Artificial Intelligence for Engineering Design, Analysis and Manufacturing (2002), *16*, 39–53. Printed in the USA.

DOI: 10.1017.S0890060401020054

Design of combinational logic circuits through an evolutionary multiobjective optimization approach

CARLOS A. COELLO COELLO[1] AND ARTURO HERNÁNDEZ AGUIRRE[2]

[1]CINVESTAV-IPN, Depto. de Ingeniería Eléctrica, Sección de Computación, Av. Instituto Politécnico Nacional No. 2508, Col. San Pedro Zacatenco, México, D. F. 07300

[2]Department of Electrical Engineering and Computer Science, Tulane University, New Orleans, LA 70118, USA

(RECEIVED January 29, 1999; REVISED July 18, 2000; ACCEPTED July 26, 2001)

Abstract

In this paper, we propose a population-based evolutionary multiobjective optimization approach to design combinational circuits. Our results indicate that the proposed approach can significantly reduce the computational effort required by a genetic algorithm (GA) to design circuits at a gate level while generating equivalent or even better solutions (i.e., circuits with a lower number of gates) than a human designer or even other GAs. Several examples taken from the literature are used to evaluate the performance of the proposed approach.

Keywords: Circuit Design; Evolvable Hardware; Evolutionary Multiobjective Optimization; Genetic Algorithms; Multiobjective Optimization

1. INTRODUCTION

The genetic algorithm (GA) has been widely used for optimization tasks (Goldberg, 1989) and is known to be a very powerful tool in certain domains. In our current work we wish to find a way to use the GA as a design tool, with particular emphasis in the design of digital combinational circuits.

As is known, there are several standard graphical design aids such as the Karnaugh Maps (Veitch, 1952; Karnaugh, 1953), which are widely used by human designers. There are also other tools more suitable for computer implementation such as the Quine–McCluskey Method (Quine, 1955; McCluskey, 1956), Espresso (Brayton et al., 1984) and MisII (Brayton et al., 1987).

Despite the drawbacks of classical circuit design techniques, some of them can handle truth tables with hundreds of inputs, whereas evolutionary algorithms are restricted to relatively small truth tables (Miller et al., 2000). However, the most interesting aspect of evolutionary design is the possibility of studying its emergent patterns (Coello Coello et al., 2000; Miller, 2000). The goals are, therefore, different when we design circuits using evolutionary algorithms. First, we aim to optimize circuits (using a certain metric) in a different way, and intuitively, we can think of producing novel designs (since there is no human intervention). Such novel designs have been shown in the past (Miller et al., 1999, 2000; Coello Coello et al., 2000). Second, it would be extremely useful to extract design patterns from such evolutionary-generated solutions. This could lead to a practical design process in which a small (optimal) circuit is used as a building block to produce complex circuits. Such a divide-and-conquer approach has also been suggested in the past (Torresen, 1998; Miller et al., 2000).

However, in the previous work on evolutionary design of combinational circuits, efficiency has been an important issue. The main approaches reported so far in the literature require a significant amount of fitness function evaluations. The motivation of this work was precisely to conceive an approach that could reduce the amount of fitness function evaluations, while keeping the capabilities of a GA to generate novel (and compact) designs. This does not mean that we claim that our approach will solve the scalability problem that has characterized evolvable hardware (Thompson et al., 1999; Miller et al., 2000). Nevertheless, we believe that approaches such as the one presented in this paper may contribute to the development of alternative techniques that could improve the performance of a GA, at least when solving relatively small

Reprint requests to: Dr. Carlos A. Coello Coello, P.O. Box 60326-394, Houston, TX 77025, USA. E-mail: ccoello@cs.cnvestav.mx

circuits (under the assumption that they could be used as building blocks to produce larger circuits).

In the past, we have approached this problem using a GA with a matrix encoding scheme, and an *n*-cardinality alphabet (after a series of experiments, we found this *n*-cardinality representation scheme to be more robust than the traditional binary representation (Coello Coello, 1996; Coello et al., 1997, 2000)).

Our original GA-based approach presents great resemblance with the one proposed by Kalganova et al. (1998) and further developed by Miller and his colleagues (1997, 1999, 2000). The two main differences between the two approaches are the encoding scheme and the fitness function, as we will explain later in this paper. However, Miller's initial work emphasized generation of functional circuits, rather than optimization. It was until recently, that Kalganova and Miller (1999) experimented with a two-stage (or multiobjective, as they call it) fitness function. We adopted that sort of fitness function from the beginning of our research in this area (Coello Coello, 1996; Coello Coello et al., 1997). However, the use of truly multiobjective optimization techniques (e.g., based on the concept of Pareto optimality (Coello Coello, 1999)) remained as an open area of research in combinational circuit design, as indicated by Kalganova and Miller (1999).

In this paper, we propose the use of an evolutionary multiobjective optimization technique (rather than just a multiobjective fitness function) to design combinational circuits. There is some (relatively scarce) previous work on using multiobjective techniques to handle constraints. This work, however, has concentrated on numerical optimization only.

Our approach is probably the first attempt to use this kind of technique in the design of circuits, and it seems to considerably reduce the amount of fitness function evaluations required by a GA [at least compared to our previous GA (Coello Coello et al., 2000) and to Miller et al.'s (1997) approach].

Our proposal is to handle each of the matches between a solution generated by a GA and the values specified by the truth table as equality constraints. This, however, introduces some dimensionality problems for conventional multiobjective optimization techniques (this is because checking for dominance is an $O(n^2)$ process), and therefore the idea of using a (more efficient) population-based approach similar to the Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1985).

The remainder of this paper is organized as follows: first, we give some basic definitions related to multiobjective optimization. Then, we describe some of the previous related work on using multiobjective optimization techniques to handle constraints. After that, we state the problem of interest to us, and introduce our approach, giving some examples of its performance. Results are compared against those produced by our previous approach (a GA with an *n*-cardinality alphabet and a two-stage fitness function that we will simply denote as NGA) and against designs produced by humans [using Karnaugh Maps (Karnaugh, 1953), the Quine–McCluskey Procedure (Quine, 1955; McCluskey, 1956)] and another GA (Miller et al., 1998). Then, we present a short discussion of our results, our conclusions and some of the possible paths of future research.

2. MULTIOBJECTIVE OPTIMIZATION

Multiobjective optimization (also called multicriteria optimization, multiperformance or vector optimization) can be defined as the problem of finding (Osyczka, 1985)

> a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria which are usually in conflict with each other. Hence, the term "optimize" means finding such a solution which would give the values of all the objective functions acceptable to the designer.

Formally, we can state the general multiobjective optimization problem (MOP) as follows.

DEFINITION 1 (GENERAL MOP). Find the vector $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ which will satisfy the m inequality constraints:

$$g_i(\vec{x}) \geq 0 \qquad i = 1,2,\ldots,m \tag{1}$$

the p equality constraints

$$h_i(\vec{x}) = 0 \qquad i = 1,2,\ldots,p \tag{2}$$

and optimizes the vector function

$$\vec{f}(\vec{x}) = [\, f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{3}$$

where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables. ■

In other words, we wish to determine from among the set $\mathcal{F}$ of all numbers which satisfy (1) and (2) the particular set $x_1^*, x_2^*, \ldots, x_n^*$ which yields the optimum values of all the k objective functions of the problem.

Another important concept is that of Pareto optimality, which was stated by Vilfredo Pareto in the 19th century (Pareto, 1896), and constitutes by itself the origin of research in multiobjective optimization:

DEFINITION 2 (PARETO OPTIMALITY). We say that $\vec{x}^* \in \mathcal{F}$, is **Pareto optimal** if for every $\vec{x} \in \Omega$ and $I = \{1,2,\ldots,k\}$ either,

$$\bigwedge_{i \in I} (\, f_i(\vec{x}) = f_i(\vec{x}^*)) \tag{4}$$

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

or there is at least one $i \in I$ such that (assuming maximization)

$$f_i(\vec{x}) \leq f_i(\vec{x}^*). \quad (5)$$

■

In words, this definition says that $\vec{x}^*$ is Pareto optimal if there exists no feasible vector $\vec{x}$ which would increase some criterion without causing a simultaneous decrement in at least one other criterion.

Pareto optimal solutions are also termed noninferior, admissible, or efficient solutions (Horn, 1997a); their corresponding vectors are termed nondominated. These solutions may have no clearly apparent relationship besides their membership in the Pareto optimal set. This is the set of all solutions whose corresponding vectors are nondominated with respect to all other comparison vectors. When plotted in objective space, the nondominated vectors are collectively known as the Pareto front.

In this paper, we will be referring to these concepts, although our approach does not necessarily produce Pareto optimal solutions. The Vector Evaluated Genetic Algorithm (VEGA) in which our approach is inspired is known to be biased towards the generation of individuals that excel in one dimension of performance [i.e., in one objective function rather than generating good "trade-offs," which is what other approaches such as Pareto ranking (Fonseca & Fleming, 1993) tend to do]. However, we argue that in the context of circuit design (as well as other design areas), the cooperative mechanism implicit in a population-based approach such as VEGA can be exploited to perform a more efficient search. Therefore, we do not really aim to generate Pareto optimal designs, but instead, we aim to approach efficiently (i.e., at a low computational cost) the feasible region of circuit design problems (a task that normally consumes a lot of CPU time). Thus, the reason why the previous concepts were included is for completeness, so that some of the related work and related concepts mentioned in this paper can be fully understood and, therefore, the paper can be self-contained.

3. HANDLING CONSTRAINTS

The idea of using multiobjective optimization techniques to handle constraints is not new. Some researchers have proposed to redefine the single-objective optimization of $f(\vec{x})$ as a multiobjective optimization problem in which we will have $m + 1$ objectives, where m is the number of constraints. Then, we can apply any multiobjective optimization technique (Fonseca & Fleming, 1995; Coello Coello, 1999) to the new vector $\vec{v} = (f(\vec{x}), f_1(\vec{x}), \ldots, f_m(\vec{x}))$, where $f_1(\vec{x}), \ldots, f_m(\vec{x})$ are the original constraints of the problem. An ideal solution $\vec{x}$ would thus have $f_i(\vec{x}) = 0$ for $1 \leq i \leq m$ and $f(\vec{x}) \geq f(\vec{y})$ for all feasible $\vec{y}$ (assuming maximization).

Surry et al. (1995), and Surry and Radcliffe (1997) proposed the use of Pareto ranking (Fonseca & Fleming, 1993) and VEGA (Schaffer, 1985) to handle constraints using this technique. In their approach, called COMOGA, the population was ranked based on constraint violations (counting the number of individuals dominated by each solution). Then, one portion of the population was selected based on constraint ranking, and the rest based on real cost (fitness) of the individuals.

Parmee and Purchase (1994) implemented a version of VEGA (Schaffer, 1985) that handled the constraints of a gas turbine problem as objectives to allow a genetic algorithm to locate a feasible region within the highly constrained search space of this application. However, VEGA was not used to further explore the feasible region, and instead Parmee and Purchase (1994) opted to use specialized operators that would create a variable-size hypercube around each feasible point to help the genetic algorithm to remain within the feasible region at all times.

Camponogara and Talukdar (1997) proposed the use of a procedure based on an evolutionary multiobjective optimization technique. Their proposal was to restate a single objective optimization problem in such a way that two objectives would be considered: The first would be to optimize the original objective function and the second would be to minimize the total amount of constraint violation of an individual.

Once the problem is redefined, nondominated solutions with respect to the two new objectives were generated. The solutions found defined a search direction $d = (x_i - x_j)/|x_i - x_j|$, where $x_i \in S_i$, $x_j \in S_j$, and S_i and S_j are Pareto sets. The direction search d is intended to simultaneously minimize all the objectives (Camponogara & Talukdar, 1997). Line search is performed in this direction so that a solution x can be found such that x dominates x_i and x_j (i.e., x is a better compromise than the two previous solutions found). Line search takes the place of crossover in this approach, and mutation is essentially the same, where the direction d is projected onto the axis of one variable j in the solution space (Camponogara & Talukdar, 1997). Additionally, a process of eliminating half of the population is applied at regular intervals (only the less fitted solutions are replaced by randomly generated points).

Jiménez and Verdegay (1999) proposed the use of a min-max approach (Chankong & Haimes, 1983) to handle constraints. The main idea of this approach is to apply a set of simple rules based on constraint violation to decide the selection process (individuals with the lowest amount of constraint violation would be preferred in a binary tournament).

In the context of combinational logic circuits design, we are not aware of any work in which the direct use of a multiobjective optimization technique had been proposed, except for the single circuit solved in Coello Coello (2000). The idea was, however, suggested by Kalganova and Miller (1999). Nevertheless, evolutionary multiobjective optimization approaches have been used by several researchers to

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

solve some other related problems. For example, Wilson and Macleod (1993) used Pareto ranking (Goldberg, 1989) to design multiplierless IIR filters; Zebulum et al. (1998) used a GA with a target vector approach (with adaptive weights) for the synthesis of low-power operational amplifiers; Harris and Ifeachor (1996) used Pareto ranking to design nonlinear Finite Impulse Response (FIR) filters; and so forth.

4. STATEMENT OF THE PROBLEM

The problem of interest to us consists of designing a circuit that performs a desired function (specified by a truth table), given a certain specified set of available logic gates.

In circuit design, one can use various criteria to define minimal-cost expressions. For example, from a mathematical perspective, one could minimize the total number of literals or the total number of binary operations or the total number of symbols in an expression. The minimization problem is difficult for all such cost criteria. In gate networks, one could minimize the total number of gates subject to such restrictions as fan-in, fan-out, number of levels, or the total number of SSI packages. In general, it is very difficult to find such minimal networks or to prove the minimality of a given network (Brzozowski & Yoeli, 1976). In spite of this, it is possible to solve a number of minimization problems using systematic techniques, provided that we are satisfied with less general solutions.

The complexity of a logic circuit is a function of the number of gates in the circuit. The complexity of a gate generally is a function of the number of inputs to it. Because a logic circuit is a realization (implementation) of a Boolean function in hardware, reducing the number of literals in the function should reduce the number of inputs to each gate and the number of gates in the circuit—thus reducing the complexity of the circuit.

In this work, we propose a GA that uses a population-based approach to design circuits. The results produced are compared against those produced by another GA [called *n*-cardinality GA or NGA (Coello Coello et al, 2000)]. We also compare our results against with those generated by a human designer using Karnaugh maps and another one using the Quine–McCluskey Procedure (unless indicated otherwise in the examples). The comparison against human designers is in many ways unfair because of differing capabilities of man and machine. For example, a human designer tends to use only the gates NOT, AND, OR and has more difficulties using XOR because the Karnaugh Map and the Quine–McCluskey Procedure do not support the identification of XOR terms as well as they support "seeing" simple product terms. The computer, using a GA-based approach, and not being restricted by human pattern recognition abilities, uses many XOR gates, often disregarding the NOT gate.

Our overall measure of circuit optimality is the total number of gates used, regardless of their kind. This is approximately proportional to the total part cost of the circuit. Obviously, we perform this analysis for only fully functional circuits.

An interesting aspect of this work relates to the analysis of the type of solutions that the GA generates. We have found in the past (Coello Coello et al., 2000) (and again in the work currently reported) that the GA tends to find certain design patterns that, through replication, can produce very compact designs. In fact, through a careful analysis of the solutions generated by a GA, we have been able to extract some of its design patterns and to use them both to improve convergence of the GA itself and to enrich the set of simplification rules normally used by human designers [see Islas Pérez et al., 2001, for details]. Some of these design aspects will be briefly discussed in Section 8.

5. THE GENETIC ALGORITHM USED

We used a matrix to represent a circuit also adopted in previous work (Coello Coello et al., 1997, 2000), as shown in Figure 1. This matrix is encoded as a fixed-length string of integers from 0 to $N-1$, where N refers to the number of rows allowed in the matrix (we call it *n*-cardinality alphabet).

More formally, we can say that any circuit can be represented as a bidimensional array of gates $S_{i,j}$, where j indicates the *level* of a gate, so that those gates closer to the inputs have lower values of j. (Level values are incremented from left to right in Figure 1.) For a fixed j, the index i varies with respect to the gates that are "next" to each other in the circuit, but without being necessarily connected. Each matrix element is a gate (there are five types of gates: AND, NOT, OR, XOR, and WIRE[1]) that receives its two inputs from any gate at the previous column as shown in Figure 1. Although our GA implementation allows gates with more inputs and these inputs might come from any previous level of the circuit, we limited ourselves to two-input gates and restricted the inputs to come only from the previous level. This restriction could, of course, be relaxed, but we adopted it to allow a fair comparison with our previous GA-based approach (it should be kept in mind that the main motivation of this work was to improve the efficiency of our previous GA).

A chromosomic string encodes the matrix shown in Figure 1 by using triplets in which the two first elements refer to each of the inputs used, and the third is the corresponding gate from the available set.

The matrix representation adopted in this work was originally proposed by Louis (1993; Louis & Rawlins, 1991, 1993). He applied his approach to a two-bit adder and to the *n*-parity check problem (for $n = 4,5,6$). This representation has also been adopted by Miller et al. (1997, 2000) with some differences. For example, the restrictions regarding the source of a certain input to be fed in a matrix element varies in each of the three approaches: Louis (1993) has

[1]WIRE basically indicates a null operation, or in other words, the absence of gate, and it is used just to keep regularity in the representation used by the GA that otherwise would have to use variable-length strings.

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

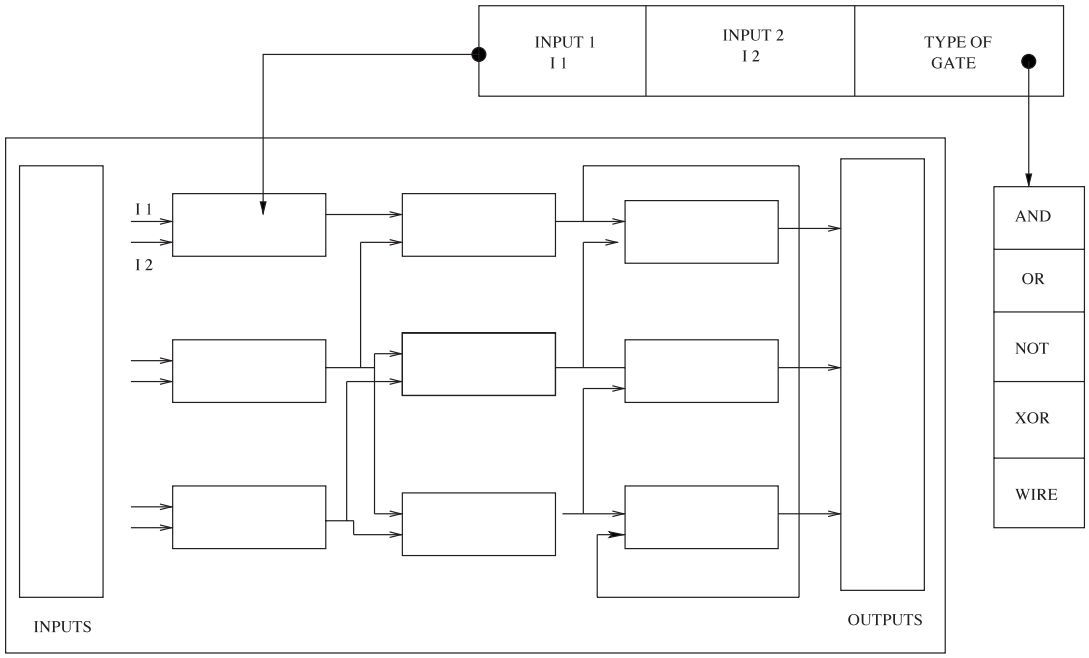

Fig. 1. Matrix used to represent a circuit. Each gate gets its inputs from either of the gates in the previous column. Note the encoding adopted for each element of the matrix as well as the set of available gates used.

strong restrictions, Miller et al. (1997) have no restrictions, and we have relatively light restrictions. The encoding is also different in all cases. Louis (1993) only encoded information regarding one input and the type of gate to be used at each matrix position. He also used binary representation. In our case, we used an *n*-cardinality alphabet and decided to encode the gate to be placed at each matrix location plus its two inputs. Miller et al. (1997) encode a full Boolean operation using a single integer. This representation is more compact, but it has the problem of requiring that mutation takes the place of crossover to introduce enough diversity in the population, so that the evolutionary algorithm can approach the feasible region. That is the reason why Miller et al. (2000) have adopted an evolutionary strategy in their recent work.

Finally, the last difference among the three approaches previously mentioned is regarding the fitness function. Louis (1993) simply maximizes the number of matches between the outputs produced by the circuit and those indicated in the truth table. We have used a fitness function that works in two stages: First, it maximizes the number of matches (as in Louis' case). However, once feasible solutions are found, we maximize the number of WIREs in the circuit. By doing this, we actually optimize the circuit in terms of the number of gates that it uses. Miller et al. (1997) did something similar to Louis until recently [they have recently introduced a two-stage fitness function like the one adopted by us (Kalganova & Miller, 1999)].

Thus, we can say that our goal was to produce a fully functional design (i.e., one that produces all the expected outputs for any combination of inputs according to the truth table given for the problem) which maximizes the number of WIREs. We also aimed to reduce the computational cost of our previous GA-based approach.

6. DESCRIPTION OF THE APPROACH

The main idea behind our proposed approach is to use a population-based multiobjective optimization technique such as VEGA (Schaffer, 1985) to handle each of the outputs of a circuit as an objective. In other words, we would have an optimization problem with m equality constraints, where m is the number of values (i.e., outputs) of the truth table that we aim to match. So, for example, a circuit with three inputs and a single output, would have $m = 2^3 = 8$ values to match.

The technique may be better illustrated by Figure 2. At each generation, the population is split into $m + 1$ subpopulations, where m is defined as indicated before (we have to add one to consider also the objective function). Each subpopulation is on charge of optimizing a constraint of the problem (in this case, an output of the circuit) and an additional subpopulation will optimize the original objective function (unconstrained). Therefore, the main goal of each subpopulation is to match its corresponding output with the value indicated by the user in the truth table. Although the size of each subpopulation may be variable, it was decided to allocate the same size to each of them in the experiments reported in this paper, but the use of different subpopulation sizes is also possible.

The objective function in our case is defined as in previous work (Coello Coello et al., 1997, 2000): it is the total number of matches (between the outputs produced by an encoded circuit and the intended values defined in the truth

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

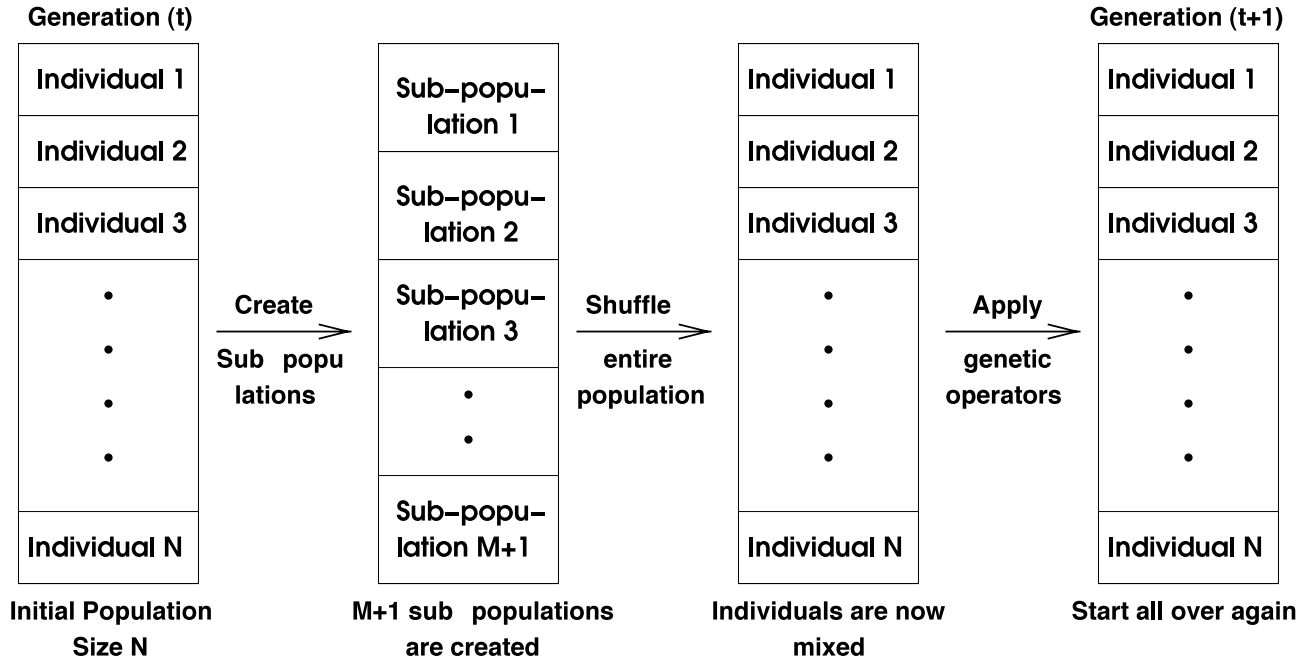

Fig. 2. Graphical representation of the approach proposed in this paper. Note that although individuals are selected using different criteria depending on the subpopulation in which they are placed, crossover is allowed between individuals of different subpopulations. The new population is generated after shuffling the old population and applying to it crossover and mutation.

table defined by the user). For each match, we increase the value of the objective function by one. If the encoded circuit is feasible (i.e., it matches the truth table completely), then we add one (the so-called "bonus") for each WIRE present in the solution.

Using the proposed scheme, a fraction of the population will be selected using the objective function as its fitness (i.e., it will try to maximize the total number of matches); another fraction will use the match of the first output as its fitness and so on (since they are all binary values, we only check if it matches or not, without computing any extra values as required in numerical optimization). The main issue here is how to handle the different situations that could arise. Fitness within each subpopulation is computed using the following scheme:

$$\begin{array}{lll} \textbf{if } o_j(\vec{x}) \neq t_j & \textbf{then} & \text{fitness}(\vec{x}) = 0 \\ \textbf{else if } v \neq 0 & \textbf{then} & \text{fitness}(\vec{x}) = -v \\ \textbf{else} & & \text{fitness}(\vec{x}) = f(\vec{x}) \end{array}$$

where $o_j(\vec{x})$ refers to the value of output j for the encoded circuit $\vec{x}$; t_j is the value specified for output j in the truth table; and v is the number of outputs that are not matched by the circuit $\vec{x}$ ($\leq m$). Finally, $f(\vec{x})$ is the fitness function described before:

$$f(\vec{x}) = h(\vec{x}) + \begin{cases} 0 & \text{if } f(\vec{x}) \text{ is infeasible} \\ w(\vec{x}) & \text{otherwise.} \end{cases}$$

In this equation, $h(\vec{x})$ refers to the number of matches between the circuit $\vec{x}$ and the values defined in the truth table, and $w(\vec{x})$ is the number of WIREs in the circuit $\vec{x}$. Therefore, selection is performed using different rules within each subpopulation. However, crossover and mutation are applied to the entire population (i.e., no "speciation" mechanism is used). This intends to recombine the chromosomic material corresponding to different partially functional circuits, as to allow convergence towards fully feasible circuits.

The algorithm of our approach is the following:

1. Generate randomly a population of size P.
2. Split the population into $m + 1$ subpopulations ($m =$ number of outputs to match).
3. Compute fitness values according to the goals of each individual within each subpopulation:
   - If the target output is not matched, fitness is zero.
   - Else, if the target output is matched, but the circuit is not functional, then fitness is the number of outputs not matched multiplied by (-1).
   - Else, if the target output is matched AND the circuit is functional, then fitness is given by the addition of the number of outputs matched plus the number of wires of the circuit.
4. Shuffle the entire population and select parents from each subpopulation based on the (previously computed) fitness value of each individual.
5. Apply crossover and mutation to the entire population. Individuals of any given subpopulation are allowed to breed with individuals of any other subpopulation. This will generate the new population P'.

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

6. If convergence criterion reached, then stop.
7. Otherwise, return to step 2.

There are a few interesting things that can be observed from this procedure. First, each subpopulation associated with an output of the circuit will try to match it with the value defined in the truth table. Once this is achieved, then the fitness function will try to maximize the number of matches of the rest of the outputs. In other words, this subpopulation will cooperate with the others that are having difficulties to match their outputs. If the circuit is feasible, then all the subpopulations will join efforts to maximize the number of WIREs in the circuit.

It is important to clarify that the current approach does not use dominance to impose an order on the constraints based on their violation [as in the case of COMOGA (Surry et al., 1995)] which is a more expensive process (in terms of CPU time) that also requires additional parameters. In fact, the current approach does not rank individuals, but it uses instead different fitness functions for each of the subpopulation allocated (whose number depends on the number of outputs in a circuit) depending on the feasibility of the individuals contained within each of them. This is easier to implement, does not require special operators to preserve feasibility [as in the case of Parmee and Purchase's approach (1994)], makes unnecessary the use of a sharing function to preserve diversity (Deb & Goldberg, 1989) [such as the traditional multiobjective optimization techniques (Fonseca & Fleming, 1995)], and does not require extra parameters to control the mixture of feasible and infeasible individuals [as in the case of COMOGA (Surry et al., 1995)].

VEGA is known to have difficulties in multiobjective optimization problems due to the fact that it tries to find individuals that excel only in one dimension regardless of the others [the so-called "middling" problem (Schaffer, 1985; Fonseca & Fleming, 1995; Coello Coello, 1999)]. However, that drawback turns out to be an advantage in this context, because what we want to find are precisely circuits that are fully functional, instead of good compromises that may not satisfy one of the outputs (which are the kinds of solutions that a Pareto ranking strategy would normally produce; Coello Coello, 1999). Also, the use of subpopulations is much more efficient than using Pareto dominance, because of the potentially high number of objectives involved (this will be illustrated in the examples shown in this paper).

7. COMPARISON OF RESULTS

We have used several circuits of different degrees of complexity to test our approach. For the purposes of this paper, five examples were chosen to illustrate our approach (called multiobjective genetic algorithm, or MGA for short), and the results produced were compared with those generated by human designers and by our previous *n*-cardinality GA (called NGA; Coello Coello et al., 1997, 2000).

In each case, the size of the matrix used to fit the circuit was determined using the following procedure:

1. Start with a square matrix of size 5.
2. If no feasible solution is found using this matrix, then increase the number of columns by one.
3. If no feasible solution is found using this matrix, then increase the number of rows by one.
4. Repeat steps 2 and 3 until a suitable matrix is produced.

As we will see in the following examples, it was normally the case that for small circuits a matrix of 5×5 was sufficient. However, in one of the examples, it was necessary to reach a matrix size of 6×7. This made necessary to run the GA for more generations, performing, in consequence, more fitness function evaluations. This situation normally arises with circuits having several outputs, although in some cases, such as in the two-bit multiplier of our fourth example, even a 5×5 matrix may be enough to find the best known circuit.

To choose the size of each subpopulation in the MGA, we started with 10, and performed 20 runs. If we did not find feasible solutions in at least one-fourth of our runs, we would increase the subpopulation size by 10 and would perform 20 more runs. This process was repeated until a suitable subpopulation size was found.

The other issue is regarding the crossover and mutation rates. After a series of experiments, we decided to use a crossover rate of 50% and a mutation rate such that each string had a 50% probability of being mutated at a certain position. Since mutation was applied on a single-gene basis, we used as our probability of mutation the result of dividing this 50% by the length of the string. For example, when a 5×5 matrix was used, the length of the chromosomic string was 75. Therefore, the probability of mutation would be 0.006667.

7.1. Example 1

Our first example is a three-even parity problem, whose truth table with three inputs and one output is shown in Table 1. In this case, the matrix used was of size 5×5, and the length of each string representing a circuit was then $3 \times 5 \times 5 = 75$. The cardinality c used for this problem was $\max(r, g)$, where r refers to the number of rows in the matrix and g to the number of allowable gates in the circuit (since only the inputs from the previous level are considered, the number of columns does not affect the cardinality used). Since $g = 5$, and $c = 5$ for this example, then the size of the intrinsic search space for this problem is $c^l = 5^{75} \approx 2.6 \times 10^{52}$. Fitness is computed in the following way: 8 (number of outputs that we must match to have a feasible circuit) + 5×5 (size of the matrix) − number of gates used (i.e., different of WIRE). Therefore, a fitness of 29 (the best value produced for this circuit) means that the circuit is feasible (otherwise, its fitness could not possibly be

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

Table 1. *Truth table for the circuit of the first example*

X	Y	Z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

above 8), and it has four gates (i.e., 21 WIREs), because $8 + (25 - 4) = 8 + 21 = 29$.

Results are compared in Table 2. Human Designer 1 used Karnaugh Maps plus Boolean algebra identities to simplify the circuit, whereas Human Designer 2 used the Quine–McCluskey Procedure. In both cases, they produced solutions with more gates than the MGA or the NGA.

A subpopulation size of 10 was enough for the MGA. Since the circuit has eight outputs, there were nine objectives. Therefore, the total population size was set to 90. We set the maximum number of generations to 300.

To make a fair comparison, the same representation scheme and the same genetic operators (two-point crossover with a probability of 0.5, and uniform mutation with a probability of 0.006667) were used for both the MGA and the NGA (for more details on the NGA, refer to Coello Coello et al., 2000).

The MGA consistently found a solution with a fitness value of 29 (75% of the time), and it produced feasible circuits 100% of the time. The average fitness of the 20 runs performed was 28.75, with a standard deviation of 0.433012. The graphical representation of this solution is depicted in Figure 3.

On the other hand, the best solution that the NGA could find using the same population size had also a fitness of 29 (i.e., a circuit with four gates), but it appeared only 10% of the time. Also, 20% of the time, the best solution found was infeasible. The average fitness of these 20 runs was 21.4, with a standard deviation of 8.438009244.

Table 3. *Truth table for the circuit of the second example*

Z	W	X	Y	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

7.2. Example 2

Our second example has four inputs and one output, as shown in Table 3. A matrix of the same size as before was used (i.e., 5×5).

The comparison of the results produced by the MGA, the NGA, a human designer using Karnaugh Maps, and Sasao's approach (1993) are shown in Table 4. Sasao (1993) has used this circuit to illustrate his circuit simplification technique based on the use of ANDs and XORs. His solution uses, however, more gates than the circuit produced by the NGA or the MGA.

Since this example has 16 outputs, there are 17 objectives for the MGA. A population size of 170 was enough to solve this circuit. The maximum number of generations in this case (for both the MGA and the NGA) was set to 400.

The MGA found a solution with a fitness value of 34 (i.e., a circuit with seven gates) 15% of the time, and solutions with eight gates were found 25% of the time. The MGA produced feasible circuits 100% of the time. The average fitness of the 20 runs performed was 32.1, with a

Table 2. *Comparison of the best solutions found by the n-cardinality GA (NGA), our multiobjective genetic algorithm (MGA), and two human designers (HD 1 and HD 2) for the circuit of the first example*

MGA	NGA	HD 1	HD 2
$F = (X + Y)Z \oplus (XY)$	$F = Z(X + Y) \oplus (XY)$	$F = Z(X \oplus Y) + Y(X \oplus Z)$	$F = X'YZ + X(Y \oplus Z)$
4 gates	4 gates	5 gates	6 gates
2 ANDs, 1 OR, 1 XOR, 1 NOT	2 ANDs, 1 OR, 1 XOR	2 ANDs, 1 OR, 2 XORs	3 ANDs, 1 OR, 1 XOR, 1 NOT

A population size of 90 was used with both GAs.

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

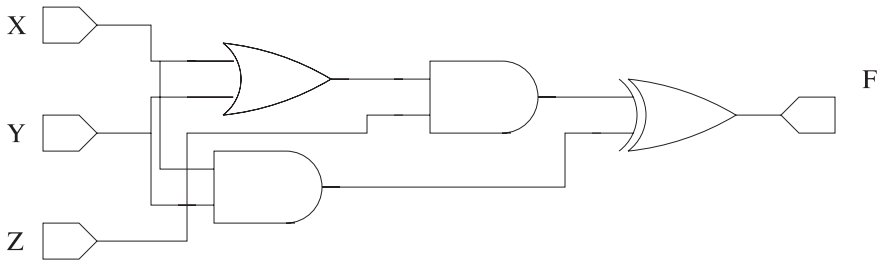

Fig. 3. Graphical representation of the best circuit found by the MGA and the NGA for the first example.

standard deviation of 1.252366. The graphical representation of the best solution found is depicted in Figure 4.

The best solution that the NGA could find using the same population size had a fitness of 31 (i.e., a circuit with 10 gates), and it appeared only once in the 20 runs performed. Also, 95% of the time, the best solution found was infeasible. The average fitness of these 20 runs was 15.55, with a standard deviation of 3.677456.

7.3. Example 3

Our third example has four inputs and four outputs, as shown in Table 5. A matrix of the same size as before was used (i.e., 5×5).

Since this example has 64 outputs, there are 65 objectives for the MGA. A subpopulation size of 10 (i.e., total population size of 650), was sufficient for the MGA. The maximum number of generations in this case (for both the MGA and the NGA) was set to 500.

The MGA found a solution with a fitness value of 82 (i.e., a circuit with seven gates) 15% of the time, and it produced feasible circuits 100% of the time. The average fitness of the 20 runs performed was 80.4, with a standard deviation of 1.142481141. The graphical representation of the best solution found by the MGA is depicted in Figure 5.

On the other hand, the best solution that the NGA could find using the same population size of 650 had a fitness of 80 (i.e., a circuit with nine gates). This solution appeared only twice in the 20 runs performed. In most cases (70% of the runs performed), the best solution found was infeasible. The average fitness of these 20 runs was 66.65, with a standard deviation of 7.638372657.

The comparison of the results produced by the MGA, the NGA, two human designers, and Miller et al. (1997) are shown in Table 6. It should be mentioned that Miller et al. (1997) considered their solution to contain only seven gates because of the way in which they encoded their Boolean functions (the reason is that they encoded NAND gates in their representation). However, since we considered each gate as a separate chromosomic element, we count each of them, including NOTs that are associated with AND and OR gates. Regardless of that fact, it is more important to point out that Miller et al. (1997) found their solution with runs of 3,000,000 fitness function evaluations each, whereas in our case, we performed runs of only 325,000 evaluations each.

7.4. Example 4

Our fourth example has four inputs and three outputs, as shown in Table 7. In this case, the matrix used was of size 6×7, and the chromosomic length was 126 ($r = 6, q = 7, t = 6 \times 7 = 42, l = 3 \times t = 126$). The cardinality $c = \max(r, g) = 6$ The size of the intrinsic search space for this problem is $c^l = 6^{126} \approx 1.1 \times 10^{98}$.

The comparison of the results produced by the MGA, the NGA, and two human designers are shown in Table 8.

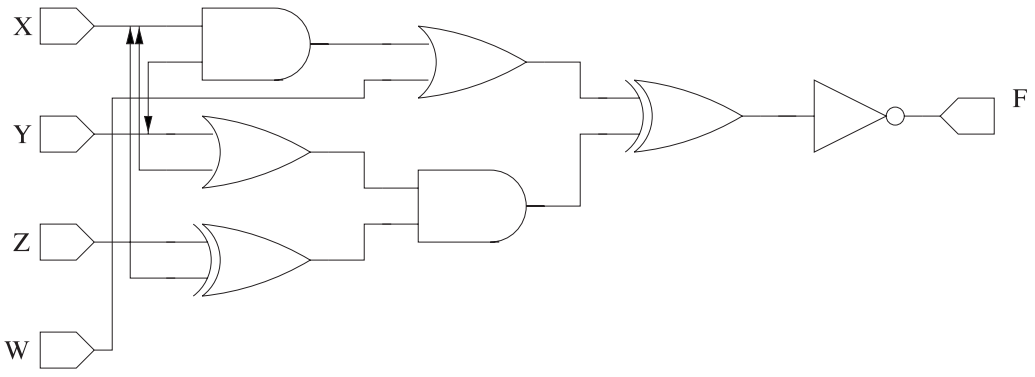

Fig. 4. Circuit produced by our MGA for the second example.

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

Table 4. *Comparison of the best solutions found by the n-cardinality GA (NGA), our multiobjective genetic algorithm (MGA), a human designer using Karnaugh Maps (HD 1), and Sasao for the circuit of the second example*

MGA	NGA	HD 1	Sasao
$F = ((W + XY) \oplus ((X + Y)\ (X \oplus Z)))'$ 7 gates 2 ANDs, 2 ORs, 2 XORs, 1 NOT	$F = (WYX' \oplus ((W + Y) \oplus Z \oplus (X + Y + Z)))'$ 10 gates 2 ANDs, 3 ORs, 3 XORs, 2 NOTs	$F = ((Z'X) \oplus (Y'W')) + ((X'Y)\ (Z \oplus W'))$ 11 gates 4 ANDs, 1 OR, 2 XORs, 4 NOTs	$F = X' \oplus Y'W' \oplus XY'Z' \oplus X'Y'W$ 12 gates 3 XORs, 5 ANDs, 4 NOTs

A population size of 170 was used with both GAs.

Table 5. *Truth table for the two-bit multiplier of the third example*

A_1	A_0	B_1	B_0	C_3	C_2	C_1	C_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	1
0	1	1	0	0	0	1	0
0	1	1	1	0	0	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0	0	1	0
1	0	1	0	0	1	0	0
1	0	1	1	0	1	1	0
1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1
1	1	1	0	0	1	1	0
1	1	1	1	1	0	0	1

Since this example has 48 outputs, there are 49 objectives for the MGA. A subpopulation size of 10 (i.e., total population size of 490), was sufficient for the MGA. The maximum number of generations in this case (for both the MGA and the NGA) was set to 2000.

The MGA found a solution with a fitness value of 81 (i.e., a circuit with nine gates) 15% of the time, and it produced feasible circuits 100% of the time (55% of the time, the MGA found better solution than the best found by the NGA). The average fitness of the 20 runs performed was 78.9, with a standard deviation of 1.020835571. The graphical representation of the best solution found by the MGA is depicted in Figure 6.

On the other hand, the best solution that the NGA could find using the same population size of 490 individuals had a fitness of 78 (i.e., a circuit with 12 gates). This solution appeared only once in the 20 runs performed. In most cases (80% of the runs performed), the best solution found was infeasible. The average fitness of these 20 runs was 52.15, with a standard deviation of 11.92641915.

8. DISCUSSION OF RESULTS

We will start by summarizing the results obtained from our experiments. Table 9 contains of summary of the best results produced by the MGA, the NGA, and the best human designer in each of the circuits analyzed. We can see that the MGA consistently outperformed its competitors, producing the lowest number of gates in each case.

Since one of the main aspects of the approach proposed in this paper is its capability to improve the efficiency of the GA to design combinational circuits, we decided to perform another comparison in which we analyzed the computational cost required by our original NGA and our proposed

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

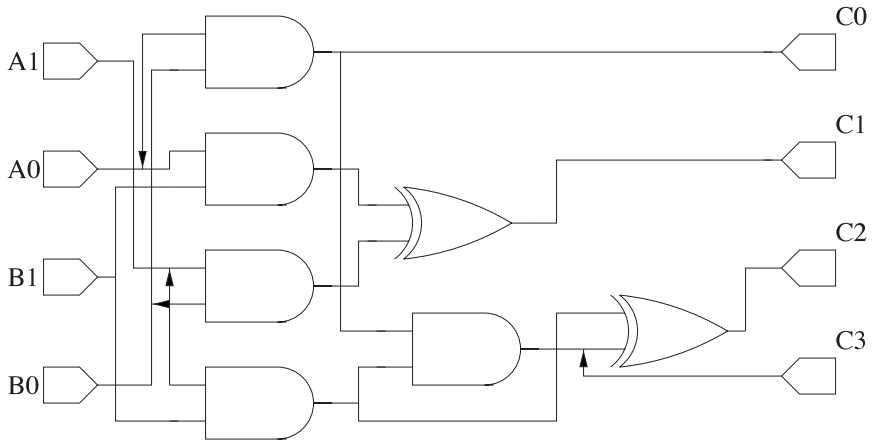

Fig. 5. Circuit produced by our MGA for the third example.

MGA to obtain equivalent results (in terms of optimality). The analysis was conducted on the five examples presented in this paper, and considering only the minimum number of fitness function evaluations required ("minimum" in this case refers to the combination of population size and maximum number of generations that produced the lowest result when multiplied). Since the best results in all cases correspond to the MGA, we established a methodology to try different parameters for the NGA, so that we could reach similar results [our methodology was similar to the one described in previous work (Coello Coello, 2000)].

The comparison of computational costs for the MGA and the NGA (reaching the best results reported in this paper for each of the five examples chose) is presented in Table 10. In all cases, the number of fitness function evaluations indicated correspond to the complete run of the GA (even if, as in most cases, convergence to the best result obtained was achieved before reaching the last generation). It can be clearly appreciated that the MGA outperforms the NGA in most cases. The difference in terms of performance, becomes more significant as we attempt to solve more complex circuits.

We believe that the good performance obtained with this algorithm is mainly due to an emergent behavior obtained from the cooperation of the different subpopulations aiming to satisfy a simple goal. This line of thought is consistent with the recent work by Potter and DeJong (2000), according to which the resolution of complex problems with evolutionary algorithms requires a cooperative effort.

Additionally, the current technique can also be considered a variation of the divide-and-conquer approach to evolvable hardware suggested by Torresen (1998). In this approach, a system is evolved through its smaller components, only that in our case, these smaller components happen to be individual outputs of a circuit. Torresen (1998) also showed that a scheme of this sort could substantially reduce the computational power required to evolve a system. The savings that this sort of population-based approach can produce could be very useful in other design domains such as structural optimization. We are in fact

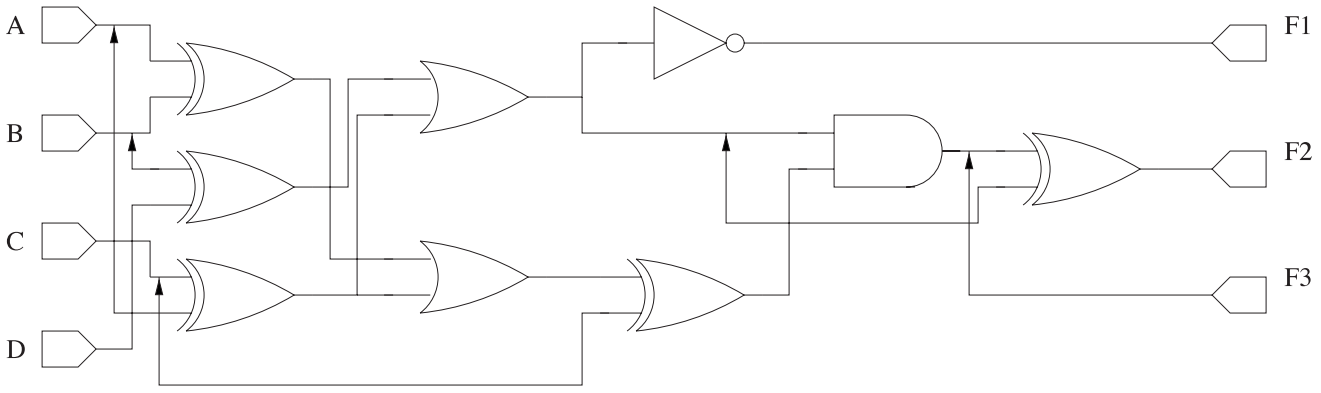

Fig. 6. Circuit produced by our MGA for the fourth example.

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

Table 6. *Comparison of the best solutions found by the n-cardinality GA (NGA), our multiobjective genetic algorithm (MGA), two human designers (HD 1 & HD 2), and Miller et al. (MIL) for the circuit of the third example*

MGA	NGA	HD 1	HD 2	MIL
$C_0 = A_0 B_0$	$C_0 = A_0 B_0$	$C_0 = A_0 B_0$	$C_0 = A_0 B_0$	$C_0 = A_0 B_0$
$C_1 = A_0 B_1 \oplus A_1 B_0$	$C_1 = A_1 A_0 B_0 B_1 \oplus (A_0 B_1 + A_1 B_0)$	$C_1 = A_0 B_1 \oplus A_1 B_0$	$C_1 = (B_1 + B_0)\,(A_1 + A_0)\,((A_1 A_0) \oplus (B_1 B_0))$	$C_1 = A_1 B_0 \oplus A_0 B_1$
$C_2 = A_1 B_1 \oplus (A_0 B_0 A_1 B_1)$	$C_2 = (A_0 B_0 + A_1 B_1) \oplus A_0 B_0$	$C_2 = A_1 B_1\,(A_0 B_0)'$	$C_2 = A_1 B_1\,(A_0 B_0)'$	$C_2 = (A_0 B_0)'\,(A_1 B_1)$
$C_3 = A_0 B_0 A_1 B_1$	$C_3 = A_1 B_1 A_0 B_0$	$C_3 = A_1 A_0 B_1 B_0$	$C_3 = A_1 B_1 A_0 B_0$	$C_3 = (A_1 B_0 \oplus A_0 B_1)'\,(A_1 B_0)$
7 gates	9 gates	8 gates	12 gates	9 gates
5 ANDs, 2 XORs	5 ANDs, 2 ORs, 2 XORs	6 ANDs, 1 XOR, 1 NOT	8 ANDs, 1 XOR, 2 ORs, 1 NOT	6 ANDs, 1 XOR, 2 NOTs

A population size of 650 was used with both the MGA and the NGA

Table 7. *Truth table for the circuit of the fourth example.*

A	B	C	D	F1	F2	F3
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	1	0	0

currently exploring the use of this type of approach in that domain.

Another interesting aspect of this work is the analysis of the design patterns used by the GA. It is important to mention that the GA does not really possess any specific domain information that could help it to bias the search. In fact, it does not even "know" anything about the simplest simplification rules existing (e.g., **NOT** (**NOT** A) = A). Nevertheless, it is able to emulate both simple and complex simplification rules used in Boolean algebra, and even produce others that tend to escape human creativity. Some of the uncommon design patterns used by the GA can be hinted by comparing its solutions against those generated by a human designer. For instance, in Example 4 from the previous section, the Boolean expression of one of the outputs is identical to the expression generated by the MGA. The two others, in contrast, are *more complex* in the case of the MGA. Then, why is the total number of gates of this circuit smaller? The answer is simple: If the solution of the MGA is carefully analyzed, it can be seen that its apparent complexity is due to the fact that it is reusing the same block to produce the three outputs. This is counterintuitive for a human using a visual aid technique such as the Karnaugh maps, but it is an emerging property of the application of natural selection to the circuit design process.

In some of our recent work, we have focused our attention to the discovery of these design patterns [other researchers such as Miller et al. (1999) and Thomson (2000) have done similar work]. To our surprise, besides rediscovering some of the most common simplification rules of Boolean algebra, and others not so simple such as a DeMorgan theorem applied to XOR gates: $(X \oplus Y')' = X \oplus Y = X' \oplus Y'$, we also discovered some more complex simplifications, such as $(A + (A \oplus B)) \oplus (A \oplus B) = AB$, which are not intuitive

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

Table 8. *Comparison of the best solutions found by the n-cardinality GA (NGA), our multiobjective genetic algorithm (MGA), and two human designers (HD 1 and HD 2) for the circuit of the fourth example*

MGA	NGA	HD 1	HD 2
$F1 = ((B \oplus D) + (A \oplus C))'$	$F1 = ((B \oplus D) + (A \oplus C))'$	$F1 = (A \oplus C)' (B \oplus D)'$	$F1 = (A \oplus C)' (B \oplus D)'$
$F3 = (((B \oplus D)) + (A \oplus C))((A \oplus C) + (A \oplus B)) \oplus C))$	$F3 = ((B \oplus D) + (A \oplus C)) ((D + (A \oplus C))' + (A' + C')$	$F3 = BD'(A + C') + AC'$	$F3 = (F1 + F2)'$
$F2 = F3 \oplus ((B \oplus D) + (A \oplus C))$	$F2 = ((B \oplus D) + (A \oplus C)) \oplus ((B \oplus D) + (A \oplus C)) ((D + (A \oplus C))' + (A' + C)')$	F2 = B'D (A' + C) + A'C	$F2 = A'C + (A \oplus C)' (B'D)$
9 gates	12 gates	19 gates	13 gates
3 XORs, 3 ORs, 2 ANDs, 2 NOTs	3 XORs, 4 ORs, 1 AND, 4 NOTs	2 XORs, 4 ORs, 7 ANDs, 6 NOTs	2 XORs, 2 ORs, 4 ANDs, 5 NOTs

A population size of 490 was used with both GAs.

Table 9. *Comparison of the number of gates contained in the best solutions produced by: our multiobjective genetic algorithm (MGA), the N-cardinality genetic algorithm (NGA), and the best human designer (BHD) for each of the examples analyzed in this paper*

Example No.	MGA	NGA	BHD
1	4	4	5
2	7	10	11
3	7	9	8
4	9	12	13

to any human designer. Through the use of case-based reasoning, we have been able to store this "knowledge" generated by the GA for further reuse. The interested reader is referred to Islas Pérez et al. (2001) for further details.

We believe that our approach can be of great help in problems that are decomposable. There are examples in the literature of cooperative search approaches designed for such problems (e.g., Murthy et al., 1999; Parmee & Watson, 1999). Since our approach is based on such a cooperative (emergent) behavior, it is highly likely that it will perform very well (and at a low computational cost) in problems that can be solved using such cooperative techniques.

It is worth mentioning one last issue that may be related to the work presented in this paper. Recently, Knowles et al. (2001) suggested that transforming certain single-objective optimization problems into multiobjective (a process that they call "multi-objectivizing") can remove local optima and therefore, become easier to solve by a heuristic. Their hypothesis was validated with a certain instance of the traveling salesperson problem. In this problem, the application of the multi-objectivizing process previously mentioned allowed for use of a simple hillclimber to solve it.

It is therefore possible that the process described in this paper is another form of multi-objectivizing single-objective circuit design problems. This transformation of the fitness

Table 10. *Comparison of the number of fitness function evaluations required to reach the optimum by each of the two GA-based approaches compared in this paper*

Example No.	MGA	NGA
1	27,000	27,000
2	68,000	500,000
3	325,000	600,000
4	980,000	5,600,000

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press

landscape (produced by the process of multi-objectivizing the problem) may transform a difficult search space into another more amenable for the application of a genetic algorithm. This allows us to find not only very good results but also to find them in a relatively reduced amount of time.

9. CONCLUSIONS AND FUTURE WORK

We have proposed a multiobjective optimization technique to design combinational logic circuits. The proposed approach uses a population-based technique to split the search task among several (small) subpopulations. The approach compared well with respect to two human designers, and a previous GA developed by us which uses and *n*-cardinality alphabet and a two-stage fitness function. Our approach, called MGA, consistently found better solutions than the human designers, and was able to find the same or even better solutions than our previous GA (called NGA), using a lower number of fitness function evaluations.

The proposed approach seems very suitable for parallelization, and that will probably be a path of research that we will explore in the near future. Also, we are interested in coupling this approach with another system based on genetic programming that is currently under development. We aim to benefit from a more powerful chromosomic representation while keeping an efficient selection mechanism.

ACKNOWLEDGMENTS

The first author acknowledges partial support from CINVESTAV through project JIRA'2001/08, and from the Mexican Consejo Nacional de Ciencia y Tecnología through CONACyT project No. 34201-A. The second author acknowledges support for this work in part by DoD EPSCoR and the Board of Regents of the State of Louisiana under grant F49620-98-1-0351, and from the NASA Jet Propulsion Laboratory, under contract No. 1230282.

REFERENCES

Brayton, R.K., Hachtel, G.D., McMullen, C.T., & Sangiovanni-Vincentelli, A.L. (1984). *Logic Minimization Algorithms for VLSI Synthesis*, Dordrecht, The Netherlands: Kluwer Academic Publishers.

Brayton, R.K., Rudell, R., Sangiovanni-Vincentelli, A., & Wang, A.R. (1987). MIS: A multiple-level logic optimization system, *IEEE Transactions on Computer-Aided Design CAD-6 (6)*, 1062–1081.

Brzozowski, J.A., & Yoeli, M. (1976). *Digital Networks*. Englewood Cliffs, NJ: Prentice Hall.

Camponogara, E., & Talukdar, S.N. (1997). A genetic algorithm for constrained and multiobjective optimization. In *Third Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, University of Vaasa, Vaasa, Finland, 49–62.

Chankong, V., & Haimes, Y.Y. (1983). *Multiobjective Decision Making: Theory and Methodology*, Amsterdam: Systems Science and Engineering, North-Holland.

Coello Coello, C.A. (1996). *An empirical study of evolutionary techniques for multiobjective optimization in engineering design*, Ph.D. Thesis, New Orleans, LA: Tulane University.

Coello Coello, C.A. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems. An International Journal 1(3)*, 269–308.

Coello Coello, C.A. (2000). Treating constraints as objectives for single-objective evolutionary optimization. *Engineering Optimization 32(3)*, 275–308.

Coello Coello, C.A., Christiansen, A.D., & Hernández Aguirre, A. (1997). Automated design of combinational logic circuits using genetic algorithms. In *Proc. Int. Conf. Artificial Neural Nets and Genetic Algorithms*. University of East Anglia, England, pp. 335–338. Austria: Springer-Verlag.

Coello Coello, C.A., Christiansen, A.D., & Hernández Aguirre, A. (2000). Use of evolutionary techniques to automate the design of combinational circuits. *International Journal of Smart Engineering System Design 2(4)*, 299–314.

Deb, K., & Goldberg, D.E. (1989). An investigation of niche and species formation in genetic function optimization. In *Proc. Third Int. Conf. Genetic Algorithms*. George Mason University, pp. 42–50. San Mateo, CA: Morgan Kaufmann Publishers.

Fonseca, C.M., & Fleming, P.J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proc. Fifth Int. Conf. Genetic Algorithms*. University of Illinois at Urbana-Champaign, pp. 416–423. San Mateo, CA: Morgan Kauffman Publishers.

Fonseca, C.M., & Fleming, P.J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation 3(1)*, 1–16.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading, MA: Addison-Wesley Publishing.

Harris, S.P., & Ifeachor, E.C. (1996). Nonlinear FIR filter design by genetic algorithm. In *1st Online Conference on Soft Computing*.

Horn, J. (1997). Multicriterion decision making. In *Handbook of Evolutionary Computation*, (Bäck, T., Fogel, D., & Michalewicz, Z., Eds.) Vol. 1, pp. F1.9:1 – F1.9:15. IOP Publishing Ltd. and Oxford University Press, UK.

Islas Pérez, E., Coello Coello, C.A., & Hernández Aguirre, A. (2001). Extraction of design patterns from evolutionary algorithms using case-based reasoning. In *Evolvable Systems: From Biology to Hardware* (Yong Liu, Kiyoshi Tanaka, Masaya Iwata, Tetsuya Higushi & Montoshi Yasunagi). Tokyo, Japan: Springer-Verlag.

Jiménez, F., & Verdegay, J.L. (1999). Evolutionary techniques for constrained optimization problems. In *Seventh European Congress on Intelligent Techniques and Soft Computing (EUFIT'99)*, Aachen, Germany: Springer-Verlag.

Kalganova, T., & Miller, J. (1999). Evolving more efficient digital circuits by allowing circuit layout and multi-objective fitness. In *Proc. First NASA/DoD Workshop on Evolvable Hardware*, pp. 54–63. Los Alamitos, CA: IEEE Computer Society Press.

Kalganova, T., Miller, J., & Fogarty, T. (1998). Some aspects of an evolvable hardware for multiple-valued combinational circuit design. In *Proc. Second Int. Conf. Evolvable Systems (ICES'98)*, pp. 78–89. Lausanne, Switzerland: Springer-Verlag.

Karnaugh, M. (1953). A map method for synthesis of combinational logic circuits. *Transactions of the AIEE, Communications and Electronics 72 (I)*, 593–599.

Knowles, J.D., Watson, R.A., & Corne, D.W. (2001). Reducing local optima in single-objective problems by multi-objectivization. In *First Int. Conf. Evolutionary Multi-Criterion Optimization*, pp. 268–282. Lecture Notes in Computer Science No. 1993, Springer-Verlag.

Louis, S.J. (1993). *Genetic algorithms as a computational tool for design*. Ph.D. Thesis, Bloomington, IN: Department of Computer Science, Indiana University.

Louis, S.J., & Rawlins, G.J. (1991). Using Genetic Algorithms to Design Structures, Technical Report 326, Bloomington, IN: Computer Science Department, Indiana University.

Louis, S.J., & Rawlins, G.J.E. (1993). Pareto optimality, GA-easiness and deception. In *Proc. Fifth Int. Conf. Genetic Algorithms*, pp. 118–123. San Mateo, California: Morgan Kaufmann Publishers.

McCluskey, E.J. (1956). Minimization of Boolean functions. *Bell Systems Technical Journal 35(5)*, 1417–1444.

Miller, J.F., Job, D., & Vassilev, V.K. (2000). Principles in the evolutionary design of digital circuits—Part I. *Genetic Programming and Evolvable Machines 1(1/2)*, 7–35.

Miller, J.F., Thomson, P., & Fogarty, T. (1997). Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study. In *Genetic Algorithms and Evolution Strategy in Engineering and Com-*

puter Science (Quagliarella, D., Périaux, J., Poloni, C., & Winter, G., Eds.) pp. 105–131. Chichester, England: Wiley.
Miller, J., Kalganova, T., Lipnitskaya, N., & Job, D. (1999). The genetic algorithm as a discovery engine: Strange circuits and new principles. In *Proc. AISB Symposium on Creative Evolutionary Systems (CES'99)*, pp. 65–74. Edinburgh, UK.
Murthy, S., Akkiraju, R., Goodwin, R., Keskinocak, P., Rachlin, J., Wu, F., Kumaran, S., & Daigle, R. (1999). Enhancing the decision-making process for paper mill schedulers. *Tappi Journal 82(7)*, 42–47.
Osyczka, A. (1985). Multicriteria optimization for engineering design. In *Design Optimization* (Gero, J.S., Ed.), pp. 193–227. New York: Academic Press.
Pareto, V. (1896). *Cours D'Economie Politique*, Vol. I and II, Lausanne: F. Rouge.
Parmee, I.C., & Purchase, G. (1994). The development of a directed genetic search technique for heavily constrained design spaces. In *Adaptive Computing in Engineering Design and Control-'94*, University of Plymouth, Plymouth, UK, pp. 97–102.
Parmee, I.C., & Watson, A.H. (1999). Preliminary airframe design using co-evolutionary multiobjective genetic algorithms. In *Proc. Genetic and Evolutionary Computation Conference (GECCO'99)*, Vol. 2, pp. 1657–1665. San Francisco, CA: Morgan Kaufmann.
Potter, M., & DeJong, K. (2000). 'Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation 8(1)*, 1–29.
Quine, W.V. (1955). A way to simplify truth functions. *American Mathematical Monthly 62(9)*, 627–631.
Sasao, T., ed. (1993). *Logic Synthesis and Optimization*. Dordrecht, The Netherlands: Kluwer Academic Publishers.
Schaffer, J.D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Genetic Algorithms and their Applications: Proc. First Int. Conf. Genetic Algorithms*, pp. 93–100. Hillsdale, New Jersey: Lawrence Erlbaum.
Surry, P.D., & Radcliffe, N.J. (1997). The COMOGA method: Constrained optimisation by multiobjective genetic algorithms. *Control and Cybernetics 26(3)*, 391–412.
Surry, P.D., Radcliffe, N.J., & Boyd, I.D. (1995). A multi-objective approach to constrained optimisation of gas supply networks: The COMOGA method. In *Evolutionary computing. AISB workshop. Selected papers*, pp. 166–180. Sheffield, U.K.: Springer-Verlag.
Thompson, A., Layzell, P., & Zebulum, R.S. (1999). Explorations in design space: Unconventional electronics design through artificial evolution. *IEEE Transactions on Evolutionary Computation 3(3)*, 167–196.
Thomson, P. (2000). Circuit evolution and visualisation. In *Evolvable Systems: From Biology to Hardware*, (Miller, J., Thompson, A., Thomson, P., & Fogarty, T.C., Eds.), pp. 229–240. Edinburgh, Scotland: Springer-Verlag.
Torresen, J. (1998). A divide-and-conquer approach to evolvable hardware. In *Proc. Second Int. Conf. Evolvable Systems (ICES'98)*, pp. 57–65. Lausanne, Switzerland: Springer-Verlag.
Veitch, E.W. (1952). A chart method for simplifying Boolean functions. *Proceedings of the ACM*, 127–133.
Wilson, P.B., & Macleod, M.D. (1993). Low implementation cost IIR digital filter design using genetic algorithms. In *IEE/IEEE Workshop on Natural Algorithms in Signal Processing*, Chelmsford, U.K., pp. 4/1–4/8.
Zebulum, R.S., Pacheco, M.A., & Vellasco, M. (1998). A multi-objective optimisation methodology applied to the synthesis of low-power operational amplifiers. In *Proc. XIII Int. Conf. Microelectronics and Packaging*, Curitiba, Brazil, Vol. 1, pp. 264–271.

Carlos A. Coello Coello received a Ph.D. in Computer Science from Tulane University in 1996. He is currently an associate professor at CINVESTAV-IPN, in México. He has published over 50 research papers in peer-reviewed international journals and conferences (all of them on evolutionary algorithms) and has coauthored the book *Evolutionary Algorithms for Solving Multi-Objective Problems*, which is about to be published by Kluwer Academic Publishers. His current research interests are evolutionary multiobjective optimization, constraint-handling techniques for evolutionary algorithms, and evolvable hardware.

Arturo Hernández Aguirre received a Ph.D. in Computer Science from Tulane University in 1999. He worked for several years at the Arturo Rosenblueth Foundation as a Software Consultant in Mexico City and is currently a Visiting Professor at Tulane University. His research interests include computational learning, neural networks, and evolutionary computation.

https://doi.org/10.1017/S0890060401020054 Published online by Cambridge University Press