# Culturizing Differential Evolution for Constrained Optimization

Ricardo Landa Becerra and Carlos A. Coello Coello
CINVESTAV-IPN
Evolutionary Computation Group
Dpto. de Ing. Elect./Secc. Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07300, MEXICO
rlanda@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

## Abstract

*In this paper, we propose the use of differential evolution as a population space of a cultural algorithm, applied to the solution of constrained optimization problems. Differential evolution is a relatively recent evolutionary algorithm that has been found to be very robust as a search engine for real parameter optimization. Adding different knowledge sources to the variation operator of differential evolution it is possible to improve the search and reduce the computational cost necessary to approximate the global optima of different problems. The proposed technique is validated using a set of well-known constrained optimization problems commonly adopted in the specializad literature. The approach is compared with respect to two techniques that are representative of the state-of-the-art in the area.*

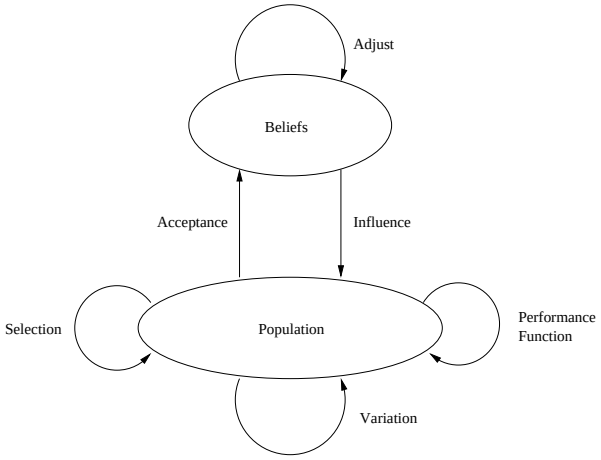


**Figure 1. Spaces of a cultural algorithm**

## 1 Introduction

Cultural algorithms are evolutionary methods that extract information of the domain of the problem during the evolutionary process itself [17]. This process of extraction and use of the information, has shown to be very effective in decreasing computational cost while approximating global optima, in unconstrained optimization, constrained optimization, and dynamic optimization [19, 2, 10, 6, 21, 4].

A cultural algorithm contains two main parts: the population space, and the belief space [18]. The **population space** consists of a set of possible solutions to the problem, and can be modeled using any population based technique, e.g. genetic algorithms [7].

The **belief space** is the information repository in which the individuals can store their experiences for the other individuals to learn them indirectly. In cultural algorithms, the information acquired by an individual can be shared with the entire population, unlike most evolutionary techniques, where the information can be only shared with the individual's offspring.

Both spaces (i.e., population space and belief space) are linked through a communication protocol, which states the rules about the individuals that can contribute to the belief space with its experiences (the acceptance function), and the way the belief space can influence to the new individuals (the influence function). Those interactions are depicted in Figure 1.

Originally, when cultural algorithms were applied to real parameter optimization, genetic algorithms were used as a population space [17]. Later on, evolutionary programming [5] appeared as a better choice [2] for the population space than genetic algorithms when dealing with constrained search spaces [3, 10]. Recently, particle swarm [11]

has also been proposed as a population space [9, 8], turning the direction to use new evolutionary methods with better performance in real parameter optimization.

Differential evolution [23] is a recently developed evolutionary algorithm, focused on solving real paramenter optimization problems. Differential evolution has been found to be a very robust optimization technique [23, 16, 14]. However, to the authors' best knowledge, this paper is the first to propose the use of differential evolution as the population space of a cultural algorithm.

The remainder of this paper is organized as follows. In Section 2, we describe some previous related work. Section 3 introduces the differential evolution algorithm and describes the specific version adopted in this paper. In Section 4, we describe our proposal to culturize differential evolution, including a description of the four knowledge sources adopted. Then, in Section 5, we compare our proposed approach with respect to two algorithms that are representative of the state-of-the-art in the area: the homomorphous maps [12] and stochastic ranking [20]. Finally, in Section 6, we provide some conclusions and some possible paths of future research.

## 2 Previous Work

Reynolds et al. [19] and Chung & Reynolds [2] have explored the use of cultural algorithms for global optimization with very encouraging results. Chung and Reynolds use a hybrid of evolutionary programming and GENOCOP in which they incorporate an interval constraint-network to represent the constraints of the problem at hand. An individual is considered as "acceptable" when it satisfies all the constraints of the problem. When that does not happen, then the belief space, *i.e.*, the intervals associated with the constraints, is adjusted. This approach is really a more sophisticated version of a repair algorithm in which an infeasible solution is made feasible by replacing its genes by a different value between its lower and upper bounds. Since GENOCOP assumes a convex search space, it is relatively easy to design operators that can exploit a search direction towards the boundary between the feasible and infeasible regions.

In more recent work, Jin and Reynolds [10] proposed an $n$-dimensional regional-based schema, called *belief-cell*, as an explicit mechanism that supports the acquisition, storage and integration of knowledge about non-linear constraints in a cultural algorithm, combined with evolutionary programming. This *belief-cell* can be used to guide the search of an evolutionary computation technique (evolutionary programming in this case) by pruning the instances of infeasible individuals and promoting the exploration of promising regions of the search space. The key aspect of this work is precisely how to represent and save the knowledge about the problem constraints in the belief space of the cultural algorithm.

The idea of Jin and Reynolds' approach is to build a map of the search space similar to the "Divide-and-Label" approaches used for robot motion planning [13]. This map is built using information derived from evaluating the constraints of each individual in the population of the evolutionary computation technique. This map is used to derive rules about how to guide the search of the evolutionary algorithm (avoiding infeasible regions and promoting the exploration of feasible regions).

Using the same population space, evolutionary programming, Saleem proposes a cultural algorithm for dealing with dynamic environments [21, 22]. Saleem adds history and domain knowledge, to the existing Chung's situational and normative knowledge, and Jin's topographical knowledge.

In [9, 8], Iacoban et al. change the evolutionary programming algorithm from the population space for a particle swarm optimizer [11]. They make an analysis of the effects of the belief space over the evolutionary process, showing the similarities with the approach in which evolutionary programming is used as a population space.

## 3 Differential Evolution

Differential evolution is an evolutionary algorithm proposed by Price and Storn [23, 16], whose main design emphasis is real parameter optimization. Differential evolution is based on a mutation operator, which adds an amount obtained by the difference of two randomly chosen individuals of the current population. The basic algorithm of differential evolution is shown in Figure 2, where the problem to solve has $n$ decision variables, $F$ and $CR$ are parameters given by the user, and $x_{i,j}$ is the $i$-th decision variable of the $j$-th individual in the population.

The authors of the differential evolution algorithm have suggested that by computing the difference between two randomly chosen individuals from the population, the algorithm is actually estimating the gradient in that zone (rather than in a point), and that is the reason why the population is always moving towards better zones. This approach is also rather efficient (computationally speaking) way to self-adapt the mutation operator.

The version of differential evolution shown in Figure 2, is called DE/rand/1/bin, and is recommended to be the first choice when trying to apply differential evolution [16]. Such version of differential evolution is the one adopted in the approach proposed in this paper.

## 4 Our Proposed Approach

The proposed approach uses differential evolution as population space. A pseudo-code of the cultured differen-

Generate initial population of size $popsize$
Do
  For each individual $j$ in the population
    Generate three random integers, $r_1$, $r_2$ and $r_3 \in (1, popsize)$, with $r_1 \neq r_2 \neq r_3 \neq j$
    Generate a random integer $i_{rand} \in (1, n)$
    For each parameter $i$

$$x'_{i,j} = \begin{cases} x_{i,r3} + F * (x_{i,r1} - x_{i,r2}) & \text{if } rand(0,1) < CR \text{ or } i = i_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}$$

    End For
    Replace $x_j$ with the child $x'_j$, if $x'_j$ is better
  End For
Until the termination condition is achieved

**Figure 2. Pseudo-code of the differential evolution algorithm adopted in this work (this version is called DE/rand/1/bin).**

Generate initial population
Evaluate initial population
Initialize the belief space
Do
  For each individual in the population
    Apply the variation operator influenced by a randomly chosen knowledge source
    Evaluate the child generated
    Replace the individual with the child, if the child is better
  End for
  Update the belief space with the accepted individuals
Until the termination condition is achieved

**Figure 3. Pseudo-code of the cultured differential evolution.**

tial evolution is shown in Figure 3.

In the initial steps of the algorithm, a population of $popsize$ individuals is created, as well as a belief space. For the children generation, the variation operator of the differential evolution is influenced by one (randomly chosen) of the four possible knowledge sources.

Since we want to solve constrained optimization problems, the fitness function by itself does not provide enough information as to guide the search properly. To determine if a child is better than its parent, and it can replace it, we use the following rules:

1. A feasible individual is always better than an infeasible one.
2. If both are feasible, the individual with the best objective function value is better.
3. If both are infeasible, the individual with less amount of constraint violations is better, measuring violations with normalized constraints.

## 4.1 The Belief Space

In our approach, the belief space is divided in four knowledge sources: situational, normative, topographical and history knowledge. Next we will describe the structure of each of the knowledge sources, as well as the influence and update functions associated to each of them.

### 4.1.1 Situational Knowledge

Situational knowledge consists of the best exemplar $E$ found along the evolutionary process. It represents a leader for the other individuals to follow.

To initialize the situational knowledge it is necessary to have an initial population, so we will be able to find the best individual and store it.

The variation operators of differential evolution are influenced in the following way:

$$x'_{i,j} = E_i + F * (x_{i,r1} - x_{i,r2})$$

where $E_i$ is the $i$-th component of the individual stored in the situational knowledge. This way, we use the leader instead a randomly chosen individual for the recombination, getting the children closer to the best point found.

| $l_1$ | $u_1$ | $l_2$ | $u_2$ | $\cdots$ | $l_n$ | $u_n$ |
|---|---|---|---|---|---|---|
| $L_1$ | $U_1$ | $L_2$ | $U_2$ | $\cdots$ | $L_n$ | $U_n$ |

| $dm_1$ | $dm_2$ | $\ldots$ | $dm_n$ |
|---|---|---|---|

**Figure 4. Structure of the normative knowledge**

The update of the situational knowledge is done by replacing the stored individual, $E$, by the best individual found in the current population, $x_{best}$, only if $x_{best}$ is better than $E$:

$$E = \begin{cases} x_{best} & \text{if } x_{best} \text{ is better than } E \\ E & \text{otherwise} \end{cases}$$

#### 4.1.2 Normative Knowledge

The normative knowledge contains the intervals for the decision variables where good solutions have been found, in order to move new solutions towards those intervals. Thus, the normative knowledge has the structure shown in Figure 4.

In Figure 4, $l_i$ and $u_i$ are the lower and upper bounds, respectively, for the $i$-th decision variable, and $L_i$ and $U_i$ are the values of the fitness function and the violations of constraints associated with that bound.

Also, the normative knowledge includes a scaling factor, $dm_i$, to influence the mutation operator adopted in differential evolution.

To initialize the normative knowledge, all the bounds are set to the intervals given as input data of the problem. $L_i$ and $U_i$ are set to $+\infty$, assuming a minimization problem, and $dm_i = u_i - l_i$, for $i = 1, 2, \ldots, n$.

The following expression shows the influence of the normative knowledge on the variation operators:

$$x_{i,j}^{'} = \begin{cases} x_{i,r3} + F * |x_{i,r1} - x_{i,r2}| & \text{if } x_{i,r3} < l_i \\ x_{i,r3} - F * |x_{i,r1} - x_{i,r2}| & \text{if } x_{i,r3} > u_i \\ x_{i,r3} + \frac{u_i - l_i}{dm_i} * F * (x_{i,r1} - x_{i,r2}) & \text{otherwise} \end{cases}$$

We introduce the scaling factor $\frac{u_i - l_i}{dm_i}$ for the mutation to be proportional to the interval of the normative knowledge for the $i$-th decision variable. The values $dm_i$ are initialized with $u_i - l_i$ to have a null influence at the first generation.

The update of the normative knowledge can reduce or expand the intervals stored on it. An expansion takes place when the accepted individuals do not fit in the current interval, while a reduction occurs when all the accepted individuals lie inside the current interval, and the extreme values have a better fitness and are feasible.

The values $dm_i$ are updated with the greatest difference $|x_{i,r1} - x_{i,r2}|$ found during application of the variation operators of the previous generation.

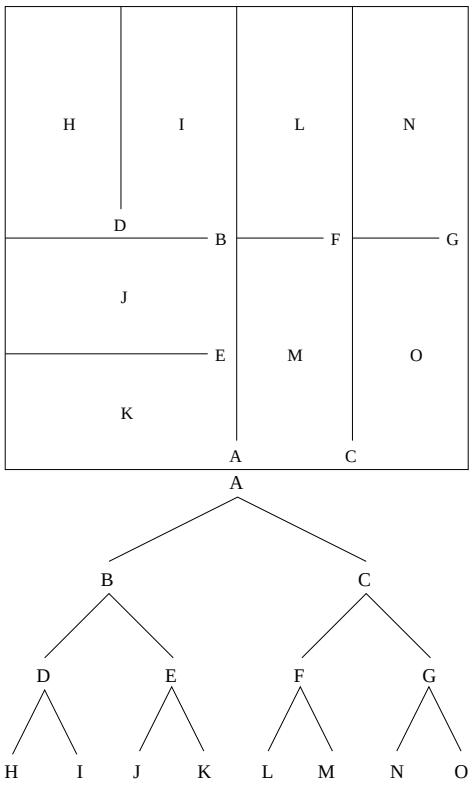


**Figure 5. Example of the partition of a two dimensional space by a $k$-d tree**

#### 4.1.3 Topographical Knowledge

The use of the topographical knowledge is to create a map of the fitness landscape of the problem during the evolutionary process. It consists of a set of cells, and the best individual found on each cell. The topographical knowledge, also, has an ordered list of the best $b$ cells, based on the fitness value of the best individual on each of them.

For the sake of a more efficient memory management, in the presence of high dimensionality (i.e., too many decision variables), we use an spatial data structure, called $k$-d tree, or $k$-dimensional binary tree [1]. In $k$-d trees, each node can only have two children (or none, if it is a leaf node), and represents a division in half for any of the $k$ dimensions (see Figure 5).

To initialize the topographical knowledge, we only create the root node, which represents the entire search space, and contains the best solution found in the initial population.

The influence function tries to move the children to any of the $b$ cells in the list:

$$x_{i,j}^{'} = \begin{cases} x_{i,r3} + F * |x_{i,r1} - x_{i,r2}| & \text{if } x_{i,r3} < l_{i,c} \\ x_{i,r3} - F * |x_{i,r1} - x_{i,r2}| & \text{if } x_{i,r3} > u_{i,c} \\ x_{i,r3} + F * (x_{i,r1} - x_{i,r2}) & \text{otherwise} \end{cases}$$

| $e_1$ | $\cdots$ | $e_i$ | $\cdots$ | $e_w$ |
|---|---|---|---|---|
| $ds_1$ | $ds_2$ | $\cdots$ | $ds_n$ | |
| $dr_1$ | $dr_2$ | $\cdots$ | $dr_n$ | |

**Figure 6. Structure of the history knowledge**

where $l_{i,c}$ and $u_{i,c}$ are the lower and upper bounds of the cell $c$, randomly chosen from the list of the $b$ best cells.

The update function splits a node if a better solution is found in that cell, and if the tree has not reached its maximum depth. The dimension in which the division is done, is the one that has a greater difference between the solution stored and the new reference solution (i.e., the new solution considered as the "best" found so far).

#### 4.1.4 History Knowledge

This knowledge source was originally proposed for dynamic objective functions, and it was used to find patterns in the environmental changes. As we use static objective functions, we can use this knowledge source to find patterns of the positions of local aptima. In its original form, history knowledge records in a list, the location of the best individual found before each environmental change. That list has a maximum size $w$. The history knowledge also contains the average distance and direction of the changes for each decision variable, as shown in Figure 6.

In Figure 6, $e_i$ is the best individual found before the $i$-th environmental change, $ds_i$ is the average distance of the changes for parameter $i$, and $dr_i$ is the average direction if there are changes for parameter $i$. In our approach, instead of detecting changes of the environment, we store a solution if it remains as the best one by the last $p$ generations. If this happens, we assume that we are trapped in a local optimum. To initialize this part, the list of best individuals is empty, and the average distances and directions is zero. The influence function is divided in three parts, the first is for trying to move the generated individual to the direction $dr_i$; this is done the $\alpha\%$ of the times. The second is for trying to move the child a distance $ds_i$; this is done the $\beta\%$ of the times. The third is only to inject diversity, keeping in mind that this knowledge source must be useful when we are trapped in local optima. The expression is:

$$x'_{i,j} = \begin{cases} x_{i,e_w} + dr_i * F * |x_{i,r1} - x_{i,r2}| & \\ \qquad \text{if } rand(0,1) < \alpha & \\ x_{i,e_w} + \frac{1.5 * ds_i}{dm_i} * (x_{i,r1} - x_{i,r2}) & \\ \qquad \text{if } rand(0,1) < \beta & \\ rand(lb_i, ub_i) & \\ \qquad \text{otherwise} & \end{cases}$$

where $x_{i,e_w}$ is the $i$-th decision variable of the previous best $e_w$ stored in the list of the history knowledge, $dr_i$ and $ds_i$ are the average direction and distance of the changes in the $i$-th variable, $dm_i$ is the maximum difference for the $i$-th variable, stored in the normative knowledge, $lb_i$ and $ub_i$ are the lower and upper bounds of the variable $x_i$, given as input for the problem, and $rand(a, b)$ is a uniform distributed random number between $a$ and $b$.

To update the history knowledge, we add to the list any local optimum found during the evolutionary process. If the list has reached its maximum length $w$, the oldest element in the list is discarded. The average distances and directions of change are calculated by:

$$ds_i = \frac{\sum_{k=1}^{w-1} \left| x_{i,e_{k+1}} - x_{i,e_k} \right|}{w-1}$$

$$dr_i = sgn\left( \sum_{k=1}^{w-1} sgn\left( x_{i,e_{k+1}} - x_{i,e_k} \right) \right)$$

where the function $sgn$ is defined by:

$$sgn(a) = \begin{cases} 1 & \text{if } a > 0 \\ -1 & \text{if } a < 0 \\ 0 & \text{otherwise} \end{cases}$$

### 4.2 Acceptance Function

The number of individuals accepted for the update of the belief space is computed according the design of a dynamic acceptance function by Saleem [22]. The number of accepted individuals decreases while the number of generation increases.

Saleem [22] suggests to reset the number of accepted individuals when an environmental change occurs. In our case, all functions are static, but this mechanism is useful also when the population converges to a local optimum. Then, we reset the number of accepted individuals when the best solution has not changed in the last $p$ generations.

We get the number of accepted individuals, $n_{accepted}$, with the following expression:

$$n_{accepted} = \left\lfloor p\% * popsize + \frac{p\% * popsize}{g} \right\rfloor$$

where $p\%$ is a parameter given by the user, in $(0, 0.5]$; Saleem [22] suggests using 0.2. $g$ is the generation counter, but is reset to 1 when the best solution has no changed in the last $p$ generations.

## 5 Comparison of Results

To validate our approach, we have used the well-known benchmark proposed in [15] and extended in [20] which has been often used in the literature to validate new constraint-handling techniques. The characteristics of the test functions are sumarized in Table 1, similar to the table reported

**Table 4. Results reported for the homomorphous maps [12]. N.A. = Not available**

| TF | Optimal | Best | Mean | Worst |
|---|---|---|---|---|
| g01 | **-15** | -14.7864 | -14.7082 | -14.6154 |
| g02 | **0.803619** | 0.79953 | 0.79671 | 0.79119 |
| g03 | **1** | 0.9997 | 0.9989 | 0.9978 |
| g04 | **-30665.539** | -30664.5 | -30655.3 | -30645.9 |
| g05 | **5126.4981** | — | — | — |
| g06 | **-6961.8138** | -6952.1 | -6342.6 | -5473.9 |
| g07 | **24.3062091** | 24.620 | 24.826 | 25.069 |
| g08 | **0.095825** | **0.095825** | 0.089157 | 0.029144 |
| g09 | **680.6300573** | 680.91 | 681.16 | 683.18 |
| g10 | **7049.330** | 7147.9 | 8163.6 | 9659.3 |
| g11 | **0.75** | **0.75** | **0.75** | **0.75** |
| g12 | **1** | 0.999999 | 0.999135 | 0.991950 |
| g13 | **0.0539498** | N.A. | N.A. | N.A. |

in [12]. Table 1 shows, for each problem, the number of variables $n$, the type of objective function $f$, the ratio $\rho$ between the feasible region and the whole search space, the number of linear inequalities $LI$, the number of nonlinear inequalities $NI$, the number of nonlinear equations $NE$, and the number of active constraints at the optimum (or best known) $a$. The ratio $\rho$ is an estimation done by generating 10,000,000 random points in the search space, and checking their feasibility. For a full description of the test functions adopted, the reader should refer to [20].

The parameters used by our approach are the following: $popsize = 100$, maximum number of generations = 1000, the factors of differential evolution are $F = 0.5$ and $CR = 1$, maximum depth of the $k$-d tree = 12, length of the best cells list $b = 10$, the size of the list in the history knowledge $w = 5$, $\alpha = \beta = 0.45$, and $\%p = 0.2$. These parameters were derived empirically after numerous experiments.

Our results are compared to the homomorphous maps of Koziel & Michalewicz [12] and to stochastic ranking of Runarsson & Yao [20] (these are the two best constraint-handling techniques proposed for evolutionary algorithms known to date). The results of Koziel & Michalewicz were obtained with 1,400,000 fitness function evaluations. The results of Runarsson & Yao were obtained with 350,000 evaluations. Our approach required only 100,100 evaluations.

Our approach performs better than the homomorphous maps in most of the problems. It is worth noticing that in g05 the homomorphous maps were not able to find any feasible solution, whereas our cultured differential evolution algorithm found feasible solutions in all the runs performed. Although our approach did not reach the global optimum in this case, its best result is very close from it. Another good example is g10, where our approach not only reached the global optimum but systematically converged to solutions very close to it. In contrast, the homomorphous maps were not able to converge to the global optimum and it showed a considerably higher variability of results from run to run. Our approach showed its worst performance in g02, where it was unable to reach the best known solution. This may be due to the fact that g02 has a fairly large feasible region and the number of iterations performed may had been insufficient for allowing our approach to reach the global optimum. Note that the authors of the homomorphous maps did not use g13 in their paper. That is the reason why no results are reported for such problem.

Stochastic ranking is the most competitive constraint-handling technique known to date as can be seen from the results shown in Table 3. When compared with respect to our approach, we can see that our cultured differential evolution algorithm is very competitive. Our approach reached the global optimum in seven problems, and stochastic ranking did it in nine. However, with the exception of g02 and g01 (where stochastic ranking was a clear winner), in all the other problems the results obtained by our approach are very close to the global optimum. An additional aspect that we found quite interesting is that our approach presented in most cases a low standard deviation. A remarkable example is g10, where stochastic ranking was not able to reach the global optimum and presented a high variability of results. Another example is g06, where stochastic ranking also presented a higher variability than our approach. In contrast, stochastic ranking showed a more robust behavior than our approach in g01, g05, g11 and g13.

## 6 Conclusions and Future Work

We have presented the first attempt to use differential evolution as a population space of a cultural algorithm. This approach is applied in the solution of constrained numerical optimization problems.

Differential evolution has been found before to be a very succesful and robust search engine for optimization problems. As a consequence, and fulfilling our expectations, the culturization of a differential evolution algorithm has given rise to an approach that requires a low number of fitness function evaluations. Our comparison of results with respect to two approaches representative of the state-of-the-art in the area has shown that such an approach produces highly competitive results and is very robust.

One of the main limitations of our current version of the algorithm presented in this paper has to do with the lack of diversity that occurs due to the high selection pressure of our approach. This occurs despite the fact that the knowledge sources adopted by our cultural algorithm provide additional diversity to the approach.

**Table 1. Characteristics of the test functions**

| TF | $n$ | Type of $f$ | $\rho$ | $LI$ | $NI$ | $NE$ | $a$ |
|---|---|---|---|---|---|---|---|
| g01 | 13 | quadratic | 0.000235% | 9 | 0 | 0 | 6 |
| g02 | 20 | nonlinear | 99.996503% | 1 | 1 | 0 | 1 |
| g03 | 10 | polynomial | 0.000000% | 0 | 0 | 1 | 1 |
| g04 | 5 | quadratic | 26.962511% | 0 | 6 | 0 | 2 |
| g05 | 4 | cubic | 0.000000% | 2 | 0 | 3 | 3 |
| g06 | 2 | cubic | 0.006679% | 0 | 2 | 0 | 2 |
| g07 | 10 | quadratic | 0.000103% | 3 | 5 | 0 | 2 |
| g08 | 2 | nonlinear | 0.859082% | 0 | 2 | 0 | 0 |
| g09 | 7 | polynomial | 0.524450% | 0 | 4 | 0 | 2 |
| g10 | 8 | linear | 0.000522% | 3 | 3 | 0 | 3 |
| g11 | 2 | quadratic | 0.000000% | 0 | 0 | 1 | 1 |
| g12 | 3 | quadratic | 4.775265% | 0 | 1 | 0 | 0 |
| g13 | 5 | nonlinear | 0.000000% | 0 | 0 | 3 | 3 |

**Table 2. Results obtained by our cultured differential evolution approach**

| TF | Optimal | Best | Mean | Worst | Std Dev |
|---|---|---|---|---|---|
| g01 | **-15** | -14.996953 | -13.214513 | -5.999896 | 2.985388 |
| g02 | **0.803619** | 0.616900 | 0.517901 | 0.419959 | 0.066237 |
| g03 | **1** | **1.000000** | 0.821397 | 0.600900 | 0.144609 |
| g04 | **-30665.539** | **-30665.539177** | -30665.538824 | -30665.538672 | 0.000244 |
| g05 | **5126.4981** | 5126.563220 | 5136.862081 | 5184.827897 | 19.569988 |
| g06 | **-6961.8138** | **-6961.813876** | **-6961.813876** | **-6961.813876** | 0.000000 |
| g07 | **24.3062091** | 24.575671 | 24.585679 | 24.650253 | 0.023298 |
| g08 | **0.095825** | **0.095825** | **0.095825** | **0.095825** | 0.000000 |
| g09 | **680.6300573** | **680.630057** | **680.630057** | **680.630057** | 0.000000 |
| g10 | **7049.25** | **7049.251189** | 7049.284777 | 7049.372205 | 0.040707 |
| g11 | **0.75** | 0.757500 | 0.779440 | 0.854357 | 0.039593 |
| g12 | **1** | **1.000000** | **1.000000** | **1.000000** | 0.000000 |
| g13 | **0.0539498** | 0.054903 | 0.314341 | 0.426815 | 0.181099 |

**Table 3. Results reported for stochastic ranking [20]**

| TF | Optimal | Best | Mean | Worst | Std Dev |
|---|---|---|---|---|---|
| g01 | **-15** | **-15.000** | **-15.000** | **-15.000** | 0.0 |
| g02 | **0.803619** | 0.803515 | 0.781975 | 0.726288 | 0.020 |
| g03 | **1** | **1.000** | **1.000** | **1.000** | 0.00019 |
| g04 | **-30665.539** | **-30665.539** | **-30665.539** | **-30665.539** | 0.00002 |
| g05 | **5126.4981** | **5126.497** | 5128.881 | 5142.472 | 3.5 |
| g06 | **-6961.8138** | **-6961.814** | -6875.940 | -6350.262 | 160 |
| g07 | **24.3062091** | 24.307 | 24.374 | 24.642 | 0.066 |
| g08 | **0.095825** | **0.095825** | **0.095825** | **0.095825** | 0.000000 |
| g09 | **680.6300573** | **680.630** | 680.656 | 680.763 | 0.034 |
| g10 | **7049.330** | 7054.316 | 7559.192 | 8835.655 | 530 |
| g11 | **0.75** | **0.750** | **0.750** | **0.750** | 0.00008 |
| g12 | **1** | **1.000000** | **1.000000** | **1.000000** | 0.0 |
| g13 | **0.0539498** | 0.053957 | 0.057006 | 0.216915 | 0.031 |

As part of our future work, we are interested in investigating the specific role played by each of the knowledge sources adopted. We want to measure the impact of each of these knowledge sources on the evolutionary process as to allow a more appropriate balance. We hypothesize that such a balance will improve the quality of the results obtained by our approach. Finally, we are also interested in adopting additional diversity maintenance mechanisms to counterbalance the high selection pressure of our approach.

## Acknowledgements

The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Section of the Electrical Engineering Department at CINVESTAV-IPN. The second author gratefully acknowledges support from CONACyT through project 42435-Y.

## References

[1] J. L. Bentley and J. H. Friedman. Data Structures for Range Searching. *ACM Computing Surveys*, 11(4):397–409, December 1979.

[2] C.-J. Chung and R. G. Reynolds. A Testbed for Solving Optimization Problems using Cultural Algorithms. In L. J. Fogel, P. J. Angeline, and T. Bäck, editors, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, Cambridge, Massachusetts, 1996. MIT Press.

[3] C. A. Coello Coello and R. Landa Becerra. Adding knowledge and effi cient data structures to evolutionary programming: A cultural algorithm for constrained optimization. In E. C.-P. et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pages 201–209, San Francisco, California, July 2002. Morgan Kaufmann Publishers.

[4] C. A. Coello Coello and R. Landa Becerra. Evolutionary Multiobjective Optimization using a Cultural Algorithm. In *2003 IEEE Swarm Intelligence Symposium Proceedings*, pages 6–13, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.

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

[6] B. Franklin and M. Bergerman. Cultural algorithms: Concepts and experiments. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 1245–1251, Piscataway, New Jersey, 2000. IEEE Service Center.

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

[8] R. Iacoban, R. G. Reynolds, and J. Brewster. Cultural Swarms: Assessing the Impact of Culture on Social Interaction and Problem Solving. In *2003 IEEE Swarm Intelligence Symposium Proceedings*, pages 212–219, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.

[9] R. Iacoban, R. G. Reynolds, and J. Brewster. Cultural Swarms: Modeling the Impact of Culture on Social Interaction and Problem Solving. In *2003 IEEE Swarm Intelligence Symposium Proceedings*, pages 205–211, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.

[10] X. Jin and R. 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.

[11] J. Kennedy and R. C. Eberhart. Particle swarm: Social adaptation in information-processing systems. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 379–387. McGraw-Hill, London, UK, 1999.

[12] S. Koziel and Z. Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[13] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, Massachusetts, 1993.

[14] E. Mezura-Montes, C. A. C. Coello, and E. I. Tun-Morales. Simple Feasibility Rules and Differential Evolution for Constrained Optimization. In R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, and H. Sossa, editors, *Proceedings of the Third Mexican International Conference on Artificial Intelligence (MICAI'2004)*, pages 707–716, Heidelberg, Germany, April 2004. M´exico City, M´exico, Springer Verlag. Lecture Notes in Artifi cial Intelligence No. 2972.

[15] Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[16] K. V. Price. An introduction to differential evolution. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 79–108. McGraw-Hill, London, UK, 1999.

[17] R. 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 Scientifi c, River Edge, New Jersey, 1994.

[18] R. G. Reynolds. Cultural algorithms: Theory and applications. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 367–377. McGraw-Hill, London, UK, 1999.

[19] R. G. Reynolds, Z. 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.

[20] T. P. Runarsson and X. Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.

[21] S. Saleem and R. Reynolds. Cultural algorithms in dynamic environments. In *Proc. of the 2000 Congress on Evolutionary Computation*, pages 1513–1520, Piscataway, NJ, July 2000. IEEE Service Center.

[22] S. M. Saleem. *Knowledge-Based Solution to Dynamic Optimization Problems using Cultural Algorithms*. PhD thesis, Wayne State University, Detroit, Michigan, 2001.

[23] R. Storn. System Design by Constraint Adaptation and Differential Evolution. *IEEE Transactions on Evolutionary Computation*, 3(1):22–34, April 1999.