Biased Mutation and Tournament Selection Approaches for Designing Combinational Logic Circuits via Cartesian Genetic Programming

José Eduardo H. da Silva[1], Francisco A.L. Manfrini[2], Heder S. Bernardino[1], Helio J.C. Barbosa[1,3]

[1]Universidade Federal de Juiz de Fora, Juiz de Fora, MG, Brazil

jehenriques@ice.ufjf.br, heder@ice.ufjf.br, hcbm@lncc.br

[2]Instituto Federal do Sudeste de Minas Gerais, Campus Juiz de Fora
Juiz de Fora, MG, Brazil

francisco.manfrini@ifsudestemg.edu.br

[3]Laboratório Nacional de Computação Científica, Petrópolis, RJ, Brazil

Abstract. *Cartesian Genetic Programming (CGP) is often applied to design combinational logic circuits. However, there is no consensus in the literature regarding the more appropriate objective function when it is desired to minimize the number of logic gates of the circuit. Thus, we analyze here two strategies: the minimization of the number of logic gates and the maximization of the number of wire gates. Additionally, a biased mutation strategy for CGP, which were previously presented and tested only to find a feasible solution, are extended in this paper for the subsequent optimization step. Several configurations were proposed and tested varying objective function and selection schemes. Computational experiments are conducted with some benchmark circuits to relatively compare the proposed methods, and the results obtained are better than those found by the other techniques considered here.*

1. Introduction

In digital electronics, the design of combinational logic circuits (CLC) is a common task. CLCs perform a desired function, defined by a truth table, and they are drawn using a set of logic gates. This task requires complex knowledge of large collections of specific rules [Miller et al. 2000]. Over the years, several rules for optimizing CLCs have emerged [Sasao 1993]. However, this task is still performed by humans, with the aid of synthesis tools especially, when the focus is on designing complex circuits. Different algorithmic methods have been developed in order to support human designers in this task, and evolutionary computation is considered one of the most successful [Thompson et al. 1996]. One important feature of the evolutionary algorithms is the reduced interference required from the specialist. Several approaches can be found in the so-called *Evolvable Hardware* (EH) [Coello et al. 2000a, Coello et al. 2003b, Coello et al. 2000d, García and Coello 2002, Miller 2011b, Manfrini et al. 2016a] literature. Those approaches aim at finding a functional solution that minimizes the number of logic gates. Recently, EH has been questioned because its techniques are not showing to be competitive when the real needs of practical applications are considered. The main justification is the scalability of

these techniques which makes the evolution of more complex circuits hard [Stepney and Adamatzky 2017].

Currently, CGP (*Cartesian Genetic Programming*) [Miller 1999] is one of the most efficient methods for the evolutionary design and optimization of combinational logic circuits [Vasicek 2015, Miller 2011b]. Although the 4-bit multiplier is the most complex circuit already evolved [Vassilev et al. 2000, Vasicek 2018], CGP is able to minimize the number of logic gates of bigger circuits [Vasicek 2015].

In addition, different formulations can be found in the literature for the optimization problem involving the design of combinational logic circuits [Coello et al. 2000a, Coello et al. 2000c, Stomeo et al. 2005, Karakatic et al. 2013]. In this context, the present work intends to present and compare different optimization problems of CLCs when using CGP, namely, (i) the minimization of the number of logic gates and (ii) the maximization of the number of wire gates in the circuit.

It is important to notice that wire gates mean a boolean function where all the outputs are equal of its inputs. We used the wire gates as an extension of a point of the circuit and so the wire gates are not considered here as logic gates.

Also, as a set of constraints defined by the truth table must be satisfied, the design of CLCs is a constrained optimization problem, and a constraint handling method has been used to assist the search technique. A common strategy when using EH [Stomeo et al. 2005] is the division of the problem into two, where the truth table is attempted at first and then a solution that minimizes a given cost function is obtained. However, it is not clear the importance of considering the objective function in the first step of the evolution when two candidate solutions compared present the same quality with respect to the constraints (error with reference to the truth table). Thus, two tournament selection strategies are also analyzed here when the candidate solutions show the same constraint violation: (i) the objective function is used to select the best candidate solution; and (ii) the best design is randomly chosen.

A common version of CGP uses only a mutation operator. The point mutation and the so-called SAM (*Single Active Mutation*) are mutation operators commonly found in the literature. In addition, a biased mutation operator called Biased SAM [Manfrini et al. 2016a] was also created, where a transition probability matrix (TPM) is used when a logic gate is replaced. We propose here the use of Biased SAM (BSAM) after a feasible solution is found, and its combination with SAM to generate the offspring.

Finally, the contributions of this work can be summarized as: (i) an analysis of two objective functions, (ii) a study of two tournament selection strategies, (iii) the use of two probability transition matrices during the search, and (iv) the combination of SAM and BSAM during the search.

The remainder of this paper is organized as follows. Section 2 presents the problem of evolving combinational logic circuits and Section 3 describes the Cartesian Genetic Programming technique. The proposed approaches are presented in Section 4 and the results found in the computational experiments are analyzed in Section 5. Finally, Section 6 concludes the paper.

2. Evolutionary Design of Logic Circuits

The design of Combinational Logic Circuits (CLC) uses the data from a truth table that lists all possible combinations of inputs with their corresponding outputs [Manfrini et al. 2016a]. A solution of this optimization problem must present the same outputs for the inputs considered. Also, a set of available gates limits the search space, and a given cost function is to be optimized. Human designers are limited to a set of conventional rules and they commonly consider hierarchically structured systems. Simplification techniques can be used, such as the Karnaugh map and the Quine-McCluskey computational method [McCluskey 1956], in addition to heuristics, such as Espresso [Brayton et al. 1984]. These traditional techniques have limitations either in the number of variables allowed or in the high computational cost. On the other hand, the automated design makes it possible to transcend these restrictions and to explore a much richer space of possibilities through evolutionary computation. In the meantime other difficulties appear; for example, the problem of scalability that is related to the increment in the size of the chromosomes required to represent complex solutions [Vasicek and Sekanina 2015]. Moreover, another difficulty is related to the processing time of the objective function evaluations, which grows exponentially with the number of inputs [Miller 1999].

Different ways of evaluating individuals and selecting them are found in the literature. The selection of the best design from a set of candidate solutions is important as evolutionary techniques frequently require this decision to move through the search space. Coello and collaborators developed two genetic algorithms, called NGA [Coello et al. 2000c] (n-cardinality Genetic Algorithm) and MGA [Coello et al. 2000a] (Multi-objective Genetic Algorithm). Both methods use an encoding similar to that adopted in CGP but with a genotype that increases its lines and rows when the feasibility criteria is not satisfied within a given number of iterations; that is, the cardinality can be expanded during the search.

The search of NGA operates within two stages: at the beginning of the search, only the error with respect to the truth table is evaluated, and when a feasible solution is found, the fitness function considers the number of wire gates (to be maximized). In MGA, the population is split into $m + 1$ subpopulations, where m represents the number of rows of the truth table, and a penalty handling strategy that combines the values of the number of matches between the outputs of the circuit and those of the truth table with the number of wire gates of the circuit. Circuits with few inputs and few outputs (maximum 3-4) were used in their experiments and the results were compared to circuits created by humans. MGA was able to find results similar to or better than those obtained by NGA, and MGA used a lower number of fitness evaluations.

Stomeo et al. [Stomeo et al. 2005] use an evolutionary algorithm $(1+\lambda)$ where the chromosome encodes the cell functionality and its connectivity. The fitness is calculated by a dynamic function that initially counts the number of hits with respect to the truth table and later, when a feasible circuit is found, the number of logic gates is minimized.

That approach works by gradually decomposing a complex system into a series of simpler ones when the evolution does not bring any improvement in terms of fitness function value. These simpler blocks are evolved separately and then merged together once completely developed. Circuits with a maximum of 16 inputs and 128 outputs were used in the computational experiments. That approach required less fitness function eval-

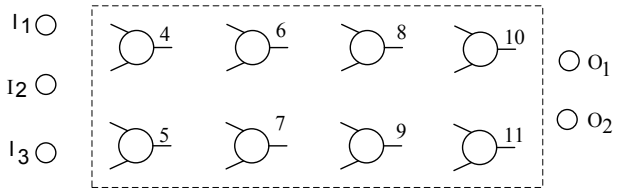

Figure 1. CGP individual with a 2×4 matrix, 3 inputs, and 2 outputs [Manfrini et al. 2016a].

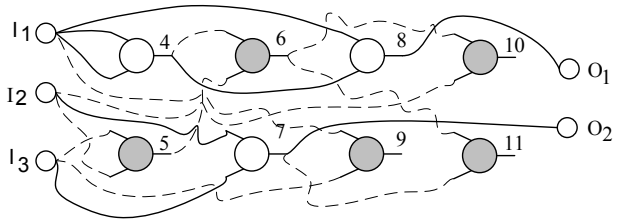

Figure 2. Example of a CGP individual with *levels-back=number of column*. Inactive nodes are in gray and the continuous lines define the phenotype.

uations than other methods from the literature when designing larger CLCs; the largest circuit obtained was a 6-bit digital multiplier.

Karakatic et al. [Karakatic et al. 2013] used a genetic programming technique where each individual generated should be unique to enforce the diversity of the population. A penalty function is used here to evaluate the candidate designs, where the objective function is the number of gates and the constraints are the errors with respect to the truth table. The same circuits used to evaluate NGA and MGA are also considered in the computational experiments of [Karakatic et al. 2013], and their approach was able to find circuits with the same number of gates of those from the literature.

In general, there is no consensus in the literature regarding the most adequate objective function and selection scheme for the design of CLCs when using EH. Therefore, the present work intends to conduct a more in-depth study of the objective functions and the tournament selection strategies. These analyses consider the approaches used in [Coello et al. 2000a, Coello et al. 2000c, Miller 2011a].

3. Cartesian Genetic Programming

Miller [Miller 1999] proposed a new form of genetic programming, called Cartesian Genetic Programming (CGP), in which programs are modeled as directed acyclic graphs (DAG). In CGP, the genes that form the chromosome are integers, and they represent the node inputs and the respective operation that the node performs. A candidate solution is illustrated in Figure 1 and its phenotype is presented in Figure 2.

CGP has three user-defined parameters to represent a program: the number of columns (n_c), the number of rows (n_r) of the matrix which encodes the graph, and levels-back (lb), which limits the elements of the left side of the current one that can be used as its inputs. Each node acts as an operator, and it can be arithmetic, logical, among others, according to the nature of the problem addressed.

A variant of a simple evolutionary algorithm know as $(1 + \lambda)$ is widely used by

CGP and usually λ is chosen to be 4 [Miller 2011a]. In each generation, one parent generates all offspring using mutation, and the best solution among the parent and its offspring is selected to form the next generation.

In CGP, the most used mutation operator is the point mutation [Miller 2011b]. However, when using the point mutation, the changes may occur in inactive nodes leading to a lack of modifications in the phenotype.

These nodes which do not interfere in the output of the solutions are called *inactive nodes* (gray nodes in Figure 2). The single active mutation (SAM) [Goldman and Punch 2013, Goldman and Punch 2015] is another common mutation operator, where (i) one node and one of its elements (inputs or gate) are selected at random, and (ii) the value is changed to another valid value. The steps (i) and (ii) are repeated until an active node is modified. So SAM ensures that one active node is changed.

3.1. Biased single active mutation

In SAM, a given operator in a node of a candidate design is replaced by another randomly selected using a normal distribution. Manfrini et al. [Manfrini et al. 2016a] developed a new mutation operator called biased single active mutation (Biased SAM). The proposed approach analyzes the behavior of the changes in the genotype during the evolutionary process for a given set of (more simple) problems. Based on this analysis, a bias is created to direct the modifications caused by mutation when CGP is applied to other (more complex) problems.

Biased SAM (BSAM) uses a transition probability matrix to bias the changes caused by mutation, and this matrix is created in a training procedure. For each run, every time a child has a fitness value better than that of its parent (the child proceeds to the next generation), we say that a beneficial mutation occurred. If this beneficial mutation occurred by the change of the function performed by the gene (logic gate), the transitions between operations observed in beneficial mutations are counted in a matrix. At the end of the preliminary evolutionary process, we create the frequency table of all beneficial transitions, giving rise to a probability distribution. This probability distribution is used to guide the evolutionary process. Every time a function is to be changed in a candidate design, this probability distribution is used.

4. The Proposed Approaches

In this section, the objective functions (minimize the number of logic gates and maximize the number of wire gates) and the study regarding the tournament selection of the candidate circuits are described. Also, the use of the BSAM with two matrices are presented and then, the proposed CGPs which use both SAM and BSAM are described.

4.1. Objective Functions and Tournament Selection Schemes for Handling Constraints

The first objective function considered here is the number of logic gates (to be minimized) [Miller 2011a, Karakatic et al. 2013]. As the circuit's cost is highly connected to the number of gates, it is a natural objective function when designing combinational digital circuits. On the other hand, as CGP adopts a fixed length genotype, another possibility is to maximize the number of wire gates [Coello et al. 2000a, Coello et al. 2000c]. As the

wire gates of a circuit can be discarded (because this function does not change the states from its inputs), a circuit with many wire gates corresponds to a circuit with a potentially reduced number of logic gates. Although both objectives could be considered similar, it can be noticed that when maximizing the number of wire gates in CGP one also increases the number of active nodes and, according to [Manfrini et al. 2016b], more active nodes facilitate the search for feasible circuits.

Besides the objective functions, we also consider here two ways to handle the constraints using tournament selection schemes. In the first, the best candidate circuit is the one which minimizes the error with respect to the truth table and, when the error values of the circuits are equal, the one with the best value considering the objective function is preferable. Otherwise (both values are equal), the best solution can be randomly selected. This method is similar to that used in [Coello et al. 2000c]. Formally, given the designs x_1 and x_2, the best one between them (x_b) is

$$x_\mathrm{b} = \begin{cases} x_1, & \text{if error}(x_1) < \text{error}(x_2) \\ x_2, & \text{if error}(x_1) > \text{error}(x_2) \\ x_1, & \text{if error}(x_1) = \text{error}(x_2) \text{ and objfunc}(x_1) \text{ is better than objfunc}(x_2) \\ x_2, & \text{if error}(x_1) = \text{error}(x_2) \text{ and objfunc}(x_1) \text{ is worse than objfunc}(x_2) \\ r_{1,2}, & \text{otherwise,} \end{cases} \quad (1)$$

where "error" is the number of errors of the circuit when compared to the truth table, "objfunc" is the objective function, and $r_{1,2}$ selects x_1 or x_2 randomly. We labeled this approach as "ME": tournament selection with constraint handling by minimizing the error.

Another approach is that adopted in [Miller 2011a], where the search is divided into two stages: (i) to minimize the error with respect to the truth table, and (ii) to optimize the objective function but keeping the feasibility of the best solution. In this case, initially only "error" is used to compare two candidate designs; if the error values are equal, then the best one is selected randomly. Once a feasible solution is found, "objfunc" is considered when the two solutions compared are feasible; otherwise, the unfeasible one is discarded. In other words, this approach initially searches for a feasible solution and uses the tournament selection defined by Equation 1 after one is found. This selection scheme is labeled here as "2P".

One can notice that both the minimization of the number of logic gates and the maximization of the number of wire gates can be used as "objfunc" in ME and 2P schemes. Thus, 4 strategies arise.

4.2. Biased SAM with Two Matrices

The original Biased SAM mutation operator was used to guide the search process for finding feasible circuits. Our approach extends this view and uses Biased SAM not only for finding one solution that matches all values in the truth table, but it is also adopted for the optimization stage of the search. However, it is important to highlight that the changes that generate improvement with respect to the objective function are not similar to those observed when minimizing the error of the circuits. Thus, we propose here the use of two independent probability transition matrices: one to be used when minimizing the error with respect to the truth table values (constraints) and another when the objective function is optimized. The first matrix guides the search for circuits that match the values

of the truth table (as in [Manfrini et al. 2016a]), while the second matrix conducts the optimization process to find better solutions according to the objective function.

Despite the good results presented in [Manfrini et al. 2016a], Biased SAM does not perform well in the first stage of the search (searching for a feasible solution) during our preliminary experiments. We believe that the reduced number of available gates adopted here contributed to this decrease of performance.

4.3. CGP with both SAM and Biased SAM

Finally, as both SAM and BSAM perform well in different problems, the last proposed approach is to use both during the search. ES is then modified to a $(1 + \lambda_{\text{SAM}} + \lambda_{\text{BSAM}})$ evolution strategy, where λ_{SAM} and λ_{BSAM} are the number of solutions generated by SAM and BSAM, respectively. It is important to highlight that two TPMs are used by BSAM, as in the first proposed search method: one when minimizing the error with respect to the truth table and another when a feasible solution is found. This approach combines the qualities of both SAM and BSAM. In order to maintain the standard $\lambda = 4$ of CGP, we use here $\lambda_{\text{SAM}} = \lambda_{\text{BSAM}} = 2$.

5. Computational Experiments

Computational experiments were performed to comparatively analyze the performance of (i) the proposed search techniques, (ii) the objective functions considered (number of logic gates and number of wire gates), and (iii) the constraint handling method when CGP is used to design combinational logic circuits.

5.1. Test-problems

The following test-problems were used to comparatively analyze the approaches. The parameters were: μ = 1, λ = 4, n_r = 1, n_c = 100, lb = n_c, and Γ = {AND, OR, NOT, XOR, WIRE}, 30 runs and 100,000 evaluations were allowed for problems 1-4. For Problem 5 we used n_c = 300, 10 runs and 1,000,000 evaluations, because this last one is more complex.

- **Problem 1:** It is formed by 4 inputs and one output, was addressed in [Coello et al. 2000a]. The minterms[1] are: F = $\sum m(0, 1, 3, 6, 7, 8, 10, 13)$.
- **Problem 2:** It is composed by 4 inputs and 3 outputs, was also used in [Coello et al. 2000a]. The minterms are: $F_0 = \sum m(7, 10, 11, 13, 14, 15)$, $F_1 = \sum m(2, 3, 5, 6, 8, 9, 12, 15)$ and $F_2 = \sum m(1, 3, 4, 6, 9, 11, 12, 14)$.
- **Problem 3:** This problem is common in the literature [Coello et al. 2003a, Coello et al. 1997, Coello and Aguirre 2002, Coello et al. 2000b, Coello et al. 2000c, Hilder et al. 2010, Kazarlis et al. 2016, Kazarlis et al. 2014] and represents a 2x2 bit multiplier. It has 4 inputs and 4 outputs.
- **Problem 4** It was used in [Coello et al. 2004], has 4 inputs and 2 outputs. The minterms are: $F_0 = \sum m(0, 1, 2, 4, 5, 8)$ and $F_1 = \sum m(11, 14, 15)$.
- **Problem 5** This final model is a benchmark circuit called CM85A[2], and aims to compare two 4-bit numbers and inform if the first number is greater than, equal to or smaller than the second one. This problem, used in [Wang 2014, Damiani et al. 1995], is more complex than the others used here, and the circuit is composed by 8 inputs and 3 outputs.

[1] Minterms are the product terms where the function value is 1.

[2] http://ddd.fit.cvut.cz/prj/Benchmarks/LGSynth91.pdf

5.2. Objective Functions and Tournament Selection

The computational experiments were performed with SAM when minimizing the number of logic gates (SAM+G) and maximizing the number of wire gates (SAM+W). Both techniques were applied when first we are looking for feasible solutions and after the optimization (2P) and when we want to reach both objectives together (ME). The results are presented in Table 1. The success rate (SR) column means the percentage of the runs that achieved feasible solutions.

Table 1. Number of gates of the best solution and number of evaluations for finding the first feasible solution.

Method	Number of Gates					Number of Evaluations					SR
	Best	Median	Mean	Std	Worst	Best	Median	Mean	Std	Worst	
Problem 1											
SAM+G+2P	7	10	10.33	1.77	14	1649	8030,5	12925,97	10.51×10^3	39477	100%
SAM+W+2P	7	12	11.87	1.98	16	1232	4922	8054.27	7.24×10^3	26703	100%
SAM+G+ME	7	9	8.79	0.92	11	3273	27248	36201.16	29.33×10^3	91964	63.30%
SAM+W+ME	8	11	11.84	2.76	18	753	12770	21532.88	23.16×10^3	94830	83.30%
Problem 2											
SAM+G+2P	7	9	9.13	2.01	13	2923	10509	17981.87	18.34×10^3	76488	100%
SAM+W+2P	10	14	13.93	1.89	17	1567	8692.5	14783.5	16.53×10^3	86262	100%
SAM+G+ME	7	8	7.81	0.83	10	3132	24737	28461.37	23.63×10^3	95637	90%
SAM+W+ME	10	14	14.1	2.45	20	2183	15060.5	27084.05	24.43×10^3	79025	66.67%
Problem 3											
SAM+G+2P	7	8	9.10	2.54	16	1937	9422	11790.77	11.02×10^3	59275	100%
SAM+W+2P	10	13	13.8	2.14	18	2265	9285.5	11333.47	9.94×10^3	48803	100%
SAM+G+ME	7	8	8.39	1.29	13	6402	28064.5	32263.46	23.01×10^3	95637	93.33%
SAM+W+ME	8	14	14.48	2.99	19	4319	10550	29387.76	30.75×10^3	86984	70%
Problem 4											
SAM+G+2P	7	8	8.10	1.16	11	714	8192.5	10095.53	8.98×10^3	29699	100%
SAM+W+2P	9	13	13	2.44	19	845	7379	12091.63	12.72×10^3	51939	100%
SAM+G+ME	7	7	7.75	1.14	11	3815	28961	37299.11	27.85×10^3	96637	93.33%
SAM+W+ME	9	13	12.57	2.53	17	1619	11922.5	14582.21	14.77×10^3	52293	46.67%
Problem 5											
SAM+G+2P	22	25	26.17	3.60	31	471377	636709	671532.83	17.69×10^4	995283	80%
SAM+W+2P	34	39.5	38.67	2.8	42	336907	434590	545908.2	24×10^4	968798	60%
SAM+G+ME	-	-	-	-	-	-	-	-	-	-	0%
SAM+W+ME	-	-	-	-	-	-	-	-	-	-	0%

It was possible to observe that using the maximization of the number of wire gates (W) as objective function requires fewer fitness evaluations to find the first feasible circuit. Also, the techniques using the maximization of the number of wire gates as objective function cannot reach circuits with the same number of gates available in the literature. In addition, an increment in the number of logic gates during the optimization process was observed, as the objective function seeks to increase the number of wire gates.Thus, despite the use of the maximization of the number of wire gates as objective function in the literature [Coello et al. 2000c, Coello et al. 2000a], we concluded that this idea does not work well with CGP.

SAM minimizing the number of logic gates (G) worked well for both the two-step process (2P) and for the tournament selection over the entire search space (ME). In particular, SAM+G+2P reached feasible circuits in 100% of the runs for problems 1-4 and in 80% for Problem 5. On the other hand, SAM+G+ME presented the smallest standard deviations, showing the consistency of this method for optimizing circuits.

We concluded here that SAM+G+2P is the best variant between the approaches using SAM only. As SAM performed better with the G+2P combination, only the results using these components are presented for the remaining techniques.

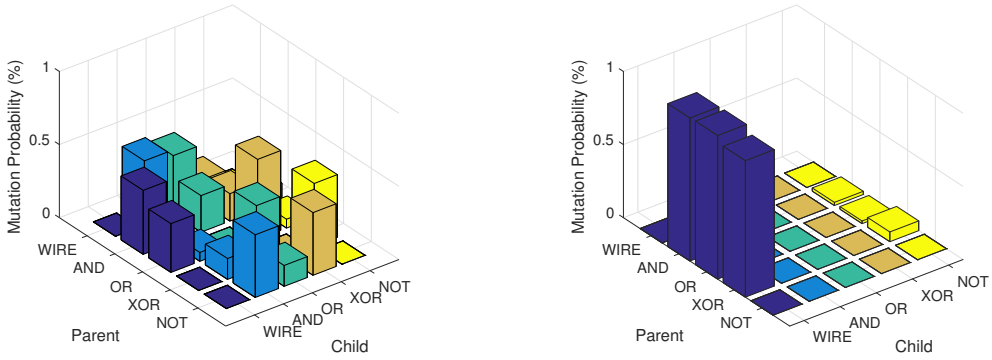

Figure 3. Bar plots of the first stage(left) and second stage(right).

Table 2. Results obtained by BSAM+G+2P+1M and BSAM+G+2P+2M.

Method	Best	Median	Mean	Std	Worst	Success Rate
Problem 1						
BSAM+G+2P+1M	7	11	10.27	1.62	13	100%
BSAM+G+2P+2M	7	9	9.62	1.42	13	100%
Problem 2						
BSAM+G+2P+1M	7	9	9.8	2.44	15	100%
BSAM+G+2P+2M	7	10	10.24	2.75	18	100%
Problem 3						
BSAM+G+2P+1M	7	9	9.73	2.27	14	100%
BSAM+G+2P+2M	8	13	12.71	2.61	18	100%
Problem 4						
BSAM+G+2P+1M	7	9	8.53	1.28	11	100%
BSAM+G+2P+2M	7	9	8.89	1.25	12	100%
Problem 5						
BSAM+G+2P+1M	24	26	27.5	4.09	34	60%
BSAM+G+2P+2M	22	23	23	1	24	60%

5.3. Biased SAM with Two Matrices

As concluded in Section 5.2, the use of minimization of the number of logic gates with the search space divided into two steps (G+2P) we propose the use of the Biased SAM with one matrix for the first step (finding a feasible solution) and another different matrix for the second step (minimizing the number of logic gates). The matrices were built with the same problems used in [Manfrini et al. 2016a] and are formed by four inputs and one output. For these problems, 30 independent runs using SAM+G+2P were performed and 100,000 evaluations were allowed. The number of beneficial mutations obtained in the first and second stages are represented in the bar plot shown in Figure 3. The parents are represented in the lines while the offspring are in the rows of this table. One can notice that the most frequent beneficial mutation is to replace an OR by an XOR. Also, changing WIRE and NOT gates to another gate with 2 inputs improves the quality of the solution.

The computational experiments were performed using only the matrix, where we want to minimize the number of logic gates (1M) and using the two matrices: finding a feasible solution and minimizing the number of logic gates (2M). The results are presented in Table 2. The results indicate that the use of the matrix on the second stage can help finding more compact circuits, mainly when we take into account bigger problems, such as Problem 5. On the other hand, the same effect is not obtained for smaller problems.

5.4. CGP with both SAM and Biased SAM

In order to take advantage of the high success rate of the SAM+G+2P and the capability of find compact circuits of the BSAM+G+2P+2M, we propose the use of the SAM with Biased SAM in both stages of the search as defined in Section 4.3. The results obtained by this approach are presented in Table 3.

Table 3. Results obtained by SAM+BSAM+G+2P.

Problem	Best	Median	Mean	Std	Worst	Success Rate
Problem 1	7	9	9.59	1.43	13	100%
Problem 2	7	9	9.33	1.86	13	100%
Problem 3	7	10	10.30	2.58	16	100%
Problem 4	7	8	8.17	0.95	10	100%
Problem 5	21	22.5	22.5	1.29	24	50%

This approach performed better than or similarly to the other techniques when (i) the number of objective function evaluations and (ii) the number of gates of the final circuit are considered. In particular, SAM+BSAM+G+2P found the designs with the smallest number of gates in all the test-problems. Also, this technique obtained the second lowest standard deviation, indicating that this method is consistent. In Problem 5, which is the most complex circuit considered here, SAM+BSAM+G+2P found the best number of gates with respect to all the metrics presented; the exception is the standard deviation. Despite the smaller success rate of SAM+BSAM+G+2P in Problem 5 when compared to the other approaches, SAM+BSAM+G+2P obtained success rates equal to 100% on the other problems. As the main focus here is to minimize the number of logic gates, SAM+BSAM+G+2P was shown to be a robust and a good performing approach.

6. Concluding Remarks

Different objective functions and constraint handling techniques can be found in the literature for designing CLC's. This paper analyzed the behavior of some of those approaches. Analyzing the results obtained in the computational experiments, we can conclude that CGP obtained the best results when the problem is solved (i) searching by a feasible solution and, in the sequence, (ii) minimizing the number of gates of the circuit.

Also, two approaches using the Biased Single Active Mutation (BSAM) are proposed here. In the first, two transition probability matrices are considered, where one is used when searching by a feasible solution and another matrix is adopted when minimizing the number of gates. The other proposed approach uses SAM and BSAM in both stages of the search. CGP with SAM and BSAM achieved the best results in all the problems tested here.

In spite of that, a deeper study of the evolutionary mechanisms is necessary, mainly when CGP is applied to more complex problems. Also, adaptive techniques are an interesting research avenue in the designing of digital circuits.

References

Brayton, R. K., Hachtel, G. D., McMullen, C., and Sangiovanni-Vincentelli, A. (1984). *Logic minimization algorithms for VLSI synthesis*. Springer Science & Business Md.

Coello, C., Aguirre, A., and Buckles, B. (2000a). Evolutionary multiobjective design of combinational logic circuits. In *Proc. of the 2nd NASA/DoD Workshop on Evolvable Hardware*, pages 161–170. IEEE.

Coello, C. A., Christiansen, A. D., and Aguirre, A. H. (1997). Automated design of combinational logic circuits using genetic algorithms. In *Intl. Conf. on Artificial Neural Nets and Genetic Algorithms*, pages 335–338.

Coello, C. A. C. and Aguirre, A. H. (2002). Design of combinational logic circuits through an evolutionary multiobjective optimization approach. *AI EDAM*, 16(1):39–53.

Coello, C. A. C., Alba, E., and Luque, G. (2003a). Comparing different serial and parallel heuristics to design combinational logic circuits. In *Proc. of the 2003 NASA/DoD Conference on Evolvable Hardware*, pages 3–12.

Coello, C. A. C., Christiansen, A. D., and Aguirre, A. H. (2000b). Towards automated evolutionary design of combinational circuits. *Computers & Electrical Engineering*, 27(1):1–28.

Coello, C. A. C., Christiansen, A. D., and Aguirre, A. H. (2000c). Use of evolutionary techniques to automate the design of combinational circuits. *Intl. J. of Smart Engineering System Design*, 2:299–314.

Coello, C. A. C., Luna, E. H., Aguirre, A. H., et al. (2003b). Use of particle swarm optimization to design combinational logic circuits. In *ICES*, pages 398–409. Springer.

Coello, C. A. C., Zavala, R. L. G., García, B. M., and Aguirre, A. H. (2000d). Ant colony system for the design of combinational logic circuits. In *Intl. Conf. on Evolvable Systems*, pages 21–30. Springer.

Coello, C. C., Luna, E. H., and Aguirre, A. H. (2004). A comparative study of encodings to design combinational logic circuits using particle swarm optimization. In *Proc. of the 2004 NASA/DoD Conference on Evolvable Hardware*, pages 71–78. IEEE.

Damiani, M., Yang, J.-Y., and De Micheli, G. (1995). Optimization of combinational logic circuits based on compatible gates. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(11):1316–1327.

García, B. M. and Coello, C. A. C. (2002). An approach based on the use of the ant system to design combinational logic circuits. *Mathware and Soft Comp*, 9(2-3):235–250.

Goldman, B. W. and Punch, W. F. (2013). Reducing wasted evaluations in cartesian genetic programming. In *European Conf. on GP*, pages 61–72. Springer.

Goldman, B. W. and Punch, W. F. (2015). Analysis of cartesian genetic programmings evolutionary mechanisms. *IEEE Trans. on Evolutionary Computation*, 19(3):359–373.

Hilder, J., Walker, J. A., and Tyrrell, A. (2010). Use of a multi-objective fitness function to improve cartesian genetic programming circuits. In *2010 NASA/ESA Conf. on Adaptive Hardware and Systems*, pages 179–185.

Karakatic, S., Podgorelec, V., and Hericko, M. (2013). Optimization of combinational logic circuits with genetic programming. *Elektronika ir Elektrotechnika*, 19(7):86–89.

Kazarlis, S., Kalomiros, J., and Kalaitzis, V. (2016). A cartesian genetic programming approach for evolving optimal digital circuits. *Journal of Engineering Science & Technology Review*, 9(5).

Kazarlis, S., Kalomiros, J., Mastorocostas, P., Petridis, V., Balouktsis, A., Kalaitzis, V., and Valais, A. (2014). A method for simulating digital circuits for evolutionary optimization. In *Intl. Joint Conf. on Computer, Information, and Systems Sciences, and Engineering (CISSE)*.

Manfrini, F. A. L., Bernardino, H. S., and Barbosa, H. J. C. (2016a). A novel efficient mutation for evolutionary design of combinational logic circuits. In *Intl. Conf. on Parallel Problem Solving from Nature*, pages 665–674. Springer.

Manfrini, F. A. L., Bernardino, H. S., and Barbosa, H. J. C. (2016b). On heuristics for seeding the initial population of cartesian genetic programming applied to combinational logic circuits. In *Genetic and Evol. Comp. Conf. (GECCO)*, pages 105–106.

McCluskey, E. J. (1956). Minimization of boolean functions. *Bell Labs Technical Journal*, 35(6):1417–1444.

Miller, J. F. (1999). An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In *Proc. of the 1st Annual Conf. on Genetic and Evolutionary Comp. Vol 2*, pages 1135–1142. Morgan Kaufmann Pub Inc.

Miller, J. F. (2011a). *Cartesian Genetic Programming*, volume 09-2011 of *Natural Computing Series*. Springer, 2011 edition.

Miller, J. F. (2011b). Cgp. *Cartesian Genetic Programming*, pages 17–34.

Miller, J. F., Job, D., and Vassilev, V. K. (2000). Principles in the evolutionary design of digital circuits, part i. *Genetic programming and evolvable machines*, 1(1-2):7–35.

Sasao, T. (1993). *Logic synthesis and optimization*, volume 1. Springer.

Stepney, S. and Adamatzky, A. (2017). *Inspired by Nature: Essays Presented to Julian F. Miller on the Occasion of His 60th Birthday*, volume 28. Springer.

Stomeo, E., Kalganova, T., Lambert, C., Lipnitsakya, N., and Yatskevich, Y. (2005). On evolution of relatively large combinational logic circuits. In *Proc. of the 2005 NASA/DoD Conference on Evolvable Hardware*, pages 59–66.

Thompson, A., Harvey, I., and Husbands, P. (1996). Unconstrained evolution and hard consequences. In *Towards evolvable hardware*, pages 136–165. Springer.

Vasicek, Z. (2015). Cartesian gp in optimization of combinational circuits with hundreds of inputs and thousands of gates. In *European Conf. on GP*, pages 139–150. Springer.

Vasicek, Z. (2018). Bridging the gap between evolvable hardware and industry using cartesian genetic programming. In *Inspired by Nature*, pages 39–55. Springer.

Vasicek, Z. and Sekanina, L. (2015). Circuit approximation using single-and multi-objective cartesian gp. In *European Conf. on GP*, pages 217–229. Springer.

Vassilev, V. K., Job, D., and Miller, J. F. (2000). Towards the automatic design of more efficient digital circuits. In *NASA/DoD Workshop on Evol. Hardware*, pages 151–160.

Wang, P. (2014). *A comprehensive technique for majority/minority logic synthesis with applications in nanotechnology*. The University of Toledo.