A Cultural Algorithm with Differential Evolution to Solve Constrained Optimization Problems

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. A cultural algorithm for constrained optimization is proposed in this paper. The main novel feature of this approach is the use of differential evolution as a population space. Differential evolution has been found to be very effective when dealing with real valued optimization problems. The knowledge sources contained in the belief space of the cultural algorithm are specifically designed according to the differential evolution population. Furthermore, we introduce an influence function that selects the source of knowledge to apply the evolutionary operators. Such influence function considerably improves the performance when compared to a previous version of the algorithm (developed by the same authors). We use a well-known set of test functions to validate the approach, and compare the results with respect to the best constraint-handling technique known to date in evolutionary optimization.

1 Introduction

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 [1]. 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 [2–5]. Cultural algorithms are made of two main components: the population space, and the belief space [6]. 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. 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.

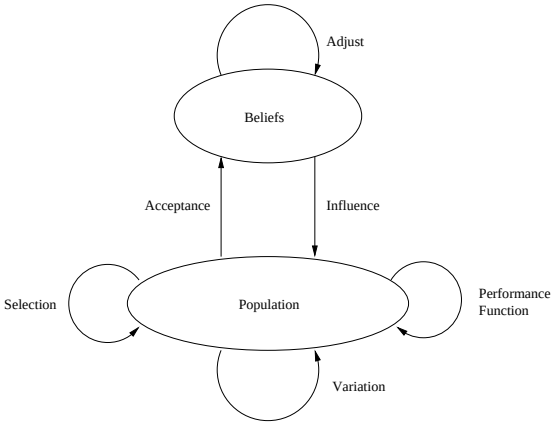

Fig. 1. Spaces of a cultural algorithm

Originally, when cultural algorithms were applied to real parameter optimization, genetic algorithms were used as a population space [1]. Later on, evolutionary programming appeared as a better choice [8] for the population space than genetic algorithms when dealing with constrained search spaces [9, 4, 3]. Recently, particle swarm [10] has also been proposed as a population space [11], turning the direction to use new evolutionary methods with better performance in real parameter optimization.

Differential evolution [12] 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 [13]. 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 [14]. This paper precisely presents an extension of our previous work in which we did a preliminary exploration of the potential of differential evolution to be "culturized" [14].

2 Previous Work

Reynolds et al. [2] and Chung & Reynolds [3] 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. In more recent work, Jin and Reynolds [4] proposed an n-dimensional regional-based schema. 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 [5]. Saleem adds history and domain knowledge to Chung's situational and normative knowledge, and Jin's topographical knowledge. In [11], Iacoban et al. change the evolutionary programming algorithm from

the population space for a particle swarm optimizer [10]. 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.

3 Differential Evolution

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

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

Differential evolution is an evolutionary algorithm proposed by Price and Storn [12], 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 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 is also rather efficient 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 [12]. That is the reason why we adopted it for the work reported in this paper.

4 Our Proposed Approach

The proposed approach uses differential evolution in the population space. A pseudo-code of the 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
```

Fig. 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 children generation, the variation operator of the differential evolution algorithm is influenced by the belief space. 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, described next.

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. 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. 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.

Normative Knowledge The normative knowledge contains the intervals for the decision variables where good solutions have been found, in order to move new solutions

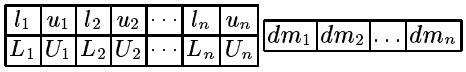

Fig. 4. Structure of the normative knowledge

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 a scaling factor, dm_i, to influence the mutation operator adopted in differential evolution. The following expression shows the influence of the normative knowledge on the variation operators:

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

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.

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 [15]. 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).

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 stored and the new reference solution (i.e., the new solution considered as the "best" found so far).

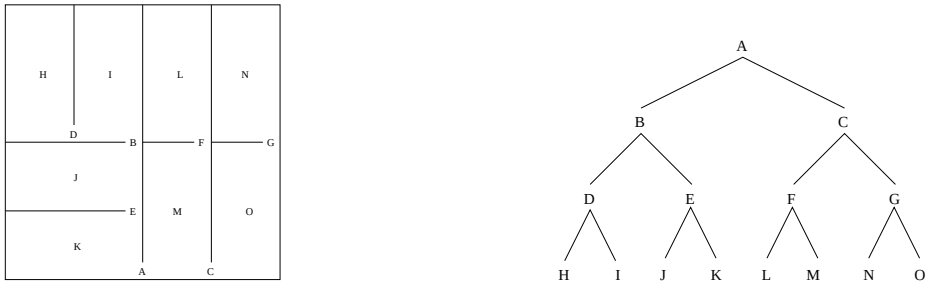

Fig. 5. Example of the partition of a two dimensional space by a k-d tree

History Knowledge This knowledge source was originally proposed for dynamic objective functions, and it was used to find patterns in the environmental changes. 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 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.

e_1	$\cdots$	e_i	$\cdots$	e_w

ds_1	ds_2	$\cdots$	ds_n
dr_1	dr_2	$\cdots$	dr_n

Fig. 6. Structure of the history knowledge

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

$$x'_{i,j} = \begin{cases} x_{i,e_w} + dr_i * F * |x_{i,r1} - x_{i,r2}| & \text{if } rand(0,1) < \alpha \\ x_{i,e_w} + \frac{1,5 * ds_i}{dm_i} * (x_{i,r1} - x_{i,r2}) & \text{if } rand(0,1) < \beta \\ rand(lb_i, ub_i) & \text{otherwise} \end{cases}$$

where x_{i,e_w} is the i-th decision variable of the previous best e_w 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:

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

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

where the function $sgn(a)$ 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 by Saleem [5]. The number of accepted individuals decreases while the number of generation increases. Saleem [5] 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, in $(0, 1]$; Saleem [5] 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.

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 the value $\%p$ is 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. This main influence function is the most important modification with respect to the previous version of this algorithm ([14]).

5 Comparison of Results

To validate our approach, we adopted the well-known benchmark included in [16] which has been often used in the literature to validate new constraint-handling techniques. For a full description of the test functions adopted, the reader should refer to [16]. 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 $b = 10$,

Table 1. Results obtained by our cultured differential evolution approach

TF	Optimal	Best	Mean	Worst	Std Dev
g01	**-15**	-14.999863	-14.999351	-14.998283	0.000333
g02	**0.803619**	0.793829	0.735590	0.620843	0.049941
g03	**1**	**1.000000**	0.896800	0.69272	0.080994
g04	**-30665.539**	**-30665.538672**	**-30665.538672**	**-30665.538672**	0.000000
g05	**5126.4981**	5126.558552	5198.202774	5323.865946	59.633275
g06	**-6961.8138**	**-6961.813876**	**-6961.813876**	**-6961.813876**	0.000000
g07	**24.3062091**	24.575518	24.575520	24.575526	0.000002
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**	**7049.248134**	7049.248489	7049.249942	0.000362
g11	**0.75**	**0.750000**	0.777469	0.898055	0.044560
g12	**1**	**1.000000**	**1.000000**	**1.000000**	0.000000
g13	**0.0539498**	0.056180	0.288324	0.39210	0.161230

the size of the list in the history knowledge $w = 5$, $\alpha = \beta = 0.45$, and $\%p = 0.2$. The values shown in tables were obtained executing 30 independent runs per problem.

Table 1 shows the results obtained by our approach. The results obtained by the stochastic ranking method (the best constraint-handling technique proposed for evolutionary algorithms known to date) are shown in Table 3. Our results are also compared to a previous version of our algorithm [14] in Table 2. The results of Runarsson and Yao were obtained with 350,000 evaluations of the fitness function. Our approach required only 100,100 evaluations (in both versions, the previous [14] and the one reported in this paper).

Table 2. Results obtained by our previous version of cultured differential evolution [14]

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

As can be seen in Tables 1 and 2, the new version of our algorithm exhibits a better performance than our previous version in all cases, except in g05, where its variablility is higher. The current version now reaches the optimum in g11, and also consistently reaches the optimum of g04. When compared to stochastic ranking (see Table 3), our cultured differential evolution algorithm turns out to be very competitive. Our approach

reached the global optimum in eight problems, and stochastic ranking did it in nine. However, with the exception of g02 and g13 (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, improving on the robustness of stochastic ranking in several cases. 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, g03, g05 and g11.

Table 3. Results reported for stochastic ranking [16]

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	**7049.25**	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

6 Conclusions and Future Work

In this paper we introduce a cultural algorithm, which uses differential evolution. This work improves on our previous attempt to develop a cultural algorithm that uses a differential evolution-based population [14]. The approach is applied to solve constrained optimization problems. Adding a belief space to the differential evolution algorithm, we were able to get a low computational cost while obtaining competitive results on a well-known benchmark adopted for evolutionary optimization. The main weakness of this approach is its apparent loss of diversity, due to its high selection pressure. Some of the knowledge sources of the belief space are designed to provide diversity, but more work remains to be done in this sense (as can be seen from the results obtained for g02). Some other directions of future work are the analysis of the impact of each knowledge source during the evolutionary process; or a comparison of different types of acceptance functions, which may allow a better exploration of the fitness landscape.

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.

References

1. Reynolds, R.G.: An Introduction to Cultural Algorithms. In Sebald, A.V., Fogel, L.J., eds.: Proceedings of the Third Annual Conference on Evolutionary Programming. World Scientific, River Edge, New Jersey (1994) 131–139
2. Reynolds, R.G., Michalewicz, Z., Cavaretta, M.: Using cultural algorithms for constraint handling in GENOCOP. In McDonnell, J.R., Reynolds, R.G., Fogel, D.B., eds.: Proceedings of the Fourth Annual Conference on Evolutionary Programming. MIT Press, Cambridge, Massachusetts (1995) 298–305
3. Chung, C.J., Reynolds, R.G.: CAEP: An Evolution-based Tool for Real-Valued Function Optimization using Cultural Algorithms. Journal on Artificial Intelligence Tools **7** (1998) 239–292
4. Jin, X., Reynolds, R.G.: Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach. In: 1999 Congress on Evolutionary Computation, Washington, D.C., IEEE Service Center (1999) 1672–1678
5. Saleem, S.M.: Knowledge-Based Solution to Dynamic Optimization Problems using Cultural Algorithms. PhD thesis, Wayne State University, Detroit, Michigan (2001)
6. Reynolds, R.G.: Cultural algorithms: Theory and applications. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw-Hill, London, UK (1999) 367–377
7. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading, Massachusetts (1989)
8. Chung, C.J., Reynolds, R.G.: A Testbed for Solving Optimization Problems using Cultural Algorithms. In Fogel, L.J., Angeline, P.J., Bäck, T., eds.: Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming, Cambridge, Massachusetts, MIT Press (1996)
9. Coello Coello, C.A., Landa Becerra, R.: Adding knowledge and efficient data structures to evolutionary programming: A cultural algorithm for constrained optimization. In et al., E.C.P., ed.: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002), San Francisco, California, Morgan Kaufmann Publishers (2002) 201–209
10. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco, California (2001)
11. Iacoban, R., Reynolds, R.G., Brewster, J.: Cultural Swarms: Modeling the Impact of Culture on Social Interaction and Problem Solving. In: 2003 IEEE Swarm Intelligence Symposium Proceedings, Indianapolis, Indiana, USA, IEEE Service Center (2003) 205–211
12. Price, K.V.: An introduction to differential evolution. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw-Hill, London, UK (1999) 79–108
13. Storn, R.: System Design by Constraint Adaptation and Differential Evolution. IEEE Transactions on Evolutionary Computation **3** (1999) 22–34
14. Landa Becerra, R., Coello Coello, C.A.: Culturizing differential evolution for constrained optimization. In: ENC'2004, IEEE Service Center (2004) Submitted.
15. Bentley, J.L., Friedman, J.H.: Data Structures for Range Searching. ACM Computing Surveys **11** (1979) 397–409
16. Runarsson, T.P., Yao, X.: Stochastic Ranking for Constrained Evolutionary Optimization. IEEE Transactions on Evolutionary Computation **4** (2000) 284–294