A Generator for Multimodal Test Functions with Multiple Global Optima

Jani Rönkkönen[1], Xiaodong Li[2], Ville Kyrki[1], and Jouni Lampinen[3]

[1] Department of Information Technology, Lappeenranta University of Technology, P.O. Box 20, Lappeenranta, FI-53851, Finland
jani.ronkkonen@lut.fi, ville.kyrki@lut.fi
[2] School of Computer Science and IT, RMIT University
xiaodong@cs.rmit.edu.au
[3] Department of Computer Science, University of Vaasa, P.O. Box 700, Vaasa, FI-65101, Finland
jouni.lampinen@uwasa.fi

Abstract. The topic of multimodal function optimization, where the aim is to locate more than one solution, has attracted a growing interest especially in the evolutionary computing research community. To experimentally evaluate the strengths and weaknesses of multimodal optimization algorithms, it is important to use test functions representing different characteristics and of various levels of difficulty. However, the available selection of multimodal test problems with multiple global optima is rather limited at the moment and no general framework exists. This paper describes our attempt in constructing a test function generator to allow the generation of easily tunable test functions. The aim is to provide a general and easily expandable environment for testing different methods of multimodal optimization. Several function families with different characteristics are included. The generator implements new parameterizable function families for generating desired landscapes and a selection of well known test functions from literature, which can be rotated and stretched. The module can be easily imported to any optimization algorithm implementation compatible with C programming language.

Keywords: Multimodal optimization, test function generator, global optimization.

1 Introduction

Real-world optimization problems often contain multiple global or local optima (i.e. solutions). Multimodal optimization aims to locate all of the global optima of a multimodal function. Evolutionary Algorithms (EAs) have become a popular choice as optimization techniques for many applications and are an interesting candidate for multimodal optimization due to their use of population, which allows multiple solutions to be searched simultaneously. However,

X. Li et al. (Eds.): SEAL 2008, LNCS 5361, pp. 239–248, 2008.

EAs in their original form are typically designed to locate a single global optimum. Many techniques for locating multiple solutions have been developed, commonly referred as *niching* methods [1]. Two most well-known niching methods are probably *crowding* [2] and *fitness sharing* [3]. Apart from *crowding* and *fitness sharing* and their variants, many other niching methods have also been developed [4,5,6,7,8]. It is noticeable, that most of these methods were often evaluated using only 1 or 2 dimensional multimodal test functions, i.e., the the functions were not scalable to higher dimensions. Furthermore, these functions are often defined in a specific way, not allowing the functions to be tunable in terms of the characteristics of the multimodal landscapes. For example, for the Shubert function used in [8], as the number of dimensions increases, the number of global optima grows exponentially ($D \cdot 3^D$, where D is the number of dimensions). There is no way to control the number of optima, neither how they are distributed. Additionally, the problem is separable and the global optima are positioned at regular intervals in the search space, both being easily exploitable features. In short, using solely the currently available selection of test functions is typically inadequate for properly analyzing the characteristics of different multimodal optimization algorithms. Especially in the light of no free lunch theorem [9], which states that no optimization algorithm can outperform another over the set of all possible problems, it becomes increasingly important to differentiate the characteristics of the subset of problems each algorithm excels in. For that, a set of parameterizable functions is required, where their characteristics can be changed independently to isolate the effects.

This paper describes an attempt to construct a multimodal test function generator, tailored specially for evaluating the performance of multimodal optimization algorithms in locating multiple globally optimal solutions. Desirable features of such a generator include the following, which have been used as guidelines for designing the generator:

1. Easy to use and tunable.
2. Functions can be transformed from separable to non-separable by rotation.
3. Regular and irregular distributions of optima.
4. Controllable number of global and local optima.
5. Scalable to different dimensions.
6. Reproducible random functions.
7. The software easily expandable and freely available.
8. Facilitates performance measures.

Several function generators have been previously presented in literature: The DF1 [10,11] and Moving Peaks [12] focus on generating dynamic multimodal landscapes, that change over time. Gaviano et.al. [13] generate differentiable multimodal functions by distorting convex functions by polynomials. Constrained test cases generator [14] generates function landscapes by dividing the search space to regions and constructing unimodal function for each, the main feature being the ability to define constraint functions for these regions. The Max set of Gaussians (MSG) landscape generator [15] and the family of generic hump

functions [16] combine several independent peaks to form the function landscape. Among all of the above, only the hump functions have been designed and analyzed in the context of multimodal optimization. The hump family has a limited usability (see section 2.4) and thus a more versatile testing environment is required, which is able to systematically generate a variety of highly tunable, scalable, and controllable multimodal test functions with multiple global optima.

2 The Function Generator

The proposed generator can be used to generate multimodal test functions for minimization, tunable with parameters to allow generation of landscape characteristics that are specifically designed for evaluating multimodal optimization algorithms by their ability to locate multiple optima. At the moment, three families of functions exist, but the generator is easily expandable and new families may be added in future. The current families are cosine, quadratic and common families.

2.1 Cosine Family

The cosine family samples two cosine curves together, of which one defines global minima and another adds local minima. The basic internal structure is regular: all minima are of similar size and shape and located in rows of similar distance from each other. The function family is defined by

$$f_{cos}(\boldsymbol{y}) = \frac{\sum_{i=1}^{D} -cos((G_i - 1)2\pi y_i) - \alpha \cdot cos((G_i - 1)2\pi L_i y_i)}{2D} \tag{1}$$

where $y_i \in [0, 1]^D$, the parameters $\boldsymbol{G}$ and $\boldsymbol{L}$ are vectors of positive integers which define the number of global and local minima for each dimension and α defines the amplitude of the sampling function (depth of the local minima).

The generator allows the function to be rotated to a random angle and use Bezier curves to stretch each dimension independently to decrease the regularity. To calculate the function value for input vector $\boldsymbol{x}$, it must be mapped to $\boldsymbol{y}$. The first step is to calculate $\boldsymbol{b}$, which is the rotated point corresponding to $\boldsymbol{x}$:

$$\boldsymbol{b} = \mathbf{O}\boldsymbol{x} \tag{2}$$

where the matrix $\mathbf{O} = [\boldsymbol{o}_1, \ldots, \boldsymbol{o}_D]$ is a randomly generated angle preserving orthogonal linear transformation as described in [17]. The domain of $\boldsymbol{x}$ (the search space) is the D-dimensional unit hypercube rotated with $\mathbf{O}$. Then $\boldsymbol{b}$ is mapped to $\boldsymbol{y}$ by applying the following Bezier formula:

$$y_i = \sum_{j=0}^{n_i} \binom{n_i}{j} P_{i,j} (1 - b_i)^{n_i - j} {b_i}^j, i = 1, \ldots, D \tag{3}$$

where n_i is the degree of the Bezier curve for dimension i and $\boldsymbol{P}_i$ are the control point vectors for Bezier stretching defined such that $P_{i,0}$ and P_{i,n_i} correspond to the lower and upper bound of y_i and the values between are strictly increasing.

The Bezier stretching will decrease the regularity of the function, but generally not completely eliminate it. The degree of regularity can be roughly estimated by considering the minimum amount of global minima points required to have a set which contains all possible differentials to jump from a neighboring minimum to the next in the axis directions. For completely regular functions, only $D+1$ points are required (as long as there are more than one minimum along each dimension). For stretched functions the required amount is $\sum_{i=1}^{D}(G_i - 1) + 1$ out of the total amount of global minima $\prod_{i=1}^{D} G_i$. So the degree of regularity increases along with the dimension. Bezier stretching also affects the shape and size of the minima, increasing the differences in their areas of attraction (AOA).

The amount of minima increases exponentially along dimension and the amount of local minima, which are not global is $\prod_{i=1}^{D} G_i + (G_i - 1)(L_i - 1)$. For each dimension, two of the outermost minima will always be located on the constraints. This means that if any of the values of $\boldsymbol{G}$ is less than 3, every minimum will be located on at least one constraint. In the unstretched case, each constraint the minimum sits on, halves the area of attraction (AOA) for that minimum compared to a minimum with one less constraint. The fraction of the AOA from full possible area is thus $1/2^{D-l}$, where $l = 0, 1, \ldots, D$ describes the amount of constraints the minimum sits on. $l = 0$ means a minimum located on no constraint (full possible AOA) and $l = D$ is a corner minimum with minimum AOA for dimension D. For methods that rely heavily on the initial points, locating the minima on corners becomes harder with increasing dimensionality, unless the information, that the minima are located on the constraints, is exploited.

Parameter α affects the depth of the local minima. Increasing the value makes the minima deeper, also increasing their area of attraction and thus slightly increasing the difficulty of the problem. Examples of cosine family functions are presented in Figure 1.

2.2 Quadratic Family

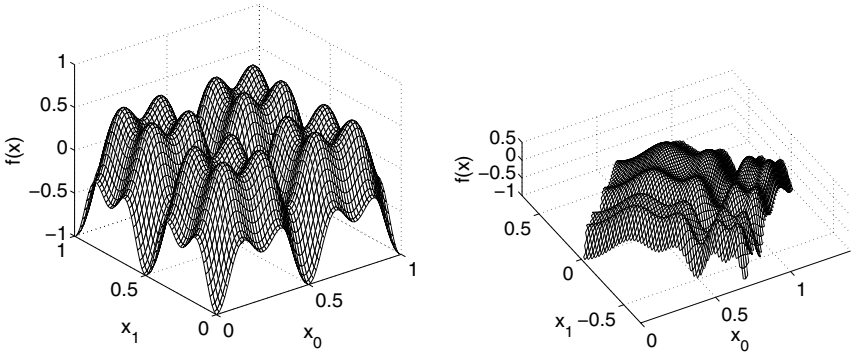
The quadratic family can be used to generate completely irregular landscapes. The function is created by combining several minima generated independently. They are described as a D dimensional general quadratic form, where a symmetric matrix $\mathbf{C}$ defines the shape of each minimum. The functions in quadratic family need not be stretched or rotated, because no additional benefit would be gained by doing that to an already irregular function. However, axis-aligned minima may be randomly rotated by rotating the matrix $\mathbf{C}$ as follows:

$$\mathbf{B} = \mathbf{OCO}^T \tag{4}$$

The functions are calculated by

$$f_{quad}(\boldsymbol{x}) = \min_{i=1,2,\ldots,q} \left((\boldsymbol{x} - \boldsymbol{p}_i)^T \boldsymbol{B}_i^{-1} (\boldsymbol{x} - \boldsymbol{p}_i) + v_i \right) \tag{5}$$

where $x_i \in [0, 1]^D$, $\boldsymbol{p}_i$ defines the location, and v_i the fitness value of a minimum point for i'th minimum. q is the amount of minima.



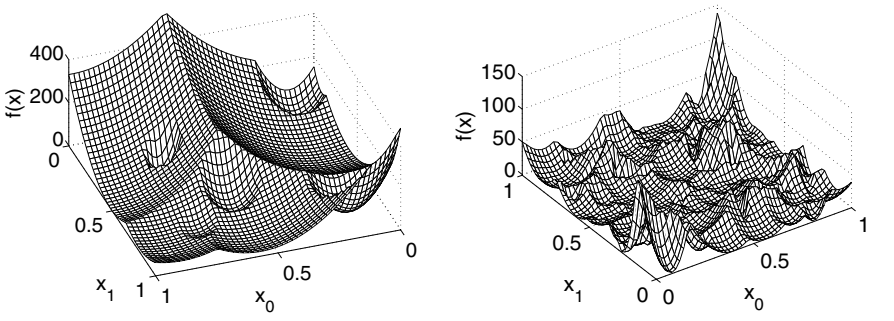
(a) Original (b) Stretched and rotated

Fig. 1. Example figures of cosine family functions using parameter values $\alpha = 0.8$, $\boldsymbol{G} = [3, 3]$, $\boldsymbol{L} = [2, 2]$ (9 global and 16 local minima). Additionally $\boldsymbol{P}_1 = [0, 0.1, 0.2, 0.5, 1]$ and $\boldsymbol{P}_2 = [0, 0.5, 0.8, 0.9, 1]$ are used for (b).

Unlike the other function families, the quadratic family allows the number of minima to be defined independently of the dimension and the user may select any number of global and local minima to be generated. The placement of minima is chosen randomly, although the minimum euclidean distance between global minima may be defined. The generator makes sure that no minimum is completely engulfed by another, deeper minimum. The user may also define the minima shape, which may be spherical, ellipsoidal or randomly rotated ellipsoidal. This selection is used for all generated minima. The shape range for global and local minima may be defined independently and the shapes are generated by using uniform random numbers from this range for each dimension in creating matrix $\mathbf{C}$. User may also define the range for the fitness values of local minima points. Figure 2 presents examples of functions from the quadratic family.

Because the locations of minima are random, and the minima will overlap, the sizes of the areas of attraction for different minima will vary greatly. Using fewer or shallower local minima will naturally leave more room for the global minima. Also forcing longer minimum euclidean distance between the global minima will leave more area for each minimum. As the dimensionality increases, the differences in shape parameters will have exponentially increasing effect to the AOAs. Thus the significance of shape range parameters will increase along dimension in deciding the problem difficulty as rather small differences in shape range can lead to large differences to the relative sizes of the AOAs in high dimensions.

To demonstrate this, random start gradient descent (RSGD) algorithm was run on 3 different sets of quadratic functions, with $D = 1, 2, \ldots, 10$. The algorithm generates random starting points and descends downhill by performing line searches in numerically estimated gradient direction using bracketing [18, p. 400] and Brent's method [18, p. 404] until all minima have been found. The



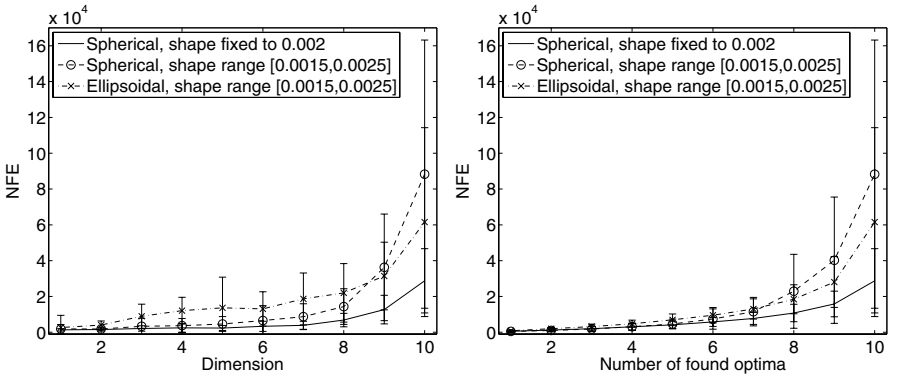
(a) Spherical, no local minima (b) Rotated ellipsoidal with local minima

Fig. 2. Examples of quadratic family functions. The minimum euclidean distance between global minima is set to 0.01, and shape range for all minima to [0.003,0.03]. The figure a has 10 global spherical minima, while figure b has 10 global and 100 local rotated ellipsoidal minima such that the local minima points have fitness value in range [-0.95,-0.15]. Globally optimal value is always -1.

average performance of RSGD is a decent estimate of the relative sizes of AOAs, because to locate a minimum, the random starting point must be located on the AOA of that particular minimum. If the sizes differ significantly, the larger ones will draw more points, slowing down the process of locating the minima with smaller AOA. Figure 3 displays the results of the runs. The first set of functions has spherical minima with identical shape. The second has also spherical and third an ellipsoidal shape, both allowing the shape range to change by $\pm 50\%$. As can be seen, the required number function evaluations (NFE) as well as the standard deviation increase notably slower along dimension for the first set compared to the other two. This is expected, because the identical shape eliminates the differences on AOA caused by shape. The ellipsoidal shape is hardest on low dimension, but varying spherical shape claims the title on $D > 8$. This is logical, because for spherical shape only one random value is generated, which is then used in all dimensions. If the value is small, it will affect all dimensions. For the ellipsoidal shape, each dimension will get a different random value and the total AOA averages better as more values are generated on higher dimensions. The slower performance on low dimensional ellipsoidal set can be explained by the fact that gradient descent tends to oscillate on non-spherical shapes and thus needs more line searches to find the minimum point compared to spherical shapes, where the gradient points directly to the minimum point.

2.3 Common Family

The common family collects some well known test problems with multiple global optima from literature and implements them inside the generator. The generator allows similar rotation and Bezier stretching, as used in cosine family, for all



(a) Function evaluations required to find all minima with accuracy 0.0001

(b) Function evaluations required to find i'th minimum in 10 dimensional case

Fig. 3. Performance of RSGD on different quadratic functions with 10 global and no local minima. The minimum euclidean distance between global minima is set to 0.1. The function evaluations are averages from 100 independent runs and figures include standard deviations. For each run, different random seed is used in generating the function.

implemented functions in the common family (the $\boldsymbol{x}$ is mapped to $\boldsymbol{y}$ using equations 2 and 3). At the moment, eight different functions are included. One of the functions is a 2 dimensional modification of the well known Rastrigin function [19] to have 4 global and 96 local minima. The function is calculated by formula:

$$f_{ras}(\boldsymbol{y}) = 20 + \sum_{i=1}^{2} \left(y_i^2 + 10\cos(2\pi y_i)\right) \tag{6}$$

where $y_i \in [-5.12, 5.12]^2$. Other implemented two dimensional functions are: Branin [20], Himmelblau [5], Shubert [8], and the Six-hump camel back [21,20]. Additionally Vincent [22] and Deb's 1st and 3rd function [5] have been included with the possibility to scale the dimensionality.

2.4 Comparison to Other Test Function Generators

Singh and Deb recently proposed a scalable multimodal test function family called generic hump functions [16]. Like quadratic family, the hump functions allow irregular landscapes to be generated and the amount of optima to be defined independently of the dimensionality. However, the hump function is defined such that each optimum peak has a fixed radius value and any values outside that are set to 0 and no two peaks can intersect. So the function surface is flat except the peaks. This kind of surface is problematic especially for methods relying on gradient information, because depending on the size and amount of the peaks, large portion of the search space is flat and gradient thus unusable.

However, the problem does not only affect gradient reliant methods, because no method can deduct meaningful direction information from a flat surface. In an extreme case, with very small radius of the peaks, we end up with a needle in a haystack problem, where the vast majority of the search space is flat, only including some very thin peaks. For such a problem, an enumeration technique would be the optimal strategy, because the only viable information available is not to visit the same location twice. In contrast, for the quadratic family, there are no flat areas in the search space and the peaks may freely intersect, making it possible to have different peak inside another peak's area of attraction. Also, for any point, a meaningful numerical estimate for gradient can be calculated. Still, when a reasonably large width for peaks is used, the hump family could provide an interesting environment to test algorithm's ability to handle flat surfaces. For this reason, the hump family would be an interesting addition to the presented generator in future.

The MSG proposed by Gallagher and Yuan [15], is another interesting test function generator. They use a similar approach, as used in quadratic family, to produce peaks independently and use the dominant one to define the function value in point $\boldsymbol{x}$. However, MSG uses Gaussian density function formula for defining the peaks, which is an exponential function. In comparison, for quadratic family, general quadratic form was selected (which is similar to the exponent part of the Gaussian density function, when the constant is removed) to describe the minima. As a result, the landscapes generated by MSG have much steeper optima shapes compared to the ones generated by quadratic family. In theory, both approaches allow landscapes with no flat areas to be generated. However, a potential problem in using landscapes with steep optima is the numerical accuracy: Because the values ascend towards zero very fast as we move away from an optimum, it is possible that areas are left in the search space where a numerical approximation of gradient rounds to zero, in practice making the area flat from an algorithm's perspective. When using the quadratic form, this is usually not an issue. Of course, the MSG generator is primarily designed for producing landscapes with only one global optimum, while the generator presented here concentrates on cases having multiple global optima, and thus the features included in the generators differ accordingly.

2.5 Implementation and Features

The generator includes an internal constraint handling method to keep the solutions at a given range. The constraints are handled by mirroring the violating value back from the violated boundary by the amount of violation. This makes the function space look continuous for the optimization approach during the run, because any minimum which is located on the boundary looks symmetrical, although in reality the value is calculated in a mirrored point inside the boundaries. The internal constraint handling can be ignored and the linear constraint functions (relevant in rotated cases) can be acquired in analytical form, if required.

Additionally the generator offers a method for deciding how many different globally minimal solutions a given population includes, with a required accuracy. Other useful features included are the possibility of initializing a population with uniform random values in a proper range for the used function, an internal counter for function evaluations and the ability to acquire the amount of minima and the exact locations of global minima. For functions that include random numbers, user may provide a seed for the internal random number generator. With the same seed, the same function is always generated.

The software is written in C programming language obeying ANSI standard and should be easy to import in most optimization algorithms written in C or C++. The generator package is freely available from: `http://www.ronkkonen.com/generator/`, and includes source codes, a simple plotter program written in Matlab for visualizing the 2D functions generated, and detailed documentation.

3 Conclusion and Future Work

In this paper, a function generator module for multimodal problems including several global optima was presented. The generator allows an easy way to construct parameterizable functions and offers a standard environment for testing multimodal optimization algorithms. To the author's knowledge, no versatile generator designed especially for producing problems with multiple global optima exists and the proposed generator aims to mend this deficiency. In addition to offering two novel families of parameterizable functions, the generator implements several well known test problems and allows an easy option to modify them by rotating and stretching. The framework allows an easy addition of new families to the module. We plan to add a variant of the hump family presented by Deb [16] in near future. Also, a variant of the current cosine family, where the global optima would not be located on the constraints, would offer a different challenge for methods that rely on constraints and initial population. For the quadratic family, an option could be added to give the global optima locations as parameters, instead of always generating them randomly.

Acknowledgments

The authors want to thank Harri Lattu and Jarmo Ilonen for their help on implementing the generator software and Jouni Sampo for mathematical consultation.

References

1. Mahfoud, S.: A comparison of parallel and sequential niching methods. In: Proceedings of 6th International Conference on Genetic Algorithms, pp. 136–143 (1995)
2. De Jong, K.: An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan (1975)

3. Goldberg, D., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Grefenstette, J. (ed.) Proc. of the Second International Conference on Genetic Algorithms, pp. 41–49 (1987)
4. Mahfoud, S.: Niching methods for genetic algorithms. PhD thesis, Urbana, IL, USA (1995)
5. Beasley, D., Bull, D., Martin, R.: A sequential niche technique for multimodal function optimization. Evolutionary Computation 1(2), 101–125 (1993)
6. Harik, G.: Finding multimodal solutions using restricted tournament selection. In: Eshelman, L. (ed.) Proc. of the Sixth International Conference on Genetic Algorithms, pp. 24–31. Morgan Kaufmann, San Francisco (1995)
7. Pétrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: Proc. of the 3rd IEEE International Conference on Evolutionary Computation, pp. 798–803 (1996)
8. Li, J., Balazs, M., Parks, G., Clarkson, P.: A species conserving genetic algorithm for multimodal function optimization. Evol. Comput. 10(3), 207–234 (2002)
9. Wolpert, D., Macready, W., William, G.: No free lunch theorems for search. Technical report, The Santa Fe Institute (1995)
10. Morrison, R., Jong, K.D.: A test problem generator for nonstationary evironments. In: Proceedings of the Congress of Evolutionary Computation, Piscataway, NJ, pp. 1843–1850. IEEE Press, Los Alamitos (1999)
11. Morrison, R.: Designing Evolutionary Algorithms for Dynamic Environments. Springer, Berlin (2004)
12. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Norwell (2002)
13. Gaviano, M., Kvasov, D., Lera, D., Sergeyev, Y.: Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. ACM Transactions on Mathematical Software 29(4), 469–480 (2003)
14. Michalewicz, Z., Deb, K., Schmidt, M., Stidsen, T.: Test-case generator for nonlinear continuous parameter optimization techniques. IEEE Trans. on Evol. Comput. 4, 197–215 (2000)
15. Gallagher, M., Yuan, B.: A general-purpose tunable landscape generator. IEEE Transactions on Evolutionary Computation 10, 590–603 (2006)
16. Singh, G., Deb, K.: Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, Seattle, WA, pp. 1305–1312. ACM Press, New York (2006)
17. Hansen, N., Ostermeier, A.: Completely derandomized self adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
18. Press, W., Flannery, B., Teukolsky, S., Vetterling, W.: Numerical Recipes in C, 2nd edn. Cambridge University Press, Cambridge (1992)
19. Törn, A., Žilinskas, A. (eds.): Global Optimization. LNCS, vol. 350. Springer, Heidelberg (1989)
20. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Heidelberg (1996)
21. Ursem, R.: Multinational evolutionary algorithms. In: Proceedings of Congress of Evolutionary Computation (CEC 1999), vol. 3. IEEE Press, Los Alamitos (1999)
22. Shir, O., Bäck, T.: Niche radius adaptation in the cma-es niching algorithm. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 142–151. Springer, Heidelberg (2006)