A COMPARATIVE ANALYSIS OF CONSTRAINT-HANDLING MECHANISMS FOR SOLVING ENGINEERING DESIGN PROBLEMS

İlker GÖLCÜK[1*]

[1]İzmir Bakırçay Üniversitesi, Mühendislik ve Mimarlık Fakültesi, Endüstri Mühendisliği Bölümü, İzmir
ORCID No: http://orcid.org/0000-0002-8430-7952

Keywords

Constraint-handling Mechanisms, Constrained Optimization, Metaheuristic Optimization, Whale Optimization Algorithm

Abstract

Optimization problems have numerous real-life applications in science and engineering. The engineering design problems are usually subject to various constraints. Although many state-of-the-art metaheuristic optimization algorithms have been developed during the last decades, these algorithms require additional constraint-handling mechanisms to cope with constrained optimization problems. Therefore, selecting a suitable constraint-handling mechanism requires extensive trial-and-error experiments, which is time-consuming and demanding. In this study, a comparative analysis of the eight constraint handling mechanisms is carried out, guiding decision-makers in their optimization practices. The constraint-handling techniques are used along with the Whale Optimization Algorithm (WOA), and 19 real-life mechanical design problems, which are also part of the CEC2020 benchmark suite, are tested in the experimental analysis. The nonparametric statistical analysis incorporating Nemenyi and Holm post-hoc procedures shows that the inverse tangent constraint-handling and eclectic penalty methods exhibit high performance in real-life mechanical design problems.

MÜHENDİSLİK TASARIM PROBLEMLERİNİ ÇÖZMEK İÇİN KISIT-YÖNETİMİ MEKANİZMALARININ KARŞILAŞTIRMALI BİR ANALİZİ

Anahtar Kelimeler

Kısıt-yönetimi Mekanizmaları, Kısıtlı Optimizasyon, Metasezgisel Optimizasyon, Balina Optimizasyon Algoritması

Öz

Optimizasyon problemlerinin bilim ve mühendislikte çok sayıda gerçek yaşam uygulaması vardır. Mühendislik tasarım problemleri genellikle çeşitli kısıtlamalara tabidir. Son on yılda birçok modern meta-sezgisel optimizasyon algoritması geliştirilmiş olsa da bu algoritmalar, kısıtlı optimizasyon problemleriyle başa çıkmak için ek kısıt-yönetimi mekanizmaları gerektirir. Bu nedenle, uygun bir kısıt-yönetimi mekanizmasının seçilmesi, zaman alıcı ve zorlu olan kapsamlı deneme yanılma deneyleri gerektirir. Bu çalışmada, karar vericilere optimizasyon uygulamalarında yol gösterecek şekilde sekiz kısıt-yönetimi mekanizmasının karşılaştırmalı bir analizi gerçekleştirilmiştir. Kısıt-yönetimi teknikleri, Balina Optimizasyon Algoritmasıyla (BOA) birlikte kullanılmış ve deneysel analizde yine CEC2020 kıyaslama paketinin bir parçası olan 19 gerçek hayat mekanik tasarım problemi test edilmiştir. Nemenyi ve Holm post-hoc prosedürlerini içeren nonparametrik istatistiksel analiz, ters tanjant kısıt-yönetimi ve eklektik ceza yöntemlerinin gerçek hayattaki mekanik tasarım problemlerinde yüksek performans sergilediğini göstermektedir.

Araştırma Makalesi		Research Article	
Başvuru Tarihi	: 15.11.2020	Submission Date	: 15.11.2020
Kabul Tarihi	: 31.05.2021	Accepted Date	: 31.05.2021

1. Introduction

Optimization problems are ubiquitous in many fields of science and engineering. The main characteristic of engineering optimization problems is that

*Sorumlu yazar; e-posta : ilker.golcuk@bakircay.edu.tr

different types of constraints are involved. The presence of the constraints makes challenging optimization problems even more complicated. The field of swarm intelligence provides a vast array of bio-inspired metaheuristic optimization algorithms to tackle with real-life optimization problems. Regrettably, these metaheuristic algorithms are almost always designed for unconstrained optimization tasks. When solving constrained optimization problems, additional constraint-handling mechanisms need to be adopted. Therefore, the selection of a suitable constraint-handling technique is a critical issue that necessitates careful analysis.

The individual solutions in a metaheuristic algorithm are divided into two categories: feasible and infeasible solutions. Feasible individuals refer to the solutions that satisfy all the constraints, while infeasible individuals fail to satisfy any constraints. The challenge in the constrained optimization problems is dealing with infeasible individuals throughout the search process (Mallipeddi & Suganthan, 2010). Because infeasible solutions may have potentially valuable information about the fitness landscape, continuing the search process by discarding infeasible solutions is not an effective strategy. Therefore, many different approaches have been proposed to deal with exploiting useful information from infeasible solutions. These approaches are known as constraint-handling mechanisms (Mezura-Montes & Coello Coello, 2011).

In the literature, different constraint handling methods have been proposed. Michalewicz and Schoenauer (1996) categorized constraint-handling mechanisms into four groups as follows: 1) penalty-function based methods, 2) separating feasible and infeasible solutions, 3) feasibility-preservation based methods, and 4) hybrid methods. In addition to these categories, constrained optimization problems are represented as a multi-objective optimization problem (Deb, 2000). However, multi-objective optimization is computationally expensive in comparison with the other counterparts. Other hybrid approaches can be found in (Coello Coello, 2002; Mezura-Montes & Coello Coello, 2011).

The salient deficiency of the constraint-handling techniques is that decision-makers should perform many trial-and-error experiments to decide on a particular constraint-handling technique. The No Free Lunch (NFL) theorem (Wolpert & Macready, 1997) states that there is no constraint-handling technique that outperforms all the other counterparts on every problem instance. Therefore, numerous trial-and-error experiments need to be carried out, which becomes time-consuming and impractical. Reported results on the performance of the constraint-handling mechanisms become highly supportive in selecting the right technique for a given problem. However, there is a research gap in the literature that very few works report the relative performance of the constraint-handling techniques.

In this study, a comprehensive evaluation of the state-of-the-art constraint-handling techniques is carried out. To this end, a recently introduced high-performing bio-inspired metaheuristic algorithm, Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016), is utilized as an optimization engine. The WOA algorithm with eight constraint handling techniques is used to solve real-life mechanical design/engineering problems. Accordingly, a total of 19 non-convex and constrained problems of the Congress on Evolutionary Computation 2020 (CEC2020) test suite (Kumar, Wu, Ali, Mallipeddi, Suganthan, Das, 2020a) are used to evaluate the performance of the constraint-handling techniques. Furthermore, nonparametric statistical tests are used to verify significant differences among constraint-handling techniques. This paper contributes to the literature in the following ways:

- A comprehensive evaluation of eight constraint-handling techniques on the real-life constrained optimization problems of the CEC2020 problems is provided.
- Performance analysis of the WOA algorithm with different constraint-handling techniques is carried out.
- The statistical analysis sheds light on the relative performance of the constraint-handling techniques so that the results can be used to select a suitable constraint-handling technique in future studies.

The organization of the paper is as follows: Section 2 presents the basics of WOA. Section 3 gives the definitions of constrained optimization problems and state-of-the-art constraint-handling techniques. Section 4 is devoted to the experimental study and statistical analysis. Finally, concluding remarks and discussion is given in Section 5.

2. Whale Optimization Algorithm

The Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016) is a recently developed metaheuristic algorithm inspired byhumpback whales' hunting practice.

The two fundamental actions of the search procedure of WOA are shrinking encircling and spiral updating. Decreasing initial step-size gradually, search agents move towards the incumbent solution in the shrinking encircling phase. The shrinking encircling is represented as follows:

$$D = |C \cdot X^*(t) - X(t)| \quad (1)$$

$$X(t+1) = X^*(t) - A \cdot D \quad (2)$$

where C and A are the coordinate vectors, $X^*(t)$represents the position vector of the incumbent solution, and $X(t)$ denotes the position vector at iteration t and $\cdot$ is the component-wise multiplication. The coefficients C and A are determined as:

$$A = 2 \cdot a \cdot r - a \quad (3)$$

$$C = 2 \cdot r \quad (4)$$

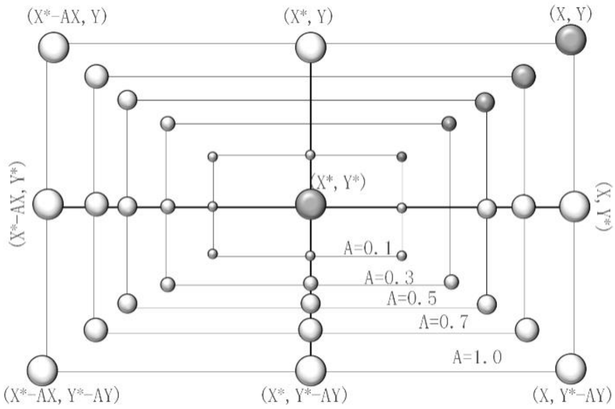
The coordinate vector a is linearly decreased from 2 to 0 by applying the formulation of $a(t) = 2 - (t - 1) \cdot (2/maxIter)$, in which the maximum number of generations is denoted by $maxIter$.

On the other hand, the helix-shaped movement of whales is modeled as bubble-net attacking. The mathematical definition of the bubble-net attacking is presented as:

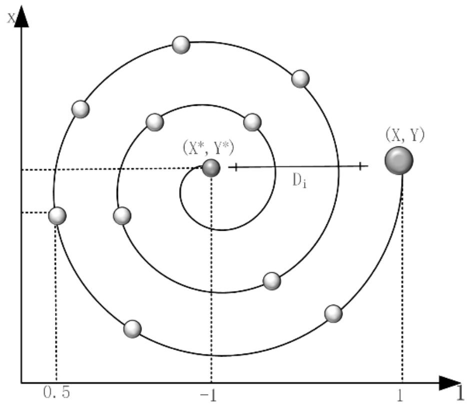
$$X(t+1) = D' \cdot e^{bl} \cdot \cos(2\pi l) + X^*(t) \quad (5)$$

where $D' = |X^*(t) - X(t)|$ shows the distance between the search agent and the incumbent solution, and b and l are being a constant and a random number between $[-1,1]$, respectively.

The spiral updating is another crucial search procedure of the WOA. The whales move towards the incumbent solution within a dwindling circle and a spiral-shaped path simultaneously. To represent this movement, it is presumed that the whales choose between either the shrinking encircling or bubble-net attacking with equal chances. These two mechanisms are illustrated in Figure 1.



a)Shrinking Encircling



b)Spiral Updating

Figure 1. Bubble-net Search Mechanism

Mathematically speaking, the exploitation phase of the WOA is represented as:

$$\boldsymbol{X}(t+1) = \begin{cases} \boldsymbol{X}^*(t) - \boldsymbol{A} \cdot \boldsymbol{D}, & \text{if } p < 0.5 \\ \boldsymbol{D}' \cdot e^{bl} \cdot \cos(2\pi l) + \boldsymbol{X}^*(t), & \text{if } p \geq 0.5 \end{cases} \quad (6)$$

where p is a racndom number in [0,1].

Instead of moving towards the incumbent solution in the exploration stage, the solution vector is updated by approaching a randomly chosen solution. Exploration mechanism is expressed by:

$$D = |C \cdot X_{rand}(t) - X(t)| \quad (7)$$

$$X(t+1) = X_{rand}(t) - A \cdot D \quad (8)$$

where X_{rand} is the position vector of a randomly chosen whale. The flowchart of the WOA algorithm is given in Figure 2.

The WOA algorithm begins with the generation of initial population, which encompasses of number of solutions. The initial solutions are randomly generated based on uniform distribution, where each dimension of the solution lies within the lower and upper bounds of the decision variables. Then, the step-size coefficients are prepared as given in Eqs. 3-

4. As mentioned before, each whale either selects shrinking encircling or bubble-net attacking. The movement of each whale is performed based on the value of a random number p and the step size coefficient A. The necessary equations are shown in Figure 2.

The steps of WOA are applied for the unconstrained optimization problems. When the constraints are defined in the model, constraint-handling techniquesmanage constraints in different ways. In the next section, constraint-handling mechanisms are introduced.

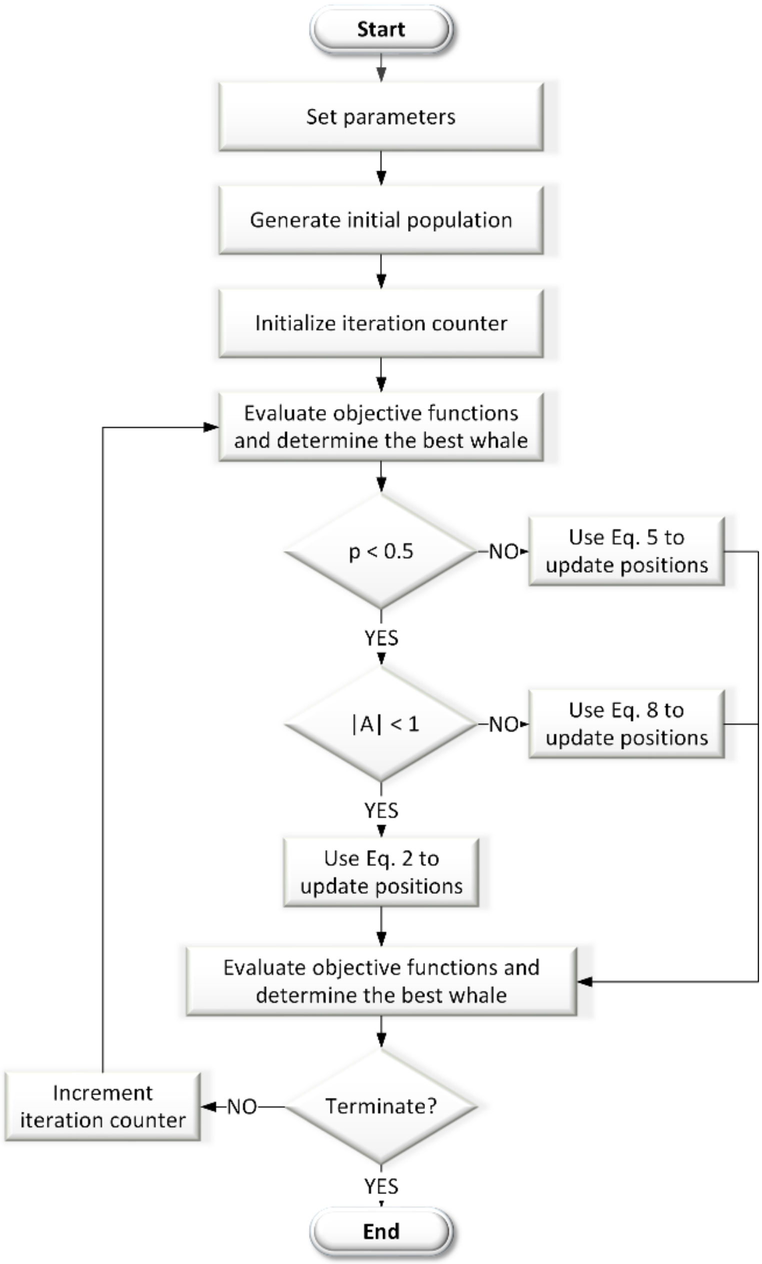


Figure 2. Flowchart of the WOA

3. Constraint-Handling Mechanisms

The constrained optimization problem is formalized as follows:

$$\begin{aligned} \min_{x} \quad & f(\boldsymbol{x}) \\ s.t. \quad & g_i(\boldsymbol{x}) \le 0, i = 1,\dots,m \\ & h_j(\boldsymbol{x}) = 0, j = 1,\dots,p \\ & \boldsymbol{x} \in S \subseteq \Re^{nDim} \end{aligned} \qquad (9)$$

There are $m+p$ constraints in which g_i denotes *i*-th inequality constraint and h_j represents *j*-th equality constraint. Here, $x \in S$ denotes *nDim* dimensional solution vector.

The set S comprises of box constraints that specify admissible intervals of each dimension. The lower and upper box constraints for each component of the solution vector are specified as *lb* and *ub*, respectively. A solution vector $x \in S$ is termed feasible if all constraints are satisfied simultaneously. The feasible set F is given by:

$$\mathrm{F} := \left\{ \boldsymbol{x} \in S : g_i(\boldsymbol{x}) \le 0 \wedge h_j(\boldsymbol{x}) = 0, \forall i, j \right\} \qquad (10)$$

where all $m+p$ constraints are satisfied by the solutions in the feasible set.

The measure of infeasibility is calculated when the constraints are violated. The constraint violation of a candidate solution is calculated as:

$$G_i(x) = \begin{cases} \max\left(0, g_i(\boldsymbol{x})\right), & \forall i \in [1, m] \\ \max\left(0, |h_i(\boldsymbol{x})| - \varepsilon\right), & \forall i \in [m+1, m+p] \end{cases} \qquad (11)$$

where ε is the tolerance parameter given by 10^{-4} and the $G_i(\boldsymbol{x})$ is the constraint violation of the *i*-th constraint.

3.1 Static Penalty Method

In the static penalty methods, the penalized cost function is given as follows (Homaifar, Qi, & Lai, 1994):

$$\min_x \quad \phi(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m+p} r_i G_i(\boldsymbol{x}) \qquad (12)$$

where r_i is the penalty factor of the *i*-th constraint and $f(x)$ is the objective function value. Here, the coefficients r_i do not change throughout iterations.

3.2 Dynamic Penalty Method

In the dynamic penalty method, penalty coefficient is given by $r_i = (ct)^{\alpha}$ where c and α are constants. The penalized cost function is given by (Joines & Houck, 1994):

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}) + (c \times iter)^{\alpha} \sum_{i=1}^{m+p} G_i(x) \qquad (13)$$

where $\phi(x)$ is the penalized cost function.

To increase the performance of the dynamic penalty method, normalization of the objective function value and the constraint violations can be performed as follows (Simon, 2013):

$$\begin{aligned} &\phi(\boldsymbol{x}) = f'(\boldsymbol{x}) + (c \times iter)^{\alpha} M'(\boldsymbol{x}) \\ &M'(\boldsymbol{x}) = \begin{cases} M(\boldsymbol{x}) / \max\limits_x M(\boldsymbol{x}), & \text{if } \max_x M(\boldsymbol{x}) > 0 \\ 0, & \text{if } \max_x M(\boldsymbol{x}) = 0 \end{cases} \\ &f'(\boldsymbol{x}) = f(\boldsymbol{x}) / \max_x |f(\boldsymbol{x})| \\ &M(\boldsymbol{x}) = \sum_{i=1}^{m+p} G_i(\boldsymbol{x}) \end{aligned} \qquad (14)$$

3.3 Dynamic Penalty MethodCombined with Superiority of Feasible Points

In the dynamic penalty method combined with the superiority of feasible points, it is ensured that all of the feasible solutions have a lower cost than the infeasible points. Therefore, the penalized cost is rewritten as (Simon, 2013):

$$\phi(\boldsymbol{x}) = \begin{cases} f'(\boldsymbol{x}), & \text{if } x \in \mathrm{F} \\ f'(\boldsymbol{x}) + (c \times iter)^{\alpha} M'(\boldsymbol{x}), & \text{if } x \notin \mathrm{F} \end{cases} \qquad (15)$$

3.4 Exponential Dynamic Penalty Method

The exponential dynamic penalty function is given as (Carlson & Shonkwiler, 1998):

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}) \exp\left(M(\boldsymbol{x}) / T\right) \qquad (16)$$

where T is a monotonically nonincreasing function of the iteration. Generally, the variable T is taken as $1/\sqrt{iter}$.

Because $f(x)$ can also become negative, the following modification is used instead,

$$\begin{aligned} &\phi(\boldsymbol{x}) = f(\boldsymbol{x}) \exp\left(\alpha M'(\boldsymbol{x}) / T\right) \\ &f'(\boldsymbol{x}) = f(\boldsymbol{x}) - \min_x f(\boldsymbol{x}) \end{aligned} \qquad (17)$$

where α is used to adjust the relative weight of the constraint violation.

3.5 Exponential Dynamic Penalty Method with the Superiority of Feasible Solutions

The superiority of the feasible solutions can be integrated into the exponential dynamic penalty method as follows(Simon, 2013):

$$\phi(\boldsymbol{x}) = \begin{cases} f'(\boldsymbol{x}), & \text{if } x \in \mathrm{F} \\ f'(\boldsymbol{x})\exp\left(\alpha M'(\boldsymbol{x})/T\right), & \text{if } x \notin \mathrm{F} \end{cases} \quad (18)$$

3.6 Adaptive Penalty Weights

The adaptive penalty weights method adjusts penalty weights by getting information from the population of individuals during the search process. Penalty weights are set as follows (Hadj-Alouane & Bean, 1997):

$$r_i(iter+1) = \begin{cases} r_i(iter)/\beta_1, & \text{if } g_{best} \text{ is feasible for k generations} \\ r_i(iter) \times \beta_2, & \text{if there is no feasible solution for k generations} \\ r_i(iter), & \text{otherwise} \end{cases} \quad (19)$$

where k, β_1, and β_2 are user-defined parameters.

3.7 Eclectic Penalty Method

The eclectic penalty method enforces the superiority of feasible solutions. The penalized cost is given as (Morales & Quezada, 1998):

$$\phi(\boldsymbol{x}) = \begin{cases} f(\boldsymbol{x}), & \text{if } x \in \mathrm{F} \\ L\left(1 - \dfrac{s(\boldsymbol{x})}{m+p}\right), & \text{if } x \notin \mathrm{F} \end{cases} \quad (20)$$

where L is a large constant, $s(x)$is the number of constraints that are satisfied by x.

3.8 Inverse Tangent Constraint Handling

Inverse tangent constraint handling (ITCH) method is reduced original constrained optimization problem to unconstrained optimization problem as follows (Kim, Maruta, & Sugie, 2010) :

$$\phi(x) = \begin{cases} \arctan[f(x)] - \frac{\pi}{2}, & \text{if } x \in \mathcal{F} \\ \max_i\left(0, G_i(x)\right), & \text{if } x \notin \mathcal{F} \end{cases} \quad (21)$$

where *arctan* is the inverse tangent function.

If the solution is feasible, then the inverse tangent function maps the solution to$[-\infty, 0]$. Otherwise, penalty functions map to positive values, which guarantees that feasible solutions are superior to infeasible solutions.

4. Experimental Study

In this section, constraint-handling mechanisms are used to solve constrained optimization problems within WOA. To this end, an extensive computational study has been conducted on mechanical design problems, which are also part of the Congress on Evolutionary Computation 2020 (CEC2020) test suite (Kumar et al., 2020a). The following problems have been tackled: weight minimization of a speed reducer(Chew & Zheng, 2012), optimal design of industrial refrigeration system(Andrei & Andrei, 2013), tension/compression spring design(Belegundu & Arora, 1985), pressure vessel design(Sandgren, 1988), welded beam design(Ragsdell & Phillips, 1976),three-bar truss design problem(Nowacki, 1973), multiple disk clutch brake design problem(Steven, 2002), planetary gear train design optimization problem(Sandgren, 1990), step-cone pulley problem(Rao, 1996), robot gripper problem(Osyczka, Krenich, & Karas, 1999), hydro-static, thrust bearing design problem(Siddall, 1982), four-stage gear box problem(Yokota, Taguchi, & Gen, 1998), 10-Bar truss optimization with frequency constraints(Grandhi, 1993), rolling element bearing(Gupta, Tiwari, & Nair, 2007), gas transmission compressor design(Beightler & Phillips, 1976), tension/compression string design problem (case 2)(Arora, 2004), gear train design problem(Sandgren, 1990), Himmelblau's function(Himmelblau, 2018), and topology optimization(Sigmund, 2001). The mathematical formulations of the problems can be found in (Kumar et al., 2020a, 2020b). The constrained optimization problems handled within the scope of the present work have a varying number of dimensions (nDim), inequality constraints (nInequal) and equality constraints (nEqual). The problem properties are summarized in Table 1.

Table 1
Details of the Problems

No	Constrained Optimization Problems	Properties		
		nDim	*nInequal*	*nEqual*
F_{01}	Weight Minimization of a Speed Reducer	7	11	0
F_{02}	Optimal Design of Industrial refrigeration System	14	15	0
F_{03}	Tension/compression spring design (case 1)	3	3	0
F_{04}	Pressure vessel design	4	4	0
F_{05}	Welded beam design	4	5	0
F_{06}	Three-bar truss design problem	2	3	0
F_{07}	Multiple disk clutch brake design problem	5	7	0
F_{08}	Planetary gear train design optimization problem	9	10	1
F_{09}	Step-cone pulley problem	5	8	3
F_{10}	Robot gripper problem	7	7	0
F_{11}	Hydro-static thrust bearing design problem	4	7	0
F_{12}	Four-stage gear box problem	22	86	0
F_{13}	10-bar truss design	10	3	0
F_{14}	Rolling element bearing	10	9	0
F_{15}	Gas Transmission Compressor Design (GTCD)	4	1	0
F_{16}	Tension/compression spring design (case 2)	3	8	0
F_{17}	Gear train design Problem	4	1	1
F_{18}	Himmelblau's Function	5	6	0
F_{19}	Topology Optimization	30	30	0

In the experimental study, population size and the maximum number of iterations are 30 and 500, respectively. On the other hand, the default parameters of the constraint-handling methods have been adopted. The penalty factor of the static penalty is set to 1e5. In dynamic penalty, C and α are set to 10 and 2, respectively. The α parameter of the exponential dynamic penalty methods is set to 10. In the adaptive penalty method, β_1, β_2, k parameters are set to 4, 3, and *nDim*, respectively. Finally, *L* parameter of the eclectic penalty method is set to 1e6.

4.1 Test Results for Engineering Design Problems

In this section, test results for the engineering design problems are given. All the constraint-handling methods have been run in the same conditions. The WOA algorithm has been run for 30 replications. The best, mean, standard deviation and worst objective function values are tabulated for each constraint-handling technique. Furthermore, the feasibility of the solution over 30 runs has been indicated. Feasible solutions are used when collecting statistics over replications. If any feasible solution cannot be produced over 30 replications, then the "Feasible" property is set to 0. Table 1 shows the results of the constraint-handling techniques. For the sake of increasing readability, the static penalty (SP), dynamic penalty (DP), dynamic penalty method combined with superiority of feasible points (DPSF), exponential dynamic penalty method (EDP), exponential dynamic penalty method with the superiority of feasible solutions (EDPSF), adaptive penalty weights (AP), eclectic penalty method (EP), and inverse tangent constraint handling (ITCH) are placed into the columns with the given abbreviation in the parentheses. Each row represents the problem and collected statistics about each constraint-handling technique. Tables2-4 show the test results of the constraint handling techniques. The best-performing constraint handling techniques in terms of the best and mean metrics are highlighted in Table 2. In Table 3, standard deviation (std_dev) values and the worst results are given. The feasibility of the solutions is given in Table 4. According to Table 4, the symbol ✗ represents the situation that any feasible solution cannot be obtained during the replications, while ✓ indicates that the constraint handling technique has managed to find at least one solution that satisfies all of the constraints.

Table 2
The Best and Mean Results

		Constraint handling techniques							
Problem	Perf.	**SP**	**DP**	**DPSF**	**EDP**	**EDPSF**	**AP**	**EP**	**ITCH**
F_{01}	*Best*	2.998E+03	3.016E+03	3.014E+03	3.014E+03	3.017E+03	3.011E+03	2.998E+03	**2.996E+03**
	Mean	3.044E+03	3.064E+03	3.116E+03	3.034E+03	3.058E+03	3.059E+03	3.102E+03	**3.011E+03**
F_{02}	*Best*	6.322E-01	3.203E+01	3.928E-01	3.835E-03	**1.382E-01**	7.228E-01	2.148E+03	2.109E-01
	Mean	8.567E+02	1.431E+05	1.391E+04	8.967E-03	**1.284E+02**	9.885E+03	2.148E+03	6.111E+03
F_{03}	*Best*	1.268E-02	1.300E-02	1.289E-02	1.320E-02	1.283E-02	1.296E-02	**1.267E-02**	1.269E-02
	Mean	1.424E-02	1.406E-02	1.404E-02	1.465E-02	1.376E-02	1.414E-02	1.318E-02	**1.313E-02**
F_{04}	*Best*	4.747E+02	7.200E+03	6.210E+03	4.757E+02	6.521E+03	4.803E+02	6.483E+03	**6.133E+03**
	Mean	4.927E+02	8.137E+03	7.851E+03	5.055E+02	7.799E+03	8.553E+02	7.016E+03	**7.011E+03**
F_{05}	*Best*	1.773E+00	1.701E+00	1.907E+00	1.046E-01	1.859E+00	2.001E+00	**1.672E+00**	1.684E+00
	Mean	2.003E+00	2.172E+00	2.367E+00	2.354E-01	2.338E+00	2.001E+00	1.964E+00	**1.912E+00**
F_{06}	*Best*	2.640E+02	2.639E+02	2.640E+02	2.520E+00	2.639E+02	1.488E+00	2.639E+02	**2.639E+02**
	Mean	2.651E+02	2.647E+02	2.648E+02	2.771E+01	2.647E+02	7.708E+00	**2.639E+02**	2.639E+02
F_{07}	*Best*	2.378E-01	2.363E-01	2.362E-01	2.377E-01	2.366E-01	1.262E-01	2.352E-01	**2.352E-01**
	Mean	2.450E-01	2.421E-01	2.426E-01	2.442E-01	2.457E-01	1.345E-01	2.353E-01	**2.353E-01**
F_{08}	*Best*	5.273E-01	5.273E-01	5.258E-01	5.273E-01	5.269E-01	5.300E-01	5.258E-01	**5.258E-01**
	Mean	5.493E-01	5.667E-01	5.570E-01	5.423E-01	5.664E-01	5.722E-01	5.305E-01	**5.298E-01**
F_{09}	*Best*	8.524E+00	1.654E+01	**1.640E+01**	8.524E+00	8.692E+00	1.672E+01	1.035E+01	1.647E+01
	Mean	1.017E+01	1.676E+01	1.661E+01	1.004E+01	1.012E+01	1.672E+01	1.899E+01	**1.661E+01**
F_{10}	*Best*	3.471E+00	3.622E+00	3.476E+00	**3.288E+00**	3.321E+00	3.839E+00	3.442E+00	3.438E+00
	Mean	6.060E+00	5.730E+00	5.748E+00	**4.032E+00**	8.164E+00	5.419E+00	4.950E+00	4.828E+00
F_{11}	*Best*	3.961E+03	2.599E+03	2.685E+03	-1.419E+27	-3.877E+29	2.535E+03	1.867E+03	**1.861E+03**
	Mean	3.961E+03	3.922E+03	3.994E+03	-1.006E+26	-3.842E+28	3.738E+03	2.668E+03	**2.649E+03**
F_{12}	*Best*	3.592E+00	8.875E+00	**5.987E+01**	0.000E+00	0.000E+00	3.224E+00	2.148E+01	6.967E+01
	Mean	6.202E+00	3.797E+01	**5.987E+01**	0.000E+00	0.000E+00	3.332E+00	6.164E+01	6.967E+01
F_{13}	*Best*	5.615E+02	5.594E+02	5.557E+02	5.652E+02	**5.533E+02**	2.028E+01	5.551E+02	5.646E+02
	Mean	6.040E+02	6.084E+02	6.018E+02	6.102E+02	6.086E+02	1.164E+02	**5.940E+02**	6.018E+02
F_{14}	*Best*	1.727E+04	1.716E+04	1.710E+04	1.717E+04	1.712E+04	1.711E+04	1.700E+04	**1.698E+04**
	Mean	1.833E+04	1.903E+04	1.844E+04	1.826E+04	1.852E+04	1.869E+04	1.738E+04	**1.709E+04**
F_{15}	*Best*	2.968E+06	2.971E+06	2.968E+06	2.976E+06	2.972E+06	1.330E+06	**2.965E+06**	2.965E+06
	Mean	3.034E+06	3.023E+06	3.040E+06	3.033E+06	3.024E+06	1.780E+06	**2.967E+06**	2.969E+06
F_{16}	*Best*	2.645E-01	2.647E-01	2.669E+00	9.258E-04	2.720E+00	1.448E-02	**2.659E+00**	2.659E+00
	Mean	2.877E-01	3.073E-01	3.076E+00	3.127E-03	3.075E+00	1.451E-01	2.919E+00	**2.866E+00**
F_{17}	*Best*	0.000E+00	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	1.611E-20
	Mean	0.000E+00	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**	3.289E-17
F_{18}	*Best*	-3.034E+04	-3.022E+04	-2.959E+04	-2.230E+04	-2.984E+04	-3.050E+04	**-3.055E+04**	-3.051E+04
	Mean	-2.997E+04	-2.988E+04	-2.853E+04	-2.230E+04	-2.861E+04	-2.990E+04	-3.023E+04	**-3.027E+04**
F_{19}	*Best*	2.640E+00	2.640E+00	2.640E+00	2.639E+00	2.640E+00	2.640E+00	2.639E+00	**2.639E+00**
	Mean	2.654E+00	2.671E+00	2.672E+00	2.674E+00	2.663E+00	2.668E+00	**2.639E+00**	2.639E+00

Table 3
Standard Deviation Values and The Worst Results

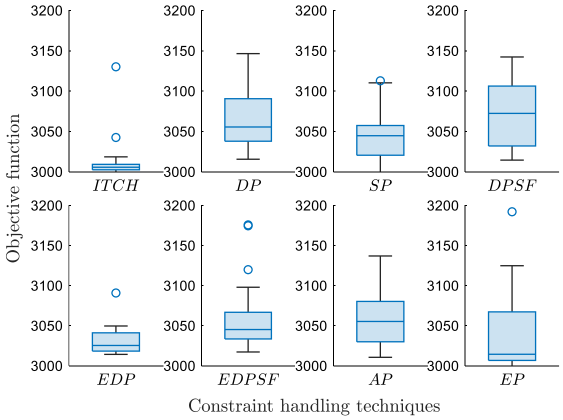
		Constraint handling techniques							
Problem	Perf.	**SP**	**DP**	**DPSF**	**EDP**	**EDPSF**	**AP**	**EP**	**ITCH**
F_{01}	*Std_Dev*	2.942E+01	3.480E+01	1.851E+02	2.319E+01	3.972E+01	3.213E+01	2.449E+02	2.398E+01
	Worst	3.113E+03	3.146E+03	4.003E+03	3.091E+03	3.176E+03	3.137E+03	4.216E+03	3.130E+03
F_{02}	*Std_Dev*	1.915E+03	5.109E+05	5.617E+04	1.008E-02	4.311E+02	4.063E+04	0.000E+00	1.876E+04
	Worst	7.917E+03	2.751E+06	2.907E+05	5.440E-02	2.108E+03	1.960E+05	2.148E+03	8.536E+04
F_{03}	*Std_Dev*	1.210E-03	1.020E-03	9.590E-04	1.230E-03	6.819E-04	8.915E-04	5.539E-04	4.416E-04
	Worst	1.879E-02	1.725E-02	1.637E-02	1.863E-02	1.550E-02	1.664E-02	1.456E-02	1.419E-02
F_{04}	*Std_Dev*	1.876E+01	8.191E+02	1.580E+03	3.664E+01	1.255E+03	9.326E+02	4.805E+02	6.755E+02
	Worst	5.692E+02	8.718E+03	1.266E+04	6.052E+02	1.229E+04	5.529E+03	8.990E+03	9.361E+03
F_{05}	*Std_Dev*	1.965E-01	3.618E-01	3.970E-01	9.745E-02	3.970E-01	0.000E+00	2.482E-01	2.145E-01
	Worst	2.248E+00	2.805E+00	3.474E+00	5.904E-01	3.387E+00	2.001E+00	2.651E+00	2.523E+00
F_{06}	*Std_Dev*	6.599E-01	6.984E-01	8.567E-01	1.992E+01	6.961E-01	5.265E+00	5.177E-02	7.372E-02
	Worst	2.659E+02	2.660E+02	2.674E+02	7.279E+01	2.665E+02	1.852E+01	2.641E+02	2.643E+02
F_{07}	*Std_Dev*	6.499E-03	3.285E-03	4.326E-03	5.782E-03	7.448E-03	5.716E-03	8.324E-05	4.607E-05
	Worst	2.639E-01	2.494E-01	2.534E-01	2.591E-01	2.657E-01	1.500E-01	2.357E-01	2.355E-01
F_{08}	*Std_Dev*	2.341E-02	4.822E-02	5.903E-02	1.926E-02	4.970E-02	4.944E-02	6.410E-03	3.352E-03
	Worst	6.283E-01	7.258E-01	8.471E-01	5.888E-01	7.327E-01	7.698E-01	5.567E-01	5.371E-01
F_{09}	*Std_Dev*	1.044E+00	2.051E-01	7.461E-02	9.379E-01	9.384E-01	0.000E+00	3.649E+00	7.345E-02
	Worst	1.282E+01	1.730E+01	1.688E+01	1.181E+01	1.222E+01	1.672E+01	2.416E+01	1.692E+01
F_{10}	*Std_Dev*	2.814E+00	1.971E+00	2.413E+00	4.868E-01	1.327E+01	1.244E+00	8.028E-01	8.164E-01
	Worst	1.976E+01	1.224E+01	1.694E+01	4.802E+00	7.703E+01	1.004E+01	7.114E+00	6.785E+00
F_{11}	*Std_Dev*	0.000E+00	1.277E+03	8.385E+02	2.974E+26	1.023E+29	1.156E+03	4.241E+02	4.886E+02
	Worst	3.961E+03	6.916E+03	5.867E+03	-1.748E+21	-1.854E+19	6.268E+03	4.183E+03	3.784E+03
F_{12}	*Std_Dev*	2.548E+00	2.482E+01	0.000E+00	0.000E+00	0.000E+00	1.006E-01	3.778E+01	0.000E+00
	Worst	1.528E+01	1.007E+02	5.987E+01	0.000E+00	0.000E+00	3.539E+00	2.035E+02	6.967E+01
F_{13}	*Std_Dev*	2.717E+01	3.346E+01	2.939E+01	2.465E+01	3.635E+01	1.396E+02	2.480E+01	3.124E+01
	Worst	6.658E+02	6.895E+02	6.737E+02	6.617E+02	6.843E+02	5.590E+02	6.424E+02	7.111E+02
F_{14}	*Std_Dev*	1.222E+03	1.484E+03	1.685E+03	1.278E+03	2.052E+03	1.398E+03	8.794E+02	1.859E+02
	Worst	2.202E+04	2.290E+04	2.400E+04	2.328E+04	2.439E+04	2.239E+04	2.147E+04	1.775E+04
F_{15}	*Std_Dev*	5.223E+04	5.124E+04	5.120E+04	4.438E+04	4.424E+04	2.523E+05	2.549E+03	9.205E+03
	Worst	3.145E+06	3.183E+06	3.165E+06	3.141E+06	3.113E+06	2.312E+06	2.978E+06	3.005E+06
F_{16}	*Std_Dev*	3.588E-02	1.460E-01	2.575E-01	1.608E-03	2.283E-01	1.386E-01	1.569E-01	1.399E-01
	Worst	3.674E-01	1.060E+00	3.728E+00	7.505E-03	3.680E+00	5.661E-01	3.302E+00	3.191E+00
F_{17}	*Std_Dev*	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	3.484E-17
	Worst	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	0.000E+00	1.080E-16
F_{18}	*Std_Dev*	2.388E+02	2.029E+02	7.175E+02	9.776E-02	1.108E+03	4.182E+02	1.539E+02	1.369E+02
	Worst	-2.955E+04	-2.960E+04	-2.691E+04	-2.230E+04	-2.558E+04	-2.908E+04	-2.984E+04	-3.001E+04
F_{19}	*Std_Dev*	1.845E-02	3.560E-02	4.185E-02	5.585E-02	2.741E-02	3.407E-02	4.186E-06	5.328E-06
	Worst	2.717E+00	2.769E+00	2.790E+00	2.910E+00	2.754E+00	2.769E+00	2.639E+00	2.639E+00

Table 4
Feasibility of the Solutions

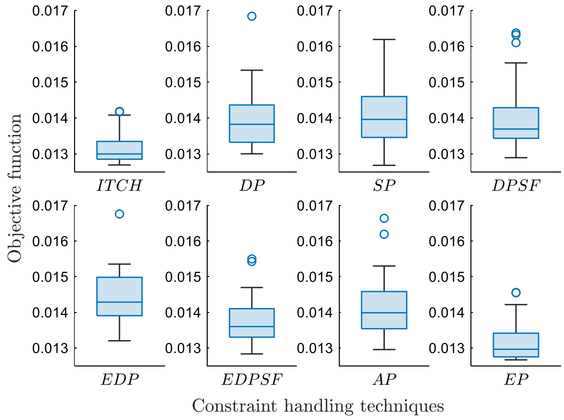
	Constraint handling techniques							
Problem	**SP**	**DP**	**DPSF**	**EDP**	**EDPSF**	**AP**	**EP**	**ITCH**
F_{01}	✓	✓	✓	✓	✓	✓	✓	✓
F_{02}	✓	✓	✓	×	✓	✓	✓	✓
F_{03}	✓	✓	✓	✓	✓	✓	✓	✓
F_{04}	×	✓	✓	×	✓	×	✓	✓
F_{05}	✓	✓	✓	×	✓	✓	✓	✓
F_{06}	✓	✓	✓	×	✓	×	✓	✓
F_{07}	✓	✓	✓	✓	✓	×	✓	✓
F_{08}	✓	✓	✓	✓	✓	✓	✓	✓
F_{09}	×	✓	✓	×	×	✓	×	✓
F_{10}	✓	✓	✓	✓	✓	✓	✓	✓
F_{11}	✓	✓	✓	×	×	✓	✓	✓
F_{12}	×	×	✓	×	×	×	×	✓
F_{13}	✓	✓	✓	✓	✓	×	✓	✓
F_{14}	✓	✓	✓	✓	✓	✓	✓	✓
F_{15}	✓	✓	✓	✓	✓	×	✓	✓
F_{16}	×	×	✓	×	✓	×	✓	✓
F_{17}	✓	✓	✓	✓	✓	✓	✓	✓
F_{18}	✓	✓	✓	×	✓	✓	✓	✓
F_{19}	✓	✓	✓	✓	✓	✓	✓	✓

According to the results, all the constraint-handling techniques have found feasible solutions for the weight minimization of a speed reducer (F_{01}) problem. ITCH has been found to be superior in terms of the best and mean results. In the optimal design of industrial refrigeration system (F_{02}) problem, EDPSF has performed superior concerning the best and mean results. Also, EDP has not achieved to find a feasible solution over the replications. In the tension/compression spring design (case 1) (F_{03}), EP has found the best solution, while ITCH has the best mean result. The ITCH has outperformed all the other algorithms in the pressure vessel design (F_{04}), multiple disk clutch brake design problem (F_{07}), planetary gear train design optimization problem (F_{08}), hydro-static thrust bearing design problem (F_{11}), and rolling element bearing (F_{14}) in terms of the best and mean results. In the welded beam design (F_{05}), EP has returned the best solution, and ITCH has produced the best mean value. The EDP algorithm has not found any feasible solution for the F_{05}. On the other hand, concerning the best and mean values for the three-bar truss design problem (F_{06}), the best performing constraint-handling techniques are ITCH and EP, respectively. For the step-cone pulley problem (F_{09}), DP and ITCH have exhibited high performance, whereas SP, EDP, EDPSF, and EP have not found any feasible solution throughout replications. In the robot gripper problem (F_{10}), EDP has become the best performing constraint-handling technique. In the four-stage gearbox problem (F_{12}), DPSF has performed the best and becomes one of the two constraint handling techniques with the ITCH that has achieved feasible solutions. In the 10-bar truss design (F13) problem, EDPSP has yielded the best feasible result, and EP has exhibited the highest performance in terms of the mean results. In the gas transmission compressor design (GTCD), EP has exhibited the best performance in terms of the best and mean results. In the tension/compression spring design (case 2) (F_{16}) and Himmelblau's function (F_{18}), the best and mean results have been achieved by EP and ITCH, respectively. In the gear train design problem (F_{17}), almost all the constraint handling techniques have performed equally well, the ITCH being slightly worse than the others. Finally, in the topology optimization (F_{19}), ITCH has exhibited the best performance while the EP has accomplished the best mean scores.

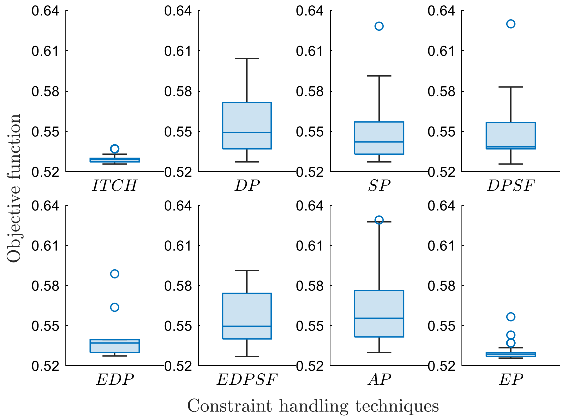
Another critical performance indicator is the standard deviation. The standard deviation shows the algorithms' robustness—the smaller the standard deviation, the more robust the constraint handling technique. Figure3 shows the boxplots of the constraint handling techniques for the most commonly studied problems in the literature, which are weight minimization of a speed reducer (F_{01}), tension/compression spring design (case 1) (F_{03}), and planetary gear train design optimization problem (F_{08}).



a) F_{01}



b) F_{03}



c) F_{08}

Figure 3. Box plots of the constraint-handling techniques

As shown in Figure3, ITCH has a considerably low standard deviation, which indicates the technique's robustness. Besides its high performance in terms of objective function value, ITCH has been found to be the most robust constraint-handling technique in the vast majority of the test instances. The ITCH algorithm has outperformed other constraint-handling techniques in 42.10% of instances in terms of the best results. Furthermore, the ITCH has dominated other algorithms in 57.89% of the test instances concerning the mean results. In the next section, nonparametric statistical analysis is used to verify the superiority of the algorithms.

4.2 Statistical verification

In the experimental analysis, the performances of the algorithms are tested by using statistical hypothesis tests. However, parametric tests require the satisfaction of assumptions such as independence, normality, and homoscedasticity. Because these assumptions cannot be satisfied in most of the cases, nonparametric statistical tests are recommended. To this end, pairwise comparisons are conducted between the algorithms. While comparing the different techniques in a pairwise manner, the probability of making false discoveries might not be controlled, which is also known as "losing control on the family-wise error rates" (Derrac, García, Molina, & Herrera, 2011). Multiple comparison tests ($1 \times N$ or $N \times N$) are recommended in the literature (Gölcük & Ozsoydan, 2020).

Therefore, this work adopts the Friedman test with multiple comparisons ($N \times N$) in order to discover statistically significant differences among constraint-handling techniques. When the null hypothesis of the Friedman test is rejected, the next step is devoted to post-hoc analysis, which serves as finding out the individual differences between constraint-handling techniques. Accordingly, this work utilizes Nemenyi and Holm tests in the post-hoc analysis. The Friedman test results, along with the average ranks, are given in Table 5.

Table 5
Average Ranks and Obtained *p*-values of (Friedman Tests)

algorithms	*overall avg. ranks*
EDP	6.2895
AP	5.5000
SP	5.1842
DP	5.0526
EDPSF	4.7632
DPSF	4.1579
EP	2.7368
ITCH	2.3158
p-values (*adj. ties*)	0.000
decision on H_0	reject H_0

In Table 5, constraint-handling techniques are sorted in descending order concerning the average ranks. According to Table 5, ITCH is the best performing constraint-handling technique with an average rank of 2.3158. The second-ranked constraint-handling technique is EP, with an average rank of 2.7368. The EP is followed by DPSF, EDPSF, DP, SP, AP, and EDP. According to the Friedman test, the p-value is found as 0.000 so that the null-hypothesis is rejected. It is inferred that at least one pair of constraint-handling techniques are significantly different from each other. Nemenyi and Holm tests have been carried out to discover the pairwise differences between constraint-handling techniques, as given in Table 6.

Table 6
Statistical Comparison of Constraint-Handling Techniques

comparisons	*z-score*	*unadjusted*	*Nemenyi*	*Holm*
ITCH vs EDP	5	0 (+)	0 (+)	0 (+)
EP vs EDP	4.47	0 (+)	0 (+)	0 (+)
ITCH vs AP	4.007	0 (+)	0.002 (+)	0.002 (+)
ITCH vs SP	3.609	0 (+)	0.009 (+)	0.008 (+)
EP vs AP	3.477	0.001 (+)	0.014 (+)	0.012 (+)
ITCH vs DP	3.444	0.001 (+)	0.016 (+)	0.013 (+)
ITCH vs EDPSF	3.08	0.002 (+)	0.058 (~)	0.046 (+)
EP vs SP	3.08	0.002 (+)	0.058 (~)	0.046 (+)
EP vs DP	2.914	0.004 (+)	0.1 (~)	0.071 (~)
DPSF vs EDP	2.682	0.007 (+)	0.205 (~)	0.139 (~)
EP vs EDPSF	2.55	0.011 (+)	0.302 (~)	0.194 (~)
ITCH vs DPSF	2.318	0.02 (+)	0.573 (~)	0.348 (~)
EDPSF vs EDP	1.921	0.055 (~)	1 (~)	0.877 (~)
EP vs DPSF	1.788	0.074 (~)	1 (~)	1 (~)
DPSF vs AP	1.689	0.091 (~)	1 (~)	1 (~)
DP vs EDP	1.556	0.12 (~)	1 (~)	1 (~)
SP vs EDP	1.391	0.164 (~)	1 (~)	1 (~)
DPSF vs SP	1.291	0.197 (~)	1 (~)	1 (~)
DPSF vs DP	1.126	0.26 (~)	1 (~)	1 (~)
AP vs EDP	0.993	0.321 (~)	1 (~)	1 (~)
EDPSF vs AP	0.927	0.354 (~)	1 (~)	1 (~)
DPSF vs EDPSF	0.762	0.446 (~)	1 (~)	1 (~)
DP vs AP	0.563	0.573 (~)	1 (~)	1 (~)
ITCH vs EP	0.53	0.596 (~)	1 (~)	1 (~)
EDPSF vs SP	0.53	0.596 (~)	1 (~)	1 (~)
SP vs AP	0.397	0.691 (~)	1 (~)	1 (~)
EDPSF vs DP	0.364	0.716 (~)	1 (~)	1 (~)
DP vs SP	0.166	0.868 (~)	1 (~)	1 (~)

In Table 6, z-scores, unadjusted *p*-values, Nemenyi, and Holm test results are tabulated. According to Nemenyi and Holm tests, the ITCH outranks EDP, AP, SP, DP, and EDPSF at the significance level of 5%. Furthermore, according to the Holm test, significant differences are detected between the EP and the EDP, AP, and SP. Note that EP is the second-best constraint handling technique in terms of average rankings. The comparison between EP and ITCH indicates that they perform equally well, and the statistical difference cannot be observed. Apart from ITCH and EP, the rest of the constraint handling techniques exhibit equal performance, and the statistical difference cannot be verified. For the statistical significance level of 10%, the statistical difference between EP and DP can be confirmed.

The post-hoc analysis clearly shows that ITCH has superior performance compared toits counterparts for solving CEC2020 engineering design problems.

5. Discussion and conclusion

In this study, research and publication ethics were followed. In this paper, the static penalty, dynamic penalty, dynamic penalty method combined with superiority of feasible points, exponential dynamic penalty method, exponential dynamic penalty method with the superiority of feasible solutions, adaptive weights, eclectic penalty method, and inverse tangent constraint handling methods are used to solve real-life mechanical design/engineering problems. The considered problems are part of the CEC2020 constrained optimization test suite. These employed constraint-handling techniques are used within WOA, proving its ability to solve real-life optimization problems with high success.

The results show that inverse tangent constraint handling is the best performing constraint handling technique. In 42.10% of instances,the best results belong to the inverse tangent constraint handling technique. Furthermore, this constraint handling technique has achieved the best-mean results in 57.89% of the test instances. On the other hand, the second-best constraint handling technique is eclectic penalty method with the average rank of 2.7368. Nonparametric statistical analysis with Nemenyi and Holm post-hoc tests show that inverse tangent constraint handling outranks all other counterparts, excepting two methods: eclectic penalty and dynamic penalty method combined with superiority of feasible points. The results frankly show that inverse tangent can be the choice of the general-purpose constraint-handling mechanisms in many situations.

The produced results may lead to new studies in various research directions. In future studies, the ensemble of constraint-handling techniques can be used to solve real-life optimization problems. Also, hyper-heuristics can be used to select constraint-handling techniques autonomously. Finally, the other bio-inspired algorithms can be integrated into the research mentioned aboveto solvethe wide-variety of real-life constrained optimization problems.

Conflict of Interest

Conflict of interest was not declared by authors.

References

Andrei, N., & Andrei, N. (2013). Nonlinear optimization applications using the GAMS technology. New York: *Springer*. Retrieved from: https://link.springer.com/book/10.1007/978-1-4614-6797-7

Arora, J. S. (2004). Introduction to optimum design. Oxford: *Elsevier*. Retrieved from: https://www.sciencedirect.com/book/9780128008065/introduction-to-optimum-design

Beightler, C. S., & Phillips, D. T. (1976). Applied geometric programming. New York: *John Wiley & Sons.* Doi: https://doi.org/10.1080/03052157708960622

Belegundu, A. D., & Arora, J. S. (1985). A study of mathematical programming methods for structural optimization. Part I: Theory. *International Journal for Numerical Methods in Engineering*, 21(9), 1583-1599. Doi: https://doi.org/10.1002/nme.1620210904

Carlson, S. E., & Shonkwiler, R. (1998). *Annealing a genetic algorithm over constraints.* Paper presented at the IEEE International Conference on Systems, Man, and Cybernetics,San Diego, CA, USA. Doi: https://doi.org/10.1109/ICSMC.1998.726702

Chew, S. H., & Zheng, Q. (2012). Integral Global Optimization: Theory, Implementation and Applications (Vol. 298): *Springer Science & Business Media.* Doi*:* https://doi.org/10.1007/978-3-642-46623-6

Coello Coello, C. A. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11), 1245-1287. Doi:https://doi.org/10.1016/S0045-7825(01)00323-1

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2), 311-338. Doi: https://doi.org/10.1016/S0045-7825(99)00389-8

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1), 3-18. Doi: https://doi.org/10.1016/j.swevo.2011.02.002

Gölcük, İ., & Ozsoydan, F. B. (2020). Evolutionary and adaptive inheritance enhanced Grey Wolf Optimization algorithm for binary domains. *Knowledge-Based Systems*, 194, 105586. Doi: https://doi.org/10.1016/j.knosys.2020.105586

Grandhi, R. (1993). Structural optimization with frequency constraints-a review. *AIAA journal*, 31(12), 2296-2303. Doi: https://doi.org/10.2514/3.11928

Gupta, S., Tiwari, R., & Nair, S. B. (2007). Multi-objective design optimisation of rolling bearings using genetic algorithms. *Mechanism and Machine Theory*, 42(10), 1418-1443. Doi: https://doi.org/10.1016/j.mechmachtheory.2006.10.002

Hadj-Alouane, A. B., & Bean, J. C. (1997). A Genetic Algorithm for the Multiple-Choice Integer Program. *Operations Research*, 45(1), 92-101. Doi: https://doi.org/10.1287/opre.45.1.92

Himmelblau, D. M. (2018). Applied nonlinear programming: *McGraw-Hill.* Retrieved from: https://www.amazon.com/Applied-Nonlinear-Programming-Mautner-Himmelblau/dp/0070289212

Homaifar, A., Qi, C. X., & Lai, S. H. (1994). Constrained Optimization Via Genetic Algorithms. *SIMULATION*, 62(4), 242-253. Doi: https://doi.org/10.1177/003754979406200405

Joines, J. A., & Houck, C. R. (1994). *On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's*. Paper presented at the Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, Orlando, FL, USA. Doi: https://doi.org/10.1109/ICEC.1994.349995

Kim, T. H., Maruta, I., & Sugie, T. (2010). A simple and efficient constrained particle swarm optimization and its application to engineering design problems. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 224(2), 389-400. Doi: https://doi.org/ 10.1243/ 09544062JMES1732

Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N., & Das, S. (2020a*). Guidelines for real-world single-objective constrained optimisation competition.* Retrieved from: https://github.com/P-N-Suganthan/2020-RW-Constrained-Optimisation

Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N., & Das, S. (2020b). A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm and Evolutionary Computation,* 56, 100693. Doi: https://doi.org/10.1016/j.swevo.2020.100693

Mallipeddi, R., & Suganthan, P. N. (2010). Ensemble of constraint handling techniques. *IEEE Transactions on Evolutionary Computation*, 14(4), 561-579. Doi: https://doi.org/10.1109/TEVC.2009.2033582

Mezura-Montes, E., & Coello Coello, C. A. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1(4), 173-194. Doi: https://doi.org/ 10.1016/ j.swevo.2011.10.001

Michalewicz, Z., & Schoenauer, M. (1996). Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1), 1-32. Doi: https://doi.org/10.1162/evco.1996.4.1.1

Mirjalili, S., & Lewis, A. (2016). The Whale Optimization Algorithm. *Advances in Engineering Software*, 95, 51-67. Doi: https://doi.org/10.1016/j.advengsoft.2016.01.008

Morales, A., & Quezada, C. (1998). *A univeral eclectic genetic algorithm for constrained optimization*. Paper presented at the 6th European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany. Retrieved from: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.468.4349&rep=rep1&type=pdf

Nowacki, H. (1973). *Optimization in pre-contract ship design*. Paper presented at the International Conference on Computer Applications in the Automation of Shipyard Operation and Ship Design, held by IFIP/IFAC/JSNA, Tokyo, Japan. Retrieved from: https://trid.trb.org/view.aspx?id=14541

Osyczka, A., Krenich, S., & Karas, K. (1999). *Optimum design of robot grippers using genetic algorithms*. Paper presented at the Proceedings of the Third World Congress of Structural and Multidisciplinary Optimization (WCSMO), Buffalo, New York. Doi: https://doi.org/10.1007/978-3-7091-2498-7_14

Ragsdell, K. M., & Phillips, D. T. (1976). Optimal design of a class of welded structures using geometric programming. *ASME. J. Eng. Ind.*, 98(3), 1021-1025. Do: https://doi.org/10.1115/1.3438995

Rao, S. S. (1996). Further topics in optimization. In Engineering Optimization: Theory and Practice (pp. 779-783): *John Wiley & Sons*. Retrieved from: https://books.google.com.tr/books/about/Engineering_Optimization.html?id=nuoryE4IwMoC&redir_esc=y

Sandgren, E. (1988). *Nonlinear integer and discrete programming in mechanical design*. Paper presented at the Proceeding of the ASME design technology conference. Doi: https://doi.org/10.1115/DETC1988-0012

Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *J. Mechn. Des.*, 112, 223-229. Doi: https://doi.org/10.1115/1.2912596

Siddall, J. N. (1982). Optimal engineering design: principles and applications: *CRC Press.* Retrieved from: https://books.google.com.tr/books/about/Optimal_Engineering_Design.html?id=i2hyniQpecYC&redir_esc=y

Sigmund, O. (2001). A 99 line topology optimization code written in Matlab. *Structural and Multidisciplinary Optimization*, 21(2), 120-127. Doi: https://doi.org/10.1007/s001580050176

Simon, D. (2013). Evolutionary optimization algorithms. New York: *John Wiley & Sons.* Retrieved from: https://www.wiley.com/en-ao/Evolutionary+Optimization+Algorithms-p-9780470937419

Steven, G. (2002). Evolutionary algorithms for single and multicriteria design optimization. A. Osyczka. *Springer Verlag*, Berlin, 2002, ISBN 3-7908-1418-01. Structural and Multidisciplinary Optimization, 24(1), 88-89. Doi: https://doi.org/10.1007/s00158-002-0218-y

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67-82. https://doi.org/10.1109/4235.585893

Yokota, T., Taguchi, T., & Gen, M. (1998). A solution method for optimal weight design problem of the gear using genetic algorithms. *Computers & Industrial Engineering*, 35(3-4), 523-526. Doi: https://doi.org/10.1016/S0360-8352(98)00149-1