Evolutionary Multiobjective Optimization using a Cultural Algorithm

Carlos A. Coello Coello and Ricardo Landa Becerra

Abstract— **In this paper, we present the first proposal to use a cultural algorithm to solve multiobjective optimization problems. Our proposal uses evolutionary programming, Pareto ranking and elitism (i.e., an external population). The approach proposed is validated using several examples taken from the specialized literature. Our results are compared with respect to the NSGA-II, which is an algorithm representative of the state-of-the-art in evolutionary multiobjective optimization. The performance of our approach indicates that cultural algorithms are a viable alternative for multiobjective optimization.**

Keywords— **cultural algorithms, multiobjective optimization, evolutionary programming.**

I. Introduction

In recent years, evolutionary algorithms have been widely used for multiobjective optimization, obtaining very promising results [3], [2]. An important advantage of evolutionary algorithms over traditional techniques used for multiobjective optimization is that the former operate over a set of solutions at a time. Therefore, evolutionary algorithms can produce several elements of the Pareto optimal set in a single run rather than generating one at a time as traditional mathematical programming approaches.

Cultural algorithms [18] are a technique that incorporates domain knowledge obtained during the evolutionary process as to make the search process more efficient. Cultural algorithms have been successfully applied to several types of optimization problems (e.g., in constrained single-objective optimization problems [20], [12], [13], [4]). However, until now, nobody had proposed a cultural algorithm for multiobjective optimization adopting Pareto ranking and elitism. Ours is then the first proposal in this direction.

II. Basic Concepts

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

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

the p *equality constraints*

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

and will optimize the vector function

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

CINVESTAV-IPN, Evolutionary Computation Group
Depto. de Ing. Elect./Sección de Computación,
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D. F. 07300,
ccoello@cs.cinvestav.mx,
rlanda@computacion.cs.cinvestav.mx

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

Definition 2 (Pareto Optimality:): *A point* $\vec{x}^* \in \Omega$ *is* **Pareto optimal** *if for every* $\vec{x} \in \Omega$ *and* $I = \{1, 2, \ldots, k\}$ *either,*

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

or, there is at least one $i \in I$ *such that*

$$f_i(\vec{x}) > f_i(\vec{x}^*) \tag{5}$$

□

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

Definition 3 (Pareto Dominance): *A vector* $\vec{u} = (u_1, \ldots, u_k)$ *is said to dominate* $\vec{v} = (v_1, \ldots, v_k)$ *(denoted by* $\vec{u} \preceq \vec{v}$*) if and only if u is partially less than v, i.e.,* $\forall i \in \{1, \ldots, k\},\ u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$. □

Definition 4 (Pareto Optimal Set): *For a given MOP* $\vec{f}(x)$*, the Pareto optimal set (*$\mathcal{P}^*$*) is defined as:*

$$\mathcal{P}^* := \{x \in \Omega \mid \neg\exists\ x' \in \Omega \ \ \vec{f}(x') \preceq \vec{f}(x)\}. \tag{6}$$

□

Definition 5 (Pareto Front:): *For a given MOP* $\vec{f}(x)$ *and Pareto optimal set* $\mathcal{P}^*$*, the Pareto front (*$\mathcal{PF}^*$*) is defined as:*

$$\mathcal{PF}^* := \{\vec{u} = \vec{f} = (f_1(x), \ldots, f_k(x)) \mid x \in \mathcal{P}^*\}. \tag{7}$$

□

III. Notions of Cultural Algorithms

Cultural algorithms were developed by Robert G. Reynolds as a complement to the metaphor adopted by evolutionary algorithms, which was mainly focused on genetic concepts and on the natural selection mechanism [18]. Cultural algorithms are based on some theories proposed in sociology and archaelogy to model cultural evolution. Such theories indicate that cultural evolution can be seen as an inheritance process that occurs at two levels: the micro-evolutionary level, and the macro-evolutionary level.

At the micro-evolutionary level, individuals are described in terms of "behavioral traits" (which can be socially acceptable or unacceptable). These behavioral traits are passed from generation to generation using several socially motivated operators. At the macro-evolutionary level, individuals are able to generate

"mappa" [17], or generalized descriptions of their experiences. Individual mappa can be merged and modified to form "group mappa" using a set of generic or problem specific operators. Both levels share a communication link.

The micro-evolutionary level refers to the knowledge acquired by individuals through generations which, once encoded and stored is used to guide the behavior of the individuals that belong to a certain population [17], [7]. Reynolds attempts to capture this double inheritance phenomenon in the cultural algorithm [18]. The goal is to increase the learning or convergence rates of the algorithm as to provide a better response to a large number of problems [10].

Cultural algorithms operate on two spaces. First, they operate on the population space as any other evolutionary computation technique in which a set of individuals (called population) is adopted. Each individual has a set of features independent from each other which allows us to determine its fitness. Through time, such individuals can be replaced by some of its descendants, obtained after applying a set of operators to the population.

The second space is the belief space, where the knowledge acquired by the individuals along the evolutionary process is stored. The information contained in this space must be accesible to any individual, so that it can use it to modify its behavior.

To unify both spaces, a communication protocol is established such that it dictates rules regarding the type of information to be exchanged between these two spaces. For example, to update the belief space, the individual experiences of a select set of individuals are incorporated. This select group of individuals is obtained with the function *acceptance* which is applied to the entire population. On the other hand, the operators that modify the population (i.e., recombination and mutation) and the selection operator are modified by the function *influence*. This function acts in such a way that the individuals resulting from the application of the operators tend to approach the desirable behavior while staying away from undesirable behaviors. Such desirable and undesirable behaviors are defined in terms of the information stored in the belief space. These two functions are used to establish the communication between the two spaces (i.e., population and belief). The interactions between these two spaces can be appreciated in Figure 1 [19].

Following this model, we have designed a cultural algorithm for multiobjective optimization, trying to take advantage of its main features.

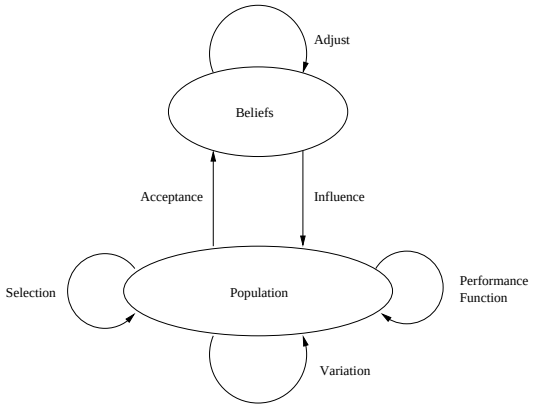

Fig. 1. Spaces of a cultural algorithm.

Algorithm 1 Basic structure of the cultural algorithm for multiobjective optimization.

Generate the initial population of size p
Evaluate the initial population
Initialize the belief space
Repeat
  Apply mutation
    to generate p offspring (we now
    have $2p$ individuals in the population)
  Evaluate each offspring
  Perform binary tournaments, randomly
    choosing c contenders for each
    individual. The decisions taken
    in the tournaments will be
    influenced by the information
    stored in the belief space.
  Select the p individuals with
    the largest number of victories
    in the tournaments, to produce the
    population of the following generation
  Add the new nondominated individuals
    to the external memory (file)
  Update belief space with the individuals
    added to the external memory
While stopping condition is not met

IV. Description of our Approach

We propose the use of a cultural algorithm combined with evolutionary programming (*Cultural Algorithm with Evolutionary Programming* [1], or CAEP). The pseudocode of our approach is shown in Algorithm 1. Here, we can clearly see the similarities of our approach with traditional evolutionary programming [8], and we also include the steps where the belief space is incorporated.

The problems that we are interested in solving have n decision variables and k objective functions. The population consists of a set of individuals, each of which represents a possible solution to the problem. Each individual contains the n decision variables of the problem to be solved. The population is initialized with p individuals randomly generated using a uniform distribution within the allowable range for each decision variable. The external memory indicated in the Algorithm 1 is an external file (or secondary population) where the nondominated individuals found along the evolutionary are stored. The final contents of this file is the set of solutions presented by the algorithm to the user. This external memory has a maximum size q that corresponds to the number of nondominated solutions that we aim to obtain. Next, we detail the structure of the belief space adopted, together with the remaining steps of the algorithm.

A. Structure of the Belief Space

The belief space consists of two parts: the phenotypic normative part and a grid which is used to emphasize the

l_{f1}	u_{f1}	l_{f2}	u_{f2}	$\cdots$	l_{fk}	u_{fk}

Fig. 2. Phenotypic normative part

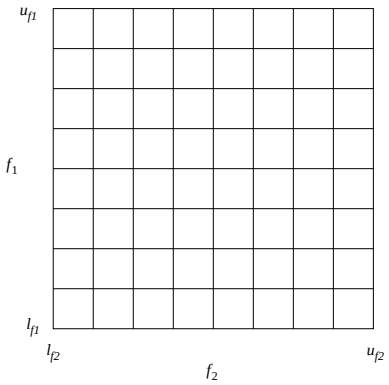

Fig. 3. Grid in the belief space for a problem with two objective functions. In this case, $s_1 = s_2 = 8$ (eight sub-intervals in each dimension).

generation of nondominated solutions that are uniformly distributed along the Pareto front. This grid is a variation of the adaptive grid proposed by Knowles and Corne [14].

The phenotypic normative part contains only the lower and upper bounds, l_{fi} and u_{fi}, of the intervals for each objective function ($i = 1, \ldots, k$) within which the grid will be built. This grid will be used to place each nondominated solution in some sort of coordinate system where the values of the objective functions are used to place each solution (see Figure 2).

Once we have these intervals, we only need to know the number of identical sub-divisions to apply to each of them (s_i, with $i = 1, \ldots, k$), so that we can build the grid in phenotypic space. An example of the grid adopted is shown in Figure 3.

As a result, we will have $s_1 s_2 \ldots s_k$ cells, which will all have the same dimensions. The values s_i are input parameters required by the algorithm. For each cell, we store the count of the number of nondominated individuals stored in the secondary memory that are contained within. This is useful to obtain an appropriate distribution of the nondominated solutions, avoiding that they all cluster together around a certain portion of the Pareto front.

B. Initialization of the Belief Space

In order to initialize the belief space is necessary to have an initial population, because we will use the nondominated individuals from that population (it can be proved that in any population of size greater than zero there is at least one nondominated individual [23]).

B.1 Initialization of the Phenotypic Normative Part

The initialization of the phenotypic normative part of the belief space consists of finding the extrema of each objective function for the nondominated individuals of the initial population. These extrema are stored in l_{fi} and u_{fi}, so that we can place the grid in the region where the nondominated individuals known so far are located.

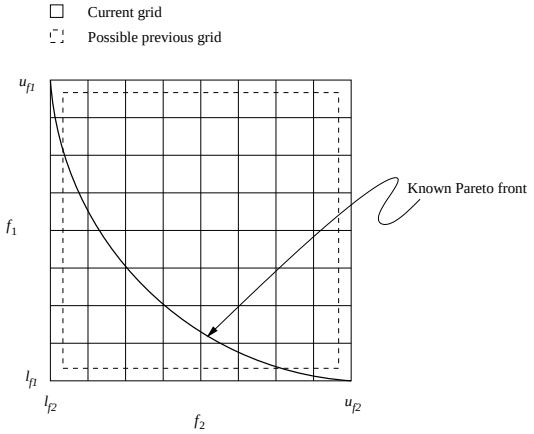

Fig. 4. The update of the phenotypic normative part of the belief space aims to contain the entire Pareto front known so far within the grid.

B.2 Initialization of the Grid

The grid is created taking as intervals the values stored in the phenotypic normative part, and we use the input paramters s_i to divide such grid. The counters of the nondominated individuals contained within each cell are initializaed to zero.

C. Updating Belief Space

The belief space grid is updated at each generation, whereas the phenotypic normative part is updated at each $g_{normative}$ generations, where $g_{normative}$ is a parameter defined by the user.

C.1 Updating the Grid

In order to update the grid, we simply increment the counters of the nondominated individuals by the number of individuals added to the external memory during the current generation. The update of the grid is very simple, and that is the reason why we do it at every generation. For the update of this part of the belief space, the acceptance function uses the population of the external memory and only chooses the new individuals within that population.

C.2 Updating the Phenotypic Normative Part

The update of the phenotypic normative part is not done at each generation, because it involves the rebuilding of the grid and, therefore, it is a process that could seriously affect the computational efficiency of the algorithm. For this update, we also use the population contained in the secondary memory.

Again, as in the initialization of this phenotypic normative part, we need to identify the extrema for each objective function contained in the external memory. These values are stored in l_{fi} and u_{fi} for $i = 1, \ldots, k$. With the new values, we build the grid, which covers the entire Pareto front known so far (see Figure 4) and that probably has limits beyond the boundaries of the previous grid.

After initializing the grid, all the counters are set to zero and it is then necessary to add all the individuals from the external memory to the counter of their corresponding cell, so that the belief space is ready again to be used.

D. Mutation

The information stored in the belief space belongs to the phenotypic space of our problem. Therefore, it is difficult to use it to self-adapt the mutation operator (since mutation operates on genotypic space). Given this inconvenience, we leave the mutation parameters as inputs for the algorithm, which must be provided by the user (i.e., no self-adaptation mechanism is adopted).

The Gaussian mutation operator adopted in our algorithm is based on the following expression:

$$x_i' = x_i + N(0, \sigma)$$

where: x_i is the i-th variable of individual x, x_i' is the i-th variable of the new individual x' obtained after applying mutation, and $N(\mu, \sigma)$ is a random variable with a normal distribution that has a mean μ and a standard deviation σ. In our case, μ will always be zero, and σ is a parameter provided by the user. Mutation is then applied for $i = 1, \ldots, n$, and it operates on the main population. Therefore, at the end of the process, we will have a population of size $2p$.

E. Tournament Selection

Tournament selection is performed considering a main population of size $2p$. Each individual is confronted against other c individuals which are randomly chosen from the main population. Tournament rules are the following:

1. If an individual dominates his contender, then the dominating individual wins.
2. If the individuals confronted are non-comparable, or if the values of their objective functions are the same, then:
   (a) If both lie within the grid of the belief space, then the individual located in the less populated cell (according to the cell counters) wins.
   (b) If one of the individuals lies outside the grid, then it automatically wins.

The first rule should be straightforward, since we are simply giving preference to nondominated individuals so that we can approach the Pareto front. In the second point, we appreciate the influence of the belief space in the decisions made during the tournament. The first subpart of the second point aims to distribute solutions in a uniform way among the cells, so that we can produce a better distributed Pareto front. Finally, if we have the case indicated in the second subpart of the second point, it means that we have found a solution that is beyond the Pareto front known so far. Since this solution will generate a new segment of the Pareto front, it is important to keep it. Figure 5 shows the case in which an individual lies outside the grid.

Once the tournaments finish, we select the individuals with the largest number of victories to be part of the following generation. We can see that these tournaments are similar to those of the NPGA [11], where each tournament competitor is compared with respect to random sample of the population. However, in the NPGA we apply Pareto ranking with respect to the sample randomly selected, whereas in our algorithm we compare individually, counting the number of victories (as it is normally done in evolutionary programming).

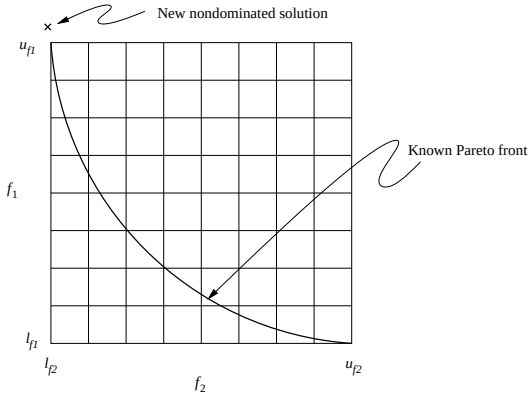

Fig. 5. Tournament in the case in which an individual lies outside the grid. The point found is a new extrema of the Pareto front known so far and it is therefore important to keep it.

F. Adding Individuals to the External Memory

The external memory must contain only nondominated individuals, without repetition. To add such individuals to the external memory, we use the following rules:

1. If the individual that we want to add is dominated by another individual contained in the external memory, then the new individual is not added.
2. If the individual that we want to add dominates some individual contained in the external memory, then it is introduced in its place, but it is still compared with respect to everybody else. If the same individual recently added dominates another one, then the dominated individual is removed from the external memory.
3. If the individual that we want to add is not dominated and it does not dominate any other individual in the external memory, and if the number of individuals in the external memory is less than the maximum allowable value q, then we add the new individual at the end.
4. If the individual that we want to add is not dominated and it does not dominate any other individual in the external memory, but the external memory is already full, then we locate an individual whose cell contains more individuals that the cell to which the new individual would belong (in case it is added). Then, we replace the old individual by the new one. This will motivate a better distribution of nondominated solutions along the Pareto front.

V. Comparison of Results

In order to validate our proposed approach, we used a set of test functions which have commonly been adopted as a benchmark in the specialized literature [3]. The first test function, **MOP1**, was proposed by Schaffer [21]; our second test function, **MOP2**, was proposed by Fonseca [9]; our third test function **MOP3** was proposed by Poloni [16]. **MOP4** was proposed by Kursawe [15]; Finally, **MOP6** was designed following Deb's methodology [5]. All these functions have variable

degrees of difficulty. There are problems with convex and concave Pareto fronts, continuous and disconnected, and problems with two and with three objective functions. The mathematical description of these problems is provided next:

MOP1. Minimize:

$$\vec{f}(x) = (f_1(x), f_2(x))$$

where:

$$\begin{aligned} f_1(x) &= x^2 \\ f_2(x) &= (x-2)^2 \end{aligned}$$

and $-10^5 \leq x \leq 10^5$.

MOP2. Minimize:

$$\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}))$$

where:

$$\begin{aligned} f_1(\vec{x}) &= 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right) \\ f_2(\vec{x}) &= 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right) \end{aligned}$$

and $-4 \leq x_i \leq 4$ $(i = 1, 2, 3)$.

MOP3. Maximize:

$$\vec{f}(\vec{x}) = (f_1(x,y), f_2(x,y))$$

where:

$$\begin{aligned} f_1(x,y) &= -[1 + (A_1 - B_1)^2 + (A_2 - B_2)^2] \\ f_2(x,y) &= -[(x+3)^2 + (y+1)^2] \end{aligned}$$

with:

$$\begin{aligned} A_1 &= 0.5\sin 1 - 2\cos 1 + \sin 2 - 1.5\cos 2 \\ A_2 &= 1.5\sin 1 - \cos 1 + 2\sin 2 - 0.5\cos 2 \\ B_1 &= 0.5\sin x - 2\cos x + \sin y - 1.5\cos y \\ B_2 &= 1.5\sin x - \cos x + 2\sin y - 0.5\cos y \end{aligned}$$

and $-\pi \leq x, y \leq \pi$.

MOP4. Minimize:

$$\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}))$$

where:

$$\begin{aligned} f_1(\vec{x}) &= \sum_{i=1}^{n-1}\left(-10e^{(-0.2)\sqrt{x_i^2 + x_{i+1}^2}}\right) \\ f_2(\vec{x}) &= \sum_{i=1}^{n}(|x_i|^a + 5\sin(x_i^b)) \end{aligned}$$

with:

$$\begin{aligned} a &= 0.8 \\ b &= 3 \end{aligned}$$

and $-5 \leq x_i \leq 5$ $(i = 1, 2, 3)$.

MOP6. Minimize:

$$\vec{f}(x,y) = (f_1(x,y), f_2(x,y))$$

where:

$$\begin{aligned} f_1(x,y) &= x \\ f_2(x,y) &= (1+10y)\left[1 - \left(\frac{x}{1+10y}\right)^{\alpha} - \right. \\ &\quad \left. \frac{x}{1+10y}\sin(2\pi q x)\right] \end{aligned}$$

with: $q = 4$, $\alpha = 2$, y $0 \leq x, y \leq 1$.

The parameters adopted by our CAEP are the following: $p = 6$, $q = 100$, $G_{max} = 35,000$, $g_{normative} = 20$, $s_i = 10$ $(i = 1, \ldots, k)$, $c = \frac{p}{2} = 5$, $\sigma = 1$. The total number of objective function evaluations during one execution of our algorithm is computed using $p(G_{max} + 1)$. For the examples used in this paper, we performed 210,006 evaluations per run.

We compared CAEP with respect to the NSGA-II [6], which used the following parameters: population size = 100, maximum number of generations = 2100, crossover rate = 0.9, mutation rate = $\frac{1}{n}$, SBX parameter = 10 y mutation parameter = 100. These values have been suggested by the author of the algorithm, except for the maximum number of generations, which was chosen in such a way that the total number of evaluationes performed by the NSGA-II was approximately the same as CAEP. The NSGA-II performed 210,100 objective function evaluations per run.

Table I shows the statistics of each of the three metrics adopted to allow a quantitative comparison: error ratio (ER) [23], generational distance (GD) [23] and spacing (SP) [22] for ten independent runs of each algorithm.

Figure 6 shows the average behavior of each of the two algorithms compared (CAEP y NSGA-II) for the first test function. Based on the results shown in Table I we can see that, in this case, CAEP obtained better results for the three metrics adopted.

Figure 7 shows the average behavior of each of the two algorithms compared in **MOP2**. In this case, our approach had a high error ratio (approximately 50%). With respect to generational distance, both algorithms had low values. With respect to spacing, the NSGA-II had values slightly higher than those of CAEP.

Figure 8 shows the average behavior of each of the two algorithms compared in **MOP3**. In this case, the NSGA-II had a lower generational distance than our approach, but a higher error ratio. This problem has a disconnected Pareto front and, therefore, spacing does not correctly reflect the performance of an algorithm. With respect to spacing, the NSGA-II had better values than CAEP.

Figure 9 shows the average behavior of the two algorithms in **MOP4**. In this case, all the metrics indicate better values for the NSGA-II, but our algorithm covers a region of the Pareto front that the NSGA-II can never cover, as Figure 9 indicates.

Figure 10 shows the average behavior of the two algorithms in **MOP6**. In this case, both algorithms have

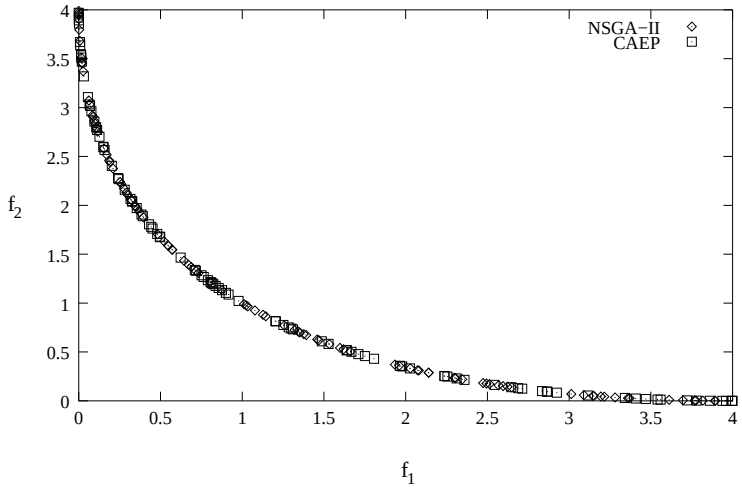

Fig. 6. Comparison of results between our algorithm (CAEP) and the NSGA-II for **MOP1**.

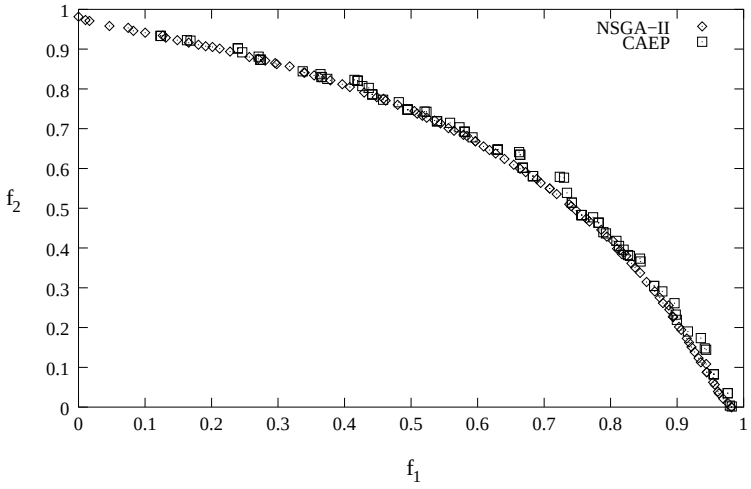

Fig. 7. Comparison of results between our algorithm (CAEP) and the NSGA-II for **MOP 2**.

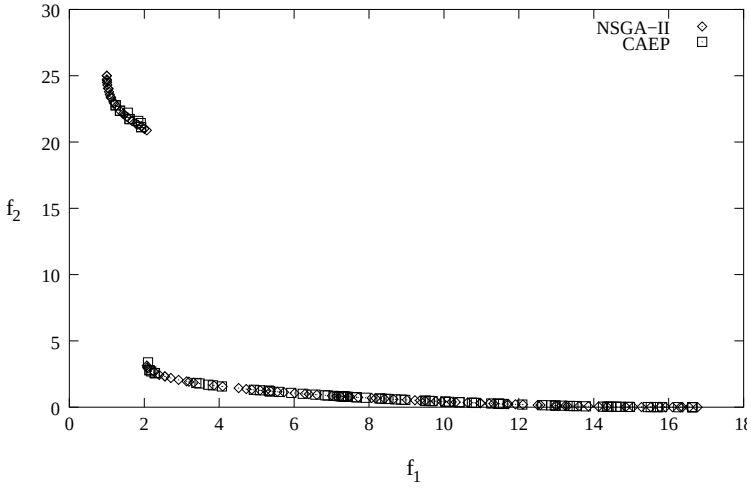

Fig. 8. Comparison of results between our algorithm (CAEP) and the NSGA-II for **MOP3**.

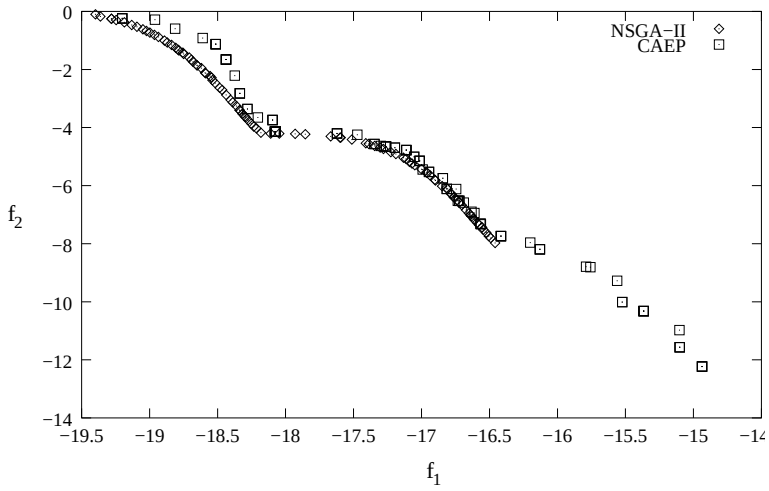

Fig. 9. Comparison of results between our algorithm (CAEP) and the NSGA-II for **MOP4**.

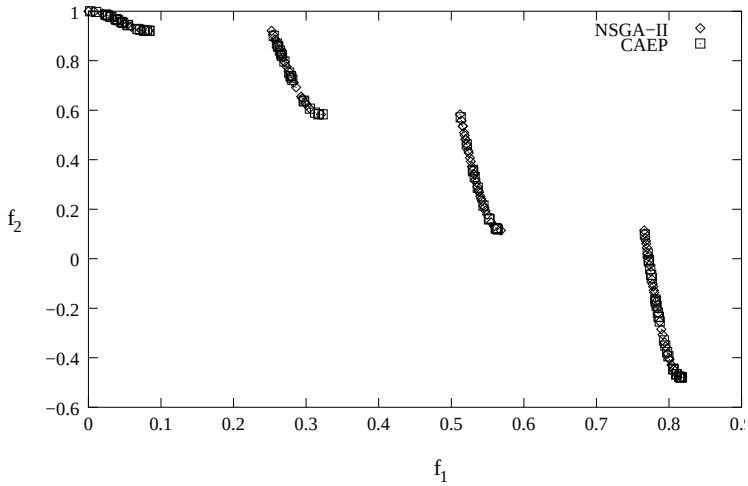

Fig. 10. Comparison of results between our algorithm (CAEP) and the NSGA-II for **MOP6**.

very low error ratio values (the NSGA-II almost always produces 0%), but our approach has a lower generational distance than the NSGA-II. Regarding spacing, we again have values slightly better for the NSGA-II, but this is caused by the disconnected Pareto front of this problem.

In summary, our results indicate that CAEP has a competitive behavior with respect to the NSGA-II, which is representative of the state-of-the-art in evolutionary multiobjective optimization.

VI. Conclusions and Future Work

In this paper we have introduced the first proposal to use cultural algorithms for multiobjective optimization incorporating Pareto-based selection. We have shown how adding a belief space to an evolutionary programming algorithm can be beneficial and represents a viable alternative to solve multiobjective problems.

The small comparative study presented in this paper validates the viability of our proposal. However, we obviously still need to do a more in-depth study in which other evolutionary multiobjective optimization techniques and more test functions are involved. Additionally, there are still a few limitations of CAEP that require a more careful analysis. For example, CAEP tends to lose diversity very quickly in some cases. In order to deal with this problem, we are currently designing an alternative scheme to maintain diversity which emphasizes efficiency (computationally speaking). We are also interested in improving the mechanism adopted to generate a uniform distribution of nondominated solutions along the Pareto front. This is motivated by the fact that our current mechanism still has some flaws and sometimes the distribution produced by our algorithm is not as good as expected.

Acknowledgments

The first author acknowledges support from the mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) through project number 34201-A. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at CINVESTAV-IPN's Electrical Engineering Department (Computer Science Section).

References

[1] Chan-Jin Chung and Robert G. Reynolds. CAEP: An Evolution-based Tool for Real-Valued Function Optimization using Cultural Algorithms. *Journal on Artificial Intelligence Tools*, 7(3):239–292, 1998.

[2] Carlos A. Coello Coello. An Updated Survey of GA-Based Multiobjective Optimization Techniques. *ACM Computing Surveys*, 32(2):109–143, June 2000.

[3] Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B.

TABLE I

COMPARISON OF RESULTS.

MOP		ER		GD		SP	
		CAEP	NSGA-II	CAEP	NSGA-II	CAEP	NSGA-II
1	Average	0.003	0.008059	0.000946	0.061548	0.037342	0.058298
	Std. Dev.	0.006749	0.005818	0.000046	0.002711	0.002674	0.009385
	Min.	0	0	0.000899	0.058940	0.034057	0.043401
	Max.	0.02	0.017857	0.001045	0.067989	0.042604	0.075381
2	Average	0.911	0.574	0.001103	0.000280	0.010931	0.007241
	Std. Dev.	0.046774	0.034059	0.000171	0.000034	0.001040	0.000741
	Min.	0.84	0.53	0.000934	0.000230	0.009493	0.006015
	Max.	0.98	0.63	0.001514	0.000349	0.012157	0.008287
3	Average	0.149	0.209	0.015321	0.001941	0.607229	0.092216
	Std. Dev.	0.069194	0.041218	0.026480	0.000078	1.025953	0.008415
	Min.	0.06	0.15	0.001942	0.001822	0.065285	0.081569
	Max.	0.28	0.28	0.076256	0.002107	2.80319	0.106568
4	Average	0.876	0.144	0.022522	0.002892	0.139053	0.038378
	Std. Dev.	0.101017	0.049035	0.008002	0.000203	0.049507	0.003837
	Min.	0.64	0.05	0.013604	0.002525	0.094707	0.031393
	Max.	0.97	0.21	0.037278	0.003199	0.242052	0.044254
6	Average	0.028	0	0.000259	0.000337	0.014476	0.008266
	Std. Dev.	0.018135	0	0.000028	0.000013	0.005168	0.000918
	Min.	0	0	0.000220	0.000318	0.009296	0.006851
	Max.	0.06	0	0.000302	0.000355	0.022920	0.010127

Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, 2002.

[4] Carlos A. Coello Coello and Ricardo Landa Becerra. Adding Knowledge and Efficient Data Structures to Evolutionary Programming: A Cultural Algorithm for Constrained Optimization. In W.B. Langdon, E.Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A.C. Schultz, J. F. Miller, E. Burke, and N.Jonoska, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 201–209, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[5] Kalyanmoy Deb. Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems. *Evolutionary Computation*, 7(3):205–230, Fall 1999.

[6] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[7] W. H. Durham. *Co-evolution: Genes, Culture, and Human Diversity*. Stanford University Press, Stanford, California, 1994.

[8] Lawrence J. Fogel. *Artificial Intelligence through Simulated Evolution. Forty Years of Evolutionary Programming*. John Wiley & Sons, Inc., New York, 1999.

[9] Carlos M. Fonseca and Peter J. Fleming. Multiobjective Genetic Algorithms Made Easy: Selection, Sharing, and Mating Restriction. In *Proceedings of the First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, pages 42–52, Sheffield, UK, September 1995. IEE.

[10] Benjamin Franklin and Marcel Bergerman. Cultural algorithms: Concepts and experiments. In *Proc. of the 2000 Congress on Evolutionary Computation*, pages 1245–1251, Piscataway, NJ, 2000. IEEE Service Center.

[11] Jeffrey Horn, Nicholas Nafpliotis, and David E. Goldberg. A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, June 1994. IEEE Service Center.

[12] Xidong Jin and Robert G. Reynolds. Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach. In *1999 Congress on Evolutionary Computation*, pages 1672–1678, Washington, D.C., July 1999. IEEE Service Center.

[13] Xidong Jin and Robert G. Reynolds. Mining Knowledge in Large-Scale Databases Using Cultural Algorithms with Constraint Handling Mechanisms. In *Proceedings of the Congress on Evolutionary Computation 2000 (CEC'2000)*, volume 2, pages 1498–1506, Piscataway, New Jersey, July 2000. IEEE Service Center.

[14] Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[15] Frank Kursawe. A variant of evolution strategies for vector optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.

[16] Carlo Poloni, Giovanni Mosetti, and Stefano Contessi. Multiobjective Optimization by GAs: Application to System and Component Design. In *Computational Methods in Applied Sciences '96: Invited Lectures and Special Technological Sessions of the Third ECCOMAS Computational Fluid Dynamics Conference and the Second ECCOMAS Conference on Numerical Methods in Engineering*, pages 258–264, Chichester, 1996. Wiley.

[17] A. C. Renfrew. Dynamic Modeling in Archaeology: What, When, and Where? In S. E. van der Leeuw, editor, *Dynamical Modeling and the Study of Change in Archaelogy*. Edinburgh University Press, Edinburgh, Scotland, 1994.

[18] Robert G. Reynolds. An Introduction to Cultural Algorithms. In A. V. Sebald and L. J. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 131–139. World Scientific, River Edge, New Jersey, 1994.

[19] Robert G. Reynolds. Cultural algorithms: Theory and applications. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 367–377. McGraw-Hill, London, 1999.

[20] Robert G. Reynolds, Zbigniew Michalewicz, and M. Cavaretta. Using cultural algorithms for constraint handling in GENOCOP. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 298–305. MIT Press, Cambridge, Massachusetts, 1995.

[21] J. David Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum, 1985.

[22] J. R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.

[23] David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.