Experimental Comparison of Constraint Handling Schemes in Particle Swarm Optimization

Mehdi Rostamian, Ali R. Kashani, Charles V. Camp, and Amir H. Gandomi

Abstract Nature-inspired optimization algorithms have been designed for unconstrained problems. However, real-world optimization problems usually deal with a lot of limitations, either boundary of design variables, or equality/inequality constraints. Therefore, an extensive number of efforts have been made to make these limitations understandable for the optimization algorithms. Here, a more important fact is that those constraint handling approaches affect the algorithms' performances considerably. In this study, some of the well-known strategies are incorporated into particle swarm optimization algorithms (PSO). The performance of the PSO algorithm is examined through several benchmarks, constrained problems, and the results discussed comprehensively.

1 Introduction

Real-world optimization problems are often very complicated, with many decision variables and practical limitations on the range of feasible solutions [10, 14, 15, 17, 26, 27]. These complexities result in optimization problems that are non-convex, discontinuous, have high dimensionality, and pose many challenges in developing algorithms that converge to the optimal global solution. Metaheuristic algorithms are designed for unconstrained optimization problems, so it is crucial to develop methods to account for constraints [6, 11–13, 16, 25, 32]. Most metaheuristic optimization algorithms are based on two crucial phases: 1-diversification, and 2-intensification.

Diversification focuses on exploring the entire search space, often in random and chaotic ways making the algorithm capable of overcoming difficulties related to the discontinuity of the solution space. On the other hand, intensification tends to focus the search on regions identified by the best solutions. Therefore, one of the

M. Rostamian · A. R. Kashani (✉) · C. V. Camp
Department of Civil Engineering, University of Memphis, Memphis, TN 38152, USA
e-mail: Kashani.alireza@ymail.com

A. H. Gandomi
Faculty of Engineering & Information Technology, University of Technology Sydney, Ultimo, NSW 2007, Australia

A. J. Kulkarni et al. (eds.), *Constraint Handling in Metaheuristics and Applications*,
https://doi.org/10.1007/978-981-33-6710-4_4

most critical features of any constraint handling scheme is limiting the influence of infeasible solutions while preserving the stability of the algorithm. Many constrained optimization problems often have the optimal solution located near the boundaries of search space, so any constraint handling approach needs to enable an algorithm to explore the boundaries effectively.

Much research has been devoted to developing efficient methods for handling the constraints in both single- and multi-objective algorithms. Homaifar et al. [22] and Hoffmeister and Sprave [21] proposed methods based on a static penalty function in which a constant penalty term is added to the objective function value. In this approach, for any violated constraint, a penalty term is added to the objective function. The resulting optimization algorithm then attempts to decrease the penalty value while also finding the global optima. Morales and Quezada [33] proposed another version of the static penalty function method called the death penalty where a predetermined large value is set as the objective function for infeasible solutions. Joines and Houck [24], Kazarlis and Petridis [28], and Petridis et al. [35] developed a dynamic penalty function where the penalty term is increased with the iteration of the algorithm. Another iteration-dependent penalty function was proposed by Carlson and Shonkwiler [4] based on simulated annealing called annealing penalty function. In this way, in the course of iterations, the temperature decreases resulting in a higher penalty. Coit and Smith [7], and Gen and Cheng [18] proposed adaptive penalty function methods for handling the constraints by permitting algorithms to explore beyond the feasible search space to some level by adjusting the penalty term during the search. Other researchers have introduced a variety of hybrid methods such as lagrangian multipliers by Adeli and Cheng [1], a hybrid interior-lagrangian penalty-based approach by Myung and Kim [34], and the application of fuzzy logic by Le [30]. Deb [9] proposed a method based on the separation of constraints and objectives. This method is based on a pair-wise comparison of solutions in every iteration. In this context, a tournament selection operator was proposed to compare every candidate solution with the following strategy: 1—any feasible solution overcome the infeasible solution, 2—between two feasible solutions, the fitter solution is the winner, 3—between two infeasible solutions, the one with lower constraint violation is preferred.

The impact of constraint handling methods on the efficiency of algorithms was also the subject of several most recent studies. Li et al. [31] explored the effect of different constraint handling strategies on the performance of evolutionary multi-objective algorithms. In this way, three constraint handling approaches as constrained-domination principle, self-adaptive penalty, and adaptive tradeoff model combined with nondominated sorting genetic algorithm II for solving 18 test problems. Jamal et al. [23] explored three constraint handling methods (i.e., ε-Level comparison, superiority of feasible solutions, and penalty function) for matrix adaptation evolutionary strategy to solve CEC-2010 benchmark constrained problems. Biswas et al. [3] tackled the problem of optimal power flow solutions using differential evolution algorithms. In this study three different constraint handling approaches were utilized as follows: 1—superiority of feasible solutions (SF), 2—self-adaptive penalty (SP), and 3—an ensemble of these two constraint handling techniques (SF and SP). Ameca-Alducin et al. [2] explored the efficiency of four constraint handling schemes (i.e., stochastic ranking, ϱ-constrained, penalty, and feasibility rules) for differential evolution algorithm to handle dynamic constrained optimization problems. Zou et al. [37] developed a new constraint handling method for solving the combined heat and power economic dispatch using an improved genetic algorithm. Datta et al. [8] proposed a novel constraint handling approach based on individual penalty approach in which all the constraints were scaled adaptively without a need for specific information from the problem. The proposed approach was examined through solving 23 benchmark test problems and two engineering problems using a hybrid evolutionary algorithm.

In this paper, we presented a review of six well-known penalty function approaches for constraint handling. Each of these schemes is incorporated into a particle swarm optimization (PSO) algorithm to evaluate their effectiveness and efficiency for a set of benchmark constrained optimization problems.

2 Particle Swarm Optimization

Kennedy and Eberhart [29] developed the particle swarm optimization (PSO) algorithm based on the behaviors of bird flocks in searching for food. In this context, the PSO algorithm searches the solution space with a population of particles. Each particle in the swarm represents a potential solution to the optimization problem. The PSO algorithm changes the position of the particles within the search space with the aim of finding more appropriate solutions. PSO determines the position of particles within the search space by two primary qualities; position and velocity. In PSO, each particles' position changes iteratively based on its current position and velocity given as:

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{1}$$

where X_i^{t+1} is the updated position of the i-th particle, X_i^t is the current position, and V_i^{t+1} is the velocity. A targeted search is conducted by a particle movement using a velocity term. Each particles' velocity connected to two important achievements in each iteration: the particles' position relative to the global best-found solution P_g and to its own best solution P_i. Clerc and Kennedy [5] proposed updating the velocity term as

$$V_i^{t+1} = \chi\left[V_i^t + C_1 r_1\left(P_i - X_i^t\right) + C_2 r_2\left(P_g - X_i^t\right)\right] \tag{2}$$

where V_i^{t+1} and V_i^t are the new velocity and old velocity of the i-th particle, respectively, C_1 and C_2 control the relative attraction to P_i and P_g, respectively, r_1 and r_2 are random numbers within [0,1], and $\chi < 1$ is the constriction factor that makes convergence rating slower and provides a better exploration of solution space (diversification).

Choosing appropriate values for parameters in the PSO algorithm is vital to obtaining the best performance. In Eq. (2), the values of χ, C_1, and C_2 impact how each particle will be attracted to its best position and the global best position. Clerc and Kennedy [5] proposed the following values: $C_1 = C_2 = 2.05$ and $\chi = 0.72984$.

3 Constraint Handling Approaches

In general, an optimization problem for the objective function $f(x)$ can be described as

$$Minimize\ f(x) \tag{3}$$

subject to

$$h_i(x) = 0, i = 1, 2, 3, \ldots, m \tag{4}$$

$$g_j(x) \le 0, j = 1, 2, 3, \ldots, p \tag{5}$$

where x is a vector of design variables, h and g are equality and inequality constraints, respectively, m and p are the number of equality and inequality constraints, respectively.

In this study, six different penalty function-based approaches are utilized to incorporate the effects of constraints into the PSO algorithm. Penalty function-based techniques are utilized to transform a constrained problem into an unconstrained one. In this way, the optimization algorithm generates a potential solution without considering its feasibility. At the next step, the constraints are checked, and a penalty value, based on the degree of violations of each constraint, is added to the objective function value. The penalty functions considered in this study are:

1. A simple static penalty function approach proposed by Homaifar et al. [22] is used to compute the penalized objective function $F(x)$ as

$$F(x) = f(x) + \sum_{i=1}^{p} R_{i,j} g_i(x)^2 \tag{6}$$

where $R_{i,j}$ is the penalty coefficient, p is the number of constraints, and $j = 1, 2, \ldots, l$, where l is the number of levels of a violation defined by the user. In this study, we used the same constant penalty coefficients for all the constraints.

2. The static penalty function method proposed by Hoffmeister and Sprave [21] is defined as

$$F(x) = f(x) + \sqrt{\sum_{i=1}^{p} \delta(-g_i(x)) g_i(x)^2} \tag{7}$$

where

$$\delta(x) = \begin{cases} 1 if x > 0 \\ 0 if x \leq 0 \end{cases} \tag{8}$$

3. The death penalty function proposed by Morales and Quezada [33] for updating the objective value is given as

$$F(x) = \begin{cases} f(x) & if\ x\ is\ feasible \\ K - \sum_{i=1}^{s} \left(\frac{K}{p}\right) otherwise \end{cases} \tag{9}$$

where K is a large constant, s is the number of satisfied constraints, and p is the total number of constraints.

4. The adaptive penalty function approach proposed by Coit and Smith [7] is utilized to alter the penalty term based on the feedback taken from the evolution process. The penalized objective function is

$$F(x) = f(x) + \left(B_{feasible} - B_{all}\right) \sum_{i=1}^{p} \left(\frac{g_i(x)}{NFT(t)} \right)^k \tag{10}$$

where $B_{feasible}$ is the best known feasible objective function at generation t, B_{all} is the best known (unpenalized) objective function at generation t, k is a constant that adjusts the "severity" of the penalty, and NFT is the so-called Near Feasibility Threshold, which is defined as the threshold distance from the feasible region as

$$NFT = \frac{NFT_0}{1 + \lambda \times t} \tag{11}$$

where NFT_0 is an upper bound for NFT, and λ is a constant which guarantees a search of the whole region between NFT_0 and zero.

5. The dynamic penalty function developed by Joines and Houck [24] is defined as

$$F(x) = f(x) + (C \times t)^{\alpha} \sum_{i=1}^{p} |g_i(x)|^{\beta} \tag{12}$$

where C, α, and β are constants defined by the user.

6. The annealing-based penalty function method developed by Joines and Houck [24] is defined as

$$F(x) = f(x) + \exp\left((C \times t)^{\alpha} \sum_{i=1}^{p} |g_i(x)|^{\beta} \right) \tag{13}$$

These methods labeled from 1 to 6 are referred to in all tables and figures as Homaifar, Hoffmeister, Death, Adaptive, Dynamic, and Annealing, respectively.

4 Numerical Simulation

In this section, we incorporate the above-mentioned constraint handling approaches into a PSO algorithm to solve several benchmark problems. In all the cases, the particle population size is 50, and the number of iterations is 1,000. Metaheuristic optimization algorithms search the solution space stochastically to find the global minimum. Therefore, we evaluated the performance of each constraint handling approach based on the best, mean, and standard deviation (STD) of solutions over a series of 50 independent runs. The best-found solutions are highlighted in bold in their relevant tables.

In all cases, parameter values for each constraint handling method are held constant. For the Homaifar approach, the constant penalty coefficient is 1013 for all the constraints. In the Death method, the K value is 109. In the Adaptive approach NFT_0, λ, and K are equal to 10, 2, and 2, respectively. In the Dynamic scheme, α, β, and C are equal to 2, 1, and 0.5, respectively. In the Annealing method, α, β, and C are equal to 1, 1, and 0.05, respectively.

4.1 Test Problems Series 1

In the following section, we solved five single-objective constrained benchmark optimization problems as follows (Wikipedia website [36]: 1—Rosenbrock function constrained with a cubic and a line, 2—Rosenbrock function constrained to a disk, 3—Mishra's Bird function, 4—Townsend function (modified), and 5—Simionescu function. Table 1 lists the objective function and constraints for each problem.

Table 2 lists the best-found solutions to the benchmark functions for each constraint handing method. Table 3 gives values for the mean ± STD of the 50 independent runs for each case. Table 4 lists the values of the constraints for the best solution for each case. The results in Table 4 indicate that none of the algorithms can satisfy the F1 constraints. Also, only two methods were successful in meeting all the F3 constraints. In all the remaining functions, all the constraints are satisfied.

For function F1, the values of the constraint violations recorded by Hoffmeister, Death, Dynamic, and Annealing are negligible. Since the Annealing constraint handing method produced both the best solution and the lowest constraint violations, it is considered the best method for function F1. However, the Hoffmeister method has the lowest mean value over multiple runs. For function F2, the Hoffmeister method produced the best solution and had the lowest mean value. Function F3 posed more of a challenge than the other functions. In this case, the Homaifar and Death methods were able to solve the problem, and recorded similar best and mean values; however, the Death method recorded a slightly lower STD. Results from F4 simulations showed that the Death penalty function method found the lowest best value, while the Dynamic method had the lowest mean value. For the F5 function, all the methods except Dynamic found equal best values; however, the Adaptive method had the lowest STD value.

4.2 Test Problems Series 2

In this section, more complicated optimization problems with numerous constraints and design variables are considered to evaluate the performance of constraint handling approaches better. To this end, we selected some benchmark optimization problems presented by Dr. Abdel-Rahman Hedar on his official website [19, 20]. Table 5 lists the objective functions and constraints for the selected optimization problems.

Numerical simulations are conducted on these functions using PSO with previously mentioned constraint handling schemes. Tables 6 and 7 tabulate the results for the best solution, and the mean ± STD over a series of independent runs, respectively. Table 8 provides constraint values from the best solution found using each constraint handling method. The results listed in Table 8 show that for functions G1 to G4, the Homaifar, Death, Adaptive, and Dynamic methods meet all the constraints successfully. A comparison of the best results for G1 shows that the Death method found the lowest objective function value and had a lower mean value than the other successful methods. For the G2 function, the Adaptive method found the highest objective function value and had the best mean and STD values. Results for the G4 function, show the lowest objective function value was recorded by the Adaptive method, while the associated mean and STD values are comparable with other successful methods. For the G6 function, the only approach that satisfied both constraints is the Adaptive

Table 1 Benchmark optimization problems 1st case

ID	Function name	Definition	Constraints	Boundary conditions
F1	Rosenbrock function constrained with a cubic and a line	$\min f(x, y) = (1-x)^2 + 100\left(y - x^2\right)^2$	$g_1 = (\mathrm{x}-1)^3 - y + 1 \leq 0$ $g_2 = x + y - 2 \leq 0$	$-1.5 \leq x \leq 1.5$ $-0.5 \leq y \leq 2.5$
F2	Rosenbrock function constrained to a disk	$\min f(x, y) = (1-x)^2 + 100\left(y - x^2\right)^2$	$g = \mathrm{x}^2 + y^2 - 2 \leq 0$	$-1.5 \leq x \leq 1.5$ $-1.5 \leq y \leq 1.5$
F3	Mishra's Bird function	$\min f(x, y) = \sin(\mathrm{y})\exp\left[(1-\cos x)^2\right] + \cos(x)\exp\left[(1-\sin y)^2\right] + (x-y)^2$	$g = (\mathrm{x}+5)^2 + (y+5)^2 - 25 \leq 0$	$-10 \leq x \leq 0$ $-6.5 \leq y \leq 0$
F4	Townsend function (modified)	$\min f(x, y) = -[cos((x-0.1)y)]^2 - x\sin(3x+y)$	$\mathrm{x}^2 + y^2 < [2\cos t - 0.5\cos 2t - 0.25\cos 3t - 0.125\cos 4t]^2 + [2\sin t]^2 \leq 0$ where $\mathrm{t} = \mathrm{arctan2(xy)}$	$-2.25 \leq x \leq 2.5$ $-2.5 \leq y \leq 1.75$
F5	Simionescu function	$\min f(x, y) = 0.1xy$	$\mathrm{x}^2 + y^2 \leq \left[1 + 0.2\cos\left(n \arctan \frac{x}{y}\right)\right]^2$	$-1.25 \leq x, y \leq 1.25$

Table 2 Best results for 1st case

Constraint handling scheme	F1	F2	F3
Homaifar	0.00214	2.17430e−29	−48.40602
Hoffmeister	2.57235e−22	1.97215e−31	−97.05423
Death	3.24207e−25	2.61868e−25	−48.40602
Adaptive	0.00032	5.85779e−28	−104.14808
Dynamic	1.00000	1	−105.09690
Annealing	2.49045e−26	6.24514e−23	−106.76454
Constraint handling scheme	F4	F5	
Homaifar	−3.37179	−0.15625	
Hoffmeister	−3.36867	−0.15625	
Death	−3.36720	−0.15625	
Adaptive	−3.37183	−0.15625	
Dynamic	−2.36984	0.84375	
Annealing	−3.36916	−0.15625	

method. The G7 function seemed to be a challenging problem; in that, none of the methods could satisfy all the constraints. In the G8 function study, all the methods except for the Hoffmeister and Annealing methods satisfied the constraints effectively. Among the successful approaches, the Dynamic method had higher best and mean values, while the other methods reached similar best values. A comparison of the mean values of the remaining successful methods demonstrated that Homaifar and Death very close, while Homaifar had a lower STD value. The results listed in Table 8 for the G9 function show that all the methods, except Annealing, were able to satisfy all the constraints. In contrast, the Hoffmeister method found the lowest best objective function value and had the lowest mean and STD values.

5 Summary

In this study, the performance of six popular penalty function-based constraint handling methods is explored. A PSO algorithm was selected as the testbed for this study because of its robustness and ability to handle complex optimization problems. Each of the six penalty function methods was incorporated into a PSO algorithm. Twelve benchmark optimization problems were solved to examine the effectiveness of each of the six constraint handling approaches. For each of the constraint handling method and objective function (total of 72 cases), the best solution was reported, and the mean and standard deviation were computed a series of 50 independent runs. For each method, the values of constraint were reported for the best solution. In

Table 3 Mean and STD for 1st case

Constraint handling scheme	F1	F2	F3
Homaifar	0.066480295 ± 7.0153E−02	5.5561E−15 ± 3.5830E−14	−48.4060 ± 1.9735E−14
Hoffmeister	1.10E−08 ± 2.9197E−08	8.8445E−19 ± 3.2715E−18	−97.0542 ± 5.5671E−14
Death	0.000122386 ± 3.5167E−04	4.7381E-07 ± 2.2950E-06	−48.4060 ± 9.1355E−15
Adaptive	0.113585377 ± 1.0286E−01	3.7770E-13 ± 2.3308E-12	−96.6595 ± 9.2323
Dynamic	1.009659277 ± 3.9075E−02	1.0000 ± 2.7925E−14	−99.3211 ± 8.7576
Annealing	3.96E−05 ± 1.2847E−04	1.3882E−05 ± 4.5058E-05	−104.8126 ± 9.6777
Constraint handling scheme	F4	F5	
Homaifar	−3.2238 ± 0.0864	−0.1563 ± 9.0765E−17	
Hoffmeister	−3.2168 ± 0.0784	−0.1563 ± 8.8928E−17	
Death	−3.2517 ± 0.0861	−0.1563 ± 8.8573E−17	
Adaptive	−3.2788 ± 0.0861	−0.1563 ± 8.8307E−17	
Dynamic	−2.2564 ± 0.0857	0.8438 ± 0	
Annealing	−3.2509 ± 0.0864	−0.1563 ± 8.8928E−17	

Table 4 Constraint values for the best solution for 1st case

Function	Constraints	Homaifar	Hoffmeister	Death	Adaptive	Dynamic	Annealing
F1	g1	−0.000634361	−2.085E−13	−1.70708E−12	−9.70381E-5	−1.89488E−10	−4.49862E−13
	g2	0.001752213	6.03739E−13	1.13776E−12	0.000629708	6.73502E−10	2.98428E−13
F2	g	−2.7978E−14	−2.6645E−15	−3.0718E−12	−1.2390E−13	−5.7046E−08	−2.7978E−14
F3	g	−5.4459	9.5974	−5.4459	9.5257	9.7568	9.8223
F4	g	−4.2072	−4.2358	−4.1844	−4.0910	−4.4387	−4.3576
F5	g	−1.685	−1.685	−1.685	−1.685	−1.685	−1.685

Table 5 Benchmark optimization problems 2nd case

ID	Definition	Constraints	Boundary conditions
G1	$\min f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$	$g_1 = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$ $g_2 = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$ $g_3 = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$ $g_4 = -8x_1 + x_{10} \leq 0$ $g_5 = -8x_2 + x_{11} \leq 0$ $g_6 = -8x_3 + x_{12} \leq 0$ $g_7 = -2x_4 - x_5 + x_{10} \leq 0$ $g_8 = -2x_6 - x_7 + x_{11} \leq 0$ $g_9 = -2x_8 - x_9 + x_{12} \leq 0$	$0 \leq x_i \leq u_i$ $i = 1, 2, \ldots, n$ $u = (1, 1, 1, 1, 1, 1, 1, 1, 1, 100, 100, 100, 1)$
G2	$\max f(x) = \left\lvert \frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right\rvert$	$g_1 = -\prod_{i=1}^{n} x_i + 0.75 \leq 0$ $g_2 = \sum_{i=1}^{n} x_i - 7.5n \leq 0$	$0 \leq x_i \leq 10$ $i = 1, 2, \ldots, 20$

(continued)

Table 5 (continued)

ID	Definition	Constraints	Boundary conditions
G4	$\min f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$	$g_1 = \upsilon(x) - 92 \leq 0$ $g_3 = \upsilon(x) - 110 \leq 0$ $g_4 = -\upsilon(x) + 90 \leq 0$ $g_5 = \omega(x) - 25 \leq 0$ $g_6 = -\omega(x) + 20 \leq 0$ $\upsilon(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5$ $\upsilon(x) = 80.51249 + 0.0071317x_2x_5) + 0.0029955x_1x_2 + 0.0021813x_3^2$ $\omega(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4$	$l_i \leq x_i \leq u_i$ $i = 1, 2, \ldots, 5$ $l = (78, 33, 27, 27, 27), u = (102, 45, 45, 45, 45)$
G6	$\min f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$	$g_1 = (x_1 - 5)^2 + (x_2 - 5)^2 + 100 \leq 0$ $g_2 = (x_1 - 5)^2 + (x_2 - 5)^2 - 82.81 \leq 0$	$l \leq x_i \leq 100$ $i = 1, 2$ $l = (13, 0)$

(continued)

Table 5 (continued)

ID	Definition	Constraints	Boundary conditions
G7	$\min f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3-10)^2 + 4(x_4-5)^2 + (x_5-3)^2 + 2(x_6-1)^2 + 5x_7^2 + 7(x_8-11)^2 + 2(x_9-10)^2 + (x_{10}-7)^2 + 45$	$g_1 = 4x_1 + 5x_2 - 3x_7 + 9x_8 - 105 \leq 0$ $g_2 = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$ $g_3 = -8x_1 + 2x_2 + 5x_9 + 2x_{10} - 12 \leq 0$ $g_4 = 3(x_1-2)^2 + 4(x_2-3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$ $g_5 = 5x_1^2 + 8x_2 + (x_3-6)^2 - 2x_4 - 40 \leq 0$ $g_6 = 0.5(x_1-8)^2 + 2(x_2-4)^2 + 3x_5^2 - x_6 - 30 \leq 0$ $g_7 = x_1^2 + 2(x_2-2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$ $g_8 = -3x_1 + 6x_2 + 12(x_9-8)^2 - 7x_{10} \leq 0$	-1 0 $\leq x_i \leq 10$ $i = 1, 2, \ldots, 10$
G8	$\max f(x) = \frac{sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$	$g_1 = x_1^2 - x_2 + 1 \leq 0$ $g_2 = 1 - x_1 + (x_2-4)^2 \leq 0$	$0 \leq x_i \leq 10$ $i = 1, 2$
G9	$\min f(x) = (x_1-10)^2 + 5(x_2-12)^2 + 3(x_4-11)^2 + x_3^4 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$	$g_1 = v_1 + 3v_2^2 + v_3 + 4v_4^2 + 5v_5 - 127 \leq 0$ $g_2 = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282 \leq 0$ $g_3 = 23x_1 + v_2 + 6x_6^2 - 8x_7 - 196 \leq 0$ $g_4 = 2v_1 + v_2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$ $v_1 = 2x_1^2 \leq 0$ $v_2 = x_2^2 \leq 0$	-1 0 $\leq x_i \leq 10$ $i = 1, 2, \ldots, 7$

Table 6 Best results for 2nd case

Constraint handling scheme	G1	G2	G4	G6
Homaifar	−18.7552	−0.4398	−33,233.6511	−6,313.0662
Hoffmeister	−29.5263	−Inf	−35,202.9264	−8,840.8022
Death	−75.6721	−0.4282	−33,257.6403	−6,854.7599
Adaptive	−30.0434	−0.4527	−33,813.9574	−1,643.5994
Dynamic	−16.3929	−0.1928	−32,955.7715	3.3224e+06
Annealing	−383.9712	−Inf	−34,320.3859	−7,988.8455
Constraint handling scheme	G7	G8	G9	
Homaifar	129.5706	−0.0958	732.8192	
Hoffmeister	44.7292	−1,541.5176	693.7655	
Death	144.4177	−0.0958	721.4620	
Adaptive	−3,381.3957	−0.0958	788.6190	
Dynamic	1.4013e+07	−0.0860	888.7828	
Annealing	1.4870e+07	−1,558.5455	539.2907	

general, the results indicated the Homaifar and Adaptive methods provide satisfactory performance, while the Hoffmeister and Annealing methods were unsuccessful in satisfying the constraints in all the cases.

Table 7 Mean and STD for 2nd case

Constraint handling scheme	G1	G2	G4	G6
Homaifar	−1.53e+01 ± 2.26e+00	−2.97e−1 ± 6.49e−02	−3.29e+04 ± 1.31e+02	−2.90e+03 ± 1.87e+03
Hoffmeister	−1.97e+1±2.64e+00	−Inf	−3.51e+04 ± 5.66e+01	−8.69e+03 ± 7.25e+01
Death	−1.79e+01 ± 9.36e+00	−2.98e−01 ± 6.1e−02	−3.28e+04 ± 1.34e+02	−6.03e+03 ± 5.17e+02
Adaptive	−8.14e+00 ± 9.36e+00	−4.31e−01 ± 6.1e−02	−3.26e+04 ± 1.34e+02	9.66e+05 ± 4.82e+05
Dynamic	−1.49e+01 ± 6.16e−01	−1.51e−01 ± 1.32e−02	−3.24e04 ± 6.14e−02	6.52e+06 ± 1.56e+06
Annealing	3.12e+06 ± 1.26e+07	−Inf	−3.25e+04 ± 7.12e+02	2.85e+07 ± 6.21e+07
Constraint handling scheme	G7	G8	G9	
Homaifar	3.31e + 02 ± 1.53e + 2	−1.00e−1 ± 6.07e−17	8.38e+02 ± 4.51e+01	
Hoffmeister	1.03e + 02 ± 3.28e + 01	−1.54e + 03 ± 5.26e-13	7.66e+02 ± 3.79e+01	
Death	3.89e + 02 ± 2.34e + 02	−1.00e−01 ± 6.24e−17	8.14e + 02 ± 5.13e + 01	
Adaptive	−3.26e+03 ± 3.44e+02	−1.00e−01 ± 6.24e−17	9.77e+02 ± 5.13e+01	
Dynamic	5.20e+07 ± 3.53e+07	5.00 ± 3.54e + 04	1.01e+03 ± 6.43e+01	
Annealing	6.33e+07 ± 2.91e+07	−4.05e+02 ± 6.91e+02	1.76e+03 ± 5.87e+02	

Table 8 Constraint values for the best solution for 2nd case

ID	Constraints	Homaifar	Hoffmeister	Death	Adaptive	Dynamic	Annealing
G1	g1	−6.8015	4.1138	−6.6545	−6.1299	−0.6733	178.9962
	g2	−6.2117	5.9941	−5.8829	−4.4511	−0.5364	178.1315
	g3	−5.7446	17.8150	−5.2382	−4.3265	−0.3819	184.7554
	g4	−0.5743	−0.5907	−0.4075	−2.5022	−5.3720	91.1861
	g5	−2.4217	11.0694	−**0.8**515	−2.6834	−5.2472	97.8100
	g6	−3.0717	11.7691	−0.7883	−2.1008	−5.1138	96.9394
	g7	−1.0595	−0.3005	−0.2981	−1.7997	−0.4049	90.5813
	g8	−1.5135	11.1074	−0.7523	−1.7912	−0.2567	97.5370
	g9	−1.4735	12.3086	−0.8344	−0.4050	−0.1208	95.2121
G2	g1	−0.3483	0.7500	−0.0000	−0.1744	−11.3301E08	0.7500
	g2	−105.5281	−150.0000	−110.2935	−110.6437	−66.9536	−150.0000
G4	g1	−91.2519	−88.8924	−91.6257	−91.7387	−91.2396	−89.5925
	g2	−0.7481	−3.1076	−0.3743	−0.2613	−0.7604	−2.4075
	g3	−6.7130	−0.6636	−7.1186	−6.8308	−6.1124	−2.4804
	g4	−13.2870	−19.3364	−12.8814	−13.1692	−13.8876	−17.5196
	g5	−0.7500	4.0644	−0.9254	−0.1964	−0.2006	3.9898
	g6	−4.2500	−9.0644	−4.0746	−4.8036	−4.7994	−8.9898
G6	g1	−0.3505	53.2423	−0.0577	−0.7250	0.2726	23.0374
	g2	25.6657	−25.0682	26.3706	−0.2570	9.3178	−19.2648
G7	g1	−50.6637	−28.2747	−45.0462	−124.5271	−33.2748	−69.9097
	g2	−90.9175	−121.4250	−110.7612	−35.9301	−119.0130	19.4864
	g3	11.3410	11.9677	10.8961	−4.9997	3.5965	12.7076
	g4	−122.8382	−118.8054	−113.1044	−23.8555	−131.0416	−71.6281
	g5	−12.2697	−6.8061	−23.6437	0.9809	3.7317	−13.6003
	g6	62.6469	23.7481	61.9868	−1.1412	38.4242	12.4138
	g7	32.7484	8.5404	31.1607	−83.0901	10.2969	−36.6248
	g8	−43.8874	−17.1694	−41.5682	−52.4224	−33.2774	14.8685
G8	g1	−1.7375	0.9996	−1.7375	−1.7390	−1.8584	1.0000
	g2	−0.1678	16.9969	−0.1678	−0.1671	−0.1101	17.0000
G9	g1	−115.6328	−104.0385	−107.5731	−2.3858	−68.4760	6212.5453
	g2	−286.1398	−262.3689	−276.4030	−253.1911	−269.3198	−200.5948
	g3	−213.0346	−187.6622	−215.3880	−170.7742	−185.4042	195.7844
	g4	−35.5948	−45.7608	−30.0437	−8.4887	−14.5741	69.3815

References

1. Adeli, H., Cheng, N.T.: Augmented Lagrangian genetic algorithm for structural optimization. J. Aerosp. Eng. **7**(1), 104–118 (1994)
2. Ameca-Alducin, M.Y., Hasani-Shoreh, M., Blaikie, W., Neumann, F., Mezura-Montes, E.: A comparison of constraint handling techniques for dynamic constrained optimization problems.

In: 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2018)
3. Biswas, P.P., Suganthan, P.N., Mallipeddi, R., Amaratunga, G.A.: Optimal power flow solutions using differential evolution algorithm integrated with effective constraint handling techniques. Eng. Appl. Artif. Intell. **68**, 81–100 (2018)
4. Carlson, S.E., Shonkwiler, R.: Annealing a genetic algorithm over constraints. In: SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218) (Vol. 4, pp. 3931–3936). IEEE (1998)
5. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**(1), 58–73 (2002)
6. Coello, C.A.C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput. Methods Appl. Mech. Eng. **191**(11–12), 1245–1287 (2002)
7. Coit, D.W., Smith, A.E.: Penalty guided genetic search for reliability design optimization. Comput. Ind. Eng. **30**(4), 895–904 (1996)
8. Datta, R., Deb, K., Kim, J.H.: CHIP: Constraint Handling with Individual Penalty approach using a hybrid evolutionary algorithm. Neural Comput. Appl. **31**(9), 5255–5271 (2019)
9. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**(2–4), 311–338 (2000)
10. Gandomi, A.H., Kashani, A.R.: Automating pseudo-static analysis of concrete cantilever retaining wall using evolutionary algorithms. Measurement **115**, 104–124 (2018)
11. Gandomi, A.H., Kashani, A.R.: Probabilistic evolutionary bound constraint handling for particle swarm optimization. Oper. Res. Int. J. **18**(3), 801–823 (2018)
12. Gandomi, A.H., Kashani, A.R.: Evolutionary bound constraint handling for particle swarm optimization. In: 2016 4th International Symposium on Computational and Business Intelligence (ISCBI), pp. 148–152. IEEE (2016)
13. Gandomi, A.H., Kashani, A.R., Mousavi, M.: Boundary constraint handling affection on slope stability analysis. In: Engineering and Applied Sciences Optimization, pp. 341–358. Springer, Cham (2015)
14. Gandomi, A.H., Kashani, A.R., Mousavi, M., Jalalvandi, M.: Slope stability analysis using evolutionary optimization techniques. Int. J. Numer. Anal. Meth. Geomech. **41**(2), 251–264 (2017)
15. Gandomi, A.H., Kashani, A.R., Zeighami, F.: Retaining wall optimization using interior search algorithm with different bound constraint handling. Int. J. Numer. Anal. Methods Geomech. (2017)
16. Gandomi, A.H., Yang, X.S.: Evolutionary boundary constraint handling scheme. Neural Comput. Appl. **21**(6), 1449–1462 (2012)
17. Gandomi, A.H., Yang, X.S., Talatahari, S., Alavi, A.H.: Metaheuristic algorithms in modeling and optimization. In: Gandomi et al. (eds.) Metaheuristic Applications in Structures and Infrastructures, pp. 1–24. Elsevier, Waltham, MA (2013)
18. Gen, M., Cheng, R.: Interval programming using genetic algorithms. In: Proceedings of the Sixth International Symposium on Robotics and Manufacturing, Montpellier, France (1996)
19. Hedar, A.-R.: Dr. Abdel-Rahman Hedar's official website (2020). http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar.html
20. Hedar, A.-R.: Test problems for con-strained global optimization (2020). http://www-opti-ma.amp.i.kyotou.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page422.htm
21. Hoffmeister, F., Sprave, J. Problem-independent handling of constraints by use of metric penalty functions. In: Proceedings of Evolutionary Programming, pp. 289–294 (1996)
22. Homaifar, A., Qi, C.X., Lai, S.H.: Constrained optimization via genetic algorithms. Simulation **62**(4), 242–253 (1994)
23. Jamal, M.., Ming, F., Zhengang, J.: Solving constrained optimization problems by using covariance matrix adaptation evolutionary strategy with constraint handling methods. In: Proceedings of the 2nd International Conference on Innovation in Artificial Intelligence, pp. 6–15 (201)
24. Joines, J.A., Houck, C.R.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In: IEEE, 1994, vol. 572, pp. 579–584 (1994)

25. Jordehi, A.R.: A review on constraint handling strategies in particle swarm optimisation. Neural Comput. Appl. **26**(6), 1265–1275 (2015)
26. Kashani, A.R., Saneirad, A., Gandomi, A.H.: Optimum design of reinforced earth walls using evolutionary optimization algorithms. Neural Comput. Appl., 1–24 (2019)
27. Kashani, A.R., Gandomi, M., Camp, C.V., Gandomi, A.H.: Optimum design of shallow foundation using evolutionary algorithms. Soft. Comput. **24**(9), 6809–6833 (2020)
28. Kazarlis, S., Petridis, V.: Varying fitness functions in genetic algorithms: Studying the rate of increase of the dynamic penalty terms. In: International conference on parallel problem solving from nature, pp. 211–220. Springer, Berlin, Heidelberg (1998)
29. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann, San Francisco, CA (2001)
30. Le, T.V.: A fuzzy evolutionary approach to constrained optimization problems. In: Proceedings of the Second IEEE Conference on Evolutionary Computation, pp. 274–278. IEEE Perth (1995)
31. Li, J.P., Wang, Y., Yang, S., Cai, Z.: A comparative study of constraint-handling techniques in evolutionary constrained multiobjective optimization. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 4175–4182. IEEE (2016)
32. Michalewicz, Z.: A survey of constraint handling techniques in evolutionary computation methods. Evol. Prog. **4**, 135–155 (1995)
33. Morales, A.K., Quezada, C.V.: A universal eclectic genetic algorithm for constrained optimization. In: Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing, vol. 1, pp. 518–522 (1998)
34. Myung, H., Kim, J.H.: Hybrid interior-lagrangian penalty based evolutionary optimization. In: International Conference on Evolutionary Programming, pp. 85–94. Springer, Berlin, Heidelberg (1998)
35. Petridis, V., Kazarlis, S., Bakirtzis, A.: Varying fitness functions in genetic algorithm constrained optimization: the cutting stock and unit commitment problems. IEEE Trans. Syst., Man, Cybern., Part B (Cybern.) **28**(5), 629–640 (1998)
36. Wikipedia.: Test functions for optimization (16 April 2020). https://en.wikipedia.org/wiki/Test_functions_for_optimization
37. Zou, D., Li, S., Kong, X., Ouyang, H., Li, Z.: Solving the combined heat and power economic dispatch problems by an improved genetic algorithm and a new constraint handling strategy. Appl. Energy **237**, 646–670 (2019)