Available online at www.sciencedirect.com

SCIENCE DIRECT®

Comput. Methods Appl. Mech. Engrg. 195 (2006) 4303–4322

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Cultured differential evolution for constrained optimization

Ricardo Landa Becerra, Carlos A. Coello Coello *

Evolutionary Computation Group (EVOCINV), CINVESTAV-IPN, Computer Science Section, Electrical Engineering Department, Av. IPN No. 2508, Col. San Pedro Zacatenco, México, D.F. 07360, Mexico

Received 27 October 2004; received in revised form 31 August 2005; accepted 1 September 2005

Abstract

A cultural algorithm with a differential evolution population is proposed in this paper. This cultural algorithm uses different knowledge sources to influence the variation operator of the differential evolution algorithm, in order to reduce the number of fitness function evaluations required to obtain competitive results. Comparisons are provided with respect to three techniques that are representative of the state-of-the-art in the area. The results obtained by our algorithm are similar (in quality) to those obtained by the other approaches with respect to which it was compared. However, our approach requires a lower number of fitness function evaluations than the others.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Evolutionary algorithms; Optimization; Differential evolution; Cultural algorithms; Evolutionary optimization

1. Introduction

Evolutionary computation techniques have been successfully applied in many types of problems, mainly related to optimization tasks [1,30]. However, evolutionary algorithms are considered "blind heuristics" because they do not normally require information about the problem but only the relative assessment of the quality of the solutions produced (through the so-called "fitness function"). This fact is also the reason of their generality, because we do not need to know any feature of the problem to solve, only to assign the fitness of any given solution.

The use of domain knowledge within an evolutionary algorithm has been successfully used to improve its performance when limiting the use of such evolutionary algorithm to a certain specific problem or group of problems [19]. By incorporating domain-specific knowledge, we obviously remove generality to the evolutionary algorithm, because such domain knowledge can be extracted and adopted only when dealing with the specific problem at hand. However, at the same time, we are replacing some of the stochastic nature of the evolutionary algorithm by deterministic information. This extra information will increase the selection pressure and will normally accelerate convergence (note, however, that this improved convergence rate may also degenerate into premature convergence, when reaching a local optimum instead of the global optimum). It is worth noticing that the incorporation of domain-specific knowledge into an evolutionary algorithm is one of the choices that have been suggested [12] to circumvent the limitations imposed by the famous "No Free Lunch" theorem, which roughly states that all heuristics are equally efficient when measuring their performance over all possible problems [39].

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.

* Corresponding author. Tel.: +52 55 5061 3800x6564; fax: +52 55 5061 3757.
E-mail addresses: rlanda@computacion.cs.cinvestav.mx (R. Landa Becerra), ccoello@cs.cinvestav.mx (C. A. C. Coello).

0045-7825/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2005.09.006

Evolutionary algorithms have been very successfully applied to constrained optimization problems [6], mainly due to some features such as the following:

- They are population-based techniques, which makes them less prone to get trapped in local optima.
- They are less susceptible to the characteristics of the function to be optimized than traditional optimization techniques (e.g., they can deal with non-differentiable objective functions).
- They are very easy to implement and use, which has contributed to make them very popular in a wide variety of engineering and scientific disciplines.

Despite the considerable amount of work on the use of evolutionary algorithms for constrained optimization, there are relatively few researchers that have devised mechanisms to incorporate knowledge from the problem in order to guide an evolutionary algorithm. As indicated before, by adding domain knowledge, we aim to accelerate the convergence of an evolutionary algorithm. The motivation of the work reported in this paper was precisely to provide a mechanism that can extract knowledge from the problem during the evolutionary process, such that we can accelerate its convergence. Such a mechanism is developed using as a basis a cultural algorithm.

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 plus two engineering optimization problems. 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 [32]. 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 [34,5,18,36].

Cultural algorithms are made of two main components: the population space, and the belief space [33]. 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 [14]. 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 (the influence function). Those interactions are depicted in Fig. 1.

Originally, when cultural algorithms were applied to real parameter optimization, genetic algorithms were used as a population space [32]. Later on, evolutionary programming appeared as a better choice [4] 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 [5,18,7], until the advent of particle swarm optimization [20] in [17],

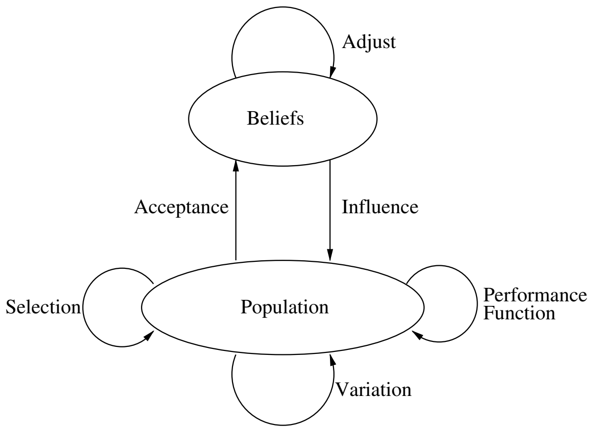

Fig. 1. Spaces of a cultural algorithm.

which shred light regarding the potential use of new evolutionary methods with better performance in real parameter optimization.

Differential evolution [31] is a recently developed evolutionary algorithm, focused on the solution of real parameter optimization problems. Differential evolution has been found to be a very robust optimization technique [38]. 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. [34] and Chung and Reynolds [5] 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 [27] in which they incorporate an interval constraint-network to represent the constraints of the problem at hand.

In [5], 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 [18] proposed an n-dimensional regional-based scheme, 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 [36], 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 [7,8], 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 [17], Iacoban et al. change the evolutionary programming algorithm of the population space for a particle swarm optimizer [20]. 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 [31], 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 evolutionary algorithms, in which the mutation operator is defined by a probability function.

The basic algorithm of differential evolution is shown in Fig. 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 ith decision variable of the jth 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 (as a matter of fact, the

```
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 $i$
            $x'_{i,j} = \begin{cases} x_{i,r3} + F * (x_{i,r1} - x_{i,r2}) & \text{if } rand_i(0,1) < CR \text{ or } i = i_{rand} \\ x_{i,j} & \text{otherwise} \end{cases}$
        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 algorithm is also capable of self-adapting both the step sizes and the step direction, since they both depend of the current solutions in the population). Furthermore, when adopting $\mathrm{CR} = 1$, the variation operator is rotationally invariant [31], which means that the differential evolution algorithm supports self-adaptation at the same level that an evolution strategy with covariance matrices to self-adapt its standard deviations (this is the most complete self-adaptation scheme available in evolution strategies).

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 Fig. 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 [31]. 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, as we will note later on.

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 [38], in which all the constraints of the problem at hand are relaxed, so that all 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 [25]. 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 (this comparison is made in the Pareto sense in the constraint violation space).

The rest of the differential evolution algorithm remains the same. The experiments are done with 10 of the 11 test functions proposed in [21]. Some previous versions of this algorithm appeared in [24,23], where the replacement rules were not as complete as in [25], and less test problems were taken into account.

A previously proposed algorithm which uses a very similar set of rules, even when it is not based in differential evolution, is the one proposed by Deb [10]. As in the case of Lampinen, this approach does not need any extra parameter for the constraint-handling mechanism, but the main search engine in this case is a genetic algorithm. A key difference between the set of rules of both algorithms is the comparison for the case of two infeasible individuals: while Lampinen makes a comparison in the Pareto sense, Deb sums all the constraint violations and compares a single value. Also, note that Deb's approach requires niching [11] to maintain diversity.

Simultaneously to Lampinen, Lin et al. [26] 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. [26] 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 Fig. 3.[1]

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.

[1] A public-domain implementation of our approach (in the C programming language under Linux) is available for download at: http://www.cs.cinvestav.mx/~EVOCINV/software/Cultured_Differential_Evolution/principal.html.

```
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.

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:

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

where $g_c(x)$ are the constr constraints of the problem, and $g_{\max c}$ is the largest violation of the constraint $g_c(x)$ found so far.

These tournament rules were inspired from previous works [25,10]. When comparing infeasible individuals, the proposed approach is more similar to the Deb's approach, since Lampinen's approach makes the comparison of violations in the Pareto sense. However, the expression above describes the normalization of the constrains; this is done with the intention that every constraint has the same contribution to the comparing value, despite of the units in which each constraint is expressed. Such a normalization was not made in the Deb's approach. Also, our approach does not use niching to maintain diversity, as in Deb's proposal [10].

A shared feature of the three approaches is the fact that the evaluation of the objective function is only needed when comparing two feasible individuals, as can be seen from the tournament rules above, reducing the amount of CPU time when solving highly constrained problems.

The rest of the algorithm, described below, is rather different from the previous approaches.

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 ith 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 [37], 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.

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

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 Fig. 4.

In Fig. 4, l_i and u_i are the lower and upper bounds, respectively, for the ith 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, $\mathrm{d}m_i$, 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 $\mathrm{d}m_i = u_i - l_i$, for $i = 1, 2, \ldots, n$.

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

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

We introduce the scaling factor $\frac{u_i - l_i}{\mathrm{d}m_i}$ for the mutation to be proportional to the interval of the normative knowledge for the ith decision variable. The values $\mathrm{d}m_i$ 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 $x_{a1}, x_{a2}, \ldots, x_{an_{\text{accepted}}}$ be the accepted individuals in the current generation, and $x_{\min_i}$ and $x_{\max_i}$, with $\min_i$, $\max_i \in \{a1, a2, \ldots, n_{\text{accepted}}\}$, 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 on 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 $\mathrm{d}m_i$ values are updated with the greatest 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 [3]. 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 Fig. 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.

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

Fig. 4. Structure of the normative knowledge.

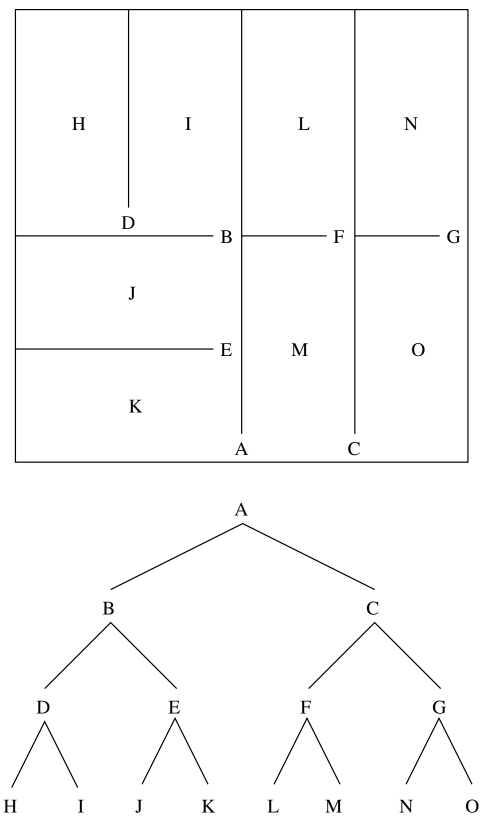

Fig. 5. Example of the partition of a two dimensional space by a k-d tree.

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).

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 [36]. 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 Fig. 6, where e_i is the best individual found before the ith 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:

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.

$$x'_{i,j} = \begin{cases} e_{i,1} + \mathrm{d}r_i * F * |x_{i,r1} - x_{i,r2}| & \text{if } \mathrm{rand}(0,1) < \alpha, \\ e_{i,1} + \dfrac{\mathrm{d}s_i}{\mathrm{d}m_i} * (x_{i,r1} - x_{i,r2}) & \text{if } \mathrm{rand}(0,1) < \beta, \\ \mathrm{rand}(lb_i, ub_i) & \text{otherwise,} \end{cases}$$

where $e_{i,1}$ is the ith decision variable of the previous best e_1 stored in the list of the history knowledge, $\mathrm{d}m_i$ is the maximum difference for the ith 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 $\mathrm{rand}(a,b)$ 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

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

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

where the function $\mathrm{sgn}(a)$ 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 [36]. The number of accepted individuals decreases as the generation number increases.

Saleem [36] 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:

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

where $\%p$ is a parameter given by the user, within the range $(0, 1]$; Saleem [36] 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, $\%p_{\text{ks}} = 1/4$, because there are four knowledge sources; but during the evolutionary process, the probability of the knowledge source ks to be applied is

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

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, $\%p_{\text{ks}} = 1/4$, as in the beginning.

4.4. Parameters of the technique

The proposed approach has several parameters that the user must set by hand in order to successfully use the algorithm. The following is a list of all the parameters and some suggestions about their settings:

- Population size, *popsize*, is the number of individuals in the population. A value as small as 10 gives good results in "easy" (i.e., with few decision variables and constraints easy to satisfy) problems. However, in general, we recommend to adopt a population size of 100. Larger sizes may be adopted, but such values are recommended only for very hard problems if one can afford the extra computational cost.

- Maximum number of generations. It is the number of iterations that the algorithm will run. This parameter, together with the population size, defines the number of fitness function evaluations that the approach will perform. For easy problems, one may start with 100 or 200 generations. Then, if necessary, this value can get increased until no improvement in the results is obtained. In general, and based on our own experience, we suggest to set this parameter to 1000.
- F and CR are the parameters of differential evolution. These parameters can be set following the suggestions in [31]. Good default values are $F = 0.5$ and $\text{CR} = 1$.
- Maximum depth of the k-d tree. The larger this value, the more accurate will be the map generated by the topographical knowledge source, but more memory will be needed. If m is this maximum depth, the tree can have up to $2^m - 1$ nodes. This parameter depends on the free memory available in the device in which the algorithm will run. With a maximum depth of 12, the algorithm can have up to 4095 nodes, which is a manageable value for most current computers. So, we suggest to set this parameter to 12.
- The length of the best cells list is the number of independent cells that will be considered for the topographical knowledge. We suggest to adopt a number of independent cells equal to the number of decision variables of the problem.
- The size of the list in the history knowledge, w, is the number of previous local optima that will be considered when looking for the next one. This parameter has little influence on the performance of the algorithm when the distribution of the local optima is not regular in the problem. We recommend to set this parameter to 5.
- α and β have little influence on performance, and can be fixed to 0.4 or 0.45.
- $\%p$ is the percentage of accepted individuals at the end of the evolutionary process. Saleem [36] suggests using 0.2, which is the value that we adopted. However, this value must be increased if the algorithm exhibits premature convergence.

5. Comparison of results

To validate our approach, we adopted the well-known benchmark originally proposed in [28] and extended in [35] which has been often used in the literature to validate new constraint-handling techniques. The 13 test problems are listed in Appendix A.

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 $\text{CR} = 1$, 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. This parameter setting was chosen to be a good compromise for all the test functions; optimal settings for each problem may exist, but they are not reported here. It is worth noticing that we did not spend too much effort in performing a very thorough parameters fine-tuning, since we found that our approach was relatively robust. However, it is expected that a better performance (even if only marginal in some cases) may be obtained with different settings. For each test function, we performed 30 independent runs (Table 1).

We compare our approach against five state-of-the-art approaches: the homomorphous mappings (HM) [21], stochastic ranking (SR) [35], the adaptive segregational constraint-handling evolutionary algorithm (ASCHEA) [16], a constraint-handling method for genetic algorithms (TS) [10], and a constraint-handling approach for differential evolution (DE) [25]. The best results obtained by each approach are shown in Table 2. The mean values provided are compared in Table 3 and the worst results are presented in Table 4. The results provided by these approaches were taken from the original references for each method.

Table 1
Results obtained by our cultured differential evolution approach

Problem	Optimal	Results of the cultured differential evolution algorithm			
		Best	Mean	Worst	Std. 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	−30,665.539	**−30,665.538672**	**−30,665.538672**	**−30,665.538672**	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 2
Comparison of the best results of the cultured differential evolution (CDE) with respect to the homomorphous maps (HM) [21], stochastic ranking (SR) [35], ASCHEA [16], a constraint-handling method for genetic algorithms (TS) [10], and a constraint-handling approach for differential evolution (DE) [25]

Problem	Optimal	Best results of the compared techniques					
		CDE	HM	SR	ASCHEA	TS	DE
g01	**−15**	**−15.000000**	−14.7864	−15.000	−15.0	−15.000	−15.000
g02	**0.803619**	**0.803619**	0.79953	0.803515	0.785	NA	NA
g03	**1**	0.995413	0.9997	1.000	1.0	NA	1.0252
g04	**−30,665.539**	**−30,665.538672**	−30,664.5	−30,665.539	−30,665.5	−30,665.537	NA
g05	**5126.498**	5126.570923	–	5126.497	5126.5	NA	5126.484
g06	**−6961.814**	**−6961.813876**	−6952.1	−6961.814	−6961.81	NA	−6961.814
g07	**24.306**	**24.306209**	24.620	24.307	24.3323	24.37248	24.306
g08	**0.095825**	**0.095825**	0.0958250	0.095825	0.095825	NA	0.095825
g09	**680.63**	**680.630057**	680.91	680.630	680.630	680.634460	680.630
g10	**7049.25**	**7049.248058**	7147.9	7054.316	7061.13	7060.221	7049.248
g11	**0.75**	**0.749900**	0.75	0.750	0.75	NA	0.74900
g12	**1.00**	**1.000000**	0.999999	1.000000	NA	NA	NA
g13	**0.053950**	0.056180	NA	0.053957	NA	0.053950	NA

"–" means no feasible solutions were found. NA = not available. A result in boldface means that our approach obtained the same or a better value than any other of the techniques.

Table 3
Comparison of the mean results of the cultured differential evolution (CDE) with respect to the homomorphous maps (HM) [21], stochastic ranking (SR) [35], ASCHEA [16], a constraint-handling method for genetic algorithms (TS) [10], and a constraint-handling approach for differential evolution (DE) [25]

Problem	Optimal	Mean results of the compared techniques					
		CDE	HM	SR	ASCHEA	TS	DE
g01	**−15**	−14.999996	−14.7082	−15.000	−14.84	−15.000	−15.000
g02	**0.803619**	0.724886	0.79671	0.781975	0.59	NA	NA
g03	**1**	0.788635	0.9989	1.000	0.99989	NA	1.0252
g04	**−30,665.539**	**−30,665.538672**	−30,655.3	−30,665.539	−30,665.5	−30,665.535	NA
g05	**5126.498**	5207.410651	–	5128.881	5141.65	NA	5126.484
g06	**−6961.814**	**−6961.813876**	−6342.6	−6875.940	−6961.81	NA	−6961.814
g07	**24.306**	**24.306210**	24.826	24.374	24.66	24.40940	24.306
g08	**0.095825**	**0.095825**	0.0891568	0.095825	0.095825	NA	0.095825
g09	**680.63**	**680.630057**	681.16	680.656	680.641	680.641724	680.630
g10	**7049.25**	**7049.248266**	8163.6	7559.192	7193.11	7220.026	7049.248
g11	**0.75**	0.757995	0.75	0.750	0.75	NA	0.74900
g12	**1.00**	**1.000000**	0.999134	1.000000	NA	NA	NA
g13	**0.053950**	0.288324	NA	0.067543	NA	0.241289	NA

"–" means no feasible solutions were found. NA = not available. A result in boldface means that our approach obtained the same or a better value than any other of the techniques.

Table 4
Comparison of the worst results of the cultured differential evolution (CDE) with respect to the homomorphous maps (HM) [21], stochastic ranking (SR) [35], ASCHEA [16], a constraint-handling method for genetic algorithms (TS) [10], and a constraint-handling approach for differential evolution (DE) [25]

Problem	Optimal	Worst results of the compared techniques					
		CDE	HM	SR	ASCHEA	TS	DE
g01	**−15**	−14.999993	−14.6154	−15.000	NA	−13.000	−15.000
g02	**0.803619**	0.590908	0.79119	0.726288	NA	NA	NA
g03	**1**	0.639920	0.9978	1.000	NA	NA	1.0252
g04	**−30,665.539**	**−30,665.538672**	−30,645.9	−30,665.539	NA	−29,846.654	NA
g05	**5126.498**	5327.390497	–	5142.472	NA	NA	5126.484
g06	**−6961.814**	**−6961.813876**	−5473.9	−6350.262	NA	NA	−6961.814
g07	**24.306**	**24.306212**	25.069	24.642	NA	25.07530	24.307
g08	**0.095825**	**0.095825**	0.0291438	0.095825	NA	NA	0.095825
g09	**680.63**	**680.630057**	683.18	680.763	NA	680.650879	680.630
g10	**7049.25**	**7049.248480**	9659.3	8835.655	NA	10230.834	7049.248
g11	**0.75**	0.796455	0.75	0.750	NA	NA	0.74900
g12	**1.00**	**1.000000**	0.991950	1.000000	NA	NA	NA
g13	**0.053950**	0.392100	NA	0.216915	NA	0.507761	NA

"–" means no feasible solutions were found. NA = not available. A result in boldface means that our approach obtained the same or a better value than any other of the techniques.

HM performs a homomorphous mapping between an n-dimensional cube and a feasible search space (either convex or non-convex). The main idea of this approach is to transform the original problem into another (topologically equivalent) function that is easier to optimize by an evolutionary algorithm. HM handles two cases: convex feasible space and non-convex feasible space. HM uses a binary-coded GA with Gray codes, proportional selection without elitism and traditional crossover and mutation operators.

The aim of SR is to balance the influence of the objective function and the penalty function when assigning fitness to a solution. SR does not require the definition of a penalty factor. Instead, a user-defined parameter called P_f sets the probability of using only the objective function to compare two solutions to sort them. The selection process is based on a ranking process. Then, when the solutions are sorted using a bubble-sort like algorithm, sometimes, depending of the P_f value, the comparison between two adjacent solutions will be performed using only the objective function. The remaining comparisons will be performed using only the penalty function that consists, in this case, of the sum of constraint violation. SR uses a $(30, 200)$-ES with global intermediate recombination applied only to the strategy parameters (not to the decision variables of the problem).

ASCHEA is based on three components: (1) an adaptive penalty function, (2) a constraint-driven recombination, and (3) a segregational selection based on feasibility. In ASCHEA's most recent version [16], the authors propose to use a penalty factor for each constraint of the problem. Also, the authors added a niching mechanism [11] to improve the performance of the algorithm in multimodal functions. Finally, the authors added a dynamic and an adaptive scheme to decrease the tolerance value used in the transformation of equality constraints into two inequality constraints. The approach uses a $(100 + 300)$-ES with standard arithmetical recombination.

The approaches TS and DE were briefly described in Section 3.1.

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. The results reported for TS and DE required a variable number of fitness function evaluations and used different parameter settings for each problem. It is worth saying that when tuning the parameters for each particular problem, our algorithm can achieve better results, but the intention here is to show the capabilities of the proposed algorithm, when not too much effort is spent in the adjustment of the algorithm's parameters to the problem at hand. TS required from **250,050** to **350,100** function evaluations (in the original source, the number of evaluations and maximum generations for the problem g01 are missing), and DE required from **10,000** to **12,000,000** function evaluations. Our approach required **100,100** evaluations in all the test problems adopted.

In the problem g01, the best result obtained was $f(x) = -15.000000$, with $x = \{1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 1.000000, 3.000000, 3.000000, 3.000000, 1.000000\}$ and the values $g_i(x) = \{0.000000, 0.000000, 0.000000, -5.000000, -5.000000, -5.000000, 0.000000, 0.000000, 0.000000\}$ for the constraints. The cultured differential evolution, SR and DE reached the optimum value in all the 30 runs performed. TS reached the optimum in its best and median case, ASCHEA reached the optimum only in its best case, and HM could not reach the optimum.

For the problem g02, SR has a lower variability than the cultured differential evolution. However, our approach can get the optimum value in some of the runs, and HM, SR and ASCHEA were unable to do it. The best value obtained was $f(x) = -0.803619$, with $x = \{3.162681, 3.128106, 3.095032, 3.061492, 3.028413, 2.994410, 2.957990, 2.921746, 0.495094, 0.488762, 0.481956, 0.476491, 0.472845, 0.465291, 0.461291, 0.457043, 0.452091, 0.448045, 0.444050, 0.440310\}$ and $g_i(x) = \{0.000000, -120.066861\}$.

In g03, the best result obtained by the cultured differential evolution was $f(x) = 0.995413$, with $x = \{0.304887, 0.329917, 0.319260, 0.328069, 0.326023, 0.302707, 0.305104, 0.315312, 0.322047, 0.309009\}$ and $g_i(x) = \{0.000991\}$, which is very close of the global optimum. SR and DE are clear winners in this problem, because they were able to reach the optimum in all the runs reported. HM and ASCHEA also had a better performance than the cultured differential evolution in this problem.

The best result obtained for the function g04 was $f(x) = -30665.538672$, with $x = \{78.000000, 33.000000, 29.995256, 45.000000, 36.775813\}$ and $g_i(x) = \{0.000000, -92.000000, -11.159500, -8.840500, -5.000000, -0.000000\}$. This problem is "easy" to solve for all techniques, which have similar behavior obtaining a value very close to the optimum in all the runs. The results for DE reports a slightly different version of this problem, and therefore cannot be included in this comparison.

g05 is the problem in which the cultured differential evolution exhibits its highest variability of results, even when its best value is very close to the optimum: $f(x) = 5126.570923$, with $x = \{683.926335, 1021.814124, 0.116038, -0.397581\}$ and the values $g_i(x) = \{-0.036381, -1.063619, 0.000803, 0.000419, 0.000980\}$ for the constraints. SR, ASCHEA and DE also obtained a best value very close to the optimum, but their variability is lower. HM was not able to find any feasible solution for this problem.

$f(x) = -6961.813876$, with $x = \{14.095000, 0.842961\}$ and $g_i(x) = \{0.000000, -0.000002\}$ is the best result obtained for the cultured differential evolution in g06. In this problem, our approach shows a performance which is clearly better than that of HM and SR in terms of consistency to reach the optimum. Both had a large variability of results when solving this problem. ASCHEA and DE reported a mean (and worst in the case of DE) result very close to the optimum, as is the case of the cultured differential evolution.

The best result obtained by our approach in g07 is $f(x) = 24.306209$, with $x = \{2.171982, 2.363717, 8.773931, 5.095984, 0.990654, 1.430557, 1.321617, 9.828704, 8.280092, 8.376018\}$ and $g_i(x) = \{-0.000002, 0.000000, 0.000000, 0.000000, -0.000010, -0.000002, -6.148630, -50.024352\}$. The cultured differential evolution exhibits a great robustness in this problem, reaching a value very close to the optimum in all the runs. The best value obtained for any of the other techniques (except DE), is worse than the worst result produced by the cultured differential evolution. Only DE can also reach the optimum, but exhibits a slightly larger variability (as can be seen in the worst case).

The cultured differential evolution approach obtained this best value for the problem g08: $f(x) = 0.095825$, with $x = \{1.227971, 4.245373\}$ and $g_i(x) = \{-1.737460, -0.167763\}$. This problem seems easy to solve for the techniques analyzed here, because all the techniques reached the optimum in almost all the runs, except HM, which exhibits a poor performance in this problem.

In the problem g09, the best result obtained was $f(x) = 680.630057$, with $x = \{2.330499, 1.951372, -0.477541, 4.365726, -0.624487, 1.038131, 1.594227\}$ and $g_i(x) = \{-0.000045, -252.561724, -144.878190, -0.000008\}$. SR and ASCHEA were able to reach the optimum in their best cases, while the algorithm proposed here and DE obtained the optimum in all the runs performed, being more robust.

In g10, the best result obtained by the cultured differential evolution was $f(x) = 7049.248058$, with $x = \{579.380244, 1359.992748, 5109.875066, 182.023843, 295.604998, 217.976157, 286.418844, 395.604998\}$ and the values $g_i(x) = \{0.000000, 0.000000, 0.000000, 0.000000, -0.000537, -0.001600\}$ for the constraints. Again, our cultured differential evolution approach and DE obtained a value very close to the optimum in all the runs, being clear winners. However, DE required 270,000 function evaluations for this particular problem. All the other techniques exhibited a very high variability of results in this problem, and were not able to reach the optimum in their best cases.

The best result obtained for the function g11 was $f(x) = 0.749900$, with $x = \{0.707036, 0.500000\}$ and $g_i(x) = \{0.000100\}$. In this case, all the approaches analyzed here reached the optimum in their best cases. However, HM, SR, ASCHEA and DE were more robust in this problem, showing a low variability of results.

The best result obtained by our approach in g12 is $f(x) = 1.000000$, with $x = \{5.000000, 5.000000, 5.000000\}$ and $g_i(x) = \{-0.062500\}$. This problem is another example of similar performance of SR and the cultured differential evolution, because they both could reach the optimum in practically all the runs.

Finally, $f(x) = 0.056180$, with $x = \{-1.648857, 1.515826, 1.949583, 0.753371, 0.784312\}$ and $g_i(x) = \{0.000045, 0.000839, 0.000161\}$ is the best result obtained for the cultured differential evolution in g13. SR was more robust than our approach in this problem, and its best result was slightly better than ours.

In short, SR and DE are the most competitive constraint-handling techniques compared here. However, our approach reached the global optimum in 10 problems, while SR and DE did it in nine (DE was only tested in nine problems). Also, in most cases, the cultured differential evolution was more robust than SR, showing a very low standard deviation, while performing less than one third of its total number of fitness function evaluations. Also, it is worth noticing that our approach used the same number of function evaluations and parameters for all the test functions, while DE got their parameters tuned for each problem, and requires a variable number of fitness function evaluations.

5.1. Engineering optimization problems

Additionally to this standard benchmark, we tested the cultured differential evolution on an engineering optimization problem: design of trusses. Two different problems related to the design of trusses are tackled: the optimization of a 10-bar plane truss and a 200-bar plane truss. Their descriptions are provided in Appendix A.

5.2. Comparison of results

The two engineering optimization problems previously mentioned were used by Belegundu [2] to evaluate the following numerical optimization techniques: feasible directions (CONMIN and OPTDYN), Pshenichny's Recursive Quadratic Programming (LINRM), gradient projection (GRP-UI), exterior penalty function (SUMT), multiplier methods (M-3, M-4 and M-5).

The parameters adopted by our cultured differential evolution approach are the same used to solve the benchmark. Thus, we performed the same number of objective function evaluations as before (100, 100). For the 10-bar truss, the best result of the 30 independent runs is shown in Table 5, together with the results of the methods included in [2] (all the results presented are feasible). For the 200-bar truss, the best result of the 30 independent runs is shown in Table 6, with the results of other methods reported by Belegundu [2].

The best results of the cultured differential evolution are very competitive. Additionally, such results present a low variability over the 30 runs performed, as can be appreciated from Table 7.

Table 5
Results of several methods for the 10-bar plane truss (g14)

	Method				
	CONMIN	OPTDYN	LINRM	GRP-UI	SUMT
Weight	4793.0	9436.0	6151.0	5077.0	5070.0
	M-3	M-4	M-5	CDE	
Weight	4898.0	5057.0	5211.0	**4656.39**	

Our approach is labeled as CDE.

Table 6
Results of several methods for the 200-bar plane truss (g15)

	Method				
	CONMIN	OPTDYN	LINRM	GRP-UI	SUMT
Weight	34,800.0	N/A	33,315.0	N/A	27,564.0
	M-3	M-4	M-5	CDE	
Weight	26,600.0	26,654.0	26,262.0	**20,319.58**	

Our approach is labeled as CDE.

Table 7
Statistics of the results obtained by our approach for the design of trusses

TF	Best	Mean	Worst	Std. dev.
10-bar truss (g14)	4656.39	4656.52	4656.71	0.18
200-bar truss (g15)	20,319.58	25,393.37	30,269.49	2492.24

Finally, we noticed that in engineering optimization problems with large search spaces (such as the two presented in this paper), our approach could improve on the quality of the solutions produced if we allowed a larger number of fitness function evaluations. Although such increase may be unaffordable in real-world applications, this is a good indicative of the effectiveness of our approach, since, if possible, it keeps improving the solutions produced over time. To briefly illustrate this issue, we allowed our approach to run for 3500 generations (for a total of 350,100 fitness function evaluations) in the 200-bar truss problem. This setup produced a solution with a weight of only **15824.32**, which is about 28% better than the solution reported in Table 6. Note however, that the computational effort required to produce this solution (measured in terms of fitness function evaluations) is over three times the original one.

6. Conclusions and future work

In this paper we introduced a cultural algorithm, which uses differential evolution. The approach is used to solve constrained optimization problems. By adding a belief space to the differential evolution algorithm, we were able to obtain competitive results on a well-known benchmark adopted for evolutionary optimization, and on two engineering optimization problems. Our results were compared with respect to several evolutionary algorithms that are representative of the state-of-the-art in constrained optimization (the homomorphous maps, stochastic ranking and ASCHEA).

The results were obtained at a relatively low computational cost (measured in fitness function evaluations), when compared to the aforementioned evolutionary algorithms. The proposed approach only requires about 30% of the fitness evaluations of the second less expensive algorithm against which it was compared.

The algorithm supports four different knowledge sources, which make it relatively complex to implement. However, we have made the source code available so that anyone interested in using it, can download it and tailor it for his/her own application. Note that the knowledge extraction mechanisms of our approach were developed only for numerical (constrained) optimization problems, and we make no claim regarding their generality. Different types of problems (e.g., combinatorial) may require different knowledge extraction mechanisms.

One of the possible paths for future work is the analysis of the impact of each knowledge source during the evolutionary process. We are also interested in examining the influence of the different parameter settings in the performance of the algorithm, which requires a more thorough statistical analysis. To date, we have not identified a parameter that is the most critical (in terms of performance) of the approach.

Another interesting point related to the incorporation of knowledge of our approach has to do with its theoretical foundation. We did not attempt to provide a theoretical explanation of why the mechanisms that we incorporate in our evolutionary algorithm are able to accelerate convergence, as well as a theoretical insight regarding their possible limitations. However, this topic certainly deserves some further work.

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 [9,15,29,22].

Acknowledgements

The authors thank the anonymous reviewers for their comments and suggestions which greatly improved the contents of this paper. The first author acknowledges support from CONACyT to pursue graduate studies at the Computer Science Section of CINVESTAV-IPN. The second author gratefully acknowledges support from CONACyT through project 42435-Y.

Appendix A. Test functions

A.1. g01

$$\begin{aligned}
\text{Minimize}\quad & f(\vec{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \\
\text{subject to}\quad & g_1(\vec{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leqslant 0, \\
& g_2(\vec{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leqslant 0, \\
& g_3(\vec{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leqslant 0, \\
& g_4(\vec{x}) = -8x_1 + x_{10} \leqslant 0, \\
& g_5(\vec{x}) = -8x_2 + x_{11} \leqslant 0, \\
& g_6(\vec{x}) = -8x_3 + x_{12} \leqslant 0, \\
& g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \leqslant 0, \\
& g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \leqslant 0, \\
& g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \leqslant 0,
\end{aligned}$$

where $0 \leqslant x_i \leqslant 1$ $(i = 1,\ldots,9)$, $0 \leqslant x_i \leqslant 100$ $(i = 10, 11, 12)$, $0 \leqslant x_{13} \leqslant 1$.

The optimum solution is $x^* = (1,1,1,1,1,1,1,1,1,3,3,3,1)$ where $f(x^*) = -15$.

A.2. g02

$$\begin{aligned}
\text{Maximize}\quad & f(\vec{x}) = \left| \frac{\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right| \\
\text{subject to}\quad & g_1(\vec{x}) = 0.75 - \prod_{i=1}^{n} x_i \leqslant 0, \\
& g_2(\vec{x}) = \sum_{i=1}^{n} x_i - 7.5n \leqslant 0,
\end{aligned}$$

where $n = 20$, $0 \leqslant x_i \leqslant 10$ $(i = 1,\ldots,n)$.

The best known solution is $f(x^*) = 0.803619$.

A.3. g03

$$\begin{aligned}
\text{Maximize}\quad & f(\vec{x}) = (\sqrt{n})^n \prod_{i=1}^{n} x_i, \\
\text{subject to}\quad & h_1(\vec{x}) = \sum_{i=1}^{n} x_i^2 - 1 = 0,
\end{aligned}$$

where $n = 10$, $0 \leqslant x_i \leqslant 1$ $(i = 1,\ldots,n)$.

The optimum solution is $x_i^* = 1/\sqrt{n}$ $(i = 1,\ldots,n)$ where $f(x^*) = 1$.

A.4. g04

$$\begin{aligned}
\text{Minimize}\quad & f(\vec{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141\\
\text{subject to}\quad & g_1(\vec{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leqslant 0,\\
& g_2(\vec{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leqslant 0,\\
& g_3(\vec{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leqslant 0,\\
& g_4(\vec{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leqslant 0,\\
& g_5(\vec{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leqslant 0,\\
& g_6(\vec{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leqslant 0,
\end{aligned}$$

where $78 \leqslant x_1 \leqslant 102$, $33 \leqslant x_2 \leqslant 45$, $27 \leqslant x_i \leqslant 45$ $(i = 3,4,5)$.

The optimum solution is $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ where $f(x^*) = -30{,}665.539$.

A.5. g05

$$\begin{aligned}
\text{Minimize}\quad & f(\vec{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3\\
\text{subject to}\quad & g_1(\vec{x}) = -x_4 + x_3 - 0.55 \leqslant 0,\\
& g_2(\vec{x}) = -x_3 + x_4 - 0.55 \leqslant 0,\\
& h_3(\vec{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0,\\
& h_4(\vec{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0,\\
& h_5(\vec{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0,
\end{aligned}$$

where $0 \leqslant x_1 \leqslant 1200$, $0 \leqslant x_2 \leqslant 1200$, $-0.55 \leqslant x_3 \leqslant 0.55$, $-0.55 \leqslant x_4 \leqslant 0.55$.

The best known solution is $x^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$ where $f(x^*) = 5126.4981$.

A.6. g06

$$\begin{aligned}
\text{Minimize}\quad & f(\vec{x}) = (x_1 - 10)^3 + (x_2 - 20)^3\\
\text{subject to}\quad & g_1(\vec{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leqslant 0,\\
& g_2(\vec{x}) = (x_1 - 6)^2 + (x_2 - 5) - 82.81 \leqslant 0,
\end{aligned}$$

where $13 \leqslant x_1 \leqslant 100$, $0 \leqslant x_2 \leqslant 100$.

The optimum solution is $x^* = (14.095, 0.84296)$ where $f(x^*) = -6961.81388$.

A.7. g07

$$\begin{aligned}
\text{Minimize}\quad & f(\vec{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2\\
& \qquad + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45\\
\text{subject to}\quad & g_1(\vec{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leqslant 0,\\
& g_2(\vec{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leqslant 0,\\
& g_3(\vec{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leqslant 0,\\
& g_4(\vec{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leqslant 0,\\
& g_5(\vec{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leqslant 0,\\
& g_6(\vec{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leqslant 0,\\
& g_7(\vec{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leqslant 0,\\
& g_8(\vec{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leqslant 0,
\end{aligned}$$

where $-10 \leqslant x_i \leqslant 10$ $(i = 1, \ldots, 10)$.

The optimum solution is $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$ where $f(x^*) = 24.3062091$.

A.8. g08

$$\text{Maximize} \quad f(\vec{x}) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$$

$$\text{subject to} \quad g_1(\vec{x}) = x_1^2 - x_2 + 1 \leqslant 0,$$

$$g_2(\vec{x}) = 1 - x_1 + (x_2-4)^2 \leqslant 0,$$

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

The optimum solution is $x^* = (1.2279713, 4.2453733)$ where $f(x^*) = 0.095825$.

A.9. g09

$$\text{Minimize} \quad f(\vec{x}) = (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

$$\text{subject to} \quad g_1(\vec{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leqslant 0,$$

$$g_2(\vec{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leqslant 0,$$

$$g_3(\vec{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leqslant 0,$$

$$g_4(\vec{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leqslant 0,$$

where $-10 \leqslant x_i \leqslant 10$ $(i = 1, \ldots, 7)$.

The optimum solution is $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$ where $f(x^*) = 680.6300573$.

A.10. g10

$$\text{Minimize} \quad f(\vec{x}) = x_1 + x_2 + x_3$$

$$\text{subject to} \quad g_1(\vec{x}) = -1 + 0.0025(x_4 + x_6) \leqslant 0,$$

$$g_2(\vec{x}) = -1 + 0.0025(x_5 + x_7 - x_4) \leqslant 0,$$

$$g_3(\vec{x}) = -1 + 0.01(x_8 - x_5) \leqslant 0,$$

$$g_4(\vec{x}) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83,333.333 \leqslant 0,$$

$$g_5(\vec{x}) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leqslant 0,$$

$$g_6(\vec{x}) = -x_3x_8 + 1,250,000 + x_3x_5 - 2500x_5 \leqslant 0,$$

where $100 \leqslant x_1 \leqslant 10{,}000$, $1000 \leqslant x_i \leqslant 10{,}000$ $(i = 2, 3)$, $10 \leqslant x_i \leqslant 1000$ $(i = 4, \ldots, 8)$.

The optimum solution is $x^* = (579.19, 1360.13, 5109.92, 182.0174, 295.5985, 217.9799, 286.40, 395.5979)$, where $f(x^*) = 7049.25$.

A.11. g11

$$\text{Minimize} \quad f(\vec{x}) = x_1^2 + (x_2-1)^2$$

$$\text{subject to} \quad h(\vec{x}) = x_2 - x_1^2 = 0,$$

where $-1 \leqslant x_1 \leqslant 1$, $-1 \leqslant x_2 \leqslant 1$.

The optimum solution is $x^* = (\pm 1/\sqrt{2}, 1/2)$ where $f(x^*) = 0.75$.

A.12. g12

$$\text{Maximize} \quad f(\vec{x}) = \frac{(100 - (x_1-5)^2 - (x_2-5)^2 - (x_3-5)^2)}{100}$$

$$\text{subject to} \quad g(\vec{x}) = (x_1-p)^2 + (x_2-q)^2 + (x_3-r)^2 - 0.0625 \leqslant 0,$$

where $0 \leqslant x_i \leqslant 10$ $(i = 1, 2, 3)$, $p, q, r = 1, 2, \ldots, 9$.

The optimum solution is $x^* = (5, 5, 5)$ where $f(x^*) = 1$.

A.13. g13

$$\begin{aligned}
\text{Minimize} \quad & f(\vec{x}) = e^{x_1x_2x_3x_4x_5} \\
\text{subject to} \quad & h_1(\vec{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0, \\
& h_2(\vec{x}) = x_2x_3 - 5x_4x_5 = 0, \\
& h_3(\vec{x}) = x_1^3 + x_2^3 - 1 = 0,
\end{aligned}$$

where $-2.3 \leqslant x_i \leqslant 2.3$ ($i = 1,2$), $-3.2 \leqslant x_i \leqslant 3.2$ ($i = 3,4,5$).

The optimum solution is $x^* = (-1.717143, 1.595709, 1.827247, -0.7636413, -0.763645)$ where $f(x^*) = 0.0539498$.

A.14. g14: Design of a 10-bar plane truss

Consider the 10-bar plane truss shown in Fig. 7 [2]. The problem is to find the cross-sectional area of each member of this truss, such that we minimize its weight. The problem is subject to both displacement and stress constraints. The weight of the truss is given by $f(x)$.

$$f(x) = \sum_{j=1}^{10} \rho A_j L_j, \tag{1}$$

where x is the candidate solution, A_j is the cross-sectional area of the jth member, L_j is the length of the jth member, and ρ is the weight density of the material. The assumed data are: modulus of elasticity, $E = 1.09 \times 10^4$ ksi, $\rho = 0.10$ lb/in.^{3}, and a load of 100 kips in the negative y-direction is applied at nodes 2 and 4. The maximum allowable stress of each member is

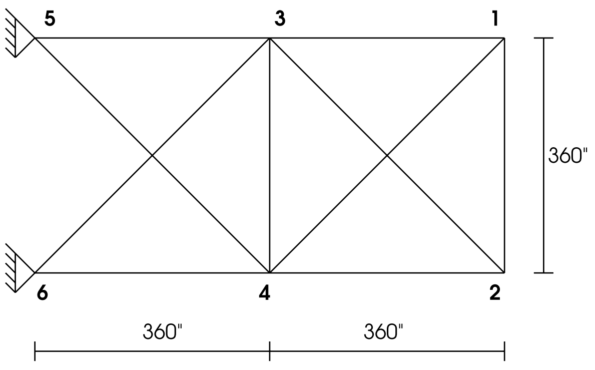

Fig. 7. 10-bar plane truss adopted for problem g14.

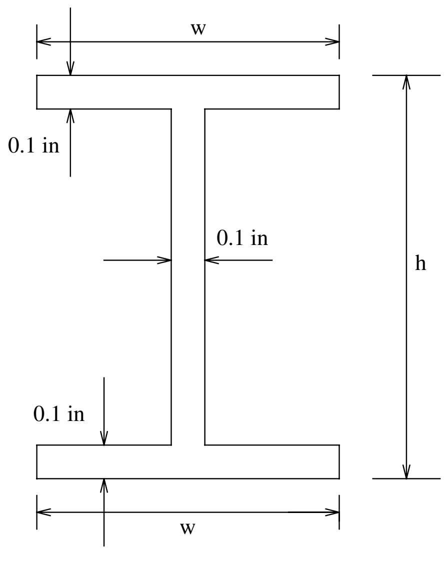

Fig. 8. Cross-section used for the 10-bar plane truss from problem g14.

called σ_a, and it is assumed to be ±25 ksi. The maximum allowable displacement of each node (horizontal and vertical) is represented by u_a, and is assumed to be 2 in. The minimum allowable cross-section area is 0.10 in.^{2} for all members. The cross-section of each element can be different, and is defined by the I-section shown in Fig. 8, with the depth and width as design variables. The web thickness and flange thickness are each kept fixed at 0.1 in. The problem has, therefore, 20 design variables.

To solve this problem, it was necessary to add a module responsible for the analysis of the plane truss. This module uses the stiffness method [13] to analyze the structure, and returns the values of the stress and displacement constraints, as well as the total weight of the structure.

A.15. g15: Design of a 200-bar plane truss

Consider the 200-bar plane truss taken from Belegundu [2] and shown in Fig. 9. The problem is to find the cross-sectional area of each member of this truss, such that we minimize its weight. The problem is subject to both displacement and stress constraints.

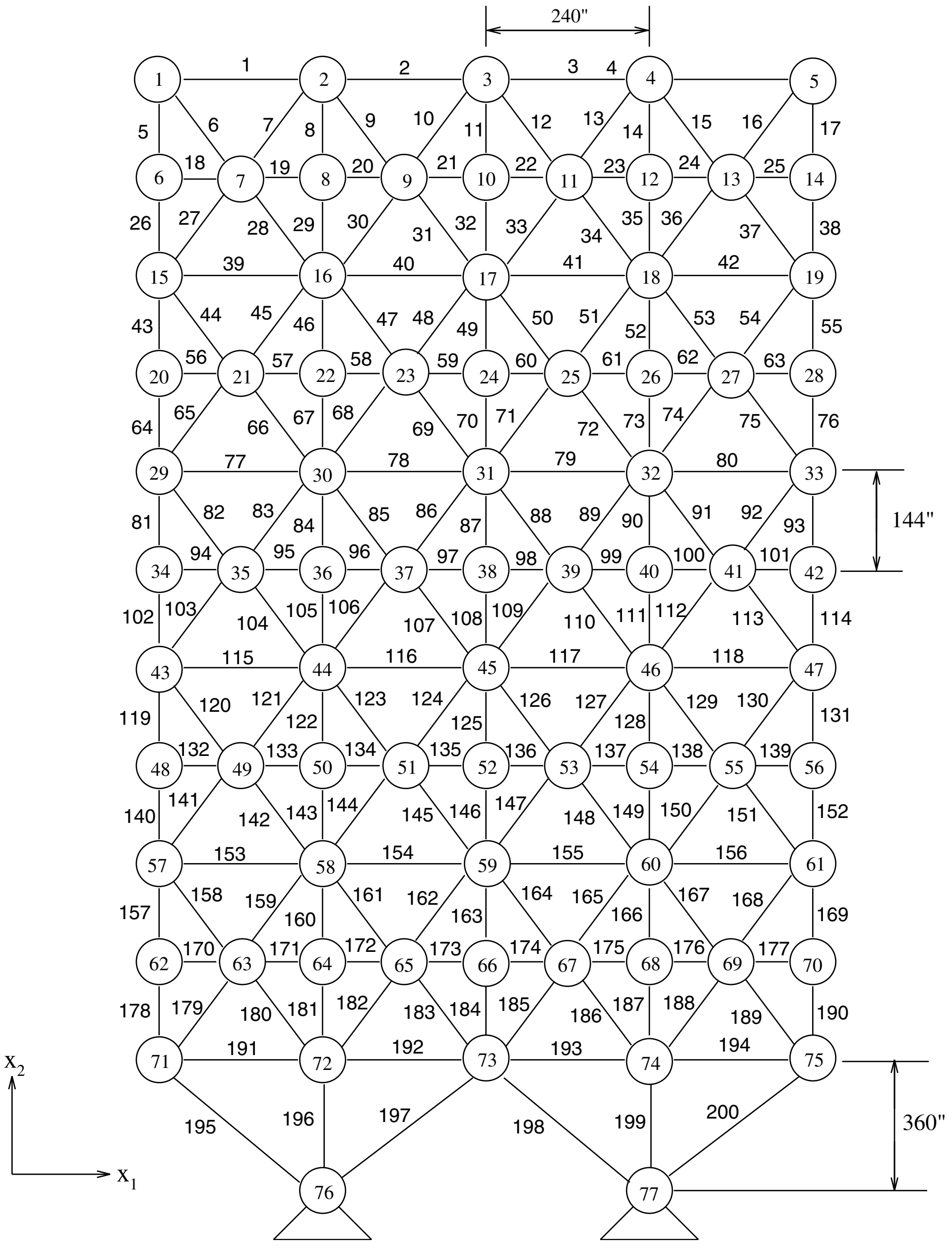

Fig. 9. 200-bar plane truss used for problem g15.

Table 8
Group membership for the 200-bar plane truss from problem g15

Group number	Member number
1	1, 2, 3, 4
2	5, 8, 11, 14, 17
3	19, 20, 21, 22, 23, 24
4	18, 25, 56, 63, 94, 101, 132, 139, 170, 177
5	26, 29, 32, 35, 38
6	6, 7, 9, 10, 12, 13, 15, 16, 27, 28, 30, 31, 33, 34, 36, 37
7	39, 40, 41, 42
8	43, 46, 49, 52, 55
9	57, 58, 59, 60, 61, 62
10	64, 67, 70, 73, 76
11	44, 45, 47, 48, 50, 51, 53, 54, 65, 66, 68, 69, 71, 72, 74, 75
12	77, 78, 79, 80
13	81, 84, 87, 90, 93
14	95, 96, 97, 98, 99, 100
15	102, 105, 108, 111, 114
16	82, 83, 85, 86, 88, 89, 91, 92, 103, 104, 106, 107, 109, 110, 112, 113
17	115, 116, 117, 118
18	119, 122, 125, 128, 131
19	133, 134, 135, 136, 137, 138
20	140, 143, 146, 149, 152
21	120, 121, 123, 124, 126, 127, 129, 130, 141, 142, 144, 145, 147, 148, 150, 151
22	153, 154, 155, 156
23	157, 160, 163, 166, 169
24	171, 172, 173, 174, 175, 176
25	178, 181, 184, 187, 190
26	158, 159, 161, 162, 164, 165, 167, 168, 179, 180, 182, 183, 185, 186, 188, 189
27	191, 192, 193, 194
28	195, 197, 198, 200
29	196, 199

There are a total of three loading conditions: (1) 1 kip acting in positive x-direction at node points 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71; (2) 10 kips acting in negative y-direction at node points 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 24, 71, 72, 73, 74, and 75; and (3) loading conditions 1 and 2 acting together. The 200 elements of this truss linked to 29 groups. The grouping information is shown in Table 8. The stress in each element is limited to a value of 10 ksi for both tension and compression members. Young's modulus of elasticity = 30,000 ksi, weight density = 0.283×10^{-3} kips/in.^{3}.

References

[1] Thomas Bäck, David Fogel, Zbigniew Michalewicz (Eds.), Handbook of Evolutionary Computation, vol. 1, IOP Publishing Ltd. and Oxford University Press, 1997.

[2] Ashok Dhondu Belegundu, A Study of Mathematical Programming Methods for Structural Optimization, Department of Civil and Environmental Engineering, University of Iowa, Iowa, 1982.

[3] Jon Louis Bentley, Jerome H. Friedman, Data structures for range searching, ACM Comput. Surveys 11 (4) (1979) 397–409.

[4] Chan-Jin Chung, Robert G. Reynolds, A testbed for solving optimization problems using cultural algorithms, in: Lawrence J. Fogel, Peter J. Angeline, Thomas Bäck (Eds.), Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1996.

[5] Chan-Jin Chung, Robert G. Reynolds, CAEP: An evolution-based tool for real-valued function optimization using cultural algorithms, J. Artif. Intell. Tools 7 (3) (1998) 239–292.

[6] Carlos A. Coello Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput. Methods Appl. Mech. Engrg. 191 (11–12) (2002) 1245–1287.

[7] Carlos A. Coello Coello, Ricardo Landa Becerra, Adding knowledge and efficient data structures to evolutionary programming: a cultural algorithm for constrained optimization, in: Erick Cantú-Paz et al. (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002), Morgan Kaufmann Publishers, San Francisco, CA, 2002, pp. 201–209.

[8] Carlos A. Coello Coello, Ricardo Landa Becerra, Efficient evolutionary optimization through the use of a cultural algorithm, Engrg. Optim. 36 (2) (2004) 219–236.

[9] Carlos A. Coello Coello, David A. Van Veldhuizen, Gary B. Lamont, Evolutionary Algorithms for Solving Multi-Objective Problems, Kluwer Academic Publishers, New York, 2002, ISBN 0-3064-6762-3.

[10] Kalyanmoy Deb, An efficient constraint handling method for genetic algorithms, Comput. Methods Appl. Mech. Engrg. 186 (2–4) (2000) 311–338.

[11] Kalyanmoy Deb, David E. Goldberg, An investigation of niche and species formation in genetic function optimization, in: J. David Schaffer (Ed.), Proceedings of the Third International Conference on Genetic Algorithms, June, Morgan Kaufmann Publishers, San Mateo, CA, 1989, pp. 42–50.

[12] A.E. Eiben, J.E. Smith, Introduction to Evolutionary Computing, Springer, 2003, ISBN 3-540-40184-9.

[13] James M. Gere, William Weaver, Analysis of Framed Structures, D. Van Nostrand Company, Inc., 1965.

[14] David E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, MA, 1989.

[15] D. Greiner, J.M. Emperador, G. Winter, Single and multiobjective frame optimization by evolutionary algorithms and the auto-adaptive rebirth operator, Comput. Methods Appl. Mech. Engrg. 193 (33–35) (2004) 3711–3743.

[16] Sana Ben Hamida, Marc Schoenauer, ASCHEA: new results using adaptive segregational constraint handlingProceedings of the Congress on Evolutionary Computation 2002 (CEC'2002), May, vol. 1, IEEE Service Center, Piscataway, NJ, 2002, pp. 884–889.

[17] Radu Iacoban, Robert G. Reynolds, Jon Brewster, Cultural swarms: modeling the impact of culture on social interaction and problem solving, in: 2003 IEEE Swarm Intelligence Symposium Proceedings, April, IEEE Service Center, Indianapolis, IN, USA, 2003, pp. 205–211.

[18] Xidong Jin, Robert G. Reynolds, Using knowledge-based evolutionary computation to solve nonlinear constraint optimization problems: a cultural algorithm approach, in: 1999 Congress on Evolutionary Computation, July, IEEE Service Center, Washington, DC, 1999, pp. 1672–1678.

[19] Yaochu Jin (Ed.), Knowledge Incorporation in Evolutionary Computation, Springer, 2005, ISBN 3-540-22902-7.

[20] James Kennedy, Russell C. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, CA, 2001.

[21] Slawomir Koziel, Zbigniew Michalewicz, Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization, Evol. Comput. 7 (1) (1999) 19–44.

[22] N.D. Lagaros, V. Plevris, M. Papadrakakis, Multi-objective design optimization using cascade evolutionary computations, Comput. Methods Appl. Mech. Engrg. 194 (30–33) (2005) 3496–3515.

[23] Jouni Lampinen, Multi-constrained optimization by the differential evolution, in: M.H. Hamza (Ed.), Proceedings of the IASTED International Conference Artificial Intelligence and Applications (AIA 2001), September 2001, pp. 177–184.

[24] Jouni Lampinen, Solving problems subject to multiple nonlinear constraints by the differential evolution, in: Radek Matousek, Pavel Osmera (Eds.), Proceedings of MENDEL 2001, 7th International Conference on Soft Computing, June 2001, pp. 50–57.

[25] Jouni Lampinen, A constraint handling approach for the differential evolution algorithm, in: Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002), May, vol. 2, IEEE Service Center, Piscataway, NJ, 2002, pp. 1468–1473.

[26] Yung-Chien Lin, Kao-Shing Hwang, Feng-Sheng Wang, Hybrid differential evolution with multiplier updating method for nonlinear constrained optimization, in: Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002), May, vol. 1, IEEE Service Center, Piscataway, NJ, 2002, pp. 872–877.

[27] Zbigniew Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, third ed., Springer-Verlag, 1996.

[28] Zbigniew Michalewicz, Marc Schoenauer, Evolutionary algorithms for constrained parameter optimization problems, Evol. Comput. 4 (1) (1996) 1–32.

[29] Costas Papadimitriou, Pareto optimal sensor locations for structural identification, Comput. Methods Appl. Mech. Engrg. 194 (12–16) (2005) 1655–1673.

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

[31] Kenneth V. Price, An introduction to differential evolution, in: David Corne, Marco Dorigo, Fred Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, 1999, pp. 79–108.

[32] Robert G. Reynolds, An introduction to cultural algorithms, in: A.V. Sebald, L.J. Fogel (Eds.), Proceedings of the Third Annual Conference on Evolutionary Programming, World Scientific, River Edge, NJ, 1994, pp. 131–139.

[33] Robert G. Reynolds, Cultural algorithms: theory and applications, in: David Corne, Marco Dorigo, Fred Glover (Eds.), New Ideas in Optimization, McGraw-Hill, London, UK, 1999, pp. 367–377.

[34] Robert G. Reynolds, Zbigniew Michalewicz, M. Cavaretta, Using cultural algorithms for constraint handling in GENOCOP, in: J.R. McDonnell, R.G. Reynolds, D.B. Fogel (Eds.), Proceedings of the Fourth Annual Conference on Evolutionary Programming, MIT Press, Cambridge, MA, 1995, pp. 298–305.

[35] Thomas P. Runarsson, Xin Yao, Stochastic ranking for constrained evolutionary optimization, IEEE Trans. Evol. Comput. 4 (3) (2000) 284–294.

[36] Saleh M. Saleem, Knowledge-based solution to dynamic optimization problems using cultural algorithms, PhD thesis, Wayne State University, Detroit, MI, 2001.

[37] Rainer Storn, On the usage of differential evolution for function optimization, in: 1996 Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS 1996), IEEE, Berkeley, 1996, pp. 519–523.

[38] Rainer Storn, System design by constraint adaptation and differential evolution, IEEE Trans. Evol. Comput. 3 (1) (1999) 22–34.

[39] David H. Wolpert, William G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (1) (1997) 67–82.