

# Mutual Information-based Fitness Functions for Evolutionary Circuit Synthesis

Arturo Hernández Aguirre  
Center for Research in Mathematics (CIMAT)  
Department of Computer Science  
A.P. 402, Guanajuato, Gto. C.P. 36000 MEXICO  
Email: artha@cimat.mx

Carlos A. Coello Coello  
CINVESTAV-IPN  
Computer Science Section, Electrical Eng. Dept.  
México, D.F., C.P. 07300 MEXICO  
Email: ccoello@cs.cinvestav.mx

**Abstract**—Entropy-based measures, such as Mutual Information and Normalized Mutual Information are investigated as tools for similarity measures between the target and evolving circuit. Three fitness functions are built over a primitive one. We show that the search landscape of Normalized Mutual Information looks more amenable for evolutionary computation algorithms than simple Mutual Information. The evolutionary synthesized circuits are compared to the known optimum size. A discussion of the potential of the Information-Theoretical approach is given.

## I. INTRODUCTION

In this article we use a pure form of Genetic Programming to synthesize logic circuits using binary multiplexer(s) (“mux” or “muxes”). That is, no knowledge is incorporated to the search mechanism other than a fitness function based on entropy.

Claude Shannon suggested the use of information entropy as a measure of the amount of information contained within a message [18]. Thus, entropy tells us there is a limit in the amount of information that can be removed from a random process without information loss. For example, music can be (losslessly) compressed and reduced up to its entropy limit. Further reduction is only possible at the expense of information loss. In a few words, entropy is a measure of disorder and it constitutes the basis of Information Theory.

In this paper we aim to explore the use of entropy-based fitness functions in a pure GP framework. Since no other knowledge than entropy will be used in the fitness function, only limited size circuits can be tested. Through some experiments, we determine what causes the lack of convergence that some researchers have experienced with information theory-based fitness functions [1]. We also propose an alternative solution that we proved quite consistent. The conclusions we draw are applicable to any evolutionary system for Boolean function synthesis based on entropy measures.

The organization of this paper is the following: In Section II, we describe some previous related work from both hybrid and pure areas. Section III provides the detailed description of the problem that we wish to solve. Some basic concepts of information theory and genetic programming are provided in Sections IV and V, respectively. The use of entropy for circuit design is discussed in Section VI. In these line of thoughts, we introduce fitness functions for evolutionary circuit design

in Section VII. A set of experiments illustrating the design of logic circuits is described in Section VIII. In Section IX, we discuss final remarks and give our conclusions.

## II. PREVIOUS WORK

Most of the previous work in this domain has been done using Genetic Algorithms (GAs). Genetic Programming is also included in this review, since this technique is really an extension of GAs in which a tree-based representation is used instead of the traditional linear binary chromosome adopted with GAs.

Two are the main approaches to Boolean function synthesis: pure methods, and hybrid methods. Pure methods are extended with procedures to cope with the poor scalability of evolutionary algorithms. Kalganova [11] proposed a two-way strategy called “bidirectional incremental evolution”, in which circuits are evolved in top-down and bottom-up fashion. Vassilev et al. [22] proposed the use of predefined circuit blocks which can improve both the convergence speed and the quality of the solutions. Although these results are promising, a big problem remaining is the definition of what is a good block for the problem in turn. Recently, Torresen [21] proposed a scalable alternative with limited success called “increased complexity evolution”. Here, training vectors are partitioned, or the training set is partitioned, solving a problem in a divide and conquer fashion. Our own approach to this problem is pure GP system [9], [8]

Hybrid methods have had more success, particularly binary decision diagram based methods. Droste [5] used GP and two heuristics: deletion and merging rules, to reduce directed acyclic graphs. His approach solved the 20-multiplexer problem for the first time ever. Another important hybrid method has been developed by Drechsler et al. [3], [4]. This GP system uses directed acyclic graphs for representing decision diagrams. Two heuristics that are representative of Dreschler’s work are: sifting and inversion.

Information Theory (IT) was early used by Hartmann et al. [6] to convert decision tables into decision trees. Boolean function minimization through IT techniques has been approached by several authors [10], [12]. Some work related to the proposal presented in this article can be found in Luba et al. [13], [14], whom address the synthesis of logic functions using

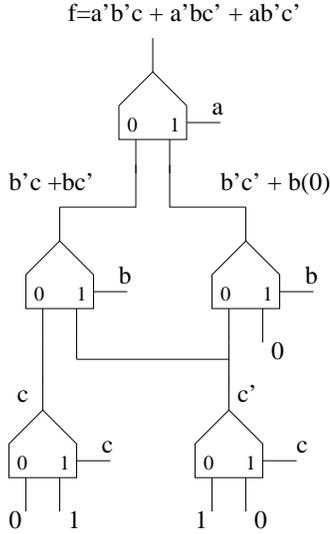


Fig. 1. Shannon expansion implemented with binary multiplexers.

a genetic algorithm and a fitness function based on conditional entropy. Their system (called *EvoDesign*) works in two phases: first, the search space is partitioned into subspaces via Shannon expansions of the initial function. Then the GA is started in the second phase. The authors claim that the partition of the space using entropy measures is the basis for their success. In their domain, a fitness function based on Mutual Information apparently worked well. Note however, that in our case such an approach did not produce good results. Conditional entropy has also been used by Cheushev et al. [1] in top-down circuit minimization methods. In fact, the formal result of Cheushev et al. [1] indicates that a Boolean function can be synthesized by using entropy measures, thus, providing a sound ground for our approach.

### III. PROBLEM STATEMENT

For the purposes of this article, let us consider a target Boolean function  $T$  specified by its truth table. The problem of interest to us is the design of the smallest possible logic circuit, with the minimum number of binary multiplexers that implements the target function. The multiplexer is the only functional unit replicated, each one being controlled by a variable of the target function. Note that only 1s and 0s are fed into the multiplexers (the analog of the Shannon's expansion shown in Figure 1). This strategy allows the implementation of the synthesized circuits by means of pass transistor logic [19]. The design metric driving the implementation is the number of components. Therefore, the best among a set of circuits with the same functionality is the one with the lowest number of components. Our goal is to find 100% functional circuits, specifying components and connections, instead of a symbolic representation of it. Thus, the approach of this paper could be classified as "gate-level synthesis".

Since the number of circuit components is unknown for most circuits, the use of a stochastic method such as Genetic Programming seems appropriate. Also, the tree-like structure

of the circuits makes Genetic Programming the most appropriate evolutionary technique.

### IV. FUNDAMENTALS OF INFORMATION THEORY

Uncertainty and its measure provide the basis for developing ideas about Information Theory [2]. The most commonly used measure of information is Shannon's entropy.

**Definition 2. Entropy:** The average information supplied by a set of  $k$  symbols whose probabilities are given by  $\{p_1, p_2, \dots, p_k\}$ , can be expressed as,

$$H(p_1, p_2, \dots, p_k) = - \sum_{s=1}^k p_k \log_2 p_k \quad (1)$$

The information shared between a transmitter and a receiver at either end of a communication channel is estimated by its Mutual Information,

$$MI(T; R) = H(T) + H(R) - H(T, R) = H(T) - H(T|R) \quad (2)$$

The conditional entropy  $H(T|R)$  can be calculated through the joint probability, as follows:

$$H(T|R) = - \sum_{i=1}^n \sum_{j=1}^n p(t_i r_j) \log_2 \frac{p(t_i r_j)}{p(r_j)} \quad (3)$$

An alternative expression of mutual information is

$$MI(T; R) = \sum_{t \in T} \sum_{r \in R} p(t, r) \log_2 \frac{p(t, r)}{p(t)p(r)} \quad (4)$$

Mutual information, Equation 2, is the difference between the marginal entropies  $H(T) + H(R)$ , and the joint entropy  $H(T, R)$ .

We can explain it as a measure of the amount of information one random variable contains about another random variable, thus it is the reduction in the uncertainty of one random variable due to the knowledge of the other [2].

Mutual information is not an invariant measure between random variables because it contains the marginal entropies. Normalized Mutual Information is a better measure of the "prediction" that one variable can do about the other [20]:

$$NMI(T; R) = \frac{H(T) + H(R)}{H(T, R)} \quad (5)$$

The joint entropy  $H(T, R)$  is calculated as follows:

$$H(T, R) = - \sum_{t \in T} \sum_{r \in R} p(t, r) \log_2 p(t, r) \quad (6)$$

Normalized MI has been used in image registration with great success [15].

**Example:** We illustrate these concepts by computing the Mutual Information between two Boolean vectors  $f$  and  $c$ ,

a	b	c	f=a'b'c+a'bc'+ab'c'
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

TABLE I

FUNCTION  $f = a'b'c + a'bc' + ab'c'$  USED TO COMPUTE  $MI(F;C)$ .

shown in Table I. Variable  $c$  is an argument of the Boolean function  $f(a, b, c) = a'b'c + a'bc' + ab'c'$ .

We wish to estimate the description the variable  $c$  can do about variable  $f$ , that is,  $MI(f; c)$ .

We use Equations 2 and 3 to calculate  $MI(f; c)$ . Thus, we need the entropy  $H(f)$  and the conditional entropy  $H(f|c)$ .

Entropy requires the discrete probabilities  $p(F = 0)$  and  $p(F = 1)$  which we find by counting their occurrences

$$H(f) = -\left(\frac{5}{8}\log_2\frac{5}{8} + \frac{3}{8}\log_2\frac{3}{8}\right) = 0.9544$$

The conditional entropy, Equation 3, uses the joint probability  $p(f_i, c_j)$ , which can be estimated through conditional probability, as follows:  $p(f, c) = p(f)p(c|f)$ . Since either vector  $f$  and  $c$  is made of two symbols, the discrete joint distribution has four entries. This is:

$$\begin{aligned} p(f = 0, c = 0) &= p(f = 0)p(c = 0|f = 0) = \frac{5}{8} \times \frac{2}{5} = 0.25 \\ p(f = 0, c = 1) &= p(f = 0)p(c = 1|f = 0) = \frac{5}{8} \times \frac{3}{5} = 0.375 \\ p(f = 1, c = 0) &= p(f = 1)p(c = 0|f = 1) = \frac{3}{8} \times \frac{2}{3} = 0.25 \\ p(f = 1, c = 1) &= p(f = 1)p(c = 1|f = 1) = \frac{3}{8} \times \frac{1}{3} = 0.125 \end{aligned}$$

Now, we can compute the conditional entropy by using Equation 3. The double summation produces four terms:

$$\begin{aligned} H(f|c) &= -\left(\frac{1}{4}\log_2\frac{1}{2} + \frac{3}{8}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{2} + \frac{1}{8}\log_2\frac{1}{4}\right) \\ H(f|c) &= 0.9056 \end{aligned}$$

Therefore,  $MI(f; c) = H(f) - H(f|c) = 0.9544 - 0.9056 = 0.0488$ . In fact, for the three arguments of the example function we get  $MI(f; a) = MI(f; b) = MI(f; c)$ . The normalized mutual information between either argument and the Boolean function is  $NMI(f; a) = NMI(f; b) = NMI(f; c) = 1.0256$ . Although no function argument seems to carry more information about the function  $f$ , we show later that the landscape of NMI contains a region implying information sharing. This region is hard to find on the MI landscape.

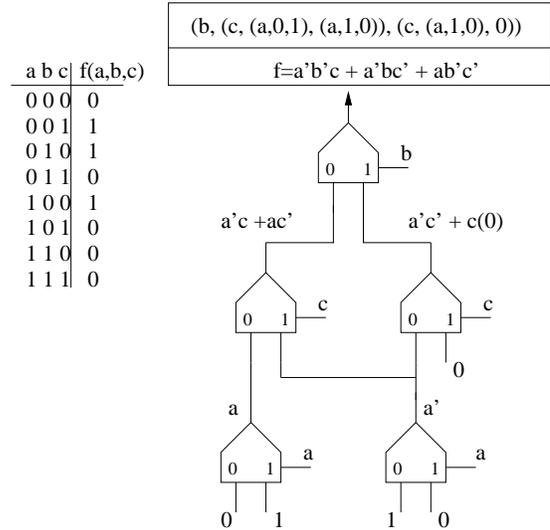


Fig. 2. Logic function specification, encoding of the circuit using lists, and the circuit.

## V. APPLYING GENETIC PROGRAMMING

A circuit is represented as a tree using binary multiplexers as node functions, and 0s and 1s for the leaves. An illustration of this kind of tree is shown in Figure 2 (the circuit was derived using the technique described in this article). The circuit is 100% functionally equivalent to the one derived by Shannon expansions in Figure 1. Note that the circuits are also structurally similar but the muxes are controlled by different variables.

Several issues regarding the application of Genetic Programming as a problem solving tool for Boolean function synthesis are discussed next.

- **Implementation language:** Although the implementation language is not relevant for the results, we chose Prolog because lists are natural structures of this language and allow the representation of trees. The evaluation of either circuit just requires one predicate that translates a list into a tree.
- **Initial population:** In genetic programming, the size of the trees plays an important role in the search. In our work, we adopted the following setting for the size of the trees which was experimentally derived: maximum tree height should not exceed half the number of variables of the Boolean function. These trees could be required to be complete binary trees but our strategy was to randomly create them as to have a rich phenotypic blend in the population.
- **Representation:** A circuit is represented using binary trees, and trees are encoded as Prolog lists. This representation is less flexible than directed acyclic graphs (used in [4]) but still suitable to generate circuits for pass transistor logic. A circuit tree is a recursive list of triplets of the form:  $(mux, left-child, right-child)$ . Mux is assigned a control variable, and either child could be a subtree or a leaf. The muxes are treated as “active high” elements,

therefore the left subtree is followed when the control is 0, and the right subtree otherwise. The tree captures the essence of the circuit topology allowing only the children to feed their parent node, as shown in Figure 2. The representation also captures with no bias the requirement for the leaves of being only 0 or 1.

- **Crossover operator:** The exchange of genetic information between two parent trees is accomplished by exchanging subtrees. Crossover points are randomly selected, therefore, node-node, node-leaf, and leaf-leaf exchanges are allowed since they produce correct circuits. The particular case when the root node is selected to be exchanged with a leaf is disallowed, so that invalid circuits are never generated.
- **Mutation operator:** Mutation is a random change with very low probability of any gene of a chromosome. The mutation of a tree can alter a mux or a leaf. If a mux is chosen then a random variable is generated anew and placed as a new gene value. The mutation of a leaf is as simple as the changing of a 0 to 1, and 1 to 0.
- **Fitness function:** The design of the fitness function using entropy principles is explained in Section VI. Nevertheless, every fitness function used in our experiments works in two stages since the goal is twofold: the synthesis of 100% functional circuits, and their minimization. At stage one, genetic programming explores the search space and builds improved solutions over partial solutions until it finds the first 100% functional circuit. The fitness function for this stage uses entropy concepts in order to reproduce the truth table. Once the first functional circuit appears in the population, a second fitness function is activated for measuring the fitness of the new circuit. Thus, if a circuit is not 100% functional its fitness value is estimated through entropy; if the circuit is 100% functional its fitness value denotes its size and smaller circuits are preferred over larger ones. The fitness value of a 100% functional circuit is always larger than the value of a non functional one, so that circuits are protected from fitness conflicts.

## VI. ENTROPY AND CIRCUITS

Entropy has to be carefully applied to the synthesis of Boolean functions. Let us assume any two Boolean functions,  $F1$  and  $F2$ , and a third  $F3$  which is the one's complement of  $F2$ , then

$$F3 \neq F2$$

For these complementary functions,

$$H(F2) = H(F3)$$

Also Mutual Information shows a similar behavior.

$$MI(F1, F2) == MI(F1, F3)$$

The implications for Evolutionary Computation are important since careless use of entropy-based measures can prevent the system from attaining convergence. Assume the target

Boolean function is  $T$ . Then,  $MI(T, F2) = MI(T, F3)$ , but only one of the circuits implementing  $F2$  and  $F3$  is close to the solution since their Boolean functions are complementary. A fitness function based on mutual information will reward both circuits with the same value, but one is better than the other. Things could get worse as evolution progresses because the mutual information increases when the circuits are closer to the solution, but in fact, two complementary circuits are then given larger rewards. The scenario is one in which the population is driven by two equally strong attractors, hence convergence is never reached.

The fitness function of that scenario is as follows. Let us assume  $T$  is the target Boolean function (must be seen as a Boolean vector), and  $C$  is the output vector of any circuit in the population. Our fitness function is either the maximization of mutual information or the minimization of the conditional entropy term. This is,

$$badfitnessfunction\#1 = MI(T, C) = H(T) - H(T|C) \quad (7)$$

The entropy term  $H(T)$  is constant since  $T$  is the target vector. Therefore, instead of maximizing mutual information, the fitness function can only minimize the conditional entropy,

$$badfitnessfunction\#2 = H(T|C) \quad (8)$$

We called *bad* to these entropy-based fitness functions because none of the experiments had the minimum visage of success. Although mutual information has been described as the “common” information shared by two random processes, the search space spawned by that kind of fitness function is not amenable for evolutionary computation. In Figure 3 we show this search space over mutual information for all possible combinations with two binary strings of 8 bits. The area shown corresponds to about  $\frac{1}{4}$  ( $[1, 150] \times [1, 150]$ ) of the whole search space of ( $[1, 254] \times [1, 254]$ ) (the values 0 and 255 were not used). Horizontal axes are decimal values of 8 bit binary strings, and height represents mutual information.

The mutual information search space, clearly full of spikes, does not favor or emphasizes any area of common information “anticipated by the theory”. For any two equal vectors, their Mutual Information lies on the line at  $45^\circ$  (over points  $\{(1, 1), (2, 2), (3, 3) \dots (n, n)\}$ ). In the next Section we continue this discussion and design fitness functions whose *landscape* seems more promising for exploration.

## VII. FITNESS FUNCTION BASED ON NORMALIZED MUTUAL INFORMATION

So far we have described the poor scenario where the search is driven by a fitness function based on the sole mutual information. We claim that fitness functions based on Normalized Mutual Information (NMI) should improve the performance of the genetic programming algorithm because of the form of the NMI landscape. This is shown in Figure 4 for two 8-bit vectors (as we did for MI in Section VI, Figure 3). Note on the figure how the search space becomes more regular

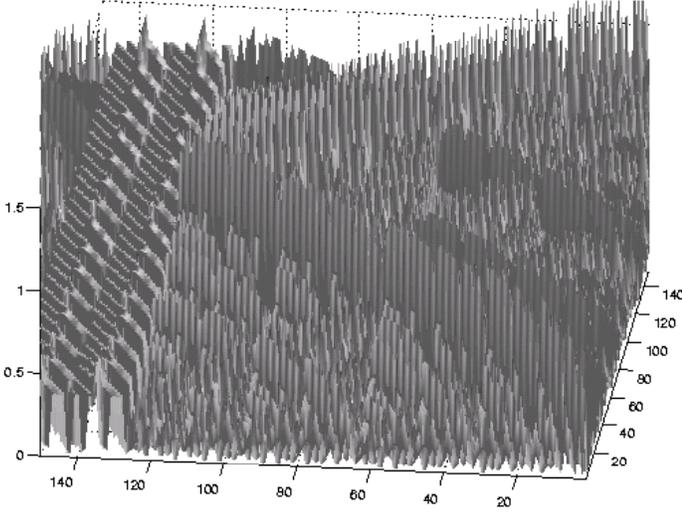


Fig. 3. The search space of Mutual Information (Equation 7).

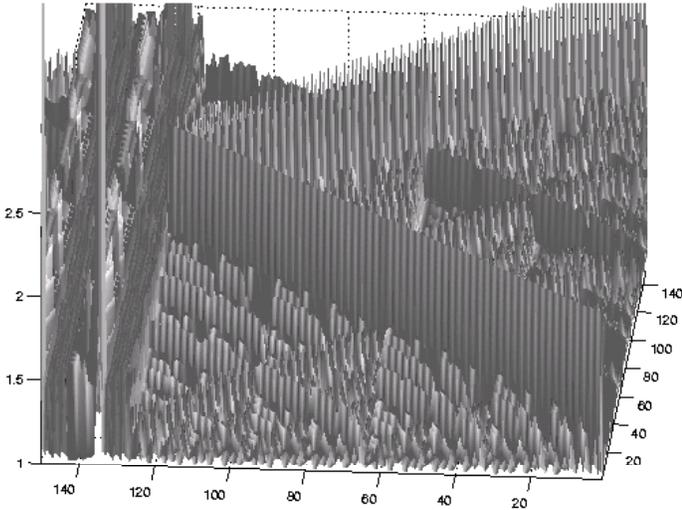


Fig. 4. The search space of Normalized Mutual Information (Equation 5).

and, more important yet, note the appearance of the *wall* at  $45^\circ$  where both strings are equal.

We propose three new fitness functions based on Normalized Mutual Information (Equation 5) and report experiments using the following three fitness functions (higher fitness means better).

Let us assume a target Boolean function of  $m$  attributes  $T(A_1, A_2, \dots, A_m)$ , and the circuit Boolean function  $C$  of the same size. In the following, we propose variations of the basic fitness function of Equation 9, and discuss the intuitive idea of their (expected) behavior.

$$fitness = (Length(T) - Hamming(T, C)) \times NMI(T, C) \quad (9)$$

We tested Equation 9 in the synthesis of several problems and the results were quite encouraging. Thus, based on this

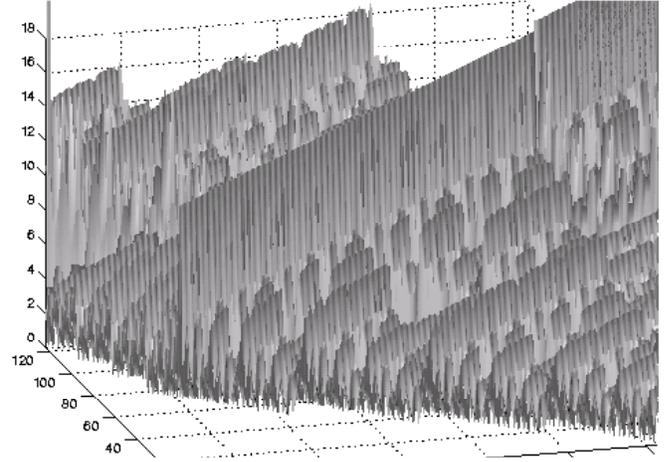


Fig. 5. Fitness landscape of:  $f = (Length(T) - Hamming(T, C)) \times NMI(T, C)$ .

primary equation we designed the following fitness functions. In Figure 5 we show the *fitness landscape* of Equation 9.

$$fitness1 = \sum_{i=1}^m \frac{fitness}{NMI(A_i, C)} \quad (10)$$

$$fitness2 = \sum_{i=1}^m fitness \times NMI(A_i, C) \quad (11)$$

$$fitness3 = (Length(T) - Hamming(T, C)) \times (10 - H(T|C)) \quad (12)$$

The function fitness, Equation 9, is driven by  $NMI(T, C)$  and adjusted by the factor  $Length(T) - Hamming(T, C)$ . This factor tends to zero when  $T$  and  $C$  are far in Hamming distance, and tends to  $Length(T)$  when  $T$  and  $C$  are close in Hamming distance. The effect of the term is to give the correct rewarding of NMI to a circuit  $C$  close to  $T$ . Equation 9 is designed to remove the convergence problems described in the previous section.

*Fitness1* and *Fitness2*, Equations 10 and 11, combine NMI of  $T$  and  $C$  with NMI of  $C$  and the attributes  $A_k$  of the target function. Thus, *fitness1* and *fitness2* look for more information available in the truth table in order to guide the search. *Fitness3* is based on conditional entropy and it uses the mentioned factor to suppress the reproduction of undesirable trees. Since conditional entropy has to be minimized we use the factor  $10 - H(T|C)$  in order to maximize fitness. Equations 8 and 10 use the conditional entropy term. Nevertheless, only Equation 10 works fine. As a preliminary discussion regarding the design of the fitness function, the noticeable difference is the use of Hamming distance to guide the search towards the aforementioned *optimum wall* of the

TABLE II

GENERATION NUMBER WHERE THE FIRST 100% FUNCTIONAL CIRCUIT IS FOUND, AND THE GENERATION WHERE THE OPTIMUM IS FOUND, FOR THE THREE PROPOSED FITNESS FUNCTIONS.

Event	Gen. at fitness1	Gen. at fitness2	Gen. at fitness3
100% Functional	13 ± 5	14 ± 7	18 ± 6
Optimum Solution	30 ± 7	30 ± 10	40 ± 20

TABLE III

GENERATION NUMBER WHERE THE FIRST 100% FUNCTIONAL CIRCUIT IS FOUND, AND THE GENERATION WHERE THE OPTIMUM IS FOUND, FOR THE THREE PROPOSED FITNESS FUNCTIONS.

Event	Gen. at fitness1	Gen. at fitness2	Gen. at fitness3
100% Functional	39 ± 12	40 ± 11	50 ± 12
Optimum Solution	160 ± 15	167 ± 15	170 ± 20

search space. The Hamming distance favors the destruction of individuals on one side of the wall, and favors the other side. Thus, in principle, there is only one attractor in the search space.

## VIII. EXPERIMENTS

In the following experiments we find and contrast the convergence of the genetic programming system for the three fitness functions of Equations 10,11,12.

### A. Experiment 1

Here we design the following (simple) Boolean function:

$$F(a, b, c, d) = \sum(0, 1, 2, 3, 4, 6, 8, 9, 12) = 1$$

We use a population size of 300 individuals, a crossover rate ( $p_c$ ) of 0.35, a mutation rate ( $p_m$ ) of 0.65, and a maximum of 100 generations. The optimal solution has 6 nodes, thus we find the generation in which the first 100% functional solution appears, and the generation number where the optimal is found. The problem was solved 20 times for each fitness function. Table II shows the results of these experiments.

### B. Experiment 2

Our second test function is:

$$F(a, b, c, d, e, f) = ab + cd + ef$$

In this case, we adopted a population size of 600 individuals,  $p_c = 0.35$ ,  $p_m = 0.65$ , and a maximum of 200 generations. The optimal solution has 14 nodes. Each problem was solved 20 times for each fitness function. Table III shows the results of these experiments.

A solution found to this problem is shown in Figure 6. The evolutionary solution is equivalent to the optimum reported by Reduced Order Binary Decision Diagram techniques.

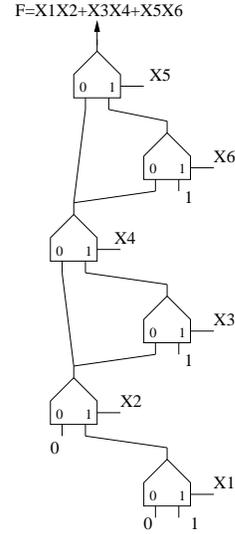


Fig. 6. Solution found by Genetic Programming to Experiment 2

TABLE IV

PARTIALLY SPECIFIED FUNCTION OF EXAMPLE 3 NEEDS  $(2 * 2k) - 1$  MUXES.

ABCD	F(ABCD)
0 0 0 0	0
0 0 0 1	1
0 0 1 0	1
0 1 0 0	1
1 0 0 0	1
0 1 1 1	1
1 0 1 1	1
1 1 0 1	1
1 1 1 0	1
1 1 1 1	0

### C. Experiment 3

Our third problem is related to partially specified Boolean functions [7]. With this experiment we address the ability of the system to design Boolean functions with “large” numbers of arguments and specific topology. For this, we have designed a synthetic problem where the topology is preserved when the number of variables increases.

Boolean functions with  $2k$  variables are implemented with  $(2 * 2k) - 1$  binary muxes if the truth table is specified as shown in Table IV.

We ran experiments for  $k = 2, 3, 4$ , thus 4, 8, and 16 variables and we contrasted these results with the best known so far for this problem (reported in [7]). For completeness, all previous results are reported together with the results of the new experiments in Table V, where we use the three fitness functions proposed before (Equations 10,11,12).

All parameters are kept with no changes for similar experiments, average is computed for 20 independent runs. In previous work we used a fitness function based on the sole Hamming distance between the current solution of an individual and the target solution of the truth table [7]. One important difference is the percentage of correct solutions

TABLE V

GENERATION NUMBER WHERE THE FIRST 100% FUNCTIONAL CIRCUIT IS FOUND, AND THE GENERATION WHERE THE OPTIMUM IS FOUND, FOR THE THREE PROPOSED FITNESS FUNCTIONS.

k	variables	size	Avg(previous)	Avg(fitness1)	Avg(fitness2)	Avg(fitness3)
2	4	7	60	60	60	60
3	8	15	200	190	195	194
4	16	31	700	740	731	748
5	32	63	2000	2150	2138	2150

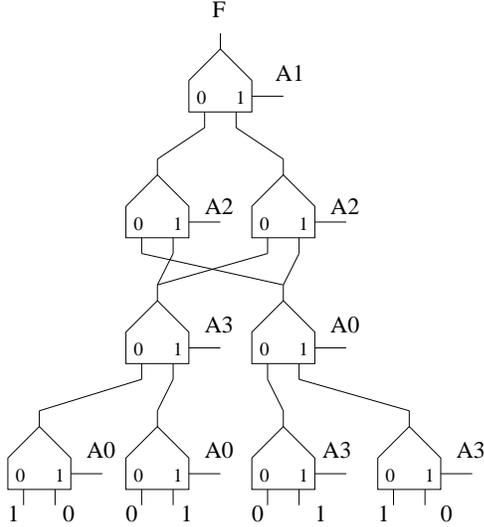


Fig. 7. Optimum solution to even 4-parity circuit problem

found. Previously, we reported that in 90% of the runs we found the solution (for the case of fitness based on Hamming distance). For the three fitness functions based on entropy we found the solution in 99% of the runs.

#### D. Experiment 4

Our fourth (and last) problem, is the synthesis of even 4-parity circuits (the output of the circuit is 1 if the number of ones assigned to the input variables is odd). This experiment is harder to solve because only *XOR* gates are used in the optimum solution. Since our approach will need to implement *XOR* gates by using muxes, or make some abstraction of the overall circuit, interesting behaviors on the fitness functions will be observed. In this case, we adopted a population size of 810 individuals,  $p_c = 0.35$ ,  $p_m = 0.65$ , and a maximum of 300 generations. Each problem was solved 30 times for each fitness function. The optimal solution has 15 nodes, which after removing similar branches gets its final form shown in Figure 7.

For this experiment we report the number of optimum solutions found as a percentage of trials (30). The results are shown in Table VI

The first three columns are similar to previous experiments, the column labeled “Hamming” indicates the use of a fitness function optimizing Hamming distance, and the column labeled “H-NMI” indicates the use of the fitness function

described in Equation 9. Except for function Fitness3, all fitness functions found functional circuits in all cases. It is important to remember that Fitness3 is based on conditional entropy and Hamming distance; the detailed results are: about 50% of the runs strayed from convergence showing an ever increasing number of nodes. In the other 50%, functional solutions were found but showing an erratic behavior. No circuit with optimum fitness solution was found.

Experiments show that Fitness H-NMI is quite similar to fitness Hamming, but Fitness2 and Fitness1 improve H-NMI and Hamming, most likely due to the normalized mutual information measured between the variables of the target and evolving functions.

#### IX. FINAL REMARKS AND CONCLUSIONS

Several experiments were tried with no success for some fitness functions based on conditional entropy. We believe this is a clear indication of a fitness function that does not take into account entropy properties. Thus, we explained how an evolutionary algorithm would not converge because there is more than one attractor in the search space. Figure 3 reveals an amorphous search MI landscape with a quite weak wall at  $45^\circ$ . The left handside of the wall seems more regular than the right handside. Although it is hard to derive any conclusions from this figure, it is clear that no attractor dominates the area and it could explain the failure of the fitness function based on the sole MI.

The landscape of Normalized Mutual Information seems less chaotic and more regular. The great advantage of a fitness function designed over NMI is the appearance of the wall at  $45^\circ$ . It is clear that that wall must appear when the random vectors are equal; as the intersection of the vectors increases so it does the MU. What we have shown in this paper is that, in spite of all the credit given to MI as the “real information” shared between two random processes, the NMI landscape is more amenable for search than the MI landscape. In the landscape of the fitness function of Figure 5, we can see the wall due to equal vectors is preserved, so we believe it is part of the landscape of three fitness functions using Equation 9.

In the first three experiments, the three fitness functions proposed in this paper worked quite well. All of them found the optimum in most cases, thus they are comparable to other fitness functions based on Hamming distance [7]. Nonetheless, Experiment 4, which is harder since the optimum solution is implemented by only using *XOR* gates, tell us a different story. Remember that Fitness1 and Fitness2 are based on NMI, and

TABLE VI

PERCENTAGE OF OPTIMUM SOLUTIONS FOUND IN 30 RUNS, FOR THE THREE PROPOSED FITNESS FUNCTIONS.

Event	Fitness1	Fitness2	Fitness3	Hamming	H-NMI
%Opt. Solutions	54.5%	44.1%	0.0%	36.6%	36.8%

that their design hypothesis is that some relevant information shared between the Boolean variables of the target function and the target function itself, could be extracted and used to guide the search. This seems to be the case of Experiment 4, since the best results are obtained by fitness functions based on NMI.

A final remark goes to the convergence time and quality of results for Experiment 4 previously reported in the specialized literature. Miller et al. [16], solved this problem using a genetic algorithm whose evolution is contained by the matrix representation used (called cartesian genetic programming). They found the optimum in 15% of the runs, each run made 4 million fitness function evaluations. In our case, we only need 240,000 fitness function evaluations, and we get the perfect fitness in 54.5% of the trials. It is not possible to derive firm conclusions from the comparison because the representation and the evolutionary technique of each approach is different, but it is worth to note how our GP based approach needs less computational resources to find perfect fitness circuits. From Tables II and III we can give some advantage to normalized mutual information over simple mutual information because it is less biased. Results from Table V and Table VI could imply that mutual information is able to capture some relationship between the data that the sole Hamming distance cannot convey to the population.

#### ACKNOWLEDGEMENTS

The first author acknowledges partial support from CON-CyTEG project No. 03-02-K118-037 and CONACyT No. 42523. The second author acknowledges support from CONACyT through project no. 42435-Y.

#### REFERENCES

- [1] Cheushev, V.; C. Moraga, S. Yanushkevich, V. Shmerko, and J. Kolodziejczyk. "Information theory method for flexible network synthesis". In *Proceedings of the IEEE 31st. International Symposium on Multiple-Valued Logic*, pages 201–206. IEEE Press, 2001.
- [2] Cover, T.M. and J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons, New York, 1991.
- [3] Drechsler, Rolf; N. Göckel, and B. Becker. Learning Heuristics for OBDD Minimization by Evolutionary Algorithms. In Hans-Michael Voigt and W. Ebeling and I. Rechenberg and H.P. Schwefel, editors, *Proceedings of the Conference Parallel Problem Solving from Nature PPSN-IV, Berlin, 1996* pages 730 –739, Springer, 1996
- [4] Drechsler, Rolf; and Bernd Becker. *Binary Decision Diagrams: Theory and Implementation*. Kluwer Academic Publishers, Boston, USA, 1998.
- [5] Droste, Stefan. "Efficient Genetic Programming for Finding Good Generalizing Boolean Functions". In John R. Koza and Kalyanmoy Deb and Marco Dorigo and David B. Fogel and Max Garzon and Hitoshi Iba and Rick L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 82–87, 1997, Morgan Kaufmann, San Francisco, CA, USA
- [6] Hartmann, C.R.P.; P.K. Varshney, K.G. Mehrotra, and C.L. Gerberich. "Application of information theory to the construction of efficient decision trees". *IEEE Transactions on Information Theory*, 28(5):565–577, 1982.
- [7] Hernández Aguirre, Arturo; Bill P. Buckles, and Carlos A. Coello Coello. "Evolutionary synthesis of logic functions using multiplexers". In C. Dagli, A.L. Buczak, and et al., editors, *Proceedings of the 10th Conference Smart Engineering System Design*, pages 311–315, New York, 2000. ASME Press.
- [8] Hernández Aguirre, Arturo; Bill P. Buckles, and Carlos A. Coello Coello. "Gate-level Synthesis of Boolean Functions using Binary Multiplexers and Genetic Programming". In *Conference on Evolutionary Computation 2000*, pages 675–682. IEEE Computer Society, 2000b.
- [9] Hernández Aguirre, Arturo; Carlos A. Coello Coello, and Bill P. Buckles. "A genetic programming approach to logic function synthesis by means of multiplexers". In Adrian Stoica, Didier Keymeulen, and Jason Lohn, editors, *Proceedings of the First NASA/DoD Workshop on Evolvable Hardware*, pages 46–53, Los Alamitos, California, 1999. IEEE Computer Society.
- [10] Kabakcioglu, A.M.; P.K. Varshney, and C.R.P. Hartmann. "Application of information theory to switching function minimization". *IEE Proceedings, Part E*, 137:387–393, 1990.
- [11] Kalganova, Tatiana. "Bidirectional Incremental Evolution in Extrinsic Evolvable Hardware". In Jason Lohn, Adrian Stoica, Didier Keymeulen, Silvano Colombano, editors, *Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware*, pages 65–74, Los Alamitos, California, 2000. IEEE Computer Society.
- [12] Lloris, A.; J.F. Gomez-Lopera, and R. Roman-Roldan. "Using decision trees for the minimization of multiple-valued functions". *International Journal of Electronics*, 75(6):1035–1041, 1993.
- [13] Luba, T.; C. Moraga, S. Yanushkevich, V. Shmerko, and J. Kolodziejczyk. "Application of design style in evolutionary multi-level network synthesis". In *Proceedings of the 26th EUROMICRO Conference Informatics: Inventing the Future*, pages 156–163. IEEE Press, 2000.
- [14] Luba, T.; C. Moraga, S. Yanushkevich, M. Opoka and V. Shmerko. "Evolutionary multi-level network synthesis in given design style". In *Proceedings of the 30th IEEE International Symposium on Multiple valued Logic*, pages 253–258. IEEE Press, 2000.
- [15] Maes, Frederik; André Collignon, Dirk Vandermeulen, Guy Marchal, and Paul Suetens. "Multimodality image registration by maximization of mutual information". *IEEE Transactions on Medical Imaging*, 16(2):187–198, April 1997.
- [16] Miller, Julian F.; Dominic Job, and Vesselin K. Vassilev. "Principles in the Evolutionary Design of Digital Circuits—Part II". *Genetic Programming and Evolvable Machines*, 1(3):259–288, July 2000.
- [17] Quinlan, J.R. "Learning efficient classification procedures and their application to chess games". In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, pages 463–482. Springer, Berlin, Heidelberg, 1983.
- [18] Shannon, Claude E.. "A Mathematical Theory of Information". *Bell System Technical Journal*, 27:379–423, July 1948.
- [19] Scholl, C.; and Bernd Becker. "On the Generation of Multiplexer Circuits for Pass Transistor Logic". In *Proceedings of Design, Automation, and Test in Europe 2000*.
- [20] Studholme, C.; D.L.G. Hill, and D.J. Hawkes. "An overlap invariant entropy measure of 3D medical image alignment". *Pattern Recognition*, 32:71–86, 1999.
- [21] Torresen, Jim. "A Scalable Approach to Evolvable Hardware". *Genetic Programming and Evolvable Machines*, 3(3):259–282, 2002.
- [22] Vassilev, Vesselin K., and Julian F. Miller. "Scalability Problems of Digital Circuit Evolution". In Jason Lohn, Adrian Stoica, Didier Keymeulen, Silvano Colombano, editors, *Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware*, pages 55–64, Los Alamitos, California, 2000. IEEE Computer Society.
- [23] Weaver, W. and C. E. Shannon. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Illinois, 1949.