Optimization with Constraints using a Cultured Differential Evolution Approach

Ricardo Landa Becerra
Evolutionary Computation Group (EVOCINV)
CINVESTAV-IPN
Computer Science Section
Electrical Engineering Department
México, D.F., 07300, MEXICO
rlanda@computacion.cs.cinvestav.mx

Carlos A. Coello Coello
Evolutionary Computation Group (EVOCINV)
CINVESTAV-IPN
Computer Science Section
Electrical Engineering Department
México, D.F., 07300, MEXICO
ccoello@cs.cinvestav.mx

ABSTRACT

In this paper we propose a cultural algorithm, where different knowledge sources modify the variation operator of a differential evolution algorithm. Differential evolution is used as a basis for the population, variation and selection processes. The experiments performed show that the cultured differential evolution is able to reduce the number of fitness function evaluations needed to obtain a good aproximation of the optimum value in constrained real-parameter optimization. Comparisons are provided with respect to three techniques that are representative of the state-of-the-art in the area.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Constrained optimization, Global optimization*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms, design

Keywords

Constraint handling, optimization, cultural algorithms, differential evolution

1. INTRODUCTION

Evolutionary computation is an increasingly popular discipline, mainly due to the successful results obtained in many types of optimization problems [1, 18]. However, evolutionary computation techniques are considered "blind heuristics" because they do not normally require information about the problem but only the assessment of the quality of the solutions produced (through the so-called "fitness function").

Cultural algorithms are evolutionary computation techniques that extract domain knowledge during the evolutionary process aiming to improve performance. In this work, we explore some of the benefits of this type of incorporation of domain knowledge when applied to constrained optimization problems.

The remainder of this paper is organized as follows. In Section 2, we provide some basic concepts related to cultural algorithms together with a brief review of the most important previous (related) work. In Section 3, we describe the basic differential evolution algorithm and we briefly review the most representative previous work on constrained optimization. In Section 4, we describe our proposed approach. In Section 5, we compare our approach with respect to techniques representative of the state-of-the-art in the area using a well-known benchmark. Finally, in Section 6, we draw our conclusions and provide some possible paths for future research.

2. CULTURAL ALGORITHMS

Cultural algorithms are techniques that add domain knowledge to evolutionary computation methods. They are based on the assumption that domain knowledge can be extracted during the evolutionary process, by means of the evaluation of each point generated [20]. This process of extraction and use of the information, has been shown to be very effective in decreasing computational cost while approximating global optima, in unconstrained, constrained and dynamic optimization [22, 4, 11, 24].

Cultural algorithms consist of two main components: the population space, and the belief space [21]. 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 [8]. 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.

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 their experiences (the acceptance function), and the way the belief space can influence to the new individuals

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'05, June 25–29, 2005, Washington, DC, USA.
Copyright 2005 ACM 1-59593-010-8/05/0006 ...$5.00.

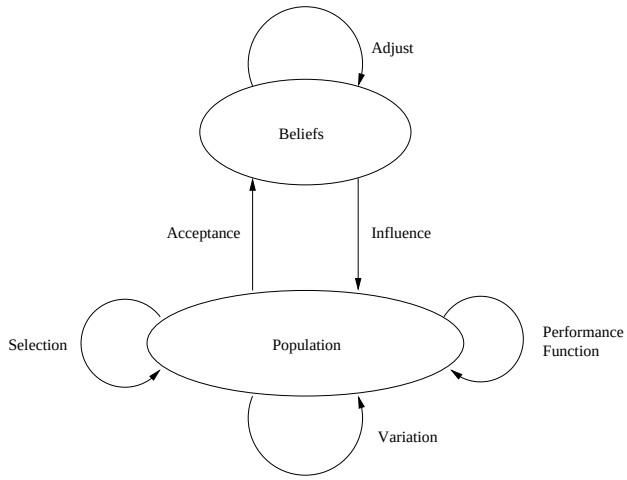

Figure 1: Spaces of a cultural algorithm

(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 [20].

Later on, evolutionary programming appeared as a better choice [3] for the population space than genetic algorithms when dealing with unconstrained search spaces. The evolutionary programming algorithm was the most commonly used search engine in cultural algorithms [4, 11, 5], until the advent of particle swarm optimization [12] in [10], which shed light regarding the potential use of new evolutionary methods with better performance in real parameter optimization.

Differential evolution [19] is a recently developed evolutionary algorithm, focused on the solution of real paramenter optimization problems. Differential evolution has been found to be a very robust optimization technique [26]. However, to the authors' best knowledge, we are the first to propose the use of differential evolution as the population space of a cultural algorithm.

2.1 Previous Work

Reynolds et al. [22] and Chung & Reynolds [4] 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 [22] in which they incorporate an interval constraint-network to represent the constraints of the problem at hand.

In [4], Chung and Reynolds use evolutionary programming with a mutation operator influenced by the best individual found so far, and the intervals where good solutions have been found. They call their approach "CAEP", or Cultural Algorithms with Evolutionary Programming.

In more recent work, Jin and Reynolds [11] 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. The idea of Jin and Reynolds' approach is to build a map of the search space which 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 [24], which adds two more ways to incorporate domain knowledge to CAEPs, in addition to the existing Chung's and Jin's proposals. These knowledge sources are designed to extract patterns in environmental changes.

Using as a basis the work of Jin and Reynolds, an algorithm for constrained optimization was developed in [5, 6], where a spatial data structure is incorporated to store the map of the feasible region, and also new rules are used in several phases of the algorithm.

In [10], Iacoban et al. change the evolutionary programming algorithm of the population space for a particle swarm optimizer [12]. 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 adopted, and identifying the phases of the search process with a belief space.

3. DIFFERENTIAL EVOLUTION

Differential evolution is an evolutionary algorithm originally proposed by Price and Storn [19, 25], 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, in contrast to most of the evolutionary algorithms, in which the mutation operator is defined by a probability function.

The basic algorithm of differential evolution is shown in Figure 2, where the problem to be solved 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 individuals randomly chosen from the population, the algorithm is actually estimating the gradient in that zone (rather than in a point). This approach also constitutes a rather efficient way to self-adapt the mutation operator.

Another important feature of the differential evolution algorithm, is the local criterion of the selection operator, which is efficient and fast.

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 to any given problem [19]. That is the reason why we adopted it for the work reported in this paper. However, there are some other versions of the differential evolution algorithm, and the modifications made here to the variation operator may have certain similarities with some of those versions.

3.1 Differential Evolution in Constrained Optimization

There have been very few approaches for handling constraints based on differential evolution. We will review next the most representative of them.

One of the original developers of differential evolution, Storn, proposed constraint adaptation [26], in which all the constraints of the problem at hand are relaxed, so that all

```
Generate initial population of size popsize
Do
    For each individual j in the population
        Generate three random integers, r1, r2 and r3 ∈ (1, popsize),
            with r1 ≠ r2 ≠ r3 ≠ j
        Generate a random integer i_rand ∈ (1, n)
        For each parameter i
            x'_{i,j} = { x_{i,r3} + F * (x_{i,r1} - x_{i,r2})   if rand(0,1) < CR or i = i_rand
                       { x_{i,j}                               otherwise
        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)

the individuals in the initial population become feasible. The constraints are reduced toward their original versions at each generation, but the individuals must always remain feasible (i.e., different relaxations are applied at each generation). The author says that this approach is not suitable for handling equality constraints, and one of its main applications is constraint satisfaction (where only constraint violation is important, and there is no objective function).

Another constraint handling technique is the one proposed by Lampinen [15]. He states some rules for the replacement made during the selection procedure, that can be summarized as follows:

- If both individuals are feasible, the one with a better value of the objective function always wins.
- If the newly generated individual is feasible, as his parent is infeasible, the new individual is used for the next generation.
- If both individuals are infeasible, the parent is replaced if the new individual has lower or equal violation for all the constraints.

The rest of the differential evolution algorithm remains the same. The experiments are done with 10 of the 11 test functions proposed in [13]. A previous version of this algorithm appeared in [14], where the replacement rules were not as complete as in [15], and less test problems were taken into account.

Simultaneously to Lampinen, Lin et al. [16] proposed the use of an augmented Lagrangian function to guide the search, with a newly developed method to update the multipliers. In a first phase, the multipliers are constant and the problem is minimized. During a second phase, the multipliers are updated and the algorithm tries to maximize the dual function.

Lin et al. [16] called their approach hybrid differential evolution, because they add some new steps to the original algorithm. Such steps are acceleration and migration, and are used when the current population has either too much or no diversity.

4. OUR PROPOSED APPROACH

Our proposed approach uses differential evolution in the population space. A pseudo-code of our approach (called cultured differential evolution) is shown in Figure 3.

```
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.

In the initial steps of the algorithm, a population of *popsize* individuals is created, as well as a belief space. For the offspring generation, the variation operator of the differential evolution algorithm is influenced by the belief space.

Since we want to solve constrained optimization problems, the objective 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, therefore, 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.

The amount of constraint violation is measured with normalized constraints, with the use of the following expression:

$$viol(x_j) = \sum_{c=1}^{constr} \frac{g_c(x_j)}{g_{maxc}}$$

where $g_c(x)$ are the *constr* constraints of the problem, and g_{maxc} is the greatest violation found of the constraint $g_c(x)$ found so far.

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

4.1 The Belief Space

In our approach, the belief space is divided in four knowledge sources, described next.

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 that we can 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 of a randomly chosen individual for the recombination. This has the effect of pushing the children closer to the best point found.

This way of influencing the variation operator was previously proposed in [25], and is called DE/best/1/bin (if the other mechanisms remain the same as in DE/rand/1/bin; in fact, the word "rand" or "best" indicates if the individual $r3$ is chosen at random or is the best of the population). The difference with that previous proposal is that we use several modifications of the variation operator, and not only one.

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 associated with that bound. Also, the normative knowledge includes the values 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 is as follows: let $x_{a1}, x_{a2}, \ldots, x_{an_{accepted}}$ be the accepted individuals in the current generation, and x_{min_i} and x_{max_i}, with $min_i, max_i \in \{a1, a2, \ldots, n_{accepted}\}$, be the individuals with minimum and maximum values for the parameter i between the accepted individuals, then

$$l_i = \begin{cases} x_{i,min_i} & \text{if } x_{i,min_i} < l_i \vee f(x_{min_i}) < L_i \\ l_i & \text{otherwise} \end{cases}$$

and

$$u_i = \begin{cases} x_{i,max_i} & \text{if } x_{i,max_i} > u_i \vee f(x_{max_i}) < U_i \\ u_i & \text{otherwise} \end{cases}$$

In words, the update will reduce or expand the intervals stored in the normative knowledge. 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.

If the values of l_i or u_i are updated, the same must be done with L_i or U_i.

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

4.1.3 Topographical Knowledge

The usefulness 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 [2]. 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:

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

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

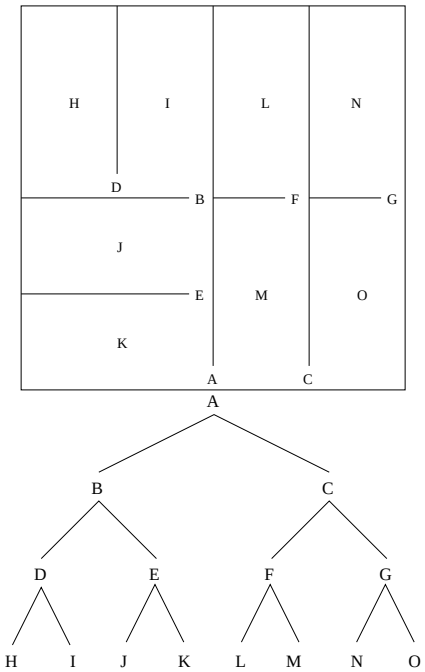

Figure 5: Example of the partition of a two dimensional space by a k-d tree

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

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, where it was used to find patterns in the environmental changes [24]. 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 structure of the history knowledge is shown in Figure 6, where 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 during the last p generations. If this happens, we assume that we are trapped in a local optimum.

The expression of the influence function of the history knowledge is the following:

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

where $e_{i,1}$ is the i-th decision variable of the previous best e_1 stored in the list of the history knowledge, 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 random number between a and b.

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

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

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

where the function $sgn(a)$ returns the sign of a.

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 proposed by Saleem [24]. The number of accepted individuals decreases as the generation number increases.

Saleem [24] suggests to reset the number of accepted individuals when an environmental change occurs. In our case, 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:

$$n_{accepted} = \left\lfloor \%p * popsize + \frac{(1 - \%p) * popsize}{g} \right\rfloor$$

where $\%p$ is a parameter given by the user, within the range $(0, 1]$; Saleem [24] suggests using 0.2. g is the generation counter, but is reset to 1 when the best solution has not changed in the last p generations.

4.3 Main Influence Function

The main influence function is responsible for choosing the knowledge source to be applied to the variation operator of differential evolution. At the beginning, all the knowledge sources have the same probability to be applied, $\%p_{ks} = \frac{1}{4}$, because there are 4 knowledge sources; but during the evolutionary process, the probability of the knowledge source ks to be applied is:

$$\%p_{ks} = 0.1 + 0.6\frac{v_{ks}}{v}$$

where v_{ks} are the times that an individual generated by the knowledge source ks outperforms its parent in the current generation, and v are the times that an individual generated (by any knowledge source) outperforms its parent in the current generation. The lower bound of $\%p$ is the arbitrary value 0.1, to ensure that any knowledge source has always a probability > 0 to be applied. If $v = 0$ during a generation, $\%p_{ks} = \frac{1}{4}$, as in the beginning.

5. COMPARISON OF RESULTS

To validate our approach, we adopted the well-known benchmark originally proposed in [17] and extended in [23] which has been often used in the literature to validate new

Problem	Optimal	Results of the cultured differential evolution algorithm			
		Best	Mean	Worst	St. Dev.
g01	-15	**-15.000000**	-14.999996	-14.999993	0.000002
g02	0.803619	**0.803619**	0.724886	0.590908	0.070125
g03	1	0.995413	0.788635	0.639920	0.115214
g04	-30665.539	**-30665.538672**	**-30665.538672**	**-30665.538672**	0.000000
g05	5126.4981	5126.570923	5207.410651	5327.390497	69.225796
g06	-6961.8138	**-6961.813876**	**-6961.813876**	**-6961.813876**	0.000000
g07	24.3062091	**24.306209**	24.306210	24.306212	0.000001
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	**7049.248058**	**7049.248266**	**7049.248480**	0.000167
g11	0.75	**0.749900**	0.757995	0.796455	0.017138
g12	1	**1.000000**	**1.000000**	**1.000000**	0.000000
g13	0.0539498	0.056180	0.288324	0.392100	0.167095

Table 1: Results obtained by our cultured differential evolution approach

constraint-handling techniques. For a further description of those problems, refer to [23].

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. For each test function, we preformed 30 independent runs.

We compare our approach against three state-of-the-art approaches: the Homomorphous Maps (HM) [13], Stochastic Ranking (SR) [23] and the Adaptive Segregational Constraint Handling Evolutionary Algorithm (ASCHEA) [9]. We do not include any differential evolution-based technique in this comparison, because the reported results do not allow us to do it. The only approach that uses the cited benchmark is the one by Lampinen [15], but he does not report the complete benchmark, and his results were obtained with a variable number of fitness function evaluations. The best results obtained by each approach are shown in Table 2a. The mean values provided are compared in Table 2b and the worst results are presented in Table 2c. The results provided by these approaches were taken from the original references for each method.

The results of HM were obtained with **1,400,000** evaluations of the fitness function, the results of SR required **350,000** evaluations, and the results of the ASCHEA technique were obtained with **1,500,000** evaluations of the fitness function. Our approach required only **100,100** evaluations.

From the results, we can say that SR is the most competitive constraint handling technique compared here. However, our approach reached the global optimum in ten problems, and SR did it in nine. Also, in most cases, the cultured differential evolution was more robust, showing a very low standard deviation, while performing less than one third of the total number of fitness function evaluations than stochastic ranking.

ASCHEA is also a very competitive technique, reaching the optimum in seven problems, but it requires the largest number of evaluations of the techniques compared here (the cultured differential evolution required less than a tenth part of the fitness function evaluations performed by ASCHEA). HM shows the worst results in this comparison, reaching the optimum in only three cases.

We can see that the approximations of the optimal values obtained by the cultured differential evolution are very competitive in most of the cases, and are obtained with a low number of fitness function evaluations. This can be attributed to a faster exploration of the influenced operators by the belief space. The robustness is a feature of the search engine (i.e. differential evolution), also increased by the knowledge sources that allows exploitation, such as the situational knowledge.

6. CONCLUSIONS AND FUTURE WORK

In this paper a cultural algorithm for constrained optimization was presented. This algorithm uses differencial evolution as a basis for its population space population. In a comparison with other three representative techniques of the state-of-the-art, and using a standard benchmark from the evolutionary optimization literature, best or equal best results were obtained in 10 of 13 benchmark problems, with the important aggregated value that a considerably smaller number of fitness funtion evaluations was required. This reduction was our main motivation in designing the belief space of the cultural algorithm.

Some directions of future work are the analysis of the impact of each knowledge source during the evolutionary process. Additionally, we want to explore the benefits of using a belief space in other types of problems. Specifically, we are interested in extending our approach so that it can deal with multiobjective optimization problems [7].

Acknowledgements

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

7. REFERENCES

[1] T. Bäck, D. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*, volume 1.

Problem	Optimal	Best Results of the compared techniques			
		CDE	**HM**	**SR**	**ASCHEA**
g01	**−15**	**−15.000000**	−14.7864	−15.000	−15.0
g02	**0.803619**	**0.803619**	0.79953	0.803515	0.785
g03	**1**	0.995413	0.9997	1.000	1.0
g04	**−30665.539**	**−30665.538672**	−30664.5	−30665.539	30665.5
g05	**5126.498**	5126.570923	–	5126.497	5126.5
g06	**−6961.814**	**−6961.813876**	−6952.1	−6961.814	−6961.81
g07	**24.306**	**24.306209**	24.620	24.307	24.3323
g08	**0.095825**	**0.095825**	0.0958250	0.095825	0.095825
g09	**680.63**	**680.630057**	680.91	680.630	680.630
g10	**7049.25**	**7049.248058**	7147.9	7054.316	7061.13
g11	**0.75**	**0.749900**	0.75	0.750	0.75
g12	**1.00**	**1.000000**	0.999999	1.000000	NA
g13	**0.053950**	0.056180	NA	0.053957	NA

a)

Problem	Optimal	Mean Results of the compared techniques			
		CDE	**HM**	**SR**	**ASCHEA**
g01	**−15**	−14.999996	−14.7082	−15.000	−14.84
g02	**0.803619**	0.724886	0.79671	0.781975	0.59
g03	**1**	0.788635	0.9989	1.000	0.99989
g04	**−30665.539**	**−30665.538672**	−30655.3	−30665.539	30665.5
g05	**5126.498**	5207.410651	–	5128.881	5141.65
g06	**−6961.814**	**−6961.813876**	−6342.6	−6875.940	−6961.81
g07	**24.306**	**24.306210**	24.826	24.374	24.66
g08	**0.095825**	**0.095825**	0.0891568	0.095825	0.095825
g09	**680.63**	**680.630057**	681.16	680.656	680.641
g10	**7049.25**	**7049.248266**	8163.6	7559.192	7193.11
g11	**0.75**	0.757995	0.75	0.750	0.75
g12	**1.00**	**1.000000**	0.999134	1.000000	NA
g13	**0.053950**	0.288324	NA	0.067543	NA

b)

Problem	Optimal	Worst Results of the compared techniques			
		CDE	**HM**	**SR**	**ASCHEA**
g01	**−15**	−14.999993	−14.6154	−15.000	NA
g02	**0.803619**	0.590908	0.79119	0.726288	NA
g03	**1**	0.639920	0.9978	1.000	NA
g04	**−30665.539**	**−30665.538672**	−30645.9	−30665.539	NA
g05	**5126.498**	5327.390497	–	5142.472	NA
g06	**−6961.814**	**−6961.813876**	−5473.9	−6350.262	NA
g07	**24.306**	**24.306212**	25.069	24.642	NA
g08	**0.095825**	**0.095825**	0.0291438	0.095825	NA
g09	**680.63**	**680.630057**	683.18	680.763	NA
g10	**7049.25**	**7049.248480**	9659.3	8835.655	NA
g11	**0.75**	0.796455	0.75	0.750	NA
g12	**1.00**	**1.000000**	0.991950	1.000000	NA
g13	**0.053950**	0.392100	NA	0.216915	NA

c)

Table 2: Comparison of the best (a), mean (b), and worst (c) results of the cultured differential evolution (CDE) with respect to the Homomorphous Maps (HM) [13], Stochastic Ranking (SR) [23] and ASCHEA [9]. "-" means no feasible solutions were found. NA = Not Available. An result in bold font means that our approach obtained the same or a better value than any other of the techniques.

IOP Publishing Ltd. and Oxford University Press, 1997.

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

[3] 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.

[4] C.-J. Chung and R. 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.

[5] C. A. Coello Coello and R. Landa Becerra. Adding knowledge and efficient 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.

[6] C. A. Coello Coello and R. Landa Becerra. Efficient Evolutionary Optimization through the use of a Cultural Algorithm. *Engineering Optimization*, 36(2):219–236, April 2004.

[7] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

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

[9] S. B. Hamida and M. Schoenauer. ASCHEA: New Results Using Adaptive Segregational Constraint Handling. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 1, pages 884–889, Piscataway, New Jersey, May 2002. IEEE Service Center.

[10] 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.

[11] 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.

[12] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.

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

[14] J. Lampinen. Solving Problems Subject to Multiple Nonlinear Constraints by the Differential Evolution. In R. M. . P. Osmera, editor, *Proceedings of MENDEL 2001, 7th International Conference on Soft Computing*, pages 50–57, June 2001.

[15] J. Lampinen. A Constraint Handling Approach for the Diifferential Evolution Algorithm. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 2, pages 1468–1473, Piscataway, New Jersey, May 2002. IEEE Service Center.

[16] Y.-C. Lin, K.-S. Hwang, and F.-S. Wang. Hybrid Differential Evolution with Multiplier Updating Method for Nonlinear Constrained Optimization. In *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002)*, volume 1, pages 872–877, Piscataway, New Jersey, May 2002. IEEE Service Center.

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

[18] I. C. Parmee. *Evolutionary and Adaptive Computing in Engineering Design*. Springer, London, 2001. ISBN 1-85233-029-5.

[19] 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.

[20] 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 Scientific, River Edge, New Jersey, 1994.

[21] 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.

[22] 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.

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

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

[25] R. Storn. On the Usage of Differential Evolution for Function Optimization. In *1996 Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS 1996)*, pages 519–523, Berkeley, 1996. IEEE.

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