Electronic Circuit Optimization Design Algorithm based on Adaptive Culture Search

Xuesong Yan[1], Qinghua Wu[2], Chengyu Hu[1*], Hong Yao[1] and Hanmin Liu[3]

[1]School of Computer Science, China University of Geosciences, Wuhan 430074, China*
[2]School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan 430073, China
[3]Wuhan Institute of Ship Building Technology, Wuhan 430050

Abstract

Electronic circuit optimization design is important research field in engineering applications. Circuit scale and the time of evaluate the circuit are the challenge problems for circuit optimization design algorithm and the traditional optimization algorithms cannot solve the two problems very well. Cultural Algorithms are a class of computational models derived from observing the cultural evolution process in nature. Aiming at the disadvantages of basic Cultural Algorithms like being trapped easily into a local optimum, this paper improves the basic Cultural Algorithms and proposes the improved cultural algorithm (ICA) to solve the overcomes of the basic Cultural Algorithms. The new algorithm keeps not only the fast convergence speed characteristic of basic Cultural Algorithms, but effectively improves the capability of global searching as well. For the case studies, the new algorithm means has proved to be efficient and the experiment results show that the new means have got the better results.

Keywords: *Circuit optimization design, Cultural algorithm, Belief space, Adaptive culture search operator, Circuit evaluation*

1. Introduction

Evolutionary Electronics applies the concepts of genetic algorithms to the evolution of electronic circuits. The main idea behind this research field is that each possible electronic circuit can be represented as an individual or a chromosome of an evolutionary process, which performs standard genetic operations over the circuits. Due to the broad scope of the area, researchers have been focusing on different problems, such as placement, Field Programmable Gate Array (FPGA) mapping, optimization of combinational and sequential digital circuits, synthesis of digital circuits, synthesis of passive and active analog circuits, synthesis of operational amplifiers, and transistor size optimization. Of great relevance are the works focusing on “intrinsic” hardware evolution in which fitness evaluation is performed in silicon, allowing a higher degree of exploration of the physical properties of the medium. This particular area is frequently called Evolvable Hardware [1-3].

In the sequence of this work, Coello, Christiansen and Aguirre presented a computer program that automatically generates high-quality circuit designs [4]. Miller, Thompson and Fogarty applied evolutionary algorithms for the design of arithmetic circuits [5]. Kalganova, Miller and Lipnitskaya proposed another technique for designing multiple-valued circuits [6]. In order to solve complex systems, Torresen proposed the method of increased complexity evolution. The idea is to evolve a system gradually as a kind of divide-and-conquer method [7]. Based on the Miller’s method, Yan applied Gene Expression Programming (GEP) [8, 9], Particle Swarm Optimization Algorithms (PSO) [10], Cultural Algorithms (CA) [11-13], Orthogonal Evolutionary Algorithm [14] and evolutionary algorithm [15, 16] for the design of electronic circuits.

A major bottleneck in the evolutionary design of electronic circuits is the problem of scale. This refers to the very fast growth of the number of gates, used in the target circuit, as the number of inputs of the evolved logic function increases. This results in a huge search space that is difficult to explore even with evolutionary techniques. Another related obstacle is the time required to calculate the fitness value of a circuit. A possible method to solve this problem is to use building blocks either than simple gates. Nevertheless, this technique leads to another difficulty, which is how to define building blocks that are suitable for evolution. This paper improves the disadvantages of basic cultural algorithms being easily trapped into a local optimum and proposed an improved cultural algorithm (ICA) with improved influence function, adaptive culture search operator and elite selection mechanism, which proved to be more simply conducted and with more efficient global searching capability.

2. Basic Cultural Algorithm

Cultural Algorithms (CA) are a class of computational models derived from observing the cultural evolution process in nature [17]. The Cultural Algorithm is a dual inheritance system that characterizes evolution in human culture at both the macro-evolutionary level, which takes place within the belief space, and at the micro-evolutionary level, which occurs at the population space. CA consists of a social population and a belief space. Experience of individuals selected from the population space by the acceptance function is used to generate problem solving knowledge that resides in the belief space. The belief space stores and manipulates the knowledge acquired from the experience of individuals in the population space. This knowledge can control the evolution of the population component by means of the influence function. As a result, CA can provide an explicit mechanism for global knowledge and a useful framework within which to model self-adaptation in an EC system. The population level component of the cultural algorithm will be Evolutionary Programming (EP). The global knowledge that has been learned by the population will be expressed in terms of both normative and situational knowledge as discussed earlier.

A pseudo-code description of the Cultural Algorithms is described as follows:

```
BEGIN
t=0;
Initialize population P(t);
Initialize belief space B(t);
Repeat
Evaluate P(t) ;
Update(B(t), accept(P(t))) ;
Generate (P(t), influence(B(t)) ;
Select P(t) from P(t-1) ;
t+=1 ;
Until (termination condition)
END
```

In this algorithm, first the belief space and the population space are initialized. Then, the algorithm will repeat processing for each generation until a termination condition is achieved. Individuals are evaluated using the performance function. The two levels of Cultural Algorithm communicate through the acceptance function and the influence function. The acceptance function determines which individuals from the current population are selected to impact the belief space. The selected individuals' experiences are generalized and applied to adjust the current beliefs in the belief space via the update function. The new beliefs can then be used to guide and influence the evolutionary process for the next generation.

In the basic cultural algorithms, the belief space uses the {S, N} structure represented [18]. The formal syntax for the belief space, B , used in this study is: $B = S \mid N \mid \langle S, N \rangle$, where S denotes structure for situational knowledge and N denotes structures for normative knowledge. The definition above means the belief space can consist of situational knowledge only, normative knowledge only, or both. The situational knowledge S is represented formally as a pair wise structure: $S = \langle < E_1, E_2, ..., E_e >, adjust_E(e) \rangle$, where E_i represent an ith best exemplar individual in the evolution history. There can be e best exemplars in S as s set that constitutes the situational knowledge. Each exemplar individual has n parameters and a performance value. $adjust_E(e)$ is the belief space operator applied to update e number of exemplar individuals in S . The normative knowledge, N , a set of interval information for each of the n parameters is defined formally as 4-tuple: $N = \langle I_j, L_j, U_j, adjust_N \rangle, j = 1, 2, ..., n$, where I_j denotes the closed interval of variable j , that is a continuous set of real numbers x represented as a ordered number pair: $I_j = [l_j, u_j] = \{x \mid l_j \le x \le u_j, x \in R\}$. l_j (lower bound) and u_j (upper bound) are initialized by the give domain values. L_j represents the performance score of the lower bound l_j for parameter j . U_j represents the performance score of the upper bound u_j for parameter j .

For the update function, we defined like this: $S = \{s_t\}$, select the best individual s_t update the situation knowledge S in belief space. The update process follows the equation (1):

$$s^{t+1} = \begin{cases} x_{best}^t & f(x_{best}^t) < f(s^t) \\ s^t & \text{ot her s} \end{cases} \tag{1}$$

where x_{best}^t denotes tth best individual.

Update the normative knowledge N in belief space uses the equation (2):

$$\begin{aligned} l_i^{t+1} &= \begin{cases} x_{j,i} & x_{j,i} \le l_i^t \text{ or } f(x_j) < L_i^t \\ l_i^t & \text{ot her s} \end{cases}, \\ L_i^{t+1} &= \begin{cases} f(x_j) & x_{j,i} \le l_i^t \text{ or } f(x_j) < L_i^t \\ L_i^t & \text{ot her s} \end{cases}, \\ u_i^{t+1} &= \begin{cases} x_{j,i} & x_{j,i} \ge u_i^t \text{ or } f(x_j) < U_i^t \\ u_i^t & \text{ot her s} \end{cases}, \\ U_i^{t+1} &= \begin{cases} f(x_j) & x_{j,i} \ge u_i^t \text{ or } f(x_j) < U_i^t \\ U_i^t & \text{ot her s} \end{cases} \end{aligned} \tag{2}$$

In basic CA, the knowledge represented in the belief space can be explicitly used to influence the creation of the offspring via an influence function. In our sliding window model, the strategy can be simply described as follows. The first is if a parent is in a promising region, the offspring are created by randomly changing the problem parameters of the parent just a little. In this case, the normative knowledge applies. The offspring $x_{j,i}^{t+1}$, will be created by using this normative knowledge as a beacon to attract the parent $x_{j,i}^{t}$ to move a copy toward the current sliding window, the influence function defined by the equation (3).

$$x_{j,i}^{t+1} = \begin{cases} x_{j,i}^{t} + \left|size(I_i) * N(0,1)\right| & x_{j,i}^{t} < l_i^t \\ x_{j,i}^{t} - \left|size(I_i) * N(0,1)\right| & x_{j,i}^{t} > u_i^t \\ x_{j,i}^{t} + \lambda_1 * size(I_i) * N(0,1) & others \end{cases} \tag{3}$$

The second is if a parent is in an unpromising region, moving a copy of the parent to a more promising region can be used to create a new offspring. In this case, the constraint knowledge applies. The creation of offspring will be affected by the characteristic of the cells within the sliding window, the influence function defined by the equation (4).

$$x_{j,i}^{t+1} = \begin{cases} x_{j,i}^{t} + \left|size(I_i) * N(0,1)\right| & x_{j,i}^{t} < S_i^t \\ x_{j,i}^{t} - \left|size(I_i) * N(0,1)\right| & x_{j,i}^{t} > S_i^t \\ x_{j,i}^{t} + \lambda_1 * size(I_i) * N(0,1) & others \end{cases} \tag{4}$$

The basic cultural algorithms being easily trapped into a local optimum, we use four benchmark functions to verify it. The details of the test function see Table 1. In the Table 1, S behalf of the range of variables, $f_{\min}$ behalf of the minimization of the function and Table 2 are the experiment results. Figure 1 to Figure 4 are the figures of the four functions.

Table 1. Test function

Function	S	f min
$f_1(x) = \sum_{i=1}^{n} x_i^2$	(-100,100)	0
$f_2(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n}\cos(\frac{x_i - 100}{\sqrt{i}}) + 1$	(-300,300)	0
$f_3(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	(-500,500)	-12569.5
$f_4(x,y) = -[x\sin(9\pi y) + y\cos(25\pi x) + 20]$	(-10,10)	-39.944506

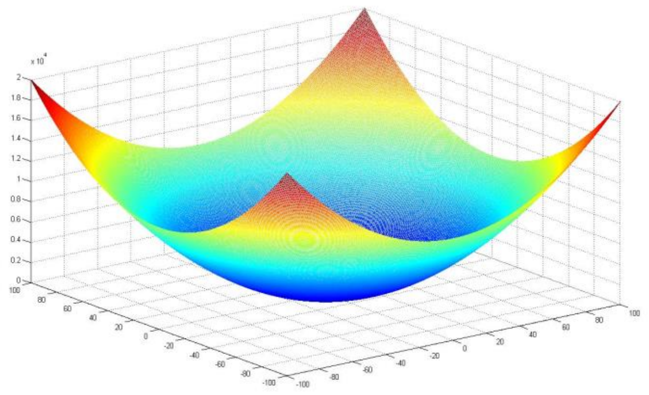


Figure 1. Test function F1

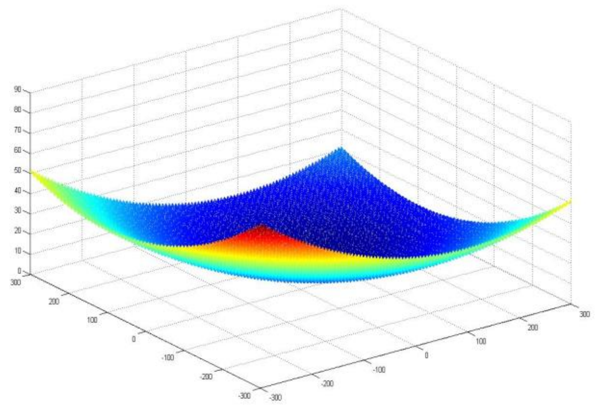


Figure 2. Test function F2

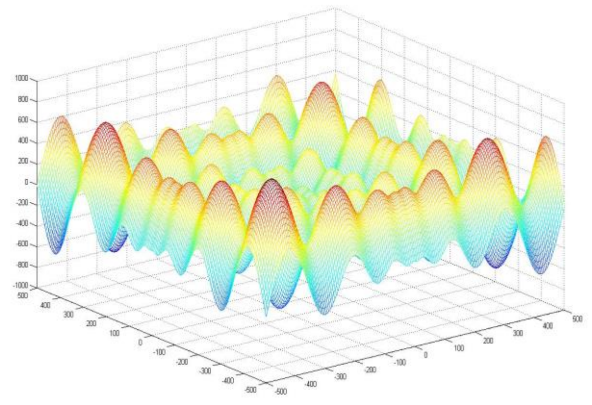


Figure 3. Test function F3

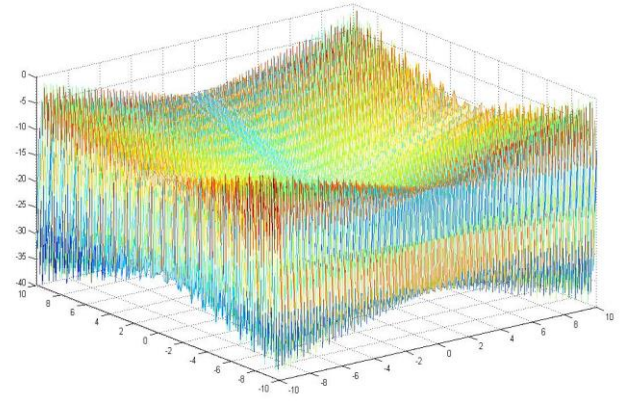


Figure 4. Test function F4

Table 2. Experimental results

Function	Best Value	Mean Value	Worst Value	$f_{\min}$
F1	1495.71	4224.77	7032.89	0
F2	72.51	101.41	123.95	0
F3	-5038.62	-4005.02	-3233.13	-12569.5
F4	-15.967876	-11.235476	-9.098765	-39.944506

3. Improved Cultural Algorithm

3.1. Adaptive cultural search operator

Local search operator has a strong local search ability, and then can solve the shortcomings of genetic algorithm has the weak ability for the local search. And the population according to the current state of adaptive evolution of the local search space adaptive local search operator will undoubtedly greatly enhance the ability of local search. In the initial stage of the evolution, the current optimal solution from the global optimum region is still relatively far away, this time the adaptive local search operator to require search a large neighborhood space to find more optimal solution, it can maintain the population diversity. When the population has evolved to the region containing the global optimum, the adaptive local search operator to require a relatively small area to search in order to improve the accuracy of the global optimal solution.

In our algorithm, the adaptive local search operator is the adaptive culture search operator. Adaptive culture search operator is aimed at the neighborhood of a point to search, so the key point is to identify a point as the center of the hypercube, the hypercube in the orthogonal test, expect to be better solution.

3.2. Improved Influence Function

In the basic CA, the convergence speed of individuals is fast, but the adjustments of cognition component and social component make individuals search in the belief space. According to influence function and update function, once the best individual in the population is trapped into a local optimum, then will attract other individuals to approach this local optimum gradually, and in the end the whole population will be converged at this position. In the basic CA, the influence function think the population space as global, and think the individuals in the population space as objects, once the best individual in the population is trapped into a local optimum, then the algorithm think it has find the global optimum and the algorithm is convergence. In the improved algorithm, we change some strategy of the influence function. In the new influence function, the best individual is considered as global and the gene of the best individual is considered as object, when the gene of the best individual changed, then the new population will be generated. The process of the operation is described as follows: X is the best individual, x_i is ith variable of X. Using equation (5) do the operation for x_i and X, then generate the new individual X'.

$$x_i' = \begin{cases} x_i + \left|size(I_i) * N(0,1)\right| & x_{j,i}^t < S_i^t \\ x_i - \left|size(I_i) * N(0,1)\right| & x_{j,i}^t > U_i^t \\ x_i + \lambda * size(I_i) * N(0,1) & others \end{cases} \tag{5}$$

For the new individual $X^{'}$, if $f(X^{'}) > f(X)$ then $X = X^{'}$. In this process, the parameter λ is very important, for the value of λ in the improved CA there has three different situations as follows:

1. When the algorithm is convergence, in order to generate the best individual, the value of the λ maybe the fraction between in (0 ， 1), then the whole procedure is actually a fine-tuning of the best individual;

2. When the algorithm can not convergence, in order to fast the convergence speed of the algorithm, the value of the λ maybe a large value, best meet $\lambda * size(I_i) >= (u_{global} - j_{global})/2$. In here, the u_{global} is the minimum ceiling of the given function's domain, the j_{global} is the maximum limit of the given function's domain. This value makes a higher degree of change about the individual or a single variable, and then the algorithm can jump out of the local optimum;

3. For some special function, the algorithm is easy trapped into the local optimum, we can combine the above two method to generate the best individual. The detail process as follows: for the generations before the p*generation, the value of the λ is the fraction between in (0, 1); for the generations after the p*generation, the value of the λ is a large value, best meets $\lambda * size(I_i) >= (u_{global} - j_{global})/2$.

3.3. Elite Selection Mechanism

Genetic algorithm is usually complete the selection operation based on the individual's fitness value, in the mechanism of elite, the population of the front generation mixed with the new population which generate through crossover and mutation operations, in the mixed population select the optimum individuals according to a certain probability. The specific procedure is as follows:

Step1: using crossover and mutation operations for population P1 which size is N then generating the next generation of sub-populations P2;

Step2: The current population P1 and the next generation of sub-populations P2 mixed together form a temporary population;

Step3: Temporary population according to fitness values in descending order, to retain the best N individuals to form new populations P1.

The characteristic of this mechanism is mainly in the following aspects. First is robust, because of using this selection strategy, even when the crossover and mutation operations to produce more inferior individuals, as the results of the majority of individual residues of the original population, does not cause lower the fitness value of the individual. The second is in genetic diversity maintaining, the operation of large populations, you can better maintain the genetic diversity of the population evolution process. Third is in the sorting method, it is good to overcome proportional to adapt to the calculation of scale.

3.4. Experiment results

We also use the four benchmark functions to test our new algorithm and the experimental result showed in Table 3. From the experimental results, we can say the new algorithm has got the better solution.

Table 3. Experimental results

Function	Algorithm	Best Value	Mean Value	Worst Value	$f_{\min}$
F1	CA	1495.71	4224.775	7032.89	0
	ICA	8.13E-29	10.46E-26	5.08E-24	0
F2	CA	72.5069	101.410452	123.954	0
	ICA	12.18E-12	0.070377	18.63E-25	0
F3	CA	-5038.62	-4005.02	-3233.13	-12569.5
	ICA	-8535.19	-7741.27	-5203.56	-12569.5
F4	CA	-15.967876	-11.235476	-9.098765	-39.944506
	ICA	-32.547698	-27.562389	-23.668734	-39.944506

In order to verify the improvement of the new algorithm, we also use other benchmarks function to test the algorithm's performance and compare the results with traditional genetic algorithm. Specific details of the test function see Table 4. In the Table 4, n behalf of the dimension number of the function, S behalf of the range of variables, $f_{\min}$ behalf of the minimization of the function.

Table 4. Test functions

Test Function	n	S	$f_{\min}$
$f_1(x)=\sum_{i=1}^{n} x_i^2$	30	-100,100	0
$f_2(x)=6\cdot\sum_{i=1}^{5}\lfloor x_i\rfloor$	30	-5.12, 5.12	0
$f_3(x)=\sum_{i=1}^{n} i\cdot x_i^4+U(0,1)$	30	-1.28,1.28	0
$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n}(x_i-100)^2-\prod_{i=1}^{n}\cos(\frac{x_i-100}{\sqrt{i}})+1$	30	-300.0,300.0	0
$f_5(x)=-20\cdot\exp(-0.2\sqrt{\frac{1}{n}\cdot\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\cdot\sum_{i=1}^{n}\cos(2\pi\cdot x_i))+20+e$	30	-32.0,32.0	0
$f_6(x)=\sum_{i=1}^{100}100((x_{i+1}-x_i^2)^2+(x_i-1)^2)$	30	-2.048,2.048	0
$f_7(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	30	-500,500	-12569.5

Table 5. ICA and CA experimental results comparison

Function	Algorithm	Best Value	Worst Value
$f_1(x)$	CA	1495.71	7032.89
	ICA	4.13731E-29	1.0882E-24
$f_2(x)$	CA	0	0
	ICA	0	0
$f_3(x)$	CA	0.00177094	0.00833963
	ICA	0.00193565	0.0103595
$f_4(x)$	CA	72.5069	123.954
	ICA	2.18559E-12	8.63194E-25
$f_5(x)$	CA	-3.19744E-14	4.4229
	ICA	-3.19744E-14	1.50229
$f_6(x)$	CA	1.84889E-28	8.63194E-25
	ICA	2.55147E-28	1.20401E-23
$f_7(x)$	CA	-5038.62	-3233.13
	ICA	-9535.19	-8203.56

The two algorithms of the same experimental parameters set. Each function in Table 4 is run 50 times with the two algorithms, their experimental results such as Table 5. By analyzing the experimental results we know, in solving function f1, f4 and f7, use the CA is easily into local optimum, but use the ICA, convergence soon, and find a better solution, the average fitness and the best fitness was both superior to CA. For the function f2, the ICA and CA all can find the global optimal, these two algorithm for this test function is very effective. For function f5, the two algorithms can find the best solutions are the same (see Table 3), and the new algorithm to get the best value of the average is better than CA. In sum, we can see that in solving function f1, f4, f5 and f7, the ICA more efficient than CA, in solving other function, the performance almost same of the two algorithms. In short, this new algorithm has the following value: better global search capability.

4. Circuit Optimization Design Problem

4.1. Circuit Representation

In this paper, the chromosome representation we use Miller's [19]. This representation is based on the FPGA of Xilinx Virtex-II. The starting point in this technique is to consider, for each potential design, a geometry (of a fixed size array) of uncommitted logic cells that exist between a set of desired inputs and outputs. The uncommitted logic cells are typical of the resource provided on the Xilinx FPGA part under consideration. An uncommitted logic cell refers to a two-input, single-output logic module with no fixed functionality. The functionality may then be chosen, at the implementation time, to be any two input variable logic function. In this technique, a chromosome is defined as a set of interconnections and gate level functionality for these cells from outputs back toward the inputs based upon a numbered rectangular grid of the cells themselves, as in Figure 5. The inputs that are made available are logic '0', logic '1', all primary inputs and primary inputs inverted. To illustrate this consider a 3 x 3 array of logic cells between two required primary inputs and two required outputs.

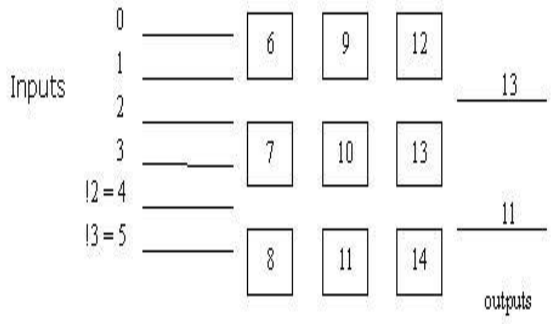
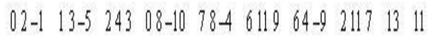


Figure 5. A 3 x 3 geometry of uncommitted logic cells with inputs, outputs and netlist numbering

The inputs 0 and 1 are standard within the chromosome, and represent the fixed values, logic '0' and logic '1' respectively. The inputs (two in this case) are numbered 2 and 3, with 2 being the most significant. The lines 4 and 5 represent the inverted inputs 2 and 3 respectively. The logic cells which form the array are numbered column-wise from 6 to 14. The outputs are numbered 13 and 11, meaning that the most significant output is connected to the output of cell 13 and the least significant output is connected to the output of cell 11.

These integer values, whilst denoting the physical location of each input, cell or output within the structure, now also represent connections or routes between the various points. In other words, this numbering system may be used to define a netlist for any combinational circuit. Thus, a chromosome is merely a list of these integer values, where the position on the list tells us the cell or output which is being referred to, while the value tells us the connection (of cell or input) to which that point is connected, and the cells functionality.

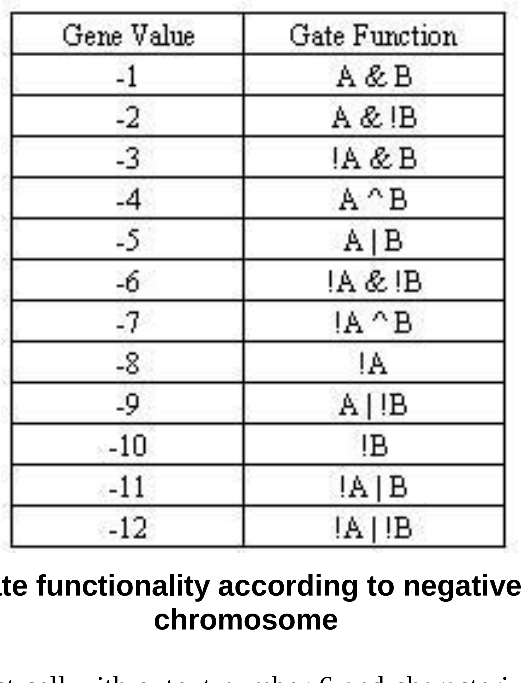
Each of the logic cells is capable of assuming the functionality of any two-input logic gate, or, alternatively a 2-1 multiplexer (MUX) with single control input. In the geometry shown in Figure 6, a sample chromosome is shown below:



0 2 -1 1 3 -5 2 4 3 0 8 -10 7 8 -4 6 11 9 6 4 -9 2 11 7 13 11

Figure 6. A typical netlist chromosome for the 3 x 3 geometry of Figure 5

Notice, in this arrangement that the chromosome is split up into groups of three integers. The first two values relate to the connections of the two inputs to the gate or MUX. The third value may either be positive - in which case it is taken to represent the control input of a MUX - or negative - in which case it is taken to represent a two-input gate, where the modulus of the number indicates the function according to Figure 7 below. The first input to the cell is called A and the second input called B for convenience. For the logic operations, the C language symbols are used: (i) & for AND, (ii) | for OR, (iii) ^ for exclusive-OR, and (iv) ! for NOT. There are only 12 entries on this table out of a possible 16 as 4 of the combinations: (i) all zeroes, (ii) all ones, (iii) input A passed straight through, and (iv) input B passed straight through are considered trivial - because these are already included among the possible input combinations, and they do not affect subsequent network connections in cascade.



Gene Value	Gate Function
-1	A & B
-2	A & !B
-3	!A & B
-4	A ^ B
-5	A | B
-6	!A & !B
-7	!A ^ B
-8	!A
-9	A | !B
-10	!B
-11	!A | B
-12	!A | !B

Figure 7. Cell gate functionality according to negative gene value in chromosome

This means that the first cell with output number 6 and characterized by the triple {0, 2, -1} has it's a input connected to '0', its B input connected to input 2, and since the third value is -1, the cell is an AND gate (thus in this case always produces a logical output of 0). Picking out the cell who's output is labeled 9, which is characterized by the triple {2, 6, 7}, it can be seen that its A input is connected to input 2 and its B input is connected to the output of cell 6, while since the third number is positive the cell is a MUX with control input connected to the output of cell 7.

4.2. Circuit evaluation

In practical applications, the evaluation of the quality of a circuit, in addition to function correctly, there are two important parameters: the delay time and power consumption. The evolution of digital circuit design process, even if the correct solution, but the circuit is not necessarily the optimal circuit functions correctly premise, the delay is the most important indicators of the evaluation circuit is good or bad, secondly, the power consumption is also to reflect the performance of the circuit important indicator of the evolution of the design of digital circuits has three goals descending order of importance: the correct function, minimize delay time and minimum power consumption. Meet the three objectives of the circuit is the best circuit. In Table 6 are the delay time and power consumption of normal logic gates.

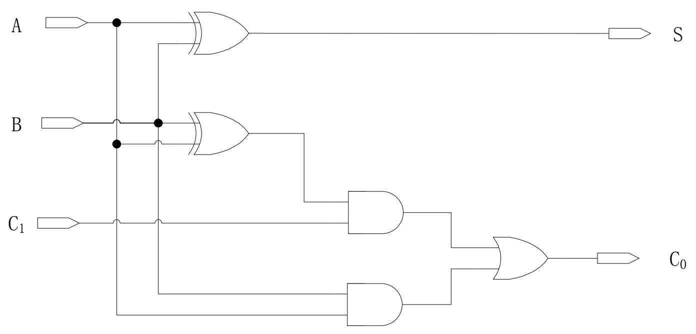


Figure 8. One-bit adder circuit

For a specific circuit, as shown in Figure 8, there may be a plurality of output terminals, there may be many different paths from the input to the output, the delay of each path are not normally equal, we calculated each path delay, as to which the maximum delay of the entire circuit, such as the Figure 8 above, a full adder, from the A input or the B input, through XOR gate, AND gate to reach the OR gate, then to C0, this road of maximum delay path and calculate the delay time as follows:

T = Tinstrinsic(XOR) + Tinstrinsic(AND) +Tinstrinsic(OR)

First XOR：t1= 0.1373(ns)

Second AND: t2= 0.0801(ns)

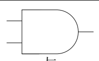
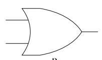
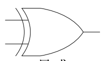
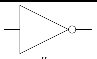
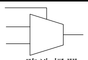
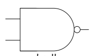
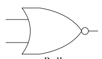
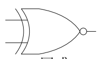
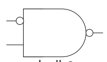
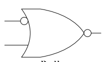
Third OR： t3= 0.0617(ns)

Total delay tiem： t= t1 + t2 + t3 = 0.2791(ns)

Calculate the power consumption is simple, to accumulate all logic gates used in the circuit power consumption. So the power consumption of the Figure 8 as:

W = W（XOR）* 2 + W（AND）* 2 + W（OR） = 0.0372 * 2 + 0.0103 * 2 + 0.0191 = 0.1141

Table 6. Delay time and power consumption of logic gates

					
Delay time	0.0801（ns）	0.0617（ns）	0.1373（ns）	0.0083（ns）	0.1318（ns）
Power consumption	0.0103（ns）	0.0191（ns）	0.0372（ns）	0.0239（ns）	0.0370（ns）
					
Delay time	0.0305（ns）	0.0383（ns）	0.1382（ns）	0.0178（ns）	0.0182（ns）
Power consumption	0.0102（ns）	0.0108（ns）	0.0383（ns）	0.0643（ns）	0.0712（ns）

According to the design requirements of the circuit gate count, power consumption, delay, and the fitness function of the circuit as follows:

$$G(x) = \begin{cases} F(x) & F < 2^n \quad (circuit \quad wrong) \\ F(x) + w * V(x)) & F = 2^n \quad (circuit \quad correct) \end{cases} \tag{6}$$

In here, $F(x)$ is the circuit x full output cell information and the truth table of the degree of matching, $V(x)$ for x is the effective number of gates of the circuit, the circuit is completely correct conditions, the smaller the effective number of gates, the better the circuit performance. w is the weight value factor, taking a very small positive number.

4.3. Circuit Optimization Design Case

Evolving the one-bit adder was easier to do on a larger geometry but resulted in a less efficient circuit. That is many genetic algorithm was able to discover 100% functional solutions was intimately related to the size of the geometry, but our algorithm use small geometry to find the fully functional solutions.

The circuit designed by CA as shown in Figure 9 (with three gates), Figure 10 is the circuit designed by ICA (with three gates). From the figure we know it is a gratifying result to obtain as it is clear that this design is an optimum solution.

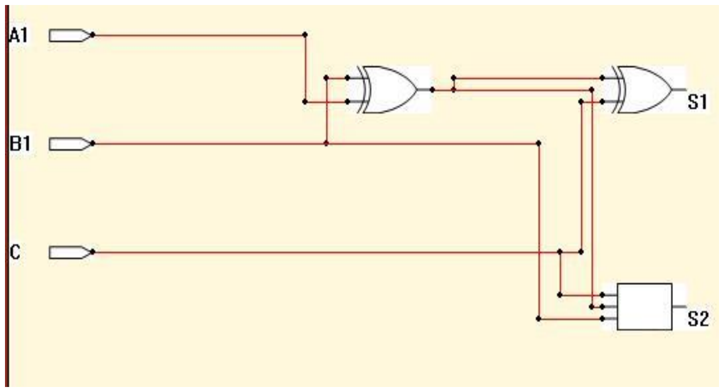


Figure 9. One-bit full adder circuit designed by CA

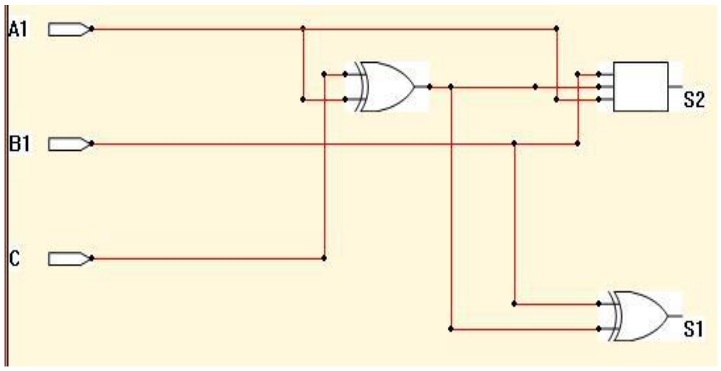


Figure 10. One-bit full adder circuit designed by ICA

A two-bit full adder circuit, which with a truth table with 5 inputs and 3 outputs. In this case, Our algorithm use small geometry to find the fully functional solutions, the matrix has a size of 4×4. The CA designed circuit is showed in Figure 11 (with eleven gates), and our resulting circuits as shown in Figure 12 (with six gates). From the figures we know it is a gratifying result to obtain as it is clear that this design is an optimum solution.

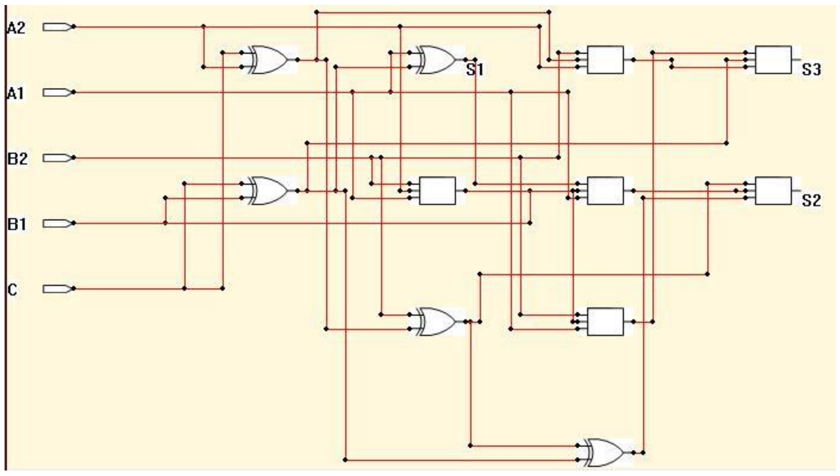


Figure 11. Two-bit full adder circuit designed by CA

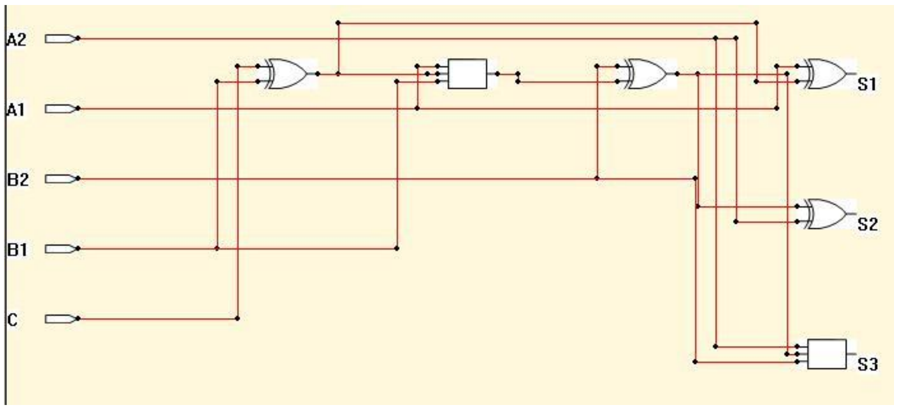


Figure 12. Two-bit full adder circuit designed by ICA

A two-bit multiplier multiplies the binary numbers (A2,A1) by (B2,B1) to produce the four-bit binary number (S4, S3, S2, S1) where A2, B2, S4 are the most significant bits.

```
                A2 A1
                B2 B1
      ______________________
              A2B1 A1B1
         A2B2 A1B2
      ______________________
      S4   S3    S2    S1
```

Figures 13-14 show the two most efficient circuits which were obtained in 50 runs of the CA and ICA with a population size of 30, each run terminating at 100,000 generations (if 100% hadn't been achieved) and the matrix has a size of 4×4.

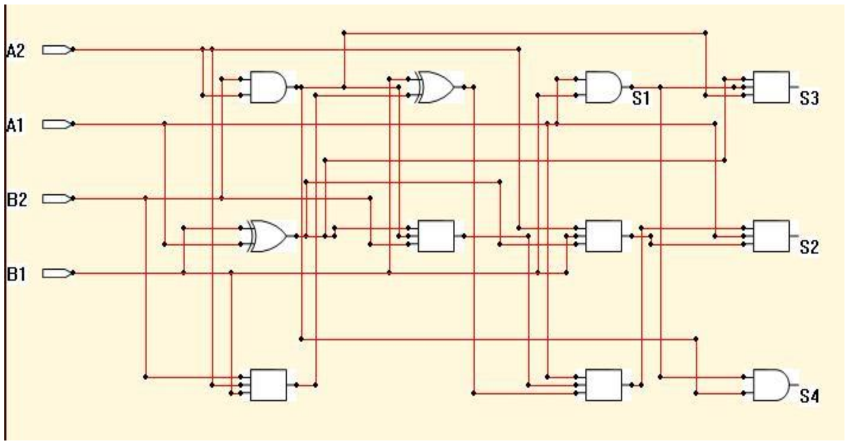


Figure 13. Two-bit full multiplier circuit designed by CA

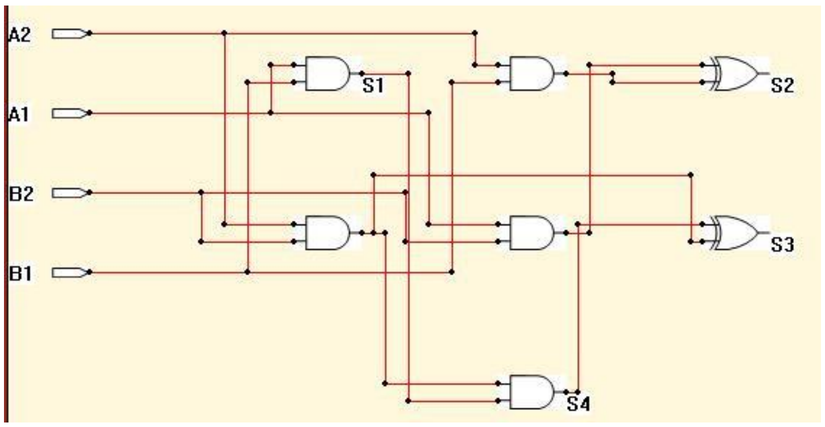


Figure 14. Two-bit full multiplier circuit designed by ICA

5. Conclusion

This paper introduce a new algorithm based on the basic CA algorithm, for the basic CA algorithm the new algorithm has done two improvements: 1. By introducing a adaptive culture search operator and elite selection mechanism, make the individuals within the population space can maintain combined with the best individuals, thus enlarge global searching space and reduce the possibility of individuals to be trapped into a local optimum; 2. By improving the influence function, decreased the possibility of being trapped into a local optimum. Compared with the basic CA algorithm, the new algorithm enlarges the searching space and the complexity is not high. The final circuit is optimized in terms of complexity (with the minimum number of gates). For the case studies this means has proved to be efficient, experiments show that we have better results.

Acknowledgements

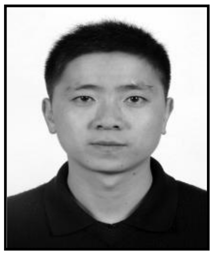
This paper is supported by Natural Science Foundation of China. (No.61272470 and No.61203307), National Civil Aerospace Pre-research Project of China, the Provincial Natural Science Foundation of Hubei (2012FFB04101), and the Fundamental Research Founds for National University, China University of Geosciences (Wuhan).

References

[1] R. S. Zebulum, M. A. Pacheco and M. M. Belasco, “Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms”, CRC Press, **(2001)**.
[2] A. Thompson and P. Layzell, “Analysis of unconventional evolved electronics”, Communications of the ACM, vol. 42, **(1999)**, pp. 71-79.
[3] S. J. Louis and G. J. Rawlins, “Designer Genetic Algorithms: Genetic Algorithms in Structure Design”, Proceedings of the Fourth International Conference on Genetic Algorithms, **(1991)**.
[4] C. A. Cello, A. D. Christiansen and A. H. Aguirre, “Using Genetic Algorithms to Design Combinational Logic Circuits”, Intelligent Engineering through Artificial Neural Networks, vol. 6, **(1996)**, pp. 391-396.
[5] J. F. Miller, P. Thompson and T. Fogarty, “Algorithms and Evolution Strategies in Engineering and Computer Science”, Recent Advancements and Industrial Applications, Chapter 6, **(1997)**.
[6] T. Kalgan ova, J. F. Miller and N. Lipnitskaya, “Multiple-Valued Combinational Circuits Synthesised using Evolvable Hardware”, Proceedings of the 7th Workshop on Post-Binary Ultra Large Scale Integration Systems, **(1998)**.

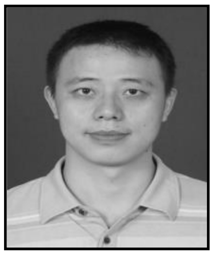
[7] J. Torresen, "A Divide-and-Conquer Approach to Evolvable Hardware", Proceedings of the Second International Conference on Evolvable Hardware, vol. 1478, **(1998)**, pp. 57-65.
[8] X. S. Yan and W. Wei, "Design Electronic Circuits by Means of Gene Expression Programming", Proceedings of the First NASA/ESA Conference on Adaptive Hardware and Systems, **(2006)**, pp. 194-199.
[9] X. S. Yan and W. Wei, "Designing Electronic Circuits by Means of Gene Expression Programming Ⅱ", Proceedings of the 7th International Conference on Evolvable Systems: From Biology to Hardware, **(2007)**, pp. 319-330.
[10] X. S. Yan and Q. Wu, "Circuit Design Based on Particle Swarm Optimization Algorithms", Key Engineering Materials, vol. 474-476, **(2011)**, pp. 1093-1098.
[11] X. S. Yan and K. Wang, "Designing Electronic Circuits Using Cultural Algorithms", Proceedings of Third International Workshop on Advanced Computational Intelligence, **(2010)**, pp. 299-303.
[12] X. S. Yan and Q. Wu, "Electronic Circuits Optimization Design Based On Cultural Algorithms", International Journal of Information Processing and Management, vol. 2, no. 1, **(2011)**, pp. 49-56.
[13] X. Yan, W. Chen, Q. Wu and H. Liu, "Research of Embedded Hardware Optimization Design Algorithm", International Journal of Computer Science Issues, vol. 9, Issue 6, no. 2, **(2012)**, pp. 70-78.
[14] X. S. Yan, Q. H. Wu and H. M. Liu, "Orthogonal Evolutionary Algorithm and its Application in Circuit Design", Przeglad Elektrotechniczny, vol. 88, Issue 05b, **(2012)**, pp. 7-10.
[15] X. S. Yan, Q. H. Wu, C. Y. Hu and Q. Z. Liang, "Circuit Optimization Design Using Evolutionary Algorithms", Advanced Materials Research, vol. 187, **(2011)**, pp. 303-308.
[16] X. Yan, H. Yao, Q. Liang and C. Hu, "Research of Digital Circuit Optimization Design Algorithm", Advances in Information Sciences and Service Sciences, vol. 4, no. 21, **(2012)**, pp. 556-563.
[17] R. Reynoids, "An introduction to cultural algorithms", Proceedings of the 3rd Annual Conference on Evolutionary Programming, Sebald, AX; Fogel, L.J. (Editors), River Edge, NJ, World Scientific Publishing, **(1994)**, pp. 131-139.
[18] C. Chung, "Knowledge-based approaches to self-adaptation in cultural algorithms", Ph.D. Thesis, Wayne State University, Detroit, Michigan, **(1997)**.
[19] J. F. Miller, D. Job and V. K. Vassilev, "Principles in the Evolutionary Design of Digital Circuits - Part I", Genetic Programming and Evolvable Machines, vol. 1, **(2000)**, pp. 8-35.
[20] X. Yan, Q. Wu, C. Zhang, W. Chen, W. Luo and W. Li, "An Efficient Function Optimization Algorithm based on Culture Evolution", International Journal of Computer Science Issues, vol. 9, Issue 5, no. 2, **(2012)**, pp. 11-18.

Authors



Xuesong Yan received the MS degree in Computer Application Technology from China University of Geosciences in 2003, and the PhD degree in Computer Software and Theory from the School of Computer Science, Wuhan University in 2006. He is currently an associate professor in China University of Geosciences. His research interests are in the areas of evolutionary computation, evolvable hardware and machine learning.

Qinghua Wu received the MS degree in Computer Application Technology from China University of Geosciences in 2003, and the PhD degree in Earth Explore and Information Technology from China University of Geosciences, in 2011. She is currently a lecturer in Wuhan Institute of Technology. Her research interests are in the areas of evolutionary computation and image processing.



Chengyu Hu received the MS degree in Automation from Wuhan University of Technology in 2003, and the PhD degree in Automation from School of Mechanical Science and Engineering, Huazhong University of Science and Technology in 2010. He is currently an associate professor in China University of Geosciences. His research interests are in the areas of intelligence computation and automation.

Hong Yao received the MS degree in Computer application technolgoy from China University of Geosciences in 2004, and the PhD degree in Computer Architecture from School of Computer Science, Huazhong University of Science and Technology in 2009. He is currently an associate professor in China University of Geosciences. His research interests are in the areas of network architecture, computer applications and information systems.

Hanmin Liu received the MS degree in Computer Technology from China University of Geosciences in 2008, and current is the PhD candidate of School of Computer Science at China University of Geosciences.

He has authored several refereed papers in journals and conference proceedings. His current research interests are biologically-inspired computational intelligence algorithms, evolutionary computation applications and computer applications. Prof. Liu is currently an associate professor of Wuhan Institute of Ship Building Technology at Wuhan, China.