Hindawi Publishing Corporation
Advances in Artificial Intelligence
Volume 2012, Article ID 540861, 7 pages
doi:10.1155/2012/540861

Research Article

A Novel Approach to Improve the Performance of Evolutionary Methods for Nonlinear Constrained Optimization

Alireza Rowhanimanesh and Sohrab Efati

Center of Excellence on Soft Computing and Intelligent Information Processing (SCIIP), Ferdowsi University of Mashhad, Mashhad, Iran

Correspondence should be addressed to Alireza Rowhanimanesh, rowhanimanesh@ieee.org

Received 24 May 2012; Revised 16 July 2012; Accepted 16 July 2012

Academic Editor: Joanna Józefowska

Evolutionary methods are well-known techniques for solving nonlinear constrained optimization problems. Due to the exploration power of evolution-based optimizers, population usually converges to a region around global optimum after several generations. Although this convergence can be efficiently used to reduce search space, in most of the existing optimization methods, search is still continued over original space and considerable time is wasted for searching ineffective regions. This paper proposes a simple and general approach based on search space reduction to improve the exploitation power of the existing evolutionary methods without adding any significant computational complexity. After a number of generations when enough exploration is performed, search space is reduced to a small subspace around the best individual, and then search is continued over this reduced space. If the space reduction parameters (red_gen and red_factor) are adjusted properly, reduced space will include global optimum. The proposed scheme can help the existing evolutionary methods to find better near-optimal solutions in a shorter time. To demonstrate the power of the new approach, it is applied to a set of benchmark constrained optimization problems and the results are compared with a previous work in the literature.

1. Introduction

A significant part of today's engineering problems are constrained optimization problems (COP). Although there exist efficient methods like Simplex for solving linear COP, solving nonlinear COP (NCOP) is still open for novel investigations. Different methods have been proposed for solving NCOP. Among them, natural optimization and especially population-based schemes are the most general and promising ones. These methods can be applied to all types of COP including convex and nonconvex, analytical and non-analytical, real-, integer- and mixed-valued problems. One of the most applied techniques for solving NCOP are evolutionary methods.

Various techniques have been introduced for handling nonlinear constrains by evolutionary optimization (EO) methods. These approaches can be grouped in four major categories [1, 2]: (1) methods based on penalty functions that are also known as indirect constraint handling, (2) methods based on a search of feasible solutions including repairing unfeasible individuals [3, 4], superiority of feasible points [5], and behavioral memory [6], (3) methods based on preserving feasibility of solutions like preserving feasibility by designing special crossover and mutation operators [7], the GENOCOP system [8], searching the boundary of feasible region [9], and homomorphous mapping [10], and (4) Hybrid methods [11–13]. Also, decoding such as transforming the search space can be considered as the fifth category which is less common. None of these approaches are complete and each of them has both advantages and weak-points. For example, although the third method (preserving feasibility) might perform very well, it is usually problem dependent and designing such a method for a given problem may be difficult, computationally expensive, and sometimes impossible. Among these approaches, the most general one is the first technique.

Penalty-based constraint handling incorporates constraints into a penalty function that is added to the main

fitness function. By this work, the main constrained problem is converted to an unconstrained problem. The main advantage of this method is its generality and simplicity (problem-independent penalty functions). Thus, this method is known as the most common approach for handling nonlinear constraints in EO.

Adding a penalty function to a fitness (objective) function creates a new unconstrained problem that might have further complexity. The introduction of penalties may transform a smooth objective function into a rugged one and the search may then become more easily trapped in local optimum [14]. Therefore, several penalty-based constraint handling methods have been proposed to improve the performance of penalty-based constrained evolutionary optimization. In [2], a survey has been performed on several types of these methods including death penalty [2, 15], static penalty [16, 17], dynamic penalty [18, 19], annealing penalty [20, 21], adaptive penalty [22–24], segregated GA [25], and coevolutionary penalty [26]. In addition to these types, other hybrid (e.g., niched-penalty approach [27]) and heuristic techniques (e.g., stochastic ranking [28]) could be found in the literature.

Due to its generality and applicability, this paper focuses on penalty-based constraint handling without loss of generality. However, the proposed approach is independent from the type of constraint handling and optimization technique. This paper demonstrates how the power of exploitation of constrained EO (CEO) can be increased by reducing the search space after enough exploration is performed. The proposed approach is simple and general and does not add any computational complexity to the original algorithm. Also, it could be applied to other optimization techniques like constrained PSO and hybrid methods.

This paper is organized as follows. In Section 2, the proposed approach is described and the details are explained and illustrated over a specific constrained optimization problem introduced in [29]. In Section 3, the performance of the proposed scheme is tested on eleven well-known test problems and the results are compared with [10].

2. Proposed Approach

A general constrained nonlinear programming problem is formulated as follows:

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g_i(x) \le 0, \quad i = 1, 2, \ldots, p \\ &&& h_i(x) = 0, \quad i = p+1, \ldots, m \\ &&& l_j \le x_j \le u_j, \quad j = 1, \ldots, n, \end{aligned} \tag{1}$$

where $x = (x_1, x_2, \ldots x_n)$ is the vector of decision variables, $f(x)$ is a scalar lower-bounded objective function, $\{g_i(x), i = 1, \ldots, p\}$ is a set of p inequality constraints, $\{h_i(x), i = p+1, \ldots, m\}$ is a set of m- p equality constraints, and $[l_j, u_j]$ is the domain of the ith variable. $f(x)$, $g_i(x)$, and $h_i(x)$ are allowed to be either linear or nonlinear, convex or nonconvex, differentiable or nondifferentiable, continuous or discrete, and analytical or nonanalytical. Also, x can be either discrete or continuous or mixed-valued. Without loss of generality, the problem is considered minimization since $\max f(x)$ is equivalent to $-\min(-f(x))$. As mentioned in [10], it is a common practice to replace equation $h_i(x) = 0$ by the inequality $|h_i(x)| \le \delta$ for some small $\delta > 0$. This replacement is considered in the rest of this paper, and, consequently, the above problem consists of m inequality constraints. For simplicity, all of the constraints are shown by $g_i(x) \le 0$ where $i = 1, 2, \ldots, m$. Note that bound constraints $l_j \le x_j \le u_j$ can be directly handled in most of the population-based optimization methods such as EO. Thus, only $g_i(x) \le 0$ $(i = 1, 2, \ldots, m)$ are considered as the constraints. Without loss of generality, to simplify the understanding of the proposed idea, its details are explained over a specific constrained optimization problem as an illustrative example chosen from the literature.

2.1. An Illustrative Example. Consider the following constrained nonlinear programming problem [29]:

$$\begin{aligned} \text{Minimize} \quad & f(x) = \left({x_1}^2 + x_2 - 11\right)^2 + \left(x_1 + {x_2}^2 - 7\right)^2 \\ \text{s.t.} \quad & -4.84 + (x_1 - 0.05)^2 + (x_2 - 2.5)^2 \le 0 \longrightarrow g_1(x) \le 0 \\ & -{x_1}^2 - (x_2 - 2.5)^2 + 4.84 \le 0 \longrightarrow g_2(x) \le 0 \\ & 0 \le x_i \le 6, \quad i = 1, 2. \end{aligned} \tag{2}$$

The feasible space and the global minimum of this problem are displayed in Figure 1. In this paper, a simple constrained GA (CGA) has been used for solving constrained optimization problems. This CGA is composed of a simple real-valued GA introduced in [30] and a static penalty function that is added to the original objective (fitness) function as follows:

$$f_p(x) = f(x) + p \sum_{i=1}^{m} \max\{0, g_i(x)\}. \tag{3}$$

In this equation, $\max\{0, g_i(x)\}$ gives the violation value of the constraint $g_i(x) \le 0$, and p is the penalty coefficient that must be determined by the designer. In some cases, assigning a proper value to p is difficult and this is known as the main disadvantage of static penalty functions. As an alternative, adaptive [22–24] and dynamic [18, 19] penalty functions have been proposed. By defining $f_p(x)$, the constrained nonlinear programming problem of (2) is converted to the following unconstrained nonlinear programming problem:

$$\begin{aligned} \text{Minimize} \quad & f_p(x) = f(x) + p \sum_{i=1}^{2} \max\{0, g_i(x)\} \\ \text{s.t.} \quad & 0 \le x_i \le 6, \quad i = 1, 2. \end{aligned} \tag{4}$$

This problem (4) is solved by the above-mentioned CGA. The CGA routine utilizes a simple real-valued GA introduced in [30] with population size of 50, selection rate of 0.5, elitism with elite rate of 0.05, single-point crossover, and uniform mutation with mutation rate of 0.2. The value of penalty coefficient is 10 and the maximum number of

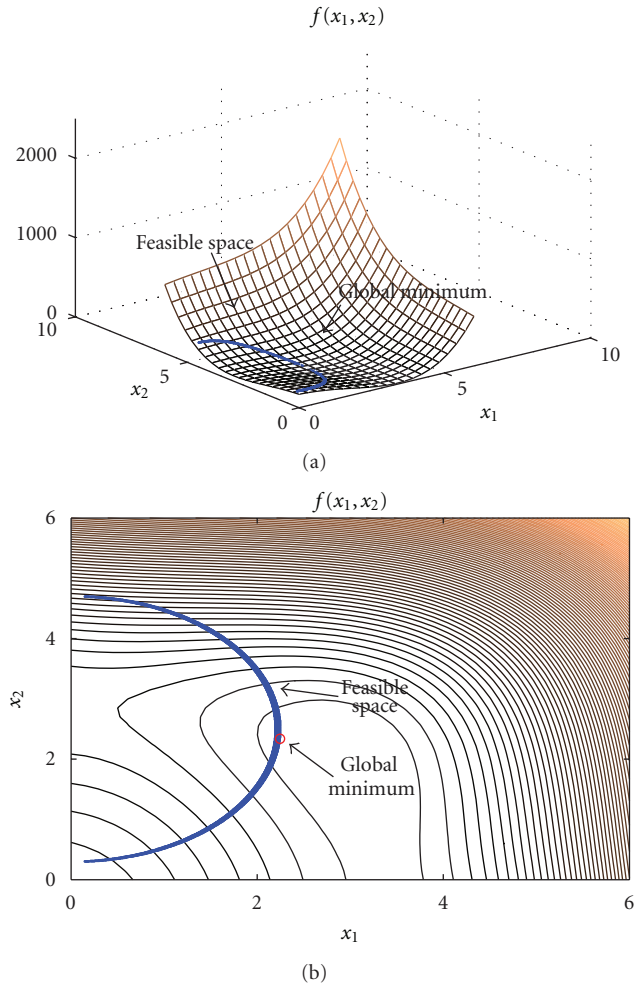

Figure 1: The feasible space and the global minimum of the problem of (2).

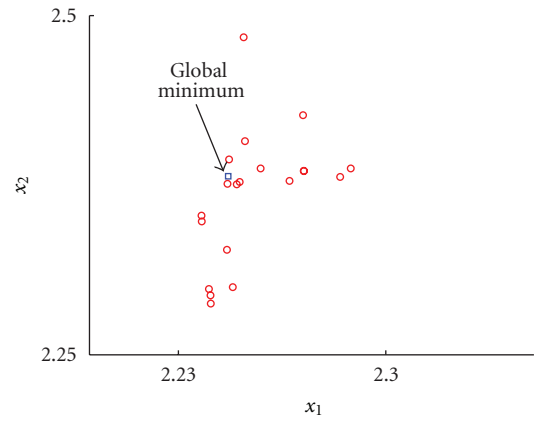

Figure 2: The best individuals (at generation 10) of 20 independent runs and the global minimum.

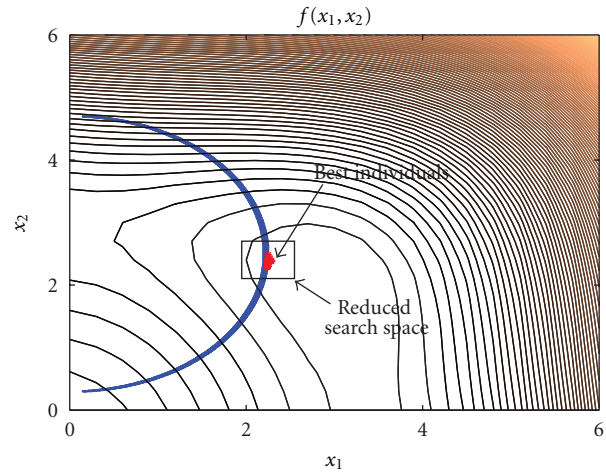

Figure 3: The original search space can be reduced to a small subspace around the global optimum after a number of generations when enough exploration has been performed.

generations is allowed to be 10. Figure 2 indicates the best individuals (solutions) of 20 independent runs. Regarding this figure, all of these individuals have been converged to the global minimum through 10 generations. Figure 3 illustrates these individuals on the original search space. A new idea is emerged from considering these figures, that is, the best individual will converge to the global optimum after a number of generations. In other words, the global optimum will locate in a small neighborhood of the best individual after a number of generations. Consequently, after a number of generations when enough exploration has been performed, the search space can be reduced to a small subspace around the best individual, and then the search can be continued over this reduced space. The experimental study of the next section demonstrates that if the space reduction parameters (red_gen and red_factor) are adjusted properly, the reduced space will include the global optimum. Here, the reduced search space is defined as follows:

$$\begin{aligned} \text{reduced\_}l_i = \max\{&\text{best\_ind}(\text{red\_gen}) \\ &-\text{red\_factor} \cdot (u_i - l_i), l_i\}, \quad i = 1, \ldots, n, \end{aligned}$$

$$\begin{aligned} \text{reduced\_}u_i = \min\{&\text{best\_ind}(\text{red\_gen}) \\ &+\text{red\_factor} \cdot (u_i - l_i), u_i\}, \quad i = 1, \ldots, n. \end{aligned} \tag{5}$$

In these equations, $[l_i, u_i]$ and $[\text{reduced\_}l_i, \text{reduced\_}u_i]$ are the original and reduced domains of variables, respectively. best_ind(red_gen) is the best individual of the generation red_gen. red_gen specifies the generation in which the reduction of search space is performed. Here, this search space reduction is done only one time at generation red_gen. The size of the reduced search space is determined by red_factor that varies between 0 and 1. Also, the value of mutation rate can be changed after this reduction.

Indeed, the value of red_gen determines the total number of generations in CGA (constrained GA) that are devoted to exploration in order to find a region around of the global minimum. The value of red_gen determines a tradeoff between exploration and exploitation and highly depends on the general characteristics of the given optimization problem.

Table 1: A comparison between the proposed approach, [22] and [29] for the same number of function evaluations in solving the problem of (2). The best results are indicated in boldface.

	Best	Mean	Worst
Proposed approach	**13.590846**	**13.61073**	**13.84861**
[29]	13.59658	—	244.11616
[22]	13.59085	30.74880	152.54840

If the global minimum is inside a narrow (sharp) valley, CGA usually needs more time to find the global valley, and thus a large value of red_gen should be considered. Also, since the valley is narrow, red_factor could be set to a small value. In contrast, if the global minimum is inside a wide (flat) valley, CGA usually needs less time to find the global valley, and therefore smaller values can be used for red_gen. Also, since the valley is wide, red_factor should be large. The value of red_gen is usually increased by increasing the number of decision variables. In most of the optimization problems, since the general characteristics of the given problem can be approximately identified by the designer, some appropriate values for red_gen and red_factor can be heuristically guessed by the designer. In general, red_gen and red_factor should be large enough to guarantee that the global optimum is included in the reduced search space.

For this example, the reduced search space is shown in Figure 3 for red_factor = 0.05. The value of red_gen must be determined by user. However, systematic or adaptive schemes can be developed in further investigations. The value of red_gen influences the power of exploration. This value should be large enough to guarantee that after red_gen number of generations, the best individual and the global optimum are close together. Moreover, the size of the reduced search space (the value of red_factor) must be large enough to include the global optimum.

This example has been solved by the proposed approach using the above-mentioned CGA. Here, the value of penalty coefficient is 20, the maximum number of generations is 50, red_factor = 0.05, and and red_gen = 5. The mutation rate is 0.2 for the first five generations and after Generation 5 (red_gen), it is changed into 0.05. The results of 50 independent runs are displayed in Table 1. The same experiment has been also done by [22, 29] with the same number of function evaluations. Table 1 compares the results of the proposed approach with these two references where the best results are indicated in boldface.Although the proposed approach has been incorporated with a simple method of CEO which is trivial in contrast to the other existing techniques of CEO in the literature, it could get better results in this example for the same number of function evaluations in comparison with [22, 29].

According to the above-mentioned explanations, the proposed method and formulation of this paper can be directly applied to different continuous (real-valued) optimization problems and incorporated with different optimization methods. Moreover, since the main contribution of this paper is the notion of search space reduction, that is general and problem-independent, the proposed approach is flexible enough to be generalized (with minor changes) to discrete and mixed-valued optimization problems such as combinatorial optimization problems where the domain is not continuous. In the next section, the performance of the new scheme is examined by a set of eleven well-known continuous test problems.

3. Experimental Study

To demonstrate the power of the proposed approach to improve the performance of constrained evolutionary methods, it is incorporated with the simple CEO of the previous section and applied to a set of eleven well-known test problems introduced in [1, 10]. These test problems consist of objective functions of different types (linear, quadratic, cubic, polynomial, and nonlinear) and various types and numbers of constraints (linear inequalities, nonlinear equations, and inequalities). The ratio between the size of the feasible search space $|F|$ and the size of the whole search space $|S|$ for these test cases vary from 0% to almost 100% and the topologies of feasible search spaces are also quite different [10]. Table 2 summarizes the main characteristics of these test cases.

The CGA routine is the same as the previous section except for the following parameters. Here, population size is 70 and the maximum number of generations is 5000. The values of mutation rate, red_gen, red_factor, and penalty coefficient are different for each test case as shown in Table 3. Equality constraints ($h(x) = 0$) were converted into inequality constraints by $\delta = 0.0001$ ($|h(x)| \leq \delta$). For each test case, 20 independent runs were performed. The same experiment has been done on this test set in [10]. It should be mentioned that the authors in [10] utilizes the third method of constraint handling (preserving feasibility by homomorphous mapping). Table 4 indicates the results of the proposed approach and its comparison with [10] for the same number of function evaluations. The best results are indicated in Boldface. For problem G5, we could not find any suitable static penalty coefficient for handling the constraints. This is due to the inadequacy of static penalty functions. However, this problem could not be solved by [10] too. It should be mentioned that this test case (G5) has been solved in some other references [22, 28].

According to comparative results of Table 4, although the proposed approach had been incorporated with a simple method of CEO which is trivial in contrast to the other existing techniques of CEO in the literature, its performance (finding a better near-optimal solution) is better than [10] in 70% of cases in this benchmark experimental study. Since the number of function evaluations is the same in this study and in the study of [10], it can be concluded that the proposed approach can also improve the efficiency (convergence speed) of the CEO in 70% of cases in contrast to [10].

Table 2: Summary of eleven test problems [10].

	Type of problem	n	Type of f	Linear inequality	Nonlinear equality	Nonlinear inequality	$\lvert F\rvert/\lvert S\rvert$	Optimum value
G1	Minimum	13	Quadratic	9	0	0	0.0111%	−15.0
G2	Maximum	20	Nonlinear	0	0	2	99.8474%	0.803553
G3	Maximum	10	Polynomial	0	1	0	0.0000%	1.0
G4	Minimum	5	Quadratic	0	0	6	52.1230%	−30655.5
G5	Minimum	4	Cubic	2	3	0	0.0000%	5126.4981
G6	Minimum	2	Cubic	0	0	2	0.0066%	−6961.8
G7	Minimum	10	Quadratic	3	0	5	0.0003%	24.306
G8	Maximum	2	Nonlinear	0	0	2	0.8560%	0.0958250
G9	Minimum	7	Polynomial	0	0	4	0.5121%	680.63
G10	Minimum	8	Linear	3	0	3	0.0010%	7049.33
G11	Minimum	2	Quadratic	0	1	0	0.0000%	0.75

Table 3: The values of mutation rate, red_gen, red_factor, and penalty coefficient for each test problem.

	Mutation rate (before and after red_gen)	red_gen	red_factor	Penalty coefficient
G1	0.2–0.05	1000	0.05	10
G2	0.2–0.05	1500	0.1	10
G3	0.2–0.1	2000	0.1	1000
G4	0.2–0.05	1000	0.05	1500
G5	—	—	—	—
G6	0.2–0.1	1000	0.02	10000
G7	0.2–0.05	2000	0.05	10
G8	0.2–0.05	1000	0.05	1000
G9	0.2–0.05	1000	0.05	10
G10	0.2–0.1	2500	0.2	15000
G11	0.2–0.05	1000	0.05	10

Table 4: The performance of the proposed approach in comparison with [10] on eleven test problems. The number of function evaluations is the same in both studies. The best results are indicated in boldface.

		Best	Mean	Worst	Optimal
G1	Proposed approach	**−14.99145**	**−14.96119**	**−14.81634**	−15.0
	[10]	−14.7207	−14.4609	−14.0566	
G2	Proposed approach	0.78727	0.74244	0.67530	0.803553
	[10]	**0.79506**	**0.79176**	**0.78427**	
G3	Proposed approach	0.98704	0.92063	0.72812	1.0
	[10]	**0.9983**	**0.9965**	**0.9917**	
G4	Proposed approach	**−30665.259**	**−30662.639**	**−30648.807**	−30655.5
	[10]	−30662.5	−30643.8	−30617.0	
G5	Proposed approach	—	—	—	5126.4981
	[10]	—	—	—	
G6	Proposed approach	**−6917.85904**	**−6862.02084**	**−6425.38018**	−6961.8
	[10]	−6901.5	−6191.2	−4236.7	
G7	Proposed approach	**24.52525**	**26.12999**	**29.24032**	24.306
	[10]	25.132	26.619	38.682	
G8	Proposed approach	**0.09582504**	**0.09582504**	**0.095825036**	0.0958250
	[10]	0.095825	0.0871551	0.0291434	
G9	Proposed approach	**680.74163**	**681.00480**	**681.53181**	680.63
	[10]	681.43	682.18	682.88	
G10	Proposed approach	**7132.98320**	**7543.48592**	**8845.85330**	7049.33
	[10]	7215.8	9141.7	11894.5	
G11	Proposed approach	**0.75**	0.75085	0.75655	0.75
	[10]	**0.75**	**0.75**	**0.75**	

4. Conclusions

This paper proposes a general and computationally simple approach to improve the performance of evolution-based optimization methods for solving nonlinear constrained optimization problems. After a number of generations when enough exploration is performed, the search space is reduced to a small subspace around the best individual, and the search is continued over this reduced space. If the reduction parameters (red_gen and red_factor) are adjusted properly, the reduced search space will include the global optimum. Here, this method was incorporated with a simple constrained GA and its performance was tested and compared with the method in [10] on a set of eleven benchmark test problems. The comparative results of the experimental study demonstrate that the proposed approach can considerably improve the performance (finding better near-optimal solutions) and efficiency (convergence speed) of the simple constrained GA in comparison with [10] without adding any considerable computational complexity to the original algorithm. The proposed scheme is general and can be incorporated with other population-based optimization methods for solving nonlinear programming problems.

References

[1] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[2] Ö. Yeniay, "Penalty function methods for constrained optimization with genetic algorithms," *Mathematical and Computational Applications*, vol. 10, no. 1, pp. 45–56, 2005.

[3] Z. Michalewicz and G. Nazhiyath, "Genocop III: a co-evolutionary algorithm for numerical optimization problems with nonlinear constraints," in *Proceedings of the 2nd IEEE International Conference on Evolutionary Computation*, pp. 647–651, December 1995.

[4] A. Rowhanimanesh, A. Khajekaramodin, and M.-R. Akbarzadeh-T, "Evolutionary constrained design of seismically excited buildings, actuators placement," in *Proceedings of the 1st Joint Congress on Intelligent and Fuzzy Systems (ISFS '07)*, pp. 297–304, Mashhad, Iran, 2007.

[5] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.

[6] M. Schouenauer and S. Xanthakis, "Constrained GA optimization," in *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 473–580, 1993.

[7] A. Rowhanimanesh, A. Khajekaramodin, and M.-R. Akbarzadeh-T, "Evolutionary constrained design of seismically excited buildings: sensor placement," *Applications of Soft Computing*, vol. 58, pp. 159–169, 2009.

[8] Z. Michalewicz and C. Z. Janikow, "Handling constraints in genetic algorithms," in *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 151–157, 1993.

[9] M. Schoenauer and Z. Michalewicz, "Evolutionary computation at the edge of feasibility," in *Proceedings of the 4rth International Conference on Parallel Problem Solving from Nature*, pp. 22–27, 1996.

[10] S. Koziel and Z. Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," *Evolutionary Computation*, vol. 7, no. 1, pp. 19–44, 1999.

[11] H. Adeli and N. T. Cheng, "Augmented lagrangian genetic algorithm for structural optimization," *Journal of Aerospace Engineering*, vol. 7, no. 1, pp. 104–118, 1994.

[12] B. W. Wah and Y. Chen, "Hybrid constrained simulated annealing and genetic algorithms for nonlinear constrained optimization," in *Proceedings of the Congress on Evolutionary Computation*, vol. 2, pp. 925–932, May 2001.

[13] J. H. Kim and H. Myung, "Evolutionary programming techniques for constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 2, pp. 129–140, 1997.

[14] T. P. Runarsson and X. Yao, "Constrained evolutionary optimization: the penalty function approach," in *Evolutionary Optimization*, R. Sarker, M. Mohammadian, and X. Yao, Eds., pp. 87–113, Kluwer Academic Publishers, 2002.

[15] T. Bäck, F. Hoffmeister, and H. P. Schwell, "A survey of evolution strategies," in *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 2–9, 1991.

[16] A. Homaifar, S. H. Y. Lai, and X. Qi, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–254, 1994.

[17] M. A. Kuri and C. C. Quezada, "A universal eclectic genetic algorithm for constrained optimization," in *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing*, pp. 518–522, 1998.

[18] J. A. Joines and C. R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pp. 579–584, June 1994.

[19] S. Kazarlis and V. Petridis, "Varying fitness functions in genetic algorithms: studying the rate of increase in the dynamic penalty terms," in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pp. 211–220, 1998.

[20] F. Mendivil and R. Shonkwiler, "Annealing a genetic algorithm for constrained optimization," *Journal of Optimization Theory and Applications*, vol. 147, no. 2, pp. 395–410, 2010.

[21] Z. Michalewicz and N. Attia, "Evolutionary optimization of constrained problems," in *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pp. 98–108, 1994.

[22] H. J. C. Barbosa and A. C. C. Lemonge, "An adaptive penalty scheme in genetic algorithms for constrained optimization problems," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02)*, pp. 287–294, 2002.

[23] M. Gen and R. Cheng, "A survey of penalty techniques in genetic algorithms," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 804–809, May 1996.

[24] A. Smith and D. Tate, "Genetic optimization using a penalty function," in *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 499–503, 1993.

[25] R. Le Riche, C. Knopf-Lenior, and R. T. Haftka, "A segregated genetic algorithm for constrained structural optimization," in *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 558–565, 1995.

[26] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Computers in Industry*, vol. 41, no. 2, pp. 113–127, 2000.

[27] K. Deb and S. Agarwal, "A niched-penalty approach for constraint handling in genetic algorithms," in *Proceedings of*

the International Conference on Adaptive and Natural Computing Algorithms (ICANNGA '99), Portoroz, Slovenia, 1999.

[28] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp. 284–294, 2000.

[29] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, 2000.

[30] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, John Wiley & Sons, 2004.