Chapter 3
Cohort Intelligence for Constrained Test Problems

Any optimization algorithm requires a technique/way to handle constraints. This is important because most of the real world problems are inherently constrained problems. There are a several traditional methods available such as feasibility-based methods, gradient projection method, reduced gradient method, Lagrange multiplier method, aggregate constraint method, feasible direction based method, penalty based method, etc. [1]. According to Vanderplaat [2], the penalty based methods can be referred to as generalized constraint handling methods. They can be easily incorporated into most of the unconstrained optimization methods and can be used to handle nonlinear constraints. Another approach is feasibility-based approach. Similar to the penalty based methods, it is also simple to use as it assists the unconstrained optimization methods to drive into feasible region and further reach in close neighborhood of the optimum solution [3–5]. Similar to other nature-/bio-inspired techniques, the performance of Cohort Intelligence (CI) methodology may degenerate when applied for solving constrained problems. As an effort in the direction of developing and further incorporating a generic constraint handling technique into the CI framework, a penalty function approach is used. The performance of the constrained CI approach is tested by successfully solving several well studied constrained test problems.

3.1 Constraint Handling Using Penalty Function Approach

Consider a general constrained problem (in the minimization sense) as follows:

$$\begin{array}{ll} \text{Minimize} & f(\mathbf{x}) \\ \text{Subject to} & g(\mathbf{x})_j \leq 0, \quad j = 1, 2, \ldots, s \\ & h(\mathbf{x})_j = 0, \quad j = 1, 2, \ldots, w \\ \text{Subject to} & \Psi_i^{lower} \leq x_i \leq \Psi_i^{upper}, \quad i = 1, \ldots, N \end{array} \tag{3.1}$$

A.J. Kulkarni et al., *Cohort Intelligence: A Socio-inspired Optimization Method*,
Intelligent Systems Reference Library 114, DOI 10.1007/978-3-319-44254-9_3

In order to incorporate the constraints into the problem, a pseudo-objective function is formed as follows:

$$\phi(\mathbf{x}) = f(\mathbf{x}) + \theta\left\{\sum_{j=1}^{s}\left[g_j^{+}(\mathbf{x})\right]^2 + \sum_{j=1}^{w}\left[h_j(\mathbf{x})\right]^2\right\} \quad \text{Subject to} \quad \Psi_i^{lower} \le x_i \le \Psi_i^{upper}, \quad i = 1, \ldots, N \tag{3.2}$$

where $g_j^{+}(\mathbf{x}) = \max\left(0, g_j(\mathbf{x})\right)$ and θ is the scalar penalty parameter which is fixed in all the runs of the CI algorithm.

3.2 Numerical Experiments and Discussion

CI methodology was applied to a variety of well known constrained test problems with penalty function approach [1–5] incorporated into it. The CI algorithm was coded in MATLAB 7.14.0.739 (R2012a) on Windows platform using Intel Core 2 Duo, 1.67 GHz processor speed and 2 GB RAM. Every problem was solved 20 times with number of candidates C chosen as 5 and number of variations in the behavior t chosen for constrained test problem is listed in Table 3.9. These parameters were derived empirically over numerous experiments.

It is evident from the results presented in Tables 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8 and 3.9, that the CI methodology is capable of efficiently handling a variety of equality as well as inequality of constraints. For detailed properties of these problems the reader is encouraged to refer to [4]. The results also demonstrated the competitiveness with other contemporary methods. Furthermore, the standard deviation presented in Table 3.9, it is evident that the approach was sufficiently robust with reasonable computational cost, i.e. function evaluations and computational time. According to Table 3.9, the solutions obtained using CI were quite close to the best reported solution so far. As mentioned before, the parameters such as number of candidates C, number of variations in the behavior t and reduction factor r were chosen empirically

Table 3.1 Characteristic of benchmark problems [4]

Problem	DV	Form of $f(\mathbf{x})$	$(\mathbf{x}\%)$	LI	NE	NI	α
G03	10	Polynomial	0.002	0	1	0	1
G04	5	Quadratic	52.123	0	0	6	2
G05	4	Cubic	0.000	2	3	0	3
G06	2	Cubic	0.006	0	0	2	2
G07	10	Quadratic	0.000	3	0	5	6
G08	2	Nonlinear	0.856	0	0	2	0
G09	7	Polynomial	0.512	0	0	4	2
G11	2	Quadratic	0.000	0	1	0	1

Table 3.2 Summary of the constrained test problem solutions

Problem		G03	G04	G05	G06	G07	G08	G09	G11
Best known solutions		1	−30,665.539	5126.498	−6961.81388	24.306	0.095825	680.6300573	0.75
Methods	Koziel et al. (1999) [23]	Best 0.9997 Avg. 0.9989 Worst 0.9960	Best −30664.5 Avg. −30655.3 Worst −30645.9	N.A. N.A. N.A.	Best −6952.1 Avg. −6342.6 Worst −5473.9	Best 24.620 Avg. 24.826 Worst 24.069	Best 0.0958250 Avg. 0.0891568 Worst 0.0291438	Best 680.91 Avg. 681.16 Worst 683.18	Best 0.75 Avg. 0.75 Worst 0.75
	Runarsson et al. (2000) [21]	1.000 1.000 1.000	−30,665.539 −30,665.539 −30,665.539	5126.497 5128.881 5142.472	−6961.814 −6875.940 −6350.262	24.307 24.374 24.642	0.095825 0.095825 0.095825	680.630 680.656 680.763	0.750 0.750 0.750
	Hamida et al. (2002) [22]	1 0.99989 N.A.	−30,665.5 −30,665.5 N.A.	5126.5 5141.65 N.A.	−6961.81 −6961.81 N.A.	24.3323 24.6636 N.A	0.09582 0.09582 N.A.	680.630 680.641 N.A.	0.75 0.75 N.A.
	Montes et al. (2003) [23]	1.001038 1.000989 1.000579	−30,665.539062 −30,665.539062 −30,665.539062	5126.599609 5174.492301 5304.166992	−6961.813965 −6961.283984 −6961.481934	24.326715 24.474926 24.842829	0.095826 0.095826 0.095826	680.631592 680.643410 680.719299	0.749090 0.749358 0.749830
	Hedar et al. (2006) [15]	1.0000015 0.9991874 0.9915186	−30,665.5380 −30,665.4665 −30,664.6880	5126.4981 5126.4981 5126.4981	−6961.81388 −6961.81388 −6961.81388	24.310571 24.379527 24.644397	0.095825 0.095825 0.095825	680.63008 680.63642 680.69832	0.7499990 0.7499990 0.7499990
	Bacerra et al. (2006) [11]	0.995143 0.788635 0.639920	−30,665.538672 −30,665.538672 −30,665.538672	5126.5709 5207.4106 5327.3904	−6961.813876 −6961.813876 −6961.813876	24.306209 24.306210 24.306212	0.095825 0.095825 0.095825	680.630057 680.630057 680.630057	0.749900 0.757995 0.796455
	Lampinen (2006) [6]	N.A. N.A. N.A.	N.A. N.A. N.A.	5126.484 5126.484 5126.484	−6961.814 −6961.814 −6961.814	N.A. N.A. N.A.	0.095825 0.095825 0.095825	680.630 680.630 680.630	0.74900 0.74900 0.74900
	Chootinan et al. (2006) [12]	0.99998 0.99979	−30,665.5386 −30,665.5386 −30,665.5386	5126.4981 5126.4981 5126.4981	−6961.8139 −6961.8139 −6961.8139	N.A. N.A. N.A.	0.0958250 0.0958250 0.0958250	680.6303 680.6381 680.6538	0.7500 0.7500 0.7500

(continued)

Table 3.2 (continued)

Problem		G03	G04	G05	G06	G07	G08	G09	G11
	Farmani et al. (2003) [4]	0.99978 0.99930 0.99830	−30,665.5000 −30,665.2000 −30,663.3000	5126.9890 5432.08 N.A.	−6961.8000 −6961.8000 −6961.8000	24.59 27.83 32.69	0.0958250 0.0958250 0.0958250	680.6400 680.7200 680.8700	0.7500 0.7500 0.7500
	Deb (2000) [5]	N.A. N.A. N.A.	−30,614.814 −30,196.404 −29,606.596	N.A. N.A. N.A.	N.A. N.A. N.A.	24.372 24.409 25.075	N.A. N.A. N.A.	680.659424 681.525635 687.188599	N.A. N.A. N.A.
	Dong et al. (2007) [14]	N.A. N.A. N.A.	−30,664.7 −30,656.1 −30,662.8	N.A. N.A. N.A.	N.A. N.A. N.A.	N.A. N.A. N.A.	N.A. N.A. N.A.	N.A. N.A. N.A.	N.A. N.A. N.A.
	He et al. (2007) [7]	N.A. N.A. N.A.	−30,665.539 −30,665.539 −30,665.539	N.A. N.A. N.A.	N.A. N.A. N.A.	N.A. N.A. N.A.	0.095825 0.095825 0.095825	N.A. N.A. N.A.	N.A. N.A. N.A.
	Hu et al. (2002) [8]	1 1 1	−30,665.5 −30,665.5 −30,665.5	N.A. N.A. N.A.	−6961.7 −6960.7 −6956.8	24.4420 26.7188 31.1843	0.09583 0.09583 0.09583	680.657 680.876 681.675	0.7500 0.7500 0.7500
	Ray et al. [24]	N.A. N.A. N.A.	−30,651.662 −30,647.105 −30,619.047	N.A. N.A. N.A.	N.A. N.A. N.A.	N.A. N.A. N.A.	N.A. N.A. N.A.	N.A. N.A. N.A.	N.A. N.A. N.A.
	CI algorithm	0.998892 0.999417 0.999762	−30,665.531736 −30,665.529486 −30,665.526082	5143.533669 5196.042840 5273.835265	−6961.812948 −6961.777472 −6961.569046	24.310498 24.357417 24.403683	0.095825 0.095825 0.095825	680.731701 680.921574 681.226784	0.749904 0.752580 0.760863

Table 3.3 Performance comparison of various algorithms solving spring design problem

Design variables	Best solutions found							
	Arora (2004) [25]	Coello (2000) [3]	Coello et al. (2002) [16]	Coello et al. (2004) [10]	He et al. (2006) [26]	He et al. (2007) [7]	Kulkarni et al. (2011) [1]	CI algorithm
x_1	0.05339	0.05148	0.05198	0.05000	0.051728	0.05170	0.05060	0.050978
x_2	0.39918	0.35166	0.36396	0.31739	0.357644	0.35712	0.32781	0.339856
x_3	9.18540	11.63220	10.89052	14.03179	11.244543	11.26508	14.05670	12.350557
$g_1(\mathbf{X})$	0.00001	−0.00330	−0.000013	0.00000	−0.000845	−0.00000	−0.05290	2.6906e −007
$g_2(\mathbf{X})$	−0.00001	−0.00010	−0.000021	−0.00007	−1.2600e−05	0.00000	−0.00740	−8.2557e −007
$g_3(\mathbf{X})$	−4.12383	−4.02630	−4.061338	−3.96796	−4.051300	−4.05460	−3.70440	−4.0191
$g_4(\mathbf{X})$	−0.69828	−0.73120	−0.722698	−0.75507	−0.727090	−0.7274	−0.74769	−0.7394
$f(\mathbf{X})$	0.01273	0.01270	0.01268	0.01272	0.0126747	0.01266	0.01350	0.012675

Table 3.4 Statistical results of different methods solving spring design problem

Methods	Best	Mean	Worst	Std.
Arora (2004) [25]	0.0127303	N.A.	N.A.	N.A.
Coello (2000) [3]	0.0127048	0.0127690	0.012822	3.9390e−005
Coello et al. (2002) [16]	0.0126810	0.0127420	0.012973	5.9000e−005
Coello et al. (2004) [10]	0.0127210	0.0135681	0.015116	8.4152e−004
He et al. (2006) [26]	0.0126747	0.0127300	0.012924	5.1985e−004
He et al. (2007) [7]	0.0126652	0.0127072	0.0127191	1.5824e−005
Kulkarni et al. (2011) [1]	0.01350	0.02607	0.05270	N.A.
CI Algorithm	0.012679	0.012719	0.012884	0.000062

over numerous experiments. Since the chosen set of parameters produced sufficiently robust results much effort was not spent in their fine-tuning. Hence, better performance may be obtained through different choice of parameters.

The CI was incorporated with penalty function approach [1, 2] and tested by solving several well studied constrained problems. The results were compared with existing contemporary approaches. A few approaches focused on overcoming the limitation of penalty approach. A self-adaptive penalty approach [3], a dynamic penalty scheme [4], GA with binary representation assisted with traditional penalty function method [5], etc. resulted in premature convergence with high sensitivity to additional parameters. PSO uses penalty factors as searching variables but it is weak in local searches. Also it is not efficient in maintaining balance between exploitation and exploration due to lack of diversity. The approach of penalty parameter was avoided by utilizing feasibility based rule in [5] solving constrained problems. It failed to produce optimum solution in every run and also required an extra fitness function. A variation of feasibility based rule [5] was proposed in [6] for solving constrained non-linear functions. In both the approaches the population in hand is the governing factor of the quality of solutions. The need for an extra fitness function was avoided by HPSO [7] by introducing the feasibility based rule [5] into PSO. The PSO [8] and the homomorphous mapping [9] required feasible solution initially along with set dependent parameters. For some problems, it is quite hard to generate feasible solutions initially and requires additional techniques. In cultural algorithm [10] and cultural differential evolution (CDE) [11] it is seen that there is a lack in diversity of the population. The gradient repair method [12] was implanted into PSO [13] and the number of solution undergoing repair [14] are the key factors of its performance. Taking directions from [15], GA was applied to find solution vector (non-dominated) [16]. In addition, Genetics Adaptive Search (Gene AS) [17], augment Lagrange multiplier method [18], geometric programming approach [19], and a branch and bound technique [20] were also used for solving various constrained benchmark problems addressed above which required additional gradient method.

Table 3.5 Performance comparison of various Algorithms solving Welded Beam Design Problem

Design var.	Coello (2000) [3]	Coello et al. (2002) [16]	Coello et al. (2004) [10]	He et al. (2006) [26]	He et al. (2007) [7]	Deb (2000) [5]	Siddall (1972) [27]	Ragsdell et al. (1976) [19]	CI algorithm
x_1	0.208800	0.205986	0.205700	0.202369	0.20573	0.2489	0.2444	0.2455	0.205730
x_2	3.420500	3.471328	3.470500	3.544214	3.47048	6.1730	6.2189	6.1960	3.635788
x_3	8.997500	9.020224	9.036600	9.048210	9.03662	8.1789	8.2915	8.2730	9.036623
x_4	0.210000	0.206480	0.205700	0.205723	0.20573	−0.2533	0.2444	0.2455	0.205729
$g_1(\mathbf{X})$	−0.337812	−0.074092	−0.00047	−12.83979	0.00010	−5758.6037	−5743.50202	−5743.82651	−0.0115
$g_2(\mathbf{X})$	−353.90260	−0.266227	−0.00156	−1.247467	−0.02656	−255.57690	−4.015209	−4.715097	−7.5194e−004
$g_3(\mathbf{X})$	−0.00120	−0.000495	0.000000	−0.001498	0.00000	−0.004400	0.000000	0.000000	−8.5103e−008
$g_4(\mathbf{X})$	−3.411865	−3.430043	−3.43298	−3.429347	−3.43298	−2.982866	−3.022561	−3.020289	−3.4182
$g_5(\mathbf{X})$	−0.08380	−0.080986	−0.08073	−0.079381	−0.08070	−0.123900	−0.119400	−0.120500	−0.0807
$g_6(\mathbf{X})$	−0.235649	−0.235514	−0.23554	−0.235536	−0.23554	−0.234160	−0.234243	−0.234208	−0.2355
$g_7(\mathbf{X})$	−363.23238	−58.66644	−0.00077	−11.68135	−0.02980	−4465.27092	−3490.46941	−3604.27500	−0.0021
$f(\mathbf{X})$	1.748309	1.728226	1.724852	1.728024	1.72485	2.43311600	2.38154338	2.38593732	1.770316

Table 3.6 Statistical results of different methods solving welded beam design problem

Methods	Best	Mean	Worst	Std.
Coello (2000) [3]	1.748309	1.771973	1.785835	0.011220
Coello et al. (2002) [16]	1.728226	1.792654	1.993408	0.074713
Coello et al. (2004) [10]	1.724852	1.971809	3.179709	0.443131
He et al. (2006) [26]	1.728024	1.748831	1.782143	0.012926
He et al. (2007) [7]	1.724852	1.749040	1.814295	0.040049
Deb (2000) [5]	2.38145	2.38263	2.38355	N.A.
Siddall (1972) [27]	2.3815	N.A.	N.A.	N.A.
Ragsdell et al. (1976) [19]	2.3859	N.A.	N.A.	N.A.
CI algorithm	1.770436	1.779802	1.816707	0.013885

In this study, we apply the CI algorithm with penalty function approach to solve the benchmark problems studied by Coello and Becerra [10]. This test problems solved include two maximization problems (G03 and G08) and six minimization problems (G04, G05, G07, G08, G09, G11). Since all constraints have explicit and simple functional forms, the gradient of the constraints can be derived directly from the constraint set. The characteristics of problems, including the number of decision variables *(DV)*, form of objective function, size of feasible region $(\mathbf{x}\%)$, and the number of linear inequality (*LI*), non-linear equality (*NE*), non-linear inequality (*NI*), and number of active constraints at the reported optimum (a), are summarized in Table 3.1. The size of feasible region, empirically determined by simulation, indicates the difficulty to randomly generate a feasible solution. These problems have been grouped in different categories [4] (refer to Table 3.1) based on the problem characteristics such as, nonlinear objective function (NLOF) [G03, G04, G05, G06, G07, G08, G09, G11], nonlinear equality constraints (NEC) (G03, G05, G11), moderated dimensionality (MD) $(n \geq 5)$ (G03, G07, G09), active constraints (AC) $(a \geq 6)$ (G07) and small feasible region (SFR) $((\mathbf{x}\%) \approx 0)$ (G03, G05, G11).

As presented in Tables 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8 and 3.9, the overall performance of CI with penalty function approach was quite comparable to other existing algorithms. The solution of to G03 problem was within 0.0583 % of the reported optimum [8, 21]. Also, the computational cost, [time and Function Evaluations (FEs)] was quite reasonable. The CI performance for solving the problem G04 was within 0.0000326 % of the reported optimum [11, 12, 21–23] with a reasonable computational cost. The CI solutions to problems G06, G08 and G09 were well within 0.0005242, 0.00000 and 0.01218 %, respectively of the reported optimum [11, 24] with reasonable computational cost and standard deviation. The CI algorithm could solve the problem G05 within 1.0356 % of the reported optimum [21] with standard deviation of 55.0329. Also, the average computational time (30 s) was comparatively higher as compared to solving other problems using CI. This underscored that CI needs to be further modified to make it

Table 3.7 Performance comparison of various algorithms solving pressure vessel design problem

Des. var.	Sandgren (1988) [20]	Kannan et al. (1994) [18]	Deb (1997) [17]	Coello (2000) [3]	Coello et al. (2002) [16]	He et al. (2006) [26]	He et al. (2007) [7]	CI algorithm
x_1	1.1250	1.1250	0.9375	0.8125	0.8125	0.8125	0.81250	14
x_2	0.6250	0.6250	0.5000	0.4375	0.4375	0.4375	0.43750	7
x_3	47.7000	58.2910	48.3290	40.3239	42.0974	42.0913	42.09840	45.3368
x_3	117.7010	43.6900	112.6790	200.0000	176.6540	176.7465	176.6366	140.2538
$g_1(\mathbf{X})$	−0.204390	0.000016	−0.004750	−0.034324	−0.000020	−0.000139	-8.8×10^{-7}	−1.7306e −007
$g_2(\mathbf{X})$	−0.169942	−0.068904	−0.038941	−0.052847	−0.035891	−0.035949	−0.03590	−0.0050
$g_3(\mathbf{X})$	54.226012	−21.22010	−3652.87683	−27.10584	−27.886075	−116.38270	3.12270	−0.0280
$g_4(\mathbf{X})$	−122.29900	−196.31000	−127.32100	−40.00000	−63.34595	−63.253500	−63.36340	−99.7461
$f(\mathbf{X})$	8129.8000	7198.0428	6410.3811	6288.7445	6059.9463	6061.0777	6059.7143	6090.5000

Table 3.8 Statistical results of different methods for solving pressure vessel design problem

Methods	Best	Mean	Worst	Std.
Sandgren (1988) [20]	8129.8000	N.A.	N.A.	N.A.
Kannan et al. (1994) [18]	7198.0428	N.A.	N.A.	N.A.
Deb (1997) [17]	6410.3811	N.A.	N.A.	N.A.
Coello (2000) [3]	6288.7445	6293.8432	6308.1497	7.4133
Coello et al. (2002) [16]	6059.9463	6177.2533	6469.3220	130.9297
He et al. (2006) [26]	6061.0777	6147.1332	6363.8041	86.4545
He et al. (2007) [7]	6059.7143	6099.9323	6288.6770	86.2022
CI algorithm	6090.52639	6090.52689	6090.52849	0.00063

solve the problems with equality constraints. A mechanism similar to the window approach proposed by Ray et al. [24] could be devised. It is also important to mention that the value of reduction factor r varied from very narrow range of 0.98–0.998. This is in contrary to the CI solutions solving unconstrained test problems (refer to Chap. 2 for details). In addition, the solutions to problem G07 and G11 were within 0.211 and 0.344 %, respectively of the reported optimum [11, 12] also required higher computational cost.

Furthermore, the performance of CI was tested solving three well known problems from the mechanical engineering design domain. The approach of CI produced better results for solving the spring design problem within 0.11848 % of the reported optimum [7]. The solution was quite robust with standard deviation 0.000062. In addition, CI solution solving the welded beam design problem and the pressure vessel design problem yielded was within 2.6357 and 0.5049 %, respectively of the reported optimum [10, 16]. Also, the standard deviations and computational cost were quite reasonable.

3.3 Conclusions

The chapter has validated the constraint handling ability of the CI methodology by solving a variety of well known test problems. This also justified the possible application of CI for solving a variety of real world problems. In all the problem solutions, the implemented CI methodology produced sufficiently robust results with reasonable computational cost. It is important to mention here that similar to the original CI approach discussed in Chap. 2, the sampling space was restored to the original one when no significant improvement in the cohort behavior was observed. This helped the solution jump out of possible local minima.

In addition to the advantages few limitations are also observed. The computational performance was essentially governed by the parameter such as sampling

Table 3.9 CI solution details

Problem	Solutions Best Mean Worst	Standard deviation	Avg. no. of function evaluations	Avg. comp. time (seconds)	Closeness to the best reported solution (%)	Reduction factor r
G03	0.998892 0.999417 0.999762	0.000297	28,125	5.5	0.0583	0.98
G04	−30,665.531736 −30,665.529486 −30,665.526082	0.001740	25,125	5.8	0.0000326	0.99
G05	5143.533669 5196.042840 5273.835265	55.032911	42,375	30	1.035657	0.99
G06	−6961.812948 −6961.777472 −6961.569046	0.075814	27,335	4.5	0.0005242	0.98
G07	24.310498 24.357417 24.403683	0.029861	348,750	42.5	0.211540	0.998
G08	0.095825 0.095825 0.095825	0.000000	6000	1.59	0	0.98
G09	680.731701 680.921574 681.226784	0.177770	24,375	15.3	0.012184	0.99
G11	0.749904 0.752580 0.760863	0.004407	25,875	12.5	0.344	0.995
Spring design problem	0.012679 0.012719 0.012884	0.000062	313,500	58	0.11848	0.99
Welded beam design problem	1.770436 1.779802 1.816707	0.013885	25,000	165	2.6359	0.983
Pressure vessel design problem	6090.526390 6090.526895 6090.528495	0.000632	294,000	85	0.5049	0.9970

interval reduction factor r, number of candidates C and number of variations t. In the future, to make the approach more generalized and to improve the rate of convergence, the quality of results, as well as reduce the computational cost, a self adaptive scheme could be developed for these parameters. The authors also see strong potential in the field of game development and mutual learning.

References

1. Kulkarni, A.J., Tai, K.: Solving constrained optimization problems using probability collectives and a penalty function approach. Int. J. Comput. Intell. Appl. **10**(4), 445–470 (2011)
2. Vanderplaat, G.N.: Numerical Optimization Techniques for Engineering Design. McGraw-Hill, New York (1984)
3. Coello Coello, C.A.: Use of self-adaptive penalty approach for engineering optimization problems. Comput. Ind. **41**, 113–127 (2000)
4. Farmani, R., Wright, J.A.: Self-adaptive fitness formulation for constrained optimization. IEEE Trans. Evol. Comput. **7**(5), 445–455 (2003)
5. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Methods Appl. Mech. Eng. **186**, 311–338 (2000)
6. Lampinen, J.: A constraint handling approach for the differential evolution algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, pp. 1468–1473 (2002)
7. He, Q., Wang, L.: A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Appl. Math. Comput. **186**, 1407–1422 (2007)
8. Hu, X., Eberhart, R.: Solving constrained nonlinear optimization problems with particle swarm optimization. In: Proceedings of the 6th World Multi-conference on Systemics, Cybernetics and Informatics (2002)
9. Koziel, S., Michalewicz, Z.: Evolutionary algorithms, homomorphous mapping, and constrained parameter optimization. Evol. Comput. **7**(1), 19–44 (1999)
10. Coello Coello, C.A., Becerra, R.L.: Efficient evolutionary optimization through the use of a cultural algorithm. Eng. Optim. **36**(2), 219–236 (2004)
11. Becerra, R.L., Coello Coello, C.A.: Cultured differential evolution for constrained optimization. Comput. Methods Appl. Mech. Eng. **195**, 4303–4322 (2006)
12. Chootinan, P., Chen, A.: Constraint handling in genetic algorithms using a gradient-based repair method. Comput. Oper. Res. **33**, 2263–2281 (2006)
13. Zahara, E., Hu, C.H.: Solving constrained optimization problems with hybrid particle swarm optimization. Eng. Optim. **40**(11), 1031–1049 (2008)
14. Dong, Y., Tang, J., Xu, B., Wang, D.: An application of swarm optimization to nonlinear programming. Comput. Math. Appl. **49**, 1655–1668 (2005)
15. Hedar, A.R., Fukushima, M.: Derivative-free simulated annealing method for constrained continuous global optimization. J. Global Optim. **35**, 521–549 (2006)
16. Coello Coello, C.A., Montes, E.M.: Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Adv. Eng. Inform. **16**, 193–203 (2002)
17. Deb, K.: GeneAS: a robust optimal design technique for mechanical component design. In: Dasgupta, D., Michalewicz, Z., (eds.) Evolutionary Algorithms in Engineering Applications, pp. 497–514. Springer, New York (1997)
18. Kannan, B.K., Kramer, S.N.: An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. ASME J. Mech. Des. **116**, 405–411 (1994)
19. Ragsdell, K.M., Phillips, D.T.: Optimal design of a class of welded structures using geometric programming. ASME J. Eng. Ind. Ser. B **98**(3), 1021–1025 (1976)
20. Sandgren, E.: Nonlinear integer and discrete programming in mechanical design. In: Proceedings of the ASME Design Technology Conference, pp. 95–105 (1988)
21. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Trans. Evol. Comput. **4**(3), 284–294 (2000)
22. Hamida, S.B., Schoenauer, M.: ASCHEA: new results using adaptive segregational constraint handling. In: Fogel, D.B., et al. (eds.) Proceedings of the IEEE Congress on Evolutionary Computation, pp. 884–889 (2002)

23. Montes, E.M., Coello Coello, C.A.: A simple multimembered evolution strategy to solve constrained optimization problems. Technical Report EVOCINV-04-2003, Evolutionary Computation Group at CINVESTAV, Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México D.F., México
24. Ray, T., Tai, K., Seow, K.C.: An evolutionary algorithm for constrained optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 771–777 (2000)
25. Arora, J.S.: Introduction to Optimum Design. Elsevier Academic Press, San Diego (2004)
26. He, Q., Wang, L.: An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng. Appl. Artif. Intell. **20**, 89–99 (2006)
27. Siddall, J.N.: Analytical Design-Making in Engineering Design. Prentice-Hall, Englewood Cliffs (1972)