A Cultural Algorithm for Constrained Optimization

Carlos A. Coello Coello and Ricardo Landa Becerra

CINVESTAV-IPN
Departamento de Ingeniería Eléctrica
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México, D.F. 07300, MEXICO
ccoello@cs.cinvestav.mx
rlanda@computacion.cs.cinvestav.mx

Abstract. In this paper, we propose a technique that exploits knowledge extracted during the search to improve the performance of an evolutionary algorithm used for global optimization. The approach is based on a cultural algorithm combined with evolutionary programming and we show that produces highly competitive results at a relatively low computational cost.

1 Introduction

Evolutionary Computation (EC) techniques have become increasingly popular in the last few years. This popularity is mainly due to the fact that EC techniques have been able to successfully solve a wide variety of complex problems [7].

However, EC techniques are normally used as "blind heuristics" in the sense that no specific domain knowledge is used or required. However, several researchers have proposed different mechanisms to extract knowledge (or certain design patterns) from an evolutionary algorithm to improve convergence of another one (e.g., [20]).

In this paper, we propose the use of a biological metaphor called "cultural algorithm" as a global optimization technique. Cultural algorithms are based on the following notion: in advanced societies, the improvement of individuals occurs beyond natural selection; besides the information that an individual posseses within his genetic code (inherited from his ancestors) there is another component called "culture". Culture can be seen as a sort of repository where individuals place the information acquired after years of experience. When a new individual has access to this library of information, it can learn things even when it has not experienced them directly. Humankind as a whole has reached its current degree of progress mainly due to culture.

2 Notions of Cultural Algorithms

Some social researchers have suggested that culture might be symbolically encoded and transmitted within and between populations, as another inheritance mechanism [5, 16]. Using this idea, Reynolds [17] developed a computational model in which cultural evolution is seen as an inheritance process that operates at two levels: the micro-evolutionary and the macro-evolutionary levels[1].

At the micro-evolutionary level, individuals are described in terms of "behavioral traits" (which could be socially acceptable or unacceptable). These behavioral traits are passed from generation to generation using several socially motivated operators. At the macro-evolutionary level, individuals are able to generate "mappa" [16], or generalized descriptions of their experiences. Individual mappa can be merged and modified to form "group mappa" using a set of generic or problem specific operators. Both levels share a communication link. Reynolds [17] proposed the use of genetic algorithms to model the micro-evolutionary process, and Version Spaces [15] to model the macro-evolutionary process of a cultural algorithm.

The main idea behind this approach is to preserve beliefs that are socially accepted and discard (or prune) unacceptable beliefs. Therefore, if we apply a cultural algorithm for global optimization, then acceptable beliefs can be seen as constraints that direct the population at the micro-evolutionary level [12].

3 Previous Work

Reynolds et al. [18] and Chung & Reynolds [2] have explored the use of cultural algorithms for global optimization with very encouraging results. Chung and Reynolds [2] use a hybrid of evolutionary programming and GENOCOP [13] in which they incorporate an interval constraint-network [3] 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 [8] 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. This *belief-cell* can be used to guide the search of an EC 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

[1] Note that other researchers have also proposed the idea of using culture to improve the performance of an evolutionary algorithm (see for example [19]).

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 [10]. This map is built using information derived from evaluating the constraints of each individual in the population of the EC technique. The map is formed by dividing the search space in sub-areas called *cells*. Each cell can be classified as: feasible (if it lies completely on a feasible region), infeasible (if it lies completely on an infeasible region), semi-feasible (if it occupies part of the feasible and part of the infeasible regions), or unknown (if that region has not been explored yet). This map is used to derive rules about how to guide the search of the EA (avoiding infeasible regions and promoting the exploration of feasible regions).

4 Constrained Optimization

In this paper, we use cultural algorithms with evolutionary programming (CAEP) [2]. The basic idea is to "influence" the mutation operator (the only operator in evolutionary programming) so that current knowledge about the properties of the search space can be properly exploited.

In a cultural algorithm there are two main spaces: the normal population adopted with evolutionary programming and the belief space. The shared acquired knowledge is stored in the belief space during the evolution of the population. The interactions between these two spaces are described as follows [2]:

1. Select an initial population of p candidate solutions, from a uniform distribution within the given domain for each parameter from 1 to n.
2. Assess the performance score of each parent solution by a given objective function f.
3. Initialize the belief space with the given problem domain and candidate solutions.
4. Generate p new offspring solutions by applying a variation operator, V as modified by the influence function, *Influence*. Now there are $2p$ solutions in the population.
5. Assess the performance score of each offspring solutions by the given objective function f.
6. For each individual, select c competitors at random from the population of size $2p$. Conduct pairwise competitions between the individual and the competitors.
7. Select the p solutions that have the greatest number of wins to be parents for the next generation.
8. Update the belief space by accepting individuals using the acceptance function.
9. Go back to step 4 unless the available execution time is exhausted or an acceptable solution has been discovered.

Most of the steps previously described are the same as in evolutionary programming [6]. The acceptance function accepts those individuals that can contribute with their knowledge to the belief space. The update function creates the new belief space with the beliefs of the accepted individuals. The idea is to add to the current knowledge the new knowledge acquired by the accepted individuals.

The function to generate offspring used in evolutionary programming is modified so that it includes the influence of the knowledge space in the generation of offspring. Evolutionary programming uses only mutation and the influence function indicates the most promising mutation direction. The remaining steps are the same used in evolutionary programming.

For unconstrained problems, Chung [1] proposes the use of two types of knowledge: (1) situational, which provides the exact point where the best individual of each generation was found; and (2) normative, which stores intervals for the decision variables of the problem, in the regions where good results were found.

5 Beliefs as Constraints

As we mentioned before, Jin and Reynolds [8] modified Chung's proposal as to include in the belief space information about feasibility of the solutions. We will explain next the changes performed in more detail, since our current proposal is an extension of Jin & Reynolds' algorithm.

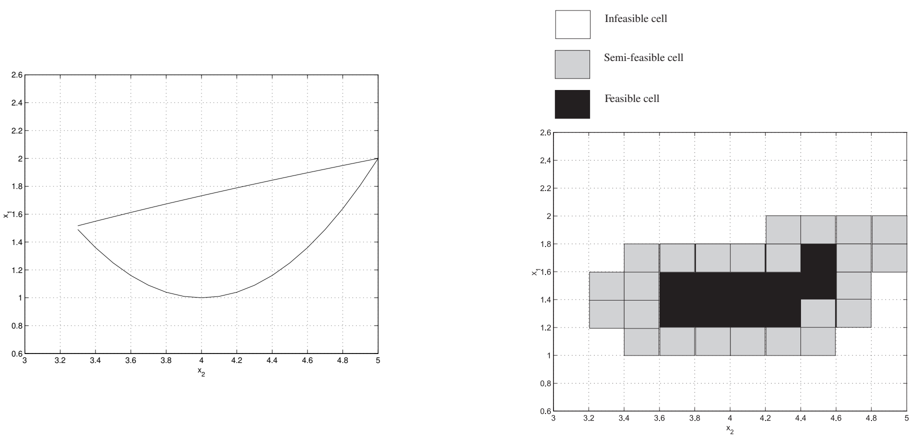

Fig. 1. The figure at the left illustrates the feasible region of a problem. In this case, the lines represent the boundary between the feasible and infeasible regions. The figure at the right illustrates the representation of the constraints part of the belief space for the search space of the same problem.

First, Jin and Reynolds eliminated the situational knowledge and added constraints knowledge. Taking advantage of the intervals of good solutions that are stored in the normative portion of the belief space, they created what they called "belief cells". These belief cells are a subdivision of the intervals of good solutions, such that feasibility of the cells can be determined. When the intervals of the variables are modified, the cells are also modified. As indicated before, there are 4 types of cells (see Figure 1)[2]: (1) feasible, (2) infeasible, (3) semi-feasible (contain part of both areas) and (4) unknown.

The influence that the belief space has on the generation of offspring consists of moving individuals that lie on infeasible cells towards feasible cells. Actually, in this process, semi-feasible cells are given preference because in most difficult constrained problems, the optimum lies on the boundary between the feasible and infeasible regions. However, Jin & Reynolds [8] do not modify the rules used to update the normative part of the belief space proposed by Chung [1]: the intervals are expanded if the accepted individuals do not fit within them; conversely, they are tightened only if the accepted individuals have a better fitness. This may reduce the intervals towards infeasible regions in which the objective function values are higher.

6 Proposed Approach

The approach proposed here is a variation of Jin & Reynolds' technique [8]. Next we will describe the main differences between traditional evolutionary programming and our approach.

6.1 Initialization of the Belief Space

The lower and upper boundaries (called l and u, respectively) of the promising intervals for each variable are stored in the normative part of the belief space, together with the fitness for each extreme of the interval (called L and U, respectively). This part is initialized putting in the boundaries of the variables the values given in the input data of the problem. The initial fitnesses in all cases are $+\infty$[3].

Regarding the constraints of the problem, the interval given in the normative part is subdivided into s subintervals such that a portion of the search space is divided in hypercubes. The following information of each hypercube is stored: number of feasible individuals (within that cell), number of infeasible individuals (within that cell), and the type of region. The type of region depends on the feasibility of the individuals within. Four types are defined:

- **if** $feasible\ individuals = 0$ **and** $infeasible\ individuals = 0$, **then** *cell type* $=$ *unknown*

[2] Other authors have also proposed the use of a map of the feasible region. See, for example, [11].

[3] This is assuming a minimization problem.

- **if** $feasible\ individuals > 0$ **and** $infeasible\ individuals = 0$, **then** $cell\ type = feasible$
- **if** $feasible\ individuals = 0$ **and** $infeasible\ individuals > 0$, **then** $cell\ type = infeasible$
- **if** $feasible\ individuals > 0$ **and** $infeasible\ individuals > 0$, **then** $cell\ type = semi-feasible$

To initialize this part, all counters are set to zero and the cell type is initialized to "unknown" (other values could be used in this case, but that would obviously affect the performance of the algorithm).

6.2 Updating the Belief Space

The constraints part of the belief space is updated at each generation, whereas the normative part is updated every k generations. The update of the constraints part consists only of adding any new individuals that fall into each region to the counter of feasible individuals. The update of the normative part is more complex (that is the reason why it is not performed at every generation). When the interval of each variable is updated, the cells or hypercubes of the restrictions part are changed and the counters of feasible and infeasible individuals are reinitialized. Furthermore, this update is done taking into consideration only a portion of the population. Such a portion is selected by the function *accept()*, taking as a parameter (given by the user) the percentage of the total population size to be used.

In the approach proposed in this paper, the conditions to reduce the intervals are stronger: an interval is reduced only if the accepted individual has a better fitness AND it is feasible. In order to make this mechanism work, it is necessary to modify the acceptance function so that feasible individuals are preferred and fitness is adopted as a secondary criterion. If this is not done, then the condition for interval reduction will not hold most of the time because the accepted individuals are more likely to be infeasible.

6.3 Influence of Beliefs in the Mutation Operator

Mutation takes place for each variable of each individual, with the influence of the belief space and in accordance with the following rules:

- If the variable j of the parent is outside the interval given by the normative part of the constraints, then we attempt to move within such interval through the use of a random variable.
- If the variable is within a feasible, a semi-feasible or an unknown hypercube, the perturbation is done trying to place it within the same hypercube or very close to it.
- Finally, if the variable is in an infeasible cell, we try to move it first to the closest semi-feasible cell. However, if none is found, we try to move it to the feasible or unknown closest cell. If that does not work either, then we move it to a random position within the interval defined by the normative part.

6.4 Tournament Selection

The rules for updating the belief space may produce that knowledge becomes specialized at a slower rate. To improve the speed of the algorithm, we take advantage of the rules for performing tournament selection. After performing mutation, we will have a population of size $2p$ (p parents generate p children). Tournament is performed considering the entire population (i.e., we use $(\mu + \lambda)$ selection). Tournaments consist of c confrontations per individual, with the c opponents randomly chosen from the entire population. When the tournaments finish, the p individuals with the larger number of victories are selected to form the following generation. The tournament rules adopted for the current proposal are very similar to those adopted by Deb in his penalty approach based on feasibility [4]. However, unlike Deb's approach, in our case, we never add violated constraints (as normally done with penalty-based approaches).

The new tournament rules adopted by our approach are the following:

1. If both individuals are feasible, or both are infeasible, then the individual with the best fitness value wins.
2. Otherwise, the feasible individual always wins.

7 Comparison of Results

To validate our approach, we have used the well-known benchmark proposed in [14] which has been often used in the literature to validate new constraint-handling techniques. The specific test functions used are the following:

1. **g04**:
   Minimize:
   $$f(\boldsymbol{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \tag{1}$$
   subject to:
   $$\begin{aligned} g_1(\boldsymbol{x}) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0 \\ g_2(\boldsymbol{x}) &= -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0 \\ g_3(\boldsymbol{x}) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0 \\ g_4(\boldsymbol{x}) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0 \\ g_5(\boldsymbol{x}) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0 \\ g_6(\boldsymbol{x}) &= -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0 \end{aligned}$$
   where: $78 \le x_1 \le 102$, $33 \le x_2 \le 45$, $27 \le x_i \le 45$ $(i = 3, 4, 5)$.
2. **g08**
   Minimize:
   $$f(\boldsymbol{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)} \tag{2}$$

subject to:

$$g_1(\boldsymbol{x}) = x_1^2 - x_2 + 1 \leq 0$$
$$g_2(\boldsymbol{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

where $0 \leq x_1 \leq 10$, $0 \leq x_2 \leq 10$.

3. **g11**
   Minimize:
   $$f(\boldsymbol{x}) = x_1^2 + (x_2 - 1)^2 \tag{3}$$
   subject to:
   $$h(\boldsymbol{x}) = x_2 - x_1^2 = 0$$
   where: $-1 \leq x_1 \leq 1$, $-1 \leq x_2 \leq 1$.
4. **g12**
   Maximize:
   $$f(\boldsymbol{x}) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100 \tag{4}$$
   subject to:
   $$g(\boldsymbol{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$
   where: $0 \leq x_1 \leq 10$ $(i = 1, 2, 3)$, and $p, q, r = 1, 2, \ldots, 9$. The feasible region of the search space consists of 9^3 disjointed spheres. A point (x_1, x_2, x_3) is feasible if and only if there exist p, q, r such that the above inequality holds.

The parameters used by our approach are the following: population size = 20, maximum number of generations = 2500, the belief space is updated at every 20 generations, tournaments consist of 10 encounters by individual (half the population size). These parameters were derived empirically after numerous experiments.

		Best Result		**Mean Result**		**Worst Result**	
TF	**optimal**	**CAEP**	**KM**	**CAEP**	**KM**	**CAEP**	**KM**
g04	-30665.539	-30664.8	-30664.5	-30611.1	-30655.3	-30466.8	-30645.9
g08	-0.095825	-0.095825	-0.095825	-0.09525552	-0.0891568	-0.0901302	-0.0291438
g11	0.750	0.7402695	0.75	0.79299844	0.75	0.8380483	0.75
g12	1.000	1.000	0.999999857	0.99725459	0.999134613	0.9863316	0.991950498

Table 1. Comparison of the results for the test functions selected from [14].

Our results are compared to the homomorphous maps of Koziel & Michalewicz [9] (one of the best current constraint-handling techniques for evolutionary algorithms) in Table 1. The results of Koziel and Michalewicz were obtained with

1,400,000 fitness function evaluations, whereas our approach required only 50,020 fitness function evaluations.

As can be seen in Table 1, our approach produces very good results with respect to the homomorphous maps (which is considerably more difficult to implement) at a fraction of its computational cost. The main reason for this cost reduction is that the belief cells are used to guide the search of the evolutionary algorithm very efficiently, avoiding that it moves to unpromising regions of the search space.

8 Conclusions and Future Work

We have presented an approach based on cultural algorithms and evolutionary programming for constrained optimization. The approach has provided good results at a relatively low computational cost. This suggests that the proper use of domain knowledge may certainly improve the performance of an evolutionary algorithm when this is properly done. Also, it suggests that such domain knowledge may be extracted during the evolutionary process in which we aim to reach the global optimum of a problem. This contrasts with the more conventional approach of using domain knowledge extracted from previous runs of an evolutionary algorithm.

The proposed technique is relatively simple to implement, but it still requires some further implementation refinements. Some of our future work precisely involves the use of spatial data structures to store the belief cells. This should lower the memory usage of the approach which is currently its main drawback.

9 Acknowledgements

The first author acknowledges support from CONACyT through project No. 32999-A. The second author acknowledges support from CONACyT through a scholarship to pursue graduate studies at CINVESTAV-IPN.

References

1. Chan-Jin Chung. *Knowledge-Based Approaches to Self-Adaptation in Cultural Algorithms*. PhD thesis, Wayne State University, Detroit, Michigan, 1997.
2. Chan-Jin Chung and Robert G. Reynolds. A Testbed for Solving Optimization Problems using Cultural Algorithms. In Lawrence J. Fogel, Peter J. Angeline, and Thomas Bäck, editors, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, Cambridge, Massachusetts, 1996. MIT Press.
3. Ernest Davis. Constraint propagation with interval labels. *Artificial Intelligence*, 32:281–331, 1987.
4. Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.

5. W. H. Durham. *Co-evolution: Genes, Culture, and Human Diversity.* Stanford University Press, Stanford, California, 1994.
6. Lawrence J. Fogel. *Artificial Intelligence through Simulated Evolution. Forty Years of Evolutionary Programming.* John Wiley & Sons, Inc., New York, 1999.
7. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
8. Xidong Jin and Robert G. Reynolds. Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach. In *1999 Congress on Evolutionary Computation*, pages 1672–1678, Washington, D.C., July 1999. IEEE Service Center.
9. Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.
10. Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Norwell, Massachusetts, 1993.
11. Carlos E. Mariano and Eduardo F. Morales. Distributed Reinforcement Learning for Multiple Objective Optimization Problems. In *2000 Congress on Evolutionary Computation*, volume 1, pages 188–195, Piscataway, New Jersey, July 2000. IEEE Service Center.
12. Zbigniew Michalewicz. A Survey of Constraint Handling Techniques in Evolutionary Computation Methods. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pages 135–155. The MIT Press, Cambridge, Massachusetts, 1995.
13. Zbigniew Michalewicz and Cezary Z. Janikow. Handling Constraints in Genetic Algorithms. In R. K. Belew and L. B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 151–157, San Mateo, California, 1991. Morgan Kaufmann Publishers.
14. Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.
15. Tom Mitchell. *Version Spaces: An Approach to Concept Learning.* PhD thesis, Computer Science Department, Stanford University, Stanford, California, 1978.
16. A. C. Renfrew. Dynamic Modeling in Archaeology: What, When, and Where? In S. E. van der Leeuw, editor, *Dynamical Modeling and the Study of Change in Archaelogy.* Edinburgh University Press, Edinburgh, Scotland, 1994.
17. Robert G. Reynolds. An Introduction to Cultural Algorithms. In A. V. Sebald and L. J. Fogel, editors, *Proceedings of the Third Annual Conference on Evolutionary Programming*, pages 131–139. World Scientific, River Edge, New Jersey, 1994.
18. Robert G. Reynolds, Zbigniew Michalewicz, and M. Cavaretta. Using cultural algorithms for constraint handling in GENOCOP. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, pages 298–305. MIT Press, Cambridge, Massachusetts, 1995.
19. Lee Spector and Sean Luke. Cultural Transmission of Information in Genetic Programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 209–214, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
20. Zhiming Zhang and T. Warren Liao. Combining Case-Based Reasoning with Genetic Algorithms. In Scott Brave and Annie S. Wu, editors, *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, pages 305–310, Orlando, Florida, 1999.