EVOLUTIONARY SYNTHESIS OF QCA CIRCUITS: A CRITIQUE OF EVOLUTIONARY SEARCH METHODS BASED ON THE HAMMING ORACLE

R. Salas Machado, J. Castellanos, R. Lahoz-Beltra

***Abstract**: This paper introduces a discussion about evolutionary search methods based on Hamming oracle. In many optimization problems, the design of the fitness function includes the Hamming distance being referred this kind of functions as Hamming oracle. In this paper we adopt a critical look and ask ourselves to what extent genetic algorithms and other related evolutionary methods truly mimic evolution. We tested three evolutionary search methods taken as a case study the evolutionary synthesis of quantum-dot cellular automata circuits. Our main conclusion is that evolutionary search methods do not mimetic Darwinian evolution because knowledge is not obtained from the evolutionary surface exploration: evolution is the result of the 'knowledge' embedded by the researcher or human expert into the fitness function. Maybe a more appropriate denomination would be "combinatorial search algorithms" such as Minimax, Alpha-beta pruning, etc.*

***Keywords**: Evolutionary search methods, genetic algorithms, Dawkins weasel program, Hamming oracle*

***ACM Classification Keywords**: I.6 Simulation and Modeling*

Introduction

One of the key tasks in genetic and evolutionary algorithms is the evaluation of the quality, goodness or merit of a given solution, represented by an array, which is referred to chromosome. Generally, this evaluation is performed by an objective function or fitness that maps each chromosome or solution onto a real number, representing a measure of the optimality of a chromosome. In many optimization problems, the design of the fitness function includes the Hamming distance being referred this kind of functions as *Hamming oracle* (Figure 1). According to [Dembsky et al., 2007] an array or string, i.e. the chromosome, is then presented (*input*) to the Hamming oracle which assigns the string a rank, based on its proximity to a desired target (the *output* dependent on Hamming distance). In such cases, the desired target or optimal chromosome, could be defined by setting directly the optimal solution, e.g. in a theoretical study of the convergence of an evolutionary algorithm. In other cases, the desired target is defined by setting the quantitative and qualitative features for a given optimal solution, e.g. in industrial design optimization we could define the most appropriate strength, weight, cost, and durability of a product. However, today one of the biggest criticisms received by genetic algorithms and related evolutionary search techniques is that in many practical applications the fitness function contains a lot of information about the optimal solution, such 'knowledge' being provided by a human expert.

In this paper we adopt a critical look and ask ourselves to what extent genetic algorithms (GAs) and other related evolutionary methods truly mimic the evolution in the search for an optimal or near-optimal solution. We tested three evolutionary search methods taken as a case study the evolutionary synthesis of quantum-dot cellular automata circuits. Our assumption is that the researcher or human expert feeds the Hamming oracle with an excess of knowledge, so the fitness landscape reduces its area significantly.

During the last decade there has been an attempt to apply GAs to the efficient and optimal design of QCA circuits. In all the following examples, the fitness is the result of comparing the output of the evaluated circuit with a predefined truth table or desired target: evolution is guided by a Hamming oracle. For instance, [Kamrani et al. 2012] applied GAs to design QCA circuits optimizing the number of gates and clock cycles by utilization of NAND and inverter gates. In this approach a tree structure is used to represent a chromosome, i.e. a candidate QCA circuit. The fitness function evaluates as positive effects a size reduction of the tree, and therefore a small number of QCA cells. Consequently, the negative effects are a result of any enlargement of the tree, and thus a greater number of QCA cells. A similar approach is used by [Bonyadi et al., 2007] and [Houshmand et al., 2009, 2011] representing a chromosome as a tree structure. However, the tree is constructed with majority and inverter gates and the leaves can be either logical value 1 or Boolean variables. Once again, the fitness function minimize the number of gates through promoting the rule that 'fewer nodes is a better solution'. Applying a more sophisticated methodology [Vilela Neto et al., 2007] introduced a coevolutionary model of GA optimizing the desired logic by the evolution of circuit topology, type cell and clock zone. The fitness is calculated by comparing the output of the candidate circuit with a truth table. However, the 'oracle needs some extra help' including the fitness function some terms –depending on the specific problem - to prevent local trapping problems.

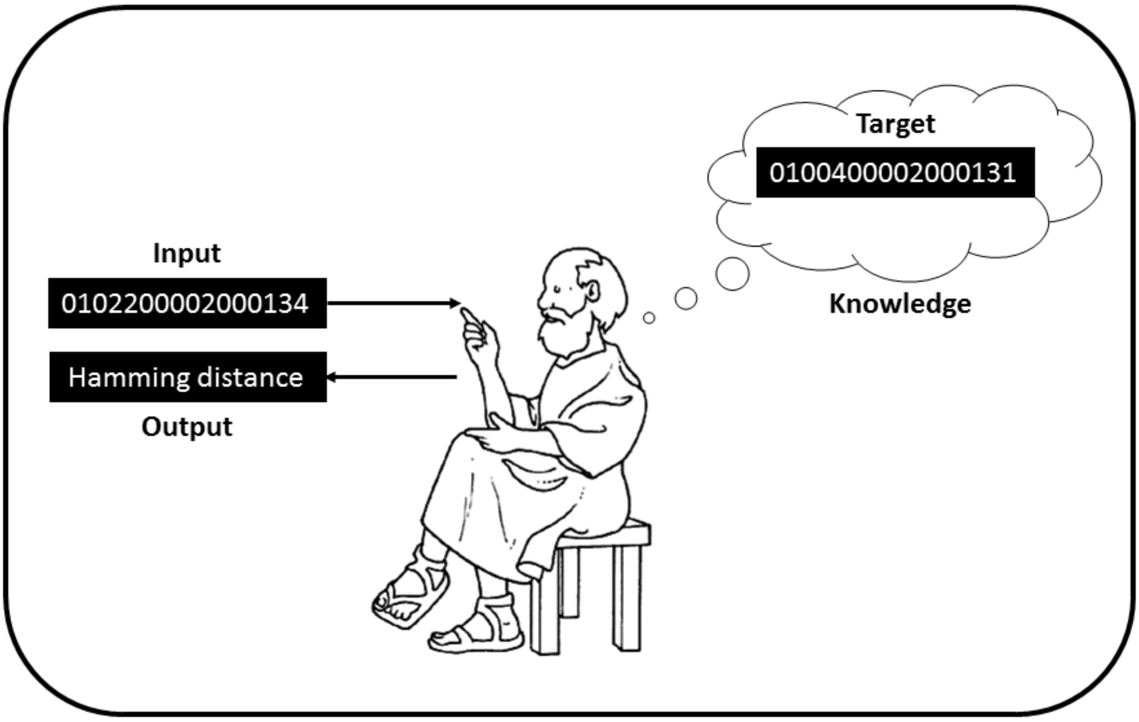

Figure 1.- Hamming oracle (for explanation see text)

Simulation experiments

Quantum-dot cellular automata (QCA) are a nanoscale technology based on Coulombic forces instead of current (e.g. CMOS) technology. A QCA cell (Figure 2) is the basic element being composed of four dots and two electrons. Electrons due to tunnel and the Coulomb repulsion effects occupy the dots, leading to two stable arrangements or polarizations which are encoded as 0 or 1. Thus, electrons arrange diagonally in order to be at a maximum distance from each other (Figure 2). QCA circuits are combinational logic circuits constructed from the binding of QCA cells, being the fundamental QCA logic devices: QCA wire, QCA majority gate and QCA inverter. Once a QCA circuit is designed the logical functionality of the circuit is tested through simulation. At present one of the most used tools to create and conduct the simulation of a novel QCA circuit is QCADesigner [Walus et al, 2004] [Walus Group, 2009].

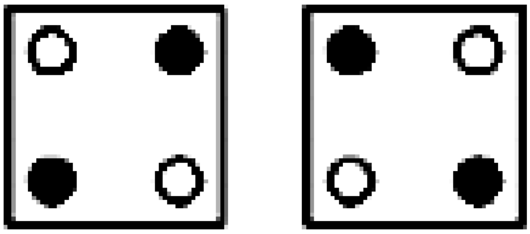

Figure 2.- QCA cell showing (left) polarization *P*=+1 ('bit state 1')
and (right) polarization *P*=-1 ('bit state 0')

Simulations experiments were conducting studying the synthesis of four QCA circuits (Figure 3): XOR gate, inverted XOR gate, AND gate, 5-input majority gate [Angizi et al., 2015]. After choosing these simple circuits, the circuits were designed with QCADesigner verifying their logical functionality with a bistable simulation engine [Walus et al., 2004], i.e. assuming that each cell is a simple two-state system. The two-state model assumes the following Hamiltonian:

$$H_i = \sum_j \begin{bmatrix} -\frac{1}{2} P_j E_{i,j}^k & -\gamma_i \\ -\gamma_i & \frac{1}{2} P_j E_{i,j}^k \end{bmatrix} \quad (1)$$

where $E_{i,j}^k$ is the kink energy between cell i and j, P_j is the cell polarization and γ is the tunneling energy. Using the Schrodinger equation the simulation engine obtains the state of each cell with respect to other cells that are close in the circuit, thus:

$$P_i = \frac{\frac{E_{i,j}^k}{2\gamma} \sum_j P_j}{\sqrt{1 + \left(\frac{E_{i,j}^k}{2\gamma} \sum_j P_j \right)}} \quad (2)$$

where P_i is the polarization state of cell i and P_j the polarization state of the cells in the neighborhood.

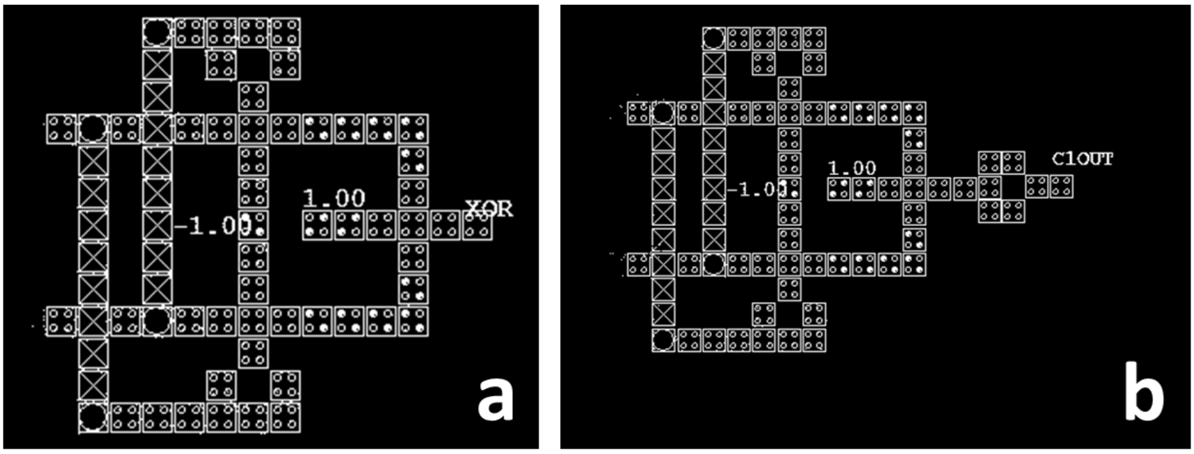

Figure 3.- QCA circuits designed and tested with QCADesigner: (a) XOR gate (b) inverted XOR.

Since QCA circuits are clocked, different clock zones were assigned to QCA cells, pumping the information through the circuit according to QCADesigner signal clocking. QCA cells perform computations using a four-phase clock representing such cells in different color according to its clock zone. Following, each circuit was transformed into an array of 13 rows and 19 columns (Figure 4), assigning to each cell a value: 0 (empty cell or layout without QCA cell), 1 (QCA cell with clock zone 0), 2 (QCA cell with clock zone 1), 3 (QCA cell with clock zone 2) or 4 (QCA cell with clock zone 3). According to Figure 4 each of arrays would be representing the desired or target QCA circuit.

```
#XOR

0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1, 0, 0, 2, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1, 0, 0, 2, 0, 3, 3, 4, 4, 4, 4, 0, 0, 0, 0,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 4, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1, 0, 0, 2, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0,
1, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

```
#INVERTED XOR

0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1, 0, 0, 2, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 4, 0, 0, 4, 4, 0, 0,
0, 0, 1, 0, 1, 0, 0, 2, 0, 3, 3, 4, 4, 4, 4, 4, 0, 4, 4,
0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 4, 0, 0, 4, 4, 0, 0,
0, 0, 1, 0, 1, 0, 0, 2, 0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0,
1, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 3, 3, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

```
#AND

0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 2, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

```
#5 INPUT MAJORITY

0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 2, 2, 2, 2, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 2, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
```

Figure 4.- QCA circuits represented by an array (for explanation see text)

Concluded the circuit design step, we studied the evolutionary synthesis of QCA circuits by different evolutionary algorithms which details are described below.

Evolutionary search was studied testing three evolutionary search and optimization methods:

- Random search (RS).- RS is an optimization algorithm that does not use the gradient of the problem. Thus, the algorithm allows all QCA cells to mutate in any generation, but with the possibility that correct letters can be mutated again. The algorithm consists of the following steps: (1) obtains a random initial string of same length as desired or target circuit, (2) apply the evolutionary feedback loop: (2.1) obtains the mutated offspring, (2.2) selects the best circuit and pass it to the next generation and (2.3) stop the loop when a target circuit is reached.
- Partitioned search (PS).- PS is the name given by [Dembsky et al., 2007] to denote the Dawkins' weasel program: an experiment with computer described in the book by Richard Dawkins entitled *The Blind Watchmaker* [Dawkins, 1986]. Based on 'infinite monkey theorem' the PS algorithm simulates the Darwin's cumulative selection principle: Darwin's theory of evolution by natural selection in organisms with asexual reproduction (individuals reproduce by dividing –bipartition- in two 'children'). In this case, QCA cells mutate in any generation, but correct positions are locked (i.e. preserved) as soon as they appear being impossible that correct letters can be mutated again. The algorithm consists of the following steps: (1) obtains a random initial string of same length as desired or target circuit, (2) apply the evolutionary feedback loop: (2.1) obtains the mutated offspring, (2.2) selects the best circuit and pass it to the next generation and (2.3) lock correct string positions preventing further mutation (2.4) stop the loop when a target circuit is reached.
- Darwinian selection pressure (DSP).- DSP is the name we have chosen to refer to a genetic algorithm without recombination or mutation [Alajmi and Wright, 2014]. Therefore, we study the time or number of generations required for the population to be composed of the best solutions (or circuits) found in the initial generation.

In this paper, the fitness of strings, i.e. QCA circuits, is calculated in RS and PS algorithms as follows:

$$f(s) = \frac{L - d_{s,\text{target}}^{\min}}{L} \tag{3}$$

Regarding the above expression s is a label that identifies a given circuit, L the length of string depicting the circuit and $d_{s,\text{target}}^{\min}$ the distance of the circuit closest to the target circuit. Note that for $d_{s,\text{target}}^{\min} = 0$, i.e. when the target circuit is reached, the fitness *f(s)* takes the maximum value being equal to 1.

In the case of DSP algorithm the fitness is given by the following expression:

$$f(s) = L - H \tag{4}$$

where H is the Hamming distance between the circuit s and the target circuit. Note that when the target circuit is reached then H is zero, and therefore the circuit is ranked with a maximum fitness value (in our simulation experiments, 247). Just as it was done in the expression (3) we could also have standardized (4) measure (maximum fitness value equal to 1) dividing by L:

$$f(s) = \frac{L - H}{L} \tag{5}$$

Applying probability theory [Dembsky et al., 2007] obtained the expressions to estimate the median number of queries required for success - the optimum circuit or target has been found - in RS and PS search algorithms (Table 1). In this study we defined such values like Q_1 and Q_2 for RS and PS algorithms, respectively. Details and mathematical model of these two algorithms are described in [Dembsky et al., 2007]. We also include 'takeover time' (Q_3, Table 1): a DSP performance measure introduced by [Bäck, 1996]. $Q3$ is defined as the number of generations or queries for the offspring (or population) to be filled with the best solutions (or circuits) found in the initial generation but in the absence of crossover and mutation.

Table 1.- Performance evaluation of evolutionary search algorithms

Evolutionary search algorithm	Performance measure
Random search	$Q_1 = -card(A)^L \ln(1-0.5)$
Partitioned search	$Q_2 = \dfrac{\log\left(1-0.5^{\frac{1}{L}}\right)}{\log\left(1-\dfrac{1}{card(A)}\right)}$
Darwinian selection pressure	$Q_3 = \dfrac{1}{\ln(n)}\left(\ln(N)+\ln(\ln(N))\right)$

Simulation experiments were conducted using the following programs in Python 3.4.4 language: RS and PS algorithms, running *weasel_3.py* and *weasel_locked_3.py* programs respectively [Pedersen, 2009]; DSP algorithm, modifying the code of a simple genetic algorithm, i.e. *SGA.py* [Lahoz-Beltra, 2016]. All simulation experiments were performed using the following initial conditions and parameter values: L=247 (19x13, see Figure 4), the alphabet of possible symbols A={0, 1, 2, 3, 4} with $card(A)$=4, offspring size per generation (population size) N=50 and mutation rate equal to 0.08. In the particular case of DSP algorithm, n=2 represents the number of individuals in the tournament step of the algorithm. The simulation experiment, i.e. the evolutionary synthesis of QCA circuits, ends when an optimum or target circuit is achieved or maximum CPU time is reached.

Results

The results obtained in the experiments support the hypothesis that evolutionary search methods based on a Hamming oracle could not be mimicking to Darwinian evolution. A plausible explanation is because knowledge it is not obtained from the exploration of the evolutionary or fitness surface: evolution is the result of the 'knowledge' embedded by the researcher or human expert into the fitness function, i.e. the oracle (Figure 1). This conclusion is supported by the following simulation results.

First, experiments with RS and PS have yielded to the following observations. Figure 5 shows the evolution of the circuits under the RS algorithm. A striking result is that in the evolutionary synthesis of XOR and inverted XOR gates seems to be some evolutionary convergence (Figure 5 a,b) whereas such a convergence is not observed for AND as well as 5-input majority gates (Figure 5 c,d). One possible explanation could be the number of cells in the array in state 0. However, when evolution takes place by means of the PS algorithm then the evolutionary synthesis of the four QCA circuits is successfully achieved, and four QCA circuits follow the same evolutionary pattern (Figure 6). Indeed, as predicted by the theory of probability Q_1=3.0649 10^{172}, thus under RS algorithm (Figure 5) the target circuit is never reached. Moreover, the time required far exceeds the age of the universe, i.e. 13.7 10^9 according to the NASA's WMAP project and for this reason the value of fitness is always below the maximum value (i.e. 1). In contrast, and according to Q_2=26.3386 value under PS algorithm (Figure 6) evolutionary synthesis of the four QCA circuits is successful reaching in approximately 26 generations the target circuit, and hence the maximum fitness (i.e. 1). These results are confirmed if we take as an example the XOR gate simulation experiment. Table 2 shows in RS and PS search algorithms the statistical summary for the maximum fitness per generation. In case we take as a measure of centralization the median, similarly to as was done for Q_1 and Q_2, there are significant differences between the medians of RS (Me=0.48) and PS (Me=0.79) as is shown in the box and whisker plot (Figure 7).

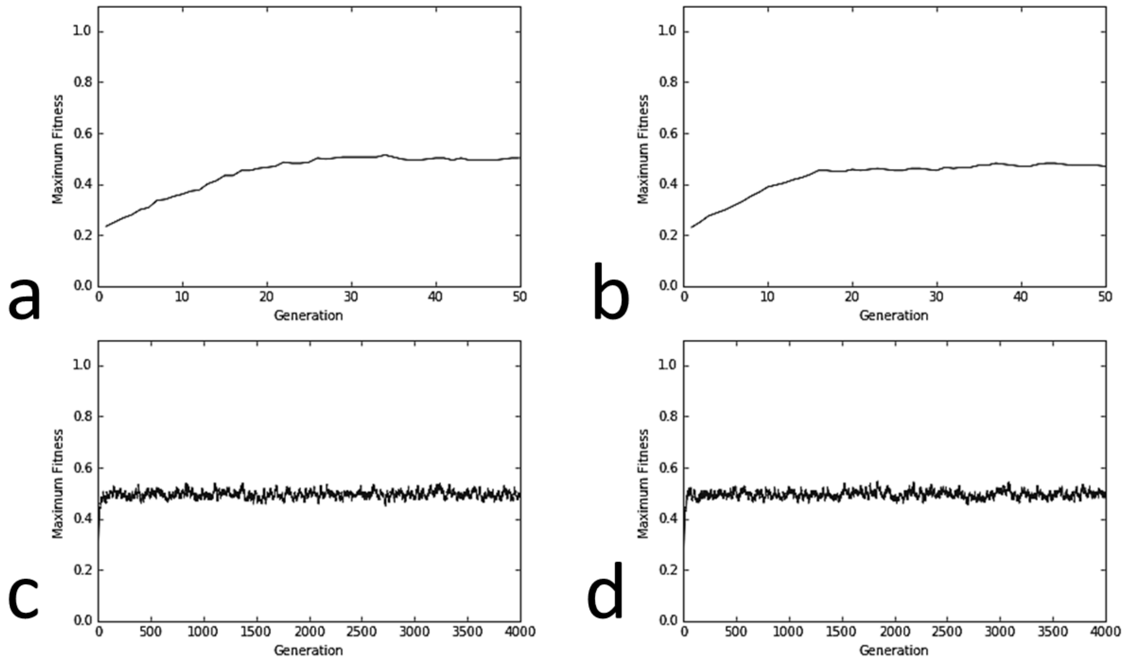

Figure 5.- Performance graph of the evolutionary synthesis of QCA circuits under RS algorithm: (a) XOR gate, (b) inverted XOR, (c) AND gate, (d) 5-input majority gate

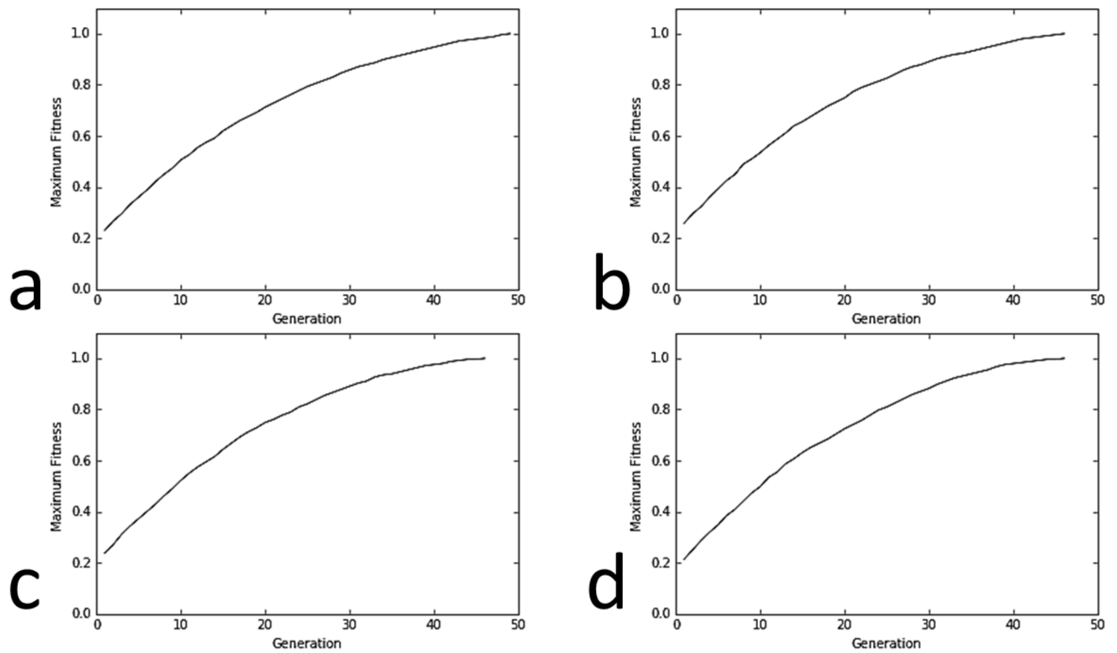

Figure 6.- Performance graph of the evolutionary synthesis of QCA circuits under PS algorithm: (a) XOR gate, (b) inverted XOR, (c) AND gate, (d) 5-input majority gate

Table 2.- Statistical summary*

	RS	PS
Sample size	200	98
Mean	0.493421	0.736305
Variance	0.000205907	0.0516711
Standard deviation	0.0143495	0.227313
Minimum	0.461538	0.222672
Maximum	0.534413	1.0
Rank	0.0728745	0.777328

*Note that in the case of RS, sample size is 200 because we get together in the same sample the fitness values of the last 100 iterations of two independent simulation experiments.

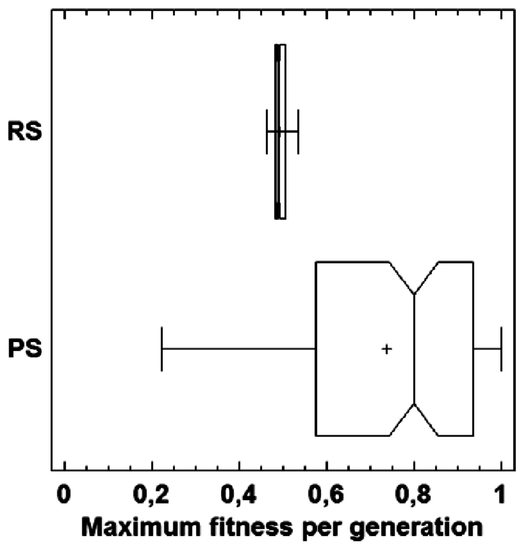

Figure 7.- Box and whisker plot of the evolutionary synthesis experiments conducted with XOR gate under RS and PS algorithms (Mann-Whitney with W = 15981.5 and *P*-value = 0.0; the notch represents a confidence interval for the median and the cross is the arithmetic mean)

Secondly, as can be seen (Figure 8) the results obtained in the study conducted with DSP algorithm, thus a genetic algorithm without crossover and mutation, show DSP search method does not find the optimum circuit. According to the theoretical model Q_3=9.1661, thus the number of generations required for the population to be composed of the best solutions is approximately equal to 9. The results show a local trapping phenomenon, without the population reaches the maximum fitness (i.e. 274). Interestingly, evolutionary convergence resembles the poor performance observed during the evolutionary synthesis of XOR and inverted XOR gates under random search (RS) algorithm (Figure 5).

Conclusion

In conclusion, evolutionary search methods based on Hamming oracle do not mimetic evolution by Darwinian natural selection because knowledge is not obtained from the exploration of the evolutionary surface: evolution is the result of the 'knowledge' embedded by the researcher or human expert into the fitness function. Since this knowledge reduces the size of the search space genetic algorithms and other related evolutionary methods based on Hamming oracle should be termed as "combinatorial search algorithms" together with techniques such as Minimax, Alpha-beta pruning, etc.

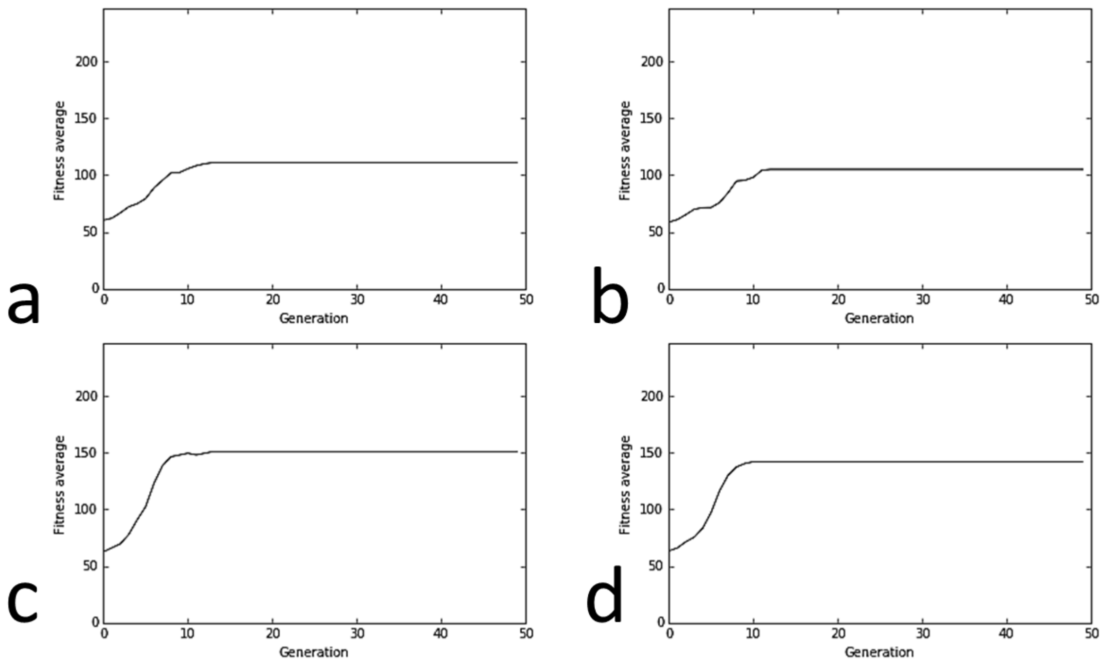

Figure 8.- Performance graph of the evolutionary synthesis of QCA circuits under DSP algorithm: (a) XOR gate, (b) inverted XOR, (c) AND gate, (d) 5-input majority gate

Bibliography

[Alajmi and Wright, 2014] A. Alajmi, J. Wright. 2014. Selecting the most efficient genetic algorithm sets in solving unconstrained building optimization problem. International Journal of Sustainable Built Environment 3: 18-26.

[Angizi et al., 2015] S. Angizi, S. Sarmadi, S. Sayedsalehi, K. Navi. 2015. Design and evaluation of new majority gate-based RAM cell in quantum-dot cellular autómata. Microelectronics Journal 46: 43-51.

[Bäck, 1996] T. Bäck. 1996. Evolutionary Algorithms in Theory and Practise. New York: Oxford University Press.

[Bonyadi et al., 2007] M.R. Bonyadi, S.M.R. Azghadi, N.M. Rad, K. Navi, E. Afjei. 2007. Logic optimization for majority gate-based nanoelectronic circuits based on genetic algorithm. In: Proceedings International Conference on Electrical Engineering, 2007. ICEE '07:1-5.

[Dawkins, 1986] R. Dawkins. 1986. The Blind Watchmaker. New York: W. W. Norton & Company.

[Dembsky et al., 2007] W.A. Dembsky, W. Ewert, R.J. Marks II. The Evolutionary Informatics Lab. http://www.evoinfo.org/index/

[Houshmand et al., 2009]. M. Houshmand, S.H. Khayat, R. Rezaei. 2009. Genetic algorithm based logic optimization for multi-output majority gate-based nano-electronic circuits. In: Proceedings of the IEEE International Conference on Intelligent Comput-ing and Intelligent Systems, 20–22 November 2009, Shanghai, China, IEEE: 584–588.

[Houshmand et al., 2011] M. Houshmand, R. Rezaee Saleh, M. Houshmand. 2011. Logic minimization of QCA circuits using genetic algorithms. In: Soft Computing in Industrial Applications, AISC 96 (Eds. A. Gaspar-Cunha et al.) Springer-Verlag, Berlin, Heidelberg: 393–403.

[Kamrani et al., 2012] M. Kamrani, H. Khademolhosseini, A. Roohi, P. Aloustanimirmahalleh. 2012. A novel genetic algorithm based method for efficient QCA circuit design. In: Advances in Computer Science, Eng. & Appl., AISC 166 (Eds. D.C. Wyld et al.) Springer-Verlag, Berlin, Heidelberg: 433-442.

[Lahoz-Beltra, 2016] R. Lahoz-Beltra. 2016. Simple genetic algorithm (SGA). figshare. dx.doi.org/10.6084/ m9.figshare.3397714.v2

[Pedersen, 2009] A.G. Pedersen. 2009. Weasel programs in Python. http://www.cbs.dtu.dk/courses/27615.mol/weasel.php

[Vilela Neto et al., 2007] O.P. Vilela Neto, M.A.C. Pacheco, C.R. Hall Barbosa. 2007. Neural network simulation and evolutionary synthesis of QCA circuits. IEEE Transactions on Computers 56: 191-201.

[Walus et al, 2004] K. Walus, T.J. Dysart, G.A. Jullien, R.A. Budiman. 2004. QCADesigner: a rapid design and Simulation tool for quantum-dot cellular autómata. IEEE Transactions on Nanotechnology 3: 26-31.

[Walus Group, 2009] QCADesigner. Walus Group at the University of British Columbia. http://www.mina.ubc.ca/qcadesigner.

Authors' Information

Ramses Salas Machado *– Research Scholar at Carlos III University of Madrid and member of the Grupo de Computación Natural, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid ; e-mail: ramsesjsm@gmail.com*

Major Fields of Scientific Research: Natural computing

Juan B. Castellanos Peñuela *– Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Spain; e-mail: jcastellanos@fi.upm.es*

Major Fields of Scientific Research: Natural computing, membrane computing, molecular computing and artificial neural networks.

Rafael Lahoz–Beltra *– Department of Applied Mathematics, Faculty of Biological Sciences, Complutense University of Madrid, 28040 Madrid, Spain; e-mail: lahozraf@ucm.es*

Major Fields of Scientific Research: Evolutionary computation, bioinspired algorithms