation rate *MR* is the percentage of the population *P* that can mutate in each generation.

2.3. The Fitness Function

The calculation of the fitness function *F* in Eq. (1) has two parts, f_1 and f_2, where f_1 measures the functionality and f_2 measures the simplicity. 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 occurs when we have a functional circuit. Once the circuit is functional, in a second phase, the algorithm 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:

- First phase, circuit functionality:

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

$$\begin{array}{l} f_{11} = f_{11} + 1, \\ \textit{if } \{\textit{bit } i \textit{ of } \mathbf{Y}\} = \{\textit{bit } i \textit{ of } \mathbf{Y_R}\} \\ i = 1,\ldots,f_{10} \end{array} \quad \ldots\ldots \quad (1b)$$

$$\begin{array}{l} f_1 = f_{11} - \delta, \\ \textit{if } error_i \neq error_{i-1}, \quad i = 1,\ldots,f_{10} \end{array} \quad \ldots \quad (1c)$$

(when measuring discontinuity)

- Second phase, circuit simplicity:

$$f_2 = f_2 + 1, \quad \textit{if gate type} = \textit{wire} \quad \ldots\ldots \quad (1d)$$

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

where $i = 1,\ldots,f_{10}$, *ni* and *no* represent the number of inputs and outputs of the circuit.

3. The Memetic Algorithm

This section describes the MA. MAs are inspired by models of adaptation in natural systems that combine evolutionary adaptation of populations of individuals with individual learning within a lifetime [10]. As it is known, MAs are metaheuristics that take advantage of the evolutionary operators in determining interesting regions of the search space. Moreover, MAs adopt a local search that rapidly finds good solutions in a small region of the search space [11]. Additionally, MAs are inspired by Richard Dawkins' concept of a meme, which represents a unit of cultural evolution that can exhibit local refinement [12].

The proposed MA includes a GA and a local search algorithm, where the GA corresponds to the algorithm implemented in first stage of development.

Over the last decade, MAs have relied on the use of a variety of different methods as the local improvement procedure. Some recent studies on the choice of local search method employed have shown that this choice significantly affects the efficiency of problem searches [13].

The local search method investigates a small area around a solution and adopts the best-found solution. By other words, the procedure tries to find a fitter solution in the neighborhood of the current solution. If the algorithm finds a better solution, then the new solution replaces the current solution, and the neighborhood restarts. Local search methods are iterative algorithms that seek to enhance the solution by stepwise improvements. The simplest form of local search attempts to swap elements in combinatorial optimization problems.

In our case, it is implemented a gate type local search (GTLS) algorithm as shown in **Fig. 1**.

4. The Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a recently proposed algorithm by James Kennedy and R. C. Eberhart

```
For all population
  For the entire chromossome
    Substitute de gene gate type with a neighbour
      If the new solution has better fitness function
        New solution replaces old solution
  End for
End
```

Fig. 1. The local search algorithm.

in 1995, motivated by social behavior of organisms such as bird flocking and fish schooling. PSO algorithm is not only a tool for optimization, but also a tool for representing sociocognition of human and artificial agents, based on principles of social psychology [14]. Some scientists suggest that knowledge is optimized by social interaction and thinking is not only private but also interpersonal. PSO as an optimization tool, provides a population-based search procedure in which individuals called particles change their position (state) with time. In a PSO system, particles fly around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience, and according to the experience of a neighboring particle, making use of the best position encountered by itself and its neighbor. Thus, as in modern GAs and MAs, a PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation [15].

4.1. PSO Parameters

In the PSO, instead of using genetic operators, as in the case of GAs, each particle (individual) adjusts its flying according with its own and its companions experiences. Each particle is treated as a point in a D-dimensional space and is manipulated as described below in the original PSO algorithm:

$$v_{id} = v_{id} + c_1\, rand()(p_{id} - x_{id}) + c_2\, rand()(p_{gd} - x_{id}) \quad \dots\dots\dots\dots (2)$$

$$x_{id} = x_{id} + v_{id} \quad \dots\dots\dots\dots\dots\dots (3)$$

where c_1 and c_2 are positive constants and *rand*() is a random function in the range $[0,1]$, $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ represents the ith particle, $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ is the best previous position (the position giving the best fitness value) of the particle, the symbol g represents the index of the best particle among all particles in the population, and $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ is the rate of the position change (velocity) for particle i.

Expression (2) represents the flying trajectory of a population of particles. Eq. (2) describes how the velocity is dynamically updated and Eq. (3) the position update of the "flying" particles. Eq. (2) is divided in three parts, namely the momentum, the cognitive and the social parts. In the first part the velocity cannot be changed abruptly: it is adjusted based on the current velocity. The second part represents the learning from its own flying experience. The third part consists on the learning group flying experience [15, 16].

The initial velocity of each particle is initialized with zero. The velocities of the following generations are calculated applying Eq. (2) and the new positions result from using Eq. (3). In this way, each potential solution, called particle, flies through the problem space. For each gene is calculated the corresponding velocity. Therefore, the new positions are as many as the number of genes in the chromosome. If the new values of the input genes result out of range, then a re-insertion function is used. If the calculated gate gene is not allowed a new valid one is generated at random. These particles then have memory and each one keeps information of its previous best position (*pbest*) and its corresponding fitness. The swarm has the *pbest* of all the particles and the particle with the greatest fitness is called the global best (*gbest*).

The basic concept of the PSO technique lies in accelerating each particle towards its *pbest* and *gbest* locations with a random weighted acceleration. However, in our case we also use a kind of mutation operator that introduces a new cell in 10% of the population. This mutation operator changes the characteristics of a given cell in the matrix. Therefore, the mutation modifies the gate type and the two inputs, meaning that a completely new cell can appear in the chromosome.

To run the PSO we have also to define the number P of individuals to create the initial population of particles. This population is always the same size across the generations, until reaching the solution.

5. Computational Results

This section presents the computational results when applying, firstly the evaluation of the fitness function without the error discontinuity measure and secondly using the fitness function with the error discontinuity measure.

Due to the stochastic nature of the EAs, in order to evaluate its performance, for each gate set we perform 20 simulations. The best gate set is the one that requires the smaller average number of generations $Av(N)$ and the smaller standard deviation $S(N)$ to reach the solution.

5.1. Using the Fitness Without Discontinuity Measure

This subsection shows the implementation of four different combinational logic circuits, namely, a 2-to-1 multiplexer ($M2-1$), a one-bit full adder (*FA1*), a one-bit full subtractor (*FS1*) and a four-bit parity checker (*PC4*), using the GA, the MA and the PSO algorithms, using the fitness function described above, without the discontinuity error measure, that is with $\delta = 0$.

The first case study is the $M2-1$ circuit, with a truth table with three inputs $\{S_0, I_1, I_0\}$ and one output $\{O\}$. The matrix has a size of $r \times c = 3 \times 3$ and the length of

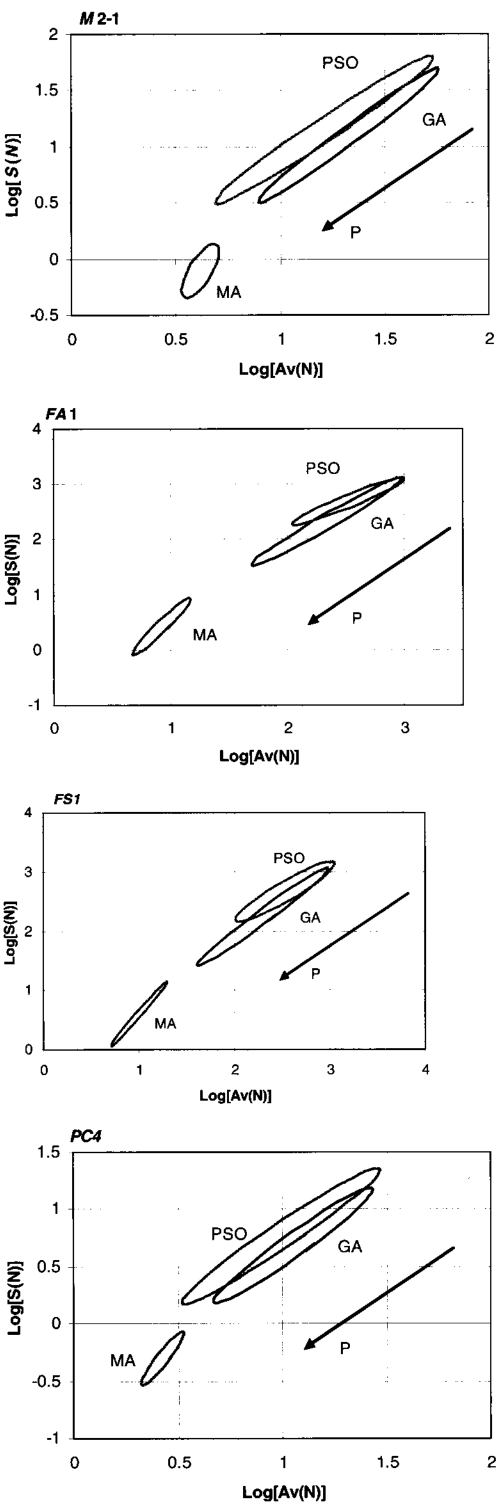


Fig. 2. Log[$S(N)$] *versus* Log[$Av(N)$] for the $M2-1$, the *FA1*, the *FS1* and the *PC4* circuits with $P = \{100, 500, 1000, 3000\}$ and $\delta = 0$.

each string representing a circuit (i.e., the chromosome length) is $CL = 27$. Since the 2-to-1 multiplexer has $ni = 3$ and $no = 1$, it results $f_{IO} = 8$ and $F \geq 12$.

The second case study is the *FA1* circuit, with a truth table with three inputs {A, B, C_{in}} and two outputs {S, C_{out}}. In this case, the matrix has a size of $r \times c = 3 \times 3$, and the chromosome length is $CL = 27$. Since the one-bit full adder has $ni = 3$ and $no = 2$, it results $f_{IO} = 16$ and $F \geq 20$.

The third case study is a *FS1* circuit, with a truth table with three inputs {A, B, B_{in}} and two outputs {S, B_{out}}. In this case, the matrix has a size of $r \times c = 3 \times 3$, and the chromosome length is $CL = 27$. Since the one-bit full adder has $ni = 3$ and $no = 2$, it results $f_{IO} = 16$ and $F \geq 20$.

The fourth case study consists on the *PC4* circuit, which has four inputs {A_3, A_2, A_1, A_0} and one output {O}. The size of the matrix is $r \times c = 4 \times 4$ and the chromosome length is $CL = 48$. In this case $ni = 4$ and $no = 1$, resulting $f_{IO} = 16$ and $F \geq 24$.

Figure 2 presents the results obtained in terms of Log[$S(N)$] *versus* Log[$Av(N)$] for the $M2-1$, the *FA1*, the *FS1* and the *PC4* circuits and $P = \{100, 500, 1000, 3000\}$ with $\delta = 0$.

The points in the space {Log[$Av(N)$], Log[$S(N)$]} are approximated by a bi-dimensional Gaussian probability distribution. The ellipses depicted in the charts represent the corresponding contour plots.

It is obvious that the MA algorithm reveals a better performance for all the combinational circuits and that both $Av(N)$ and $S(N)$ vary inversely with P. The GA and the PSO algorithms present similar results in particular for the $M2-1$ and the *PC4* circuits. For the *FA1* and the *FS1* the PSO is less sensitive to P then the GA.

Figure 3 shows Log[$S(N)$] *versus* Log[$Av(N)$] with $P = \{100, 500, 1000, 3000\}$ for the GA, the MA and the PSO algorithms, with $\delta = 0$.

Analyzing the charts is possible to classify the complexity of the combinational logic circuits in the perspective of each evolutionary algorithm. For the three algorithms, the sequence of increasing circuit complexity becomes {*PC4*, $M2-1$, *FA1*, *FS1*}. In the PSO algorithm, the circuit complexity is clearly divided in two zones, namely the {*FS1*, *FA1*} and the {*M2-1*, *PC4*} zones.

5.2. Using the Fitness with Discontinuity Measure

The experiments of this subsection consist on running the PSO algorithm to generate the one-bit full adder (*FA1*) using the fitness scheme described in Eq. (1). The circuit is generated with the gate sets presented in **Table 1** and $P = 3000$, $w = 0.5$, $c_1 = 1.5$ and $c_2 = 2$.

Figure 4 depict the average number of generations $Av(N)$ and the standard deviation of the number of generations $S(N)$ to achieve the solution *versus* δ for the PSO algorithm, the circuit {*FA1*} and the gate sets {2, 3, 4, 6}.

The results reveal that Gset 3 presents a superior performance to the other Gsets, for all values of δ. Moreover, analyzing the influence of δ we conclude that the PSO response is better mostly in the region $\delta \approx 0.5$ for the arithmetic circuit and for all gate sets.

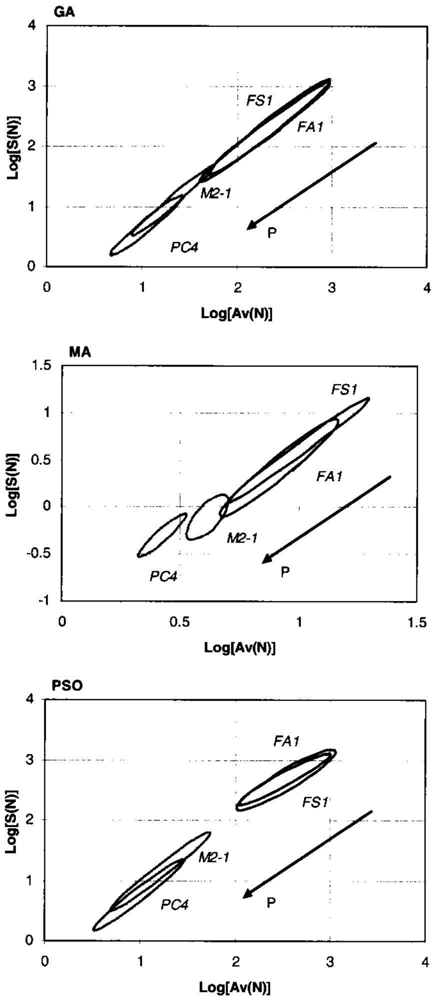


Fig. 3. Log[$S(N)$] *versus* Log[$Av(N)$] for $P = \{100, 500, 1000, 3000\}$ for the GA, the MA and the PSO algorithms, with $\delta = 0$.

6. Conclusions

This paper studied the implementation of combinational logic circuits using three evolutionary algorithms. The results reveal that the population size has influence upon the results and that Log[$S(N)$] has a linear dependence with Log[$Av(N)$], meaning that $S(N) \sim [Av(N)]^{\alpha}$.

The superior performance of the MA algorithm is obvious for all gate sets and all circuits. Moreover, the adopted methodology leads to a classification scheme for combinational logic circuits in terms of their complexity.

The PSO based algorithm for the design of arithmetic circuits is improved when the error discontinuity measure is implemented in the fitness function.

All the gate sets are suitable to the design of logic circuits being the Gset 3 the one that demonstrates higher performance in terms of $Av(N)$ and $S(N)$. Furthermore, the inclusion of a continuity measure in the fitness function leads to a superior performance in the PSO.

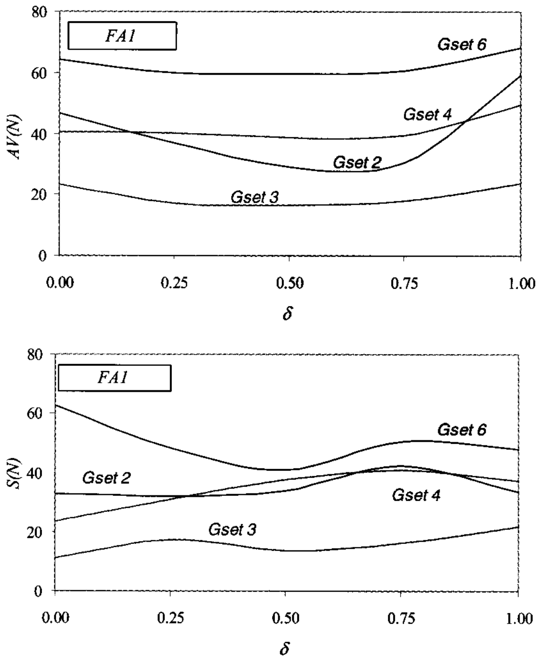


Fig. 4. Average number of generations $Av(N)$ and Standard deviation of the number of generations $S(N)$ to achieve a solution for the FA1 circuit versus $\delta \in [0,1]$, with Gsets {6, 4, 3, 2}.

References:

[1] S. Louis and G. Rawlins, "Designer Genetic Algorithms: Genetic Algorithms in Structure Design," Proc. of the Fourth Int. Conf. on Genetic Algorithms, 1991.

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

[3] R. S. Zebulum, M. A. Pacheco, and M. M. Vellasco, "Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms," CRC Press, 2001.

[4] J. R. Koza, "Genetic Programming," On the Programming of Computers by means of Natural Selection, MIT Press, 1992.

[5] C. A. Coello, A. D. Christiansen, and A. H. Aguirre, "Using Genetic Algorithms to Design Combinational Logic Circuits," Intelligent Engineering through Artificial Neural Networks, Vol.6, pp. 391-396, 1996.

[6] C. Reis, J. A. T. Machado, and J. B. Cunha, "Evolutionary Design of Combinational Logic Circuits," Journal of Advanced Computational Intelligence and Intelligent Informatics, Fuji Technology Press, Vol.8, No.5, pp. 507-513, Sept. 2004.

[7] C. Reis, J. A. T. Machado, and J. B. Cunha, "Evolutionary design of combinational circuits using fractional-order fitness," Proc. of the Fifth EUROMECH Nonlinear Dynamics Conference, pp. 1312-1321, 2005.

[8] C. Reis, J. A. T. Machado, and J. B. Cunha, "An Evolutionary Hybrid Approach in the Design of Combinational Digital Circuits," WSEAS Transactions on Systems, Issue 12, Vol.4, pp. 2338-2345.

[9] P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms," Tech. Rep. Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA, 1989.

[10] N. Krasnogor, "Studies on the Theory and Design Space of Memetic Algorithms," Ph.D. thesis, University of the West of England, Bristol, June, 2002.

[11] P. Merz and B. Freisleben, "A Comparison of Memetic Algorithms," Tabu Search, and Ant Colonies for the Quadratic Assignment Problem, Proc. of the 1999 Congress on Evolutionary Computation, IEEE Press, pp. 2063-2070, 1999.

[12] R. Dawkins, "The Selfish Gene," Oxford University Press, New York, 1976.

[13] Y. S. Ong and A. J. Keane, "Meta-Lamarckian in Memetic Algorithm," IEEE Transactions On Evolutionary Computation, Vol.8, No.2, pp. 99-110, April, 2004.

[14] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," Proc. of the IEEE Int. Conf. Neural Networks, pp. 1942-1948, 1995.

[15] Y. Shi and R. C. Eberhart, "A Modified Particle Swarm Optimizer," Proc. of the 1998 Int. Conf. on Evolutionary Computation, pp. 69-73, 1998.

[16] M. Clerc and J. Kennedy, "The Particle Swarm: explosion, stability, and convergence in a multi-dimensional complex space," IEEE Trans. on Evolutionary Comp., Vol.6, pp. 58-73, 2002.

Name:
Cecília Maria do Rio Fernandes Moreira Reis

Affiliation:
Professor, Institute of Engineering of Porto, Dept. of Electrotechnical Engineering

Address:
Rua Dr. António Bernardino de Almeida, 4200-072 Porto, Portugal

Brief Biographical History:
1992 'Licenciatura' degree in Electrical Engineering at the Institute of Engineering of Porto, Portugal
1995 MSc degree in Electrical and Computer Engineering at the Faculty of Engineering of the University of Porto, Portugal

Main Works:

- C. Reis and J. A. T. Machado, "An Evolutionary Approach to the Synthesis of Combinational Circuits," IEEE Int. Conf. on Computational Cybernetics, Hungary, 2003.
- C. Reis, J. A. T. Machado, and J. B. Cunha, "Evolutionary Design of Combinational Logic Circuits," Journal of Advanced Computational Intelligence and Intelligent Informatics, Fuji Technology Press, Vol.8, No.5, pp. 507-513, Sep. 2004.
- C. Reis, J. A. T. Machado, and J. B. Cunha, "Synthesis of Digital Circuits with Genetic Algorithms: A Fractional-Order Approach," International Journal of Computational Intelligence, Vol.1, No.4, pp. 289-294, Dec. 2004.
- C. Reis, J. A. T. Machado, and J. B. Cunha, "New Concepts Towards the Synthesis of Digital Circuits Through Genetic Algorithms," Intelligent Systems at the Service of Mankind, Vol.II, pp. 377-388, Ubooks, Dec. 2005.
- C. Reis, J. A. T. Machado, and J. B. Cunha, "Logic Circuits Synthesis Through Genetic Algorithms," WSEAS Transactions on Information Science and Applications, Issue 5, Vol.2, pp. 618-623, May 2005.
- C. Reis, J. A. T. Machado, and J. B. Cunha, "Fractional Dynamic Fitness Functions for GA-based Circuit Design," Proc. of GECCO 2005 - Genetic and Evolutionary Computation Conference, Washington, DC, USA, pp. 1571-1572, June 25-29, 2005.
- C. Reis, J. A. T. Machado, and J. B. Cunha, "Evolutionary Techniques in Circuit Design and Optimization," WSEAS Transactions on Power Systems, Issue 7, Vol.1, pp. 1337-1342, July 2006.

Name:
J. A. Tenreiro Machado

Affiliation:
Coordinator Professor, Institute of Engineering of Porto, Dept. of Electrotechnical Engineering

Address:
Rua Dr. António Bernardino de Almeida, 4200-072 Porto, Portugal

Brief Biographical History:
'Licenciatura' (1980), Ph.D. (1989), and 'Agregation' (1995) degrees in Electrical and Computer Engineering at the Faculty of Engineering of the University of Porto, Portugal

Main Works:

- J. A. T. Machado, J. L. M. de Carvalho, and A. M. S. F. Galhano, "Analysis of Robot Dynamics and Compensation Using Classical and Computed Torque Techniques," IEEE Trans. on Education, Vol.36, No.4, pp. 372-379, Nov. 1993.
- J. A. T. Machado and A. M. S. F. Galhano, "Benchmarking Computer Systems for Robot Control," IEEE Trans. on Education, Vol.38, No.3, pp. 205-210, Aug. 1995.
- J. A. T. Machado, "Fractional-Order Derivative Approximations in Discrete-Time Control Systems," Systems Analysis-Modelling-Simulation, Gordon & Breach Science Publishers, Vol.34, pp. 419-434, 1999.
- J. A. T. Machado, "Implementing Discrete-Time Fractional-Order Controllers," Journal of Advanced Computational Intelligence, Fuji Technology Press, Vol.5, No.5, pp. 279-285, ISSN 1343-0130, Oct. 2001.
- A. M. S. F. Galhano and J. A. T. Machado, "Kinematic Robustness of Manipulating Systems," Journal of Advanced Computational Intelligence, Fuji Technology Press, Vol.6, No.2, pp. 93-98, ISSN 1343-0130, June 2002.
- J. A. T. Machado, "A Probabilistic Interpretation of the Fractional-Order Differentiation," Journal of Fractional Calculus & Applied Analysis, Vol.6, No.1, pp. 73-80, 2003.
- E. J. S. Pires, J. A. T. Machado, and P. B. de Moura Oliveira, "Fractional Order Dynamics in a GA Planner," Journal of Signal Processing, Elsevier, Special Issue on Fractional Signal Processing and Applications, Vol.83, Issue 11, pp. 2377-2386, Nov. 2003.
- J. A. T. Machado and I. Jesus, "A Suggestion From the Past?" FCAA - Journal of Fractional Calculus and Applied Analysis, Vol.7, No.4, pp. 403-407, 2004.
- R. S. Barbosa, J. A. T. Machado, and I. M. Ferreira, "Controller Tuning Based on Bode's Ideal Transfer Function," Journal of Signal Processing, Elsevier, Nonlinear Dynamics, Kluwer, Vol.38, Nos.1-4, pp. 305-321, ISBN 0924-090X, Dec. 2004.
- L. Figueiredo, J. A. T. Machado, and J. Ferreira, "Dynamical Analysis of Freeway Traffic," IEEE Transactions on Intelligent Transportation Systems, Vol.5, No.4, pp. 259-266, Dec. 2004.
- N. M. F. Ferreira, J. A. T. Machado, and J. B. R. da Cunha, "Fractional-Order Position/Force Robot Control," Journal of Advanced Computational Intelligence and Intelligent Informatics, Fuji Technology Press, Vol.9, No.4, pp. 379-386, ISSN 1343-0130, Jul. 2005.
- M. F. Silva, J. A. T. Machado, and A. M. Lopes, "Modelling and Simulation of Artificial Locomotion Systems," Robotica, Vol.23, Part 5, pp. 595-606, ISSN 0263-5747, Sep. 2005.
- J. A. T. Machado, I. S. Jesus, and A. Galhano, "Fractional Order Electromagnetics," Signal Processing, Elsevier, Vol.86, pp. 2637-2644, 2006.
- E. J. S. Pires, J. A. T. Machado, and P. B. de Moura Oliveira, "Manipulator Trajectory Planning using a MOEA," Applied Soft Computing Journal, Elsevier, Vol.7, Issue 3, pp. 659-667, Jun. 2007.