Complex & Intelligent Systems (2021) 7:1565–1596
https://doi.org/10.1007/s40747-021-00283-3

ORIGINAL ARTICLE

Cohort intelligence with self-adaptive penalty function approach hybridized with colliding bodies optimization algorithm for discrete and mixed variable constrained problems

Ishaan R. Kale[1] · **Anand J. Kulkarni**[1]

Received: 5 October 2019 / Accepted: 20 January 2021 / Published online: 18 February 2021

Abstract
Recently, several socio-/bio-inspired algorithms have been proposed for solving a variety of problems. Generally, they perform well when applied for solving unconstrained problems; however, their performance degenerates when applied for solving constrained problems. Several types of penalty function approaches have been proposed so far for handling linear and non-linear constraints. Even though the approach is quite easy to understand, the precise choice of penalty parameter is very much important. It may further necessitate significant number of preliminary trials. To overcome this limitation, a new self-adaptive penalty function (SAPF) approach is proposed and incorporated into socio-inspired Cohort Intelligence (CI) algorithm. This approach is referred to as CI–SAPF. Furthermore, CI–SAPF approach is hybridized with Colliding Bodies Optimization (CBO) algorithm referred to as CI–SAPF–CBO algorithm. The performance of the CI–SAPF and CI–SAPF–CBO algorithms is validated by solving discrete and mixed variable problems from truss structure domain, design engineering domain, and several problems of linear and nonlinear in nature. Furthermore, the applicability of the proposed techniques is validated by solving two real-world applications from manufacturing engineering domain. The results obtained from CI–SAPF and CI–SAPF–CBO are promising and computationally efficient when compared with other nature inspired optimization algorithms. A non-parametric Wilcoxon's rank sum test is performed on the obtained statistical solutions to examine the significance of CI–SAPF–CBO. In addition, the effect of the penalty parameter on pseudo-objective function, penalty function and constrained violations is analyzed and discussed along with the advantages over other algorithms.

Keywords Self-adaptive penalty function approach · Cohort intelligence · Colliding bodies optimization · Discrete and mixed variable problems · Linear and nonlinear constraints

Abbreviations

AI	Artificial intelligence
CI	Cohort intelligence
SPF	Static penalty function approach
SAPF	Self-adaptive penalty function approach
CBO	Colliding bodies optimization
TSP	Travelling salesman problems
GA	Genetic algorithm
PSO	Particle swarm optimization
ACO	Ant colony optimization
BA	Bat algorithm
FA	Firefly algorithm
IA	Ideology algorithm
SOS	Symbiotic organism search
PC	Probability collectives
SELO	Socio evolution and learning optimization
IA	Ideology algorithm
EA	Election algorithm
ECO	Election campaign optimization
LCA	League championship algorithm
SLC	Soccer league computation
TLBO	Teaching learning-based optimization
SGO	Social group optimization
SLO	Social learning algorithms
SACI	Self-adaptive cohort intelligence
DPF	Dynamic penalty function
COR	Coefficient of restitution
PSOPC	PSO with passive congregation

✉ Ishaan R. Kale
ishaan.kale@sitpune.edu.in

Anand J. Kulkarni
anand.kulkarni@sitpune.edu.in

[1] Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune 412115, India

HS	Harmony search
MBA	Mine blast algorithm
FL	Fuzzy logic
B&B	Branch and bound
NLP	Nonlinear programming
ADS	Adaptive dimensional search
EAs	Evolutionary algorithms
HPSO	Hybrid particle swarm optimization
DHPSACO	Discrete heuristic particle swarm ant colony optimization
ISCSO	International student competition in structural optimization
GHN	Generalized Hopfield network
AIS-GA	Artificial immune system with genetic algorithm
AIS-GA-C	AIS-GA with clearing
SOS	Symbiotic organisms search
SA	Simulated annealing
SQP	Sequential quadratic programming
S*I*QP	Sequential penalty quadratic programming
CPSO	Co-evolutionary particle swarm optimization
NIDPM	Nonlinear integer and discrete programming
NSGA	Nondominated sorting genetic algorithm
AIA	Artificial immune algorithm
OIO	Optics-inspired optimization
RNES	Rank-niche evolution strategy
CS	Cuckoo search
AHGA	Adaptive hybrid genetic algorithm
ABC	Artificial bee colony
HGSS	Hybrid genetic simulated swarm

Introduction

The mechanical design engineering and truss structure optimization domain problems are complex and cumbersome to solve as they involve linear and nonlinear constraints. These problems become more challenging when they have discrete and mixed design variables. Several Artificial Intelligence (AI)-based optimization techniques such as Particle Swarm Optimization (PSO) [14, 59], Firefly Algorithm (FA) [23], Probability Collectives (PC) [41, 44, 45], Colliding Bodies Optimization (CBO) Kaveh and Mahdavi [38], Symbiosis Organism Search (SOS) (2014), Mine Blast Algorithm (MBA) [73], Cuckoo Search (CS) [22], Generalized Hopfield Networks (GHN) [77], Genetic Algorithm (GA) [3, 9, 18, 56, 71, 86, 87] and socio based algorithms such as Cohort Intelligence (CI) [46], Ideology Algorithm (IA) [88], Socio evolution and learning optimization algorithm (SELO) [52] have been developed so far. The real-world problems generally are constrained in nature. Several constraint handling techniques have been developed so far such as penalty-based methods, probability-based methods, feasibility-based methods, etc.

The penalty-based methods convert the constrained problem into unconstrained problem. The approach is characterized by a penalty parameter which necessitates significant number of preliminary trials to set its appropriate value. The penalty function approach is widely used due to its simple construction and easy implementation. Several penalty-based constraint handling techniques have been proposed so far, such as barrier (death) penalty function approach, which is based on elimination of infeasible solution [60], exact penalty function [30] and dynamic penalty function [34] approaches are based on setting the penalty parameter value and multiplication of factors (penalty reduction or expansion factor), respectively. Other techniques are also proposed such as annealing penalty function approach [63], Carlson et al. [5] which is based on the idea of Simulated Annealing (SA) and adaptive penalty function [25, 27, 81, 90] aimed at eliminating the setting of penalty parameter. In penalty-based segregated GA Le et al. [54], a distinct penalty parameter is set for different evaluated fitness functions. So far, these techniques have been successfully employed with nature inspired optimization techniques to deal with linear and nonlinear constraints. These techniques are simple and easy to apply for solving wide variety of constrained optimization problems [57, 91], however, as the number of constraints increase their performance degenerates [60]. An exact penalty approach is adopted by Shin et al. [80] and Wu and Chow [86] for nonlinear optimization problems having discrete design variables. For every independent problem, several preliminary trials are required to set an appropriate penalty parameter [30, 62]. Similar approach is adopted in FA [23] and CI algorithm (CI-SPF) [35] for solving discrete and mixed variable problems with linear as well as nonlinear constraints from engineering design and truss structure domains. However, it is noticed that the selection of penalty parameter becomes tedious with the increase in number of constraints.

The dynamic penalty function [40] incorporated with augmented Lagrange multiplier approach Viswanathan and Grossmann [84] is used for solving discrete and mixed variable problems from design engineering domain. In this approach, penalty parameter is multiplied by a suitable factor to penalize the cost function. Similar to the dynamic penalty function approach, Curtis and Nocedal [13] introduced flexible penalty function to handle nonlinear constraints. In this approach, the penalty parameter is arbitrarily chosen from the prescribed interval rather than a fixed value which influentially guided the convergence. Shih and Yang [77] introduced a generalized Hopfield network using extended penalty function approach. In this approach, the penalty parameter is initialized based on an arbitrary value (0 or 1) and then updated iteratively with an incremental

multiplication factor. However, if the multiplication factor is too high the objective function value may become unstable and the solution may stuck into the local minima. An adaptive penalty function approach is proposed by Nanakorn and Meesomklin [65] in which the modified binary scaling technique is employed to scale the fitness value. Broyden and Attia [4] proposed a smooth sequential penalty function incorporated with Quasi Newton approach. It is then combined with orthogonal transformation based on Jacobian constraints. A non-stationary multistage penalty function approach is implemented by Parsopoulos and Vrahatis [67]. It is then followed by Coath and Halgamuge [12] along with a feasibility preservation method for solving nonlinear problems. Coello [10] proposed a self-adaptive penalty function approach which splits the penalty function into two distinct parts such as sum of violated constraints and number of violated constraints. Nie [66] proposed a novel semi-penalty approach considering the qualities of Sequential Quadratic Programming (SQP) method and Sequential Penalty Quadratic Programming (S*l*QP) method where both equality and inequality constraints are distinctly treated.

An external penalty function scheme with relaxation strategy is incorporated into Adaptive Dimensional Search (ADS) method by [28]. In this strategy, the infeasible solution is retained to escape from the local minima. At the saturation stage, the intensity of penalty parameter is reduced by multiplying the reduction factor. After every stagnation escape period, the solution is recalculated using an updated penalty parameter and then compared with previous saturated solution. A parameter less approach referred to as niched penalty function approach is proposed by Deb and Agrawal [16]. In this approach, a feasible solution is selected based on three criteria such as accept the feasible solution rather than infeasible solution, accept best-fitted solution from two feasible solutions and accept infeasible solution based on fewer number of constraint violations. These three rules are then referred to as feasibility-based rules and are used as a constraint handling technique [17] and later implemented by Kulkarni and Tai [51]. It is further modified by Kulkarni et al. [45] in which after a stagnation period, worst solution found so far is accepted. The algorithm then restarts to help the solution jump out of local minima. It is successfully applied for solving problems from design engineering and truss structure domains.

Various constraint handling techniques associated with other nature-inspired algorithms are discussed in the literature along with their limitations. Apart from these, there are several socio-inspired optimization algorithms have been proposed so far, such as PC Wolpert et al. [85, Kulkarni and Tai [42], SOS [8], SELO [52]. SOS models symbiotic interaction] strategies that the independent agents (organisms) use to survive in the ecosystem. Political election-based socio algorithms such as IA [82], Election Algorithm (EA) Emami and Derakhshan [21], Election Campaign Optimization (ECO) (Lv et al. [61] are also proposed. ECO models the social behavior of voters where the candidates attempt to pursue maximum support from them. Based on the position of the candidates and voters, the global and local voters are considered. The uniform distribution method is used to identify the supported focus of the candidates. EA is based on the process of advertisement during the election campaign. With similar motivation, IA is proposed by Teo et al. [88]. It emphasizes the behavior of political parties aiming to improve their rank. The League Championship Algorithm (LCA) [31, 36] is inspired from distinct features of the sports activity. LCA models the social tendencies of sport competition in a league. Similar to the LCA, Soccer League Competition (SLC) algorithm is proposed by Moosavian and Roodsari [64]. It is based on the interaction of players during a soccer match. A physics-based Optic-Inspired Optimization (OIO) method [37], [32] and [33] works on the optical characteristics of concave and convex mirrors. The socio-inspired algorithm such as Teaching Learning-Based Optimization (TLBO) [68] models the influence of teaching process on students' performance. The influence of teaching process on students' outcome is modeled. A Social Group Optimization (SGO) Satapathy and Naik [75] and Social Learning Optimization (SLO) Liu et al. [58] are based on the process of propagation of human knowledge in the learning society/group to solve complex engineering problems.

The CI algorithm is proposed by Kulkarni et al. [46]. It is motivated from the socially learning behavior of the candidates such as following, interacting, cooperating and competing with every other candidate in the cohort. It is implemented for constrained problems and applied to solve combinatorial NP-hard 0–1 Knapsack problem with the number of items varying from 4 to 75 Kulkarni and Shabir [48]. The constraints involved in this problem are handled by a problem-specific probability-based constraint handling technique. The algorithm yielded competent results as compared to integer programming solutions. This approach is also applied for solving real-world combinatorial problems from healthcare and logistics domains as well as for large-sized complex problems from the Cross Border Supply Chain domain [50], Traveling Salesman Problem (TSP) [49] and several benchmark problems [76]. A Self-adaptive Cohort Intelligence (SACI) algorithm [1] is proposed using tournament mutation operator and a self-adaptive scheme to update the sampling interval. It is tested on several benchmark problems and obtained promising results. The static and dynamic penalty function approach is incorporated in CI (CI–SPF and CI–DPF) for solving several test problems and manufacturing engineering problems Kulkarni et al. [48]. The CI–SPF is adopted for solving complex problems from truss structure and mechanical engineering domain

[35]. In the current work, a self-adaptive penalty function (SAPF) approach is proposed and incorporated into the CI algorithm. This approach eliminated the effort of setting the penalty parameter and no other supporting parameter is required. Additionally, the CI–SAPF algorithm is hybridized with CBO (referred to as CI–SAPF–CBO) which eliminated the dependence of the CI algorithm on sampling space reduction factor. It is discussed in Sect. "CI-SAPF". The proposed CI–SAPF and CI–SAPF–CBO are tested for solving 10 discrete truss structure problems, 11 mixed variable design engineering problems and 17 discrete variable test problems (linear, nonlinear, global, convex and monotonous functions). The performance is validated by comparing the solutions with other contemporary techniques available in the literature. Finally, the influence of SAPF on penalty function, constraint violations, pseudo-objective function is thoroughly discussed. The proposed techniques are also applied to solve two real-world applications from manufacturing engineering such as a) multi-pass turning process problem and b) multi-pass milling process problem.

The paper is organized as follows: the mathematical representation of SAPF approach is presented in Sect. "Self-adaptive penalty function (SAPF)". Section "Cohort intelligence (CI) algorithm" describes the basic version of CI algorithm along with its characteristics. A CI–SAPF algorithm is presented in Sect. "CI-SAPF" along with its pseudo code. The detailed description of CBO algorithm and its characteristics are mentioned in Sect. "Colliding bodies optimization (CBO)". The pseudo code of the CBO algorithm is also presented in the same section. Section "Framework of CI–SAPF–CBO" describes the hybrid CI–SAPF–CBO algorithm and its mathematical expression with flowchart. Section "Test examples" discusses the discrete and mixed variable problems from truss structure, design engineering, linear and non-linear domains. In the same section, the results obtained from CI–SAPF and CI–SAPF–CBO algorithms are compared with other techniques available in the literature. Section "Test example-3: spatial 25-bar truss structure (transmission tower) [38, 42, 52, 58] discusses theoretical analysis and comparison of the results with other contemporary techniques. In Sect. "Test example-4: planer 38-bar truss structure [43, 68]", the graphical representation of variation in constraint violations, penalty parameter, penalty function and pseudo-objective function along with theoretical discussion on comparison of results is provided. The Wilcoxon's rank sum test analysis is presented in Sect. "Test example-5: planer 45-bar truss structure [2, 41]" to check the significance of solutions of the CI-SAPF-CBO over CI-SAPF. This test is conducted based on function values, function evaluation and CPU time. Finally, in Sect. "Result analysis and discussion", the applications of the proposed CI–SAPF and CI–SAPF–CBO are presented by solving multi-pass turning and milling process problems. Section "Applications" discusses the conclusions and future recommendations.

Self-adaptive penalty function (SAPF)

In general, the constrained optimization problem is expressed as follows:

$$\text{Minimize} f(\boldsymbol{X}) = f(x_1, x_2, x_3, \ldots, x_N) \tag{2.1}$$

Subject to

$$g_i(\boldsymbol{X}) \leq 0, \quad i = 1, 2, \ldots, n$$

$$h_i(\boldsymbol{X}) = 0, \quad i = 1, 2, \ldots, m$$

$$\Psi^{\text{lower}} \leq (\boldsymbol{X}) \leq \Psi^{\text{upper}}$$

A Static Penalty Function (SPF) constraint handling approach is widely used. It is expressed as follows:

$$PF = \theta \times \left(\sum_{i=1}^{n} g_i(\boldsymbol{X}) + \sum_{i=1}^{m} h_i(\boldsymbol{X}) \right), \tag{2.2}$$

where θ is a penalty parameter and $\left(\sum_{i=1}^{n} g_i(\boldsymbol{X}) + \sum_{i=1}^{m} h_i(\boldsymbol{X})\right)$ is summation of the violated constraints. However, significant number of preliminary trials are required to choose suitable value of θ. It is the major disadvantage of the SPF approach. To overcome this limitation, a Self-Adaptive Penalty Function (SAPF) approach is proposed. In the SAPF approach, the objective function $f(\boldsymbol{X})$ is itself utilized as a penalty parameter. It is expressed as follows:

$$SAPF = (\boldsymbol{X}) \times \left(\sum_{i=1}^{n} g_i(\boldsymbol{X}) + \sum_{i=1}^{m} h_i(\boldsymbol{X}) \right). \tag{2.3}$$

It further forms the pseudo-objective function $\phi(\boldsymbol{X})$ as follows:

$$\phi(\boldsymbol{X}) = f(\boldsymbol{X}) + \text{SAPF}. \tag{2.4}$$

It is important to note that when the objective function value of the problem is too small, the SAPF approach may not give best feasible solution. In such cases, an arbitrary integer (fixed) value (int) is added in the function value $f(\boldsymbol{X})$. Then, the SAPF would be calculated as follows:

$$SAPF = f(\boldsymbol{X}) + \int \times \left(\sum_{i=1}^{n} g_i(\boldsymbol{X}) + \sum_{i=1}^{m} h_i(\boldsymbol{X}) \right). \tag{2.5}$$

Cohort intelligence (CI) algorithm

The CI algorithm [46] is motivated from the social tendencies of learning candidates of a cohort. Every candidate in the cohort iteratively attempts to achieve a goal which is common to all. For this, every candidate employs roulette wheel approach and selects another candidate to follow which may result in the improvement of its own behavior. This makes every candidate learn from one another and helps the overall cohort behavior to evolve. The cohort behavior could be considered saturated, if for considerable number of learning attempts the behavior of every candidate does not improve considerably and becomes almost same. The characteristics of CI algorithm are as follows:

1. It models the learning mechanism of cohort candidates. Every candidate has inherently common goal to achieve the best behavior by improving its qualities. The interaction and competition are the two natural instincts of every cohort individual. These are achieved through roulette wheel selection and further sampling in the close neighborhood of the selected (being followed) candidate. For details refer to Kulkarni et al. [46, 49].
2. Every candidate observes itself and every other candidate in the cohort to improve its individual behavior and associated qualities.
3. In CI algorithm, at the end of every learning attempt, every candidate independently updates its search space.
4. The problem with large number of variables and constraints can be efficiently handled Kulkarni et al. [35, 50].

CI-SAPF

Consider a cohort with number of candidates C. For every individual candidate $c(c = 1,2,\ldots,C)$ the pseudo-objective function (behavior) using the CI-SAPF approach (refer Eq. 2.3) can be expressed as follows:

$$\phi\left(\boldsymbol{X}^{\boldsymbol{c}}\right) = f\left(\boldsymbol{X}^{\boldsymbol{c}}\right) + \mathrm{SAPF}\left(\boldsymbol{X}^{\boldsymbol{c}}\right), \tag{3.1}$$

where $SAPF\left(\boldsymbol{X}^{\boldsymbol{c}}\right) = f\left(\boldsymbol{X}^{\boldsymbol{c}}\right) \times \left(\sum_{i=1}^{n} g_i(\boldsymbol{X}^{c}) + \sum_{i=1}^{m} h_i \boldsymbol{c}\right)$ is the penalty function and $f\left(\boldsymbol{X}^{\boldsymbol{c}}\right)$ is the objective function of individual candidate. As the algorithm progresses, every candidate narrows down the sampling space using a sampling space reduction factor R. An independent penalty parameter $f(X^{c})$ is generated by every individual candidate c to penalize its associated behavior and subsequently updates the penalty parameter for every learning attempt of the algorithm. The pseudo code of the CI-SAPF is presented in Fig. 1. The performance of CI is dependent on the parameters such as number of candidates C and sampling space reduction factor R. As cohort is a group of learning candidates, it is necessary to decide the number of candidates C. In a cohort with fewer number of candidates (for example: 2, 3, 4), the number of choices to follow for an individual are also less. On the other hand, as the number of candidates increases (for example: 7, 8, 9 and above), the number of behavior choices also increases which certainly helps to improve the quality of the function value,however, it significantly increases the computational cost (function evaluations and CPU time). Based on the statistical analysis in Kale and Kulkarni [35], the number candidates considered here are 5, as there is no significant improvement observed in function value with more number of candidates. Another, limitation is the selection of sampling space reduction factor R. At the end of every learning attempt (iteration), every candidate updates its individual search space using the sampling space reduction factor R. The choice of R is decided based on preliminary trials. To overcome this limitation, CI-SAPF is hybridized with CBO. It is discussed in the next section.

Metaheuristic CI–SAPF–CBO

This section describes the proposed hybrid CI–SAPF–CBO algorithm. In CI–SAPF–CBO, eminent properties of CI and CBO algorithms are incorporated to enhance the applicability of algorithm towards variety of problems from different domains. The mathematical representation of CI–SAPF–CBO is presented along with its flowchart. The detailed review and characteristics of CBO algorithm are discussed.

Colliding bodies optimization (CBO)

The CBO algorithm is proposed by Kaveh and Mahdavi [38]. It is motivated from the physical behavior of colliding bodies (objects). It obeys the law of conservation of momentum and energy in which, the momentum of all the objects before collision is equal to the momentum of all the objects after collision. After the collision, two moving bodies having masses and velocities are separated with updated velocities. This causes to move an object towards better position in the search space. The pseudo code of the CBO is presented in Fig. 2. The characteristics of CBO algorithm are as follows:

1. CBO algorithm is governed by the physics law of conservation of momentum and energy. The momentum of all the objects before collision is equal to the momentum of all the objects after collision.
2. The colliding bodies are arranged in such a way that the moving objects are keen to improve. Moving objects motivate the stationary objects to explore the search space and push them towards better solution.

C Number of candidates $(c = 1, \ldots, C)$
$\boldsymbol{\Psi}$ Sampling space
R Sampling space reduction factor
P Penalty function
g Violated constraint value

Initialize $C, \boldsymbol{\Psi}, R$
While
1. Initialize the set of design variables $(\mathbf{X})$ using uniform distribution method within the original sampling space $\boldsymbol{\Psi}$.
2. Evaluate individual behavior/objective function $f(\mathbf{X}^c)$
3. Apply SAPF approach to generate a pseudo behavior/objective function:
   $\phi(\mathbf{X}^c) = f(\mathbf{X}^c) + P(\mathbf{X}^c)$
   where $P(\mathbf{X}^c) = f(\mathbf{X}^c) \times \sum_{c=1}^{C} g\,(\mathbf{X}^c)$
4. The probability p^c associated with every candidate c in the cohort is calculated as: $p^c = \frac{1/\phi^*(\mathbf{X}^c)}{\sum_{c=1}^{C} 1/\phi^*(\mathbf{X}^c)}$
5. Using roulette wheel approach every candidate c selects behavior to follow from within C available choices.
6. Every candidate c shrinks the sampling interval $\boldsymbol{\Psi}^c$ in its neighborhood using sampling space reduction parameter R and set of solutions $\mathbf{X}^c$:
   $\boldsymbol{\Psi}^c = \left[\boldsymbol{\Psi}^{c,lower}, \boldsymbol{\Psi}^{c,upper}\right] = \left[\mathbf{X}^c - \left\|\frac{\boldsymbol{\Psi}^{upper} - \boldsymbol{\Psi}^{lower}}{2}\right\| \times R,\ \mathbf{X}^c + \left\|\frac{\boldsymbol{\Psi}^{upper} - \boldsymbol{\Psi}^{lower}}{2}\right\| \times R\right]$
   Further, every candidate c selects the variable values from the updated sampling interval $\boldsymbol{\Psi}^c$.
7. **If**: No significant improvement in the behavior $\phi^*(\mathbf{X}^c)$ is observed the solution is considered to be saturated
   Every candidate c expands the sampling interval $\boldsymbol{\Psi}^c$ to its original interval $\boldsymbol{\Psi}$
   Accept the current behavior of cohort $\phi(\mathbf{X})$ and the associated attributes $\mathbf{X}$.
   Else
   Continue to Step 2
   End If

End While

Fig. 1 Pseudo Code of CI-SAPF

3. The colliding bodies are independent of computational parameters. Therefore, the preliminary process of parameter tuning is not required.
4. The Coefficient of Restitution (COR) used to keep the balance between global and local minima.

The CBO algorithm is successfully validated on certain continuous, discrete and mixed variable truss structure and design engineering domain optimization problems [38]. It is observed that the CBO is sensitive to the number of objects and needed more colliding bodies to maintain better convergence and higher level of exploration. Furthermore, [38] modified the CBO to Enhanced CBO (ECBO) to incorporate the colliding memory to save the so far best solutions; so that, it could be used for further operations by replacing current worst solutions. This helps the ECBO algorithm to enhance and provide faster convergence within less computational cost. In ECBO, two parameters are introduced randomly between [0, 1] which help the solution jump out of the local minima/maxima. The first parameter represents the change in component of each colliding body which is then compared with a second uniformly distribute random number. This further decides the modification in the positions of the colliding bodies. The hybrid CBO-PSO [38] is proposed to exploit the ability of CBO by incorporating basic features of PSO. The CBO-PSO is successfully validated by solving continuous variable truss structure problems with dynamic constraints incorporated with SPF approach.

Framework of CI–SAPF–CBO

It is necessary to generalize the problem-solving technique to explore the applicability of diversified real-world applications. The algorithm of CI has already been validated by solving large group of problems; however, the algorithm required certain preliminary trials to set a sampling space reduction factor R to avoid the solution to trap into the local minima [35]. To overcome this limitation of the CI algorithm, an important characteristic of CBO is incorporated into CI.

Initialize number of CBs

For $iter$ to $iter_{max}$

1. Evaluate the objective function $F(\boldsymbol{X}^c)$ of new position $(\mathbf{X}^c)$ and arrange these in a descending order.
2. Evaluate the mass (probability) of $m^c = \frac{1/F(\boldsymbol{X}^c)}{\sum_{c=1}^{C} 1/F(\boldsymbol{X}^c)}, c = 1,2,\ldots,C$
3. Evaluate the coefficient of restitution $(\varepsilon) = \frac{iter}{iter_{max}}$
4. First half of the CBs are referred as stationary bodies and second half of CBs are referred as moving bodies.
5. Set the velocity of the stationary bodies before collision $m^c = 0, \quad c = 1,2,\ldots,C/2$
6. Evaluate velocity of moving bodies before collision $m^c = \boldsymbol{X}^c - \boldsymbol{X}^{c-C/2}, \quad c = C/2 + 1, C/2 + 2, \ldots, C$
7. Evaluate velocity of movies bodies after collision
$$m'^c = \frac{\left(p^c - \varepsilon p^{c-C/2}\right) m^c}{p^c + p^{c-C/2}}, \qquad c = C/2 + 1, C/2 + 2, \ldots, C$$
8. Evaluate velocity of stationary bodies after collision
$$m'^c = \frac{\left(p^{c+C/2} + \varepsilon p^{c+C/2}\right) m^{c+C/2}}{p^c + p^{c+C/2}}, \qquad c = 1,2,\ldots,C/2$$
9. Evaluate the position of stationary bodies $\boldsymbol{X}^{c_{new}} = \boldsymbol{X}^c + rand.m'^c, \quad i = 1,2,\ldots,C/2$
10. Evaluate the position of stationary bodies
$\boldsymbol{X}^{c_{new}} = \boldsymbol{X}^{c-C/2} + rand.m'^c, \quad c = C/2 + 1, C/2 + 2, \ldots, C$

End

Fig. 2 Pseudo Code of the CBO

The CI–SAPF–CBO algorithm (refer Fig. 3 employs CI for global search, SAPF for constraint handling and CBO for local search. The natural tendency of CI candidates is to follow candidates chosen probabilistically using roulette wheel approach to evolve their individual behavior. Furthermore, the learning ability of CI candidates is refined (updated) using CBO. The CI-SAPF-CBO is mathematically expressed as follows:

Step 1: Consider a cohort with C number of candidates; every individual candidate $c(c = 1, 2, \ldots, C)$ belongs a set of attributes/variables $(\boldsymbol{X})^c = (x_1^c, x_2^c, \ldots, x_N^c)$ which makes the behaviour of an individual candidate $f(X^c)$. The initial solution is randomly generated as follows:

$$(\boldsymbol{X})^c = \Psi^{\text{lower}} + \left(\Psi^{\text{upper}} - \Psi^{\text{lower}}\right) \times \text{rand}(1, N). \tag{4.1}$$

The round-off integer sampling approach is employed to generate the integer value and further it helps to select the discrete variable from the predefined set.

Step 2: The SAPF approach is incorporated to handle the constraints and obtained pseudo objective function $\phi(\boldsymbol{X}^c)$ (refer Eq. 3.1).

Step 3: The probability of selecting behavior $\phi(\boldsymbol{X}^c)$ of every associated candidate $c(c = 1, 2, \ldots, C)$ is evaluated as follows:

$$p^c = \frac{1/\phi(\boldsymbol{X}^c)}{\sum_{c=1}^{C} 1/\phi(\boldsymbol{X}^c)}. \tag{4.2}$$

Step 4: Every individual candidate $c(c = 1, 2, \ldots, C)$ generates a random number $r\epsilon[0,1]$ and using roulette wheel approach decides to follow the corresponding behaviour $\phi(\boldsymbol{X}^c)$ and associated attributes $\mathbf{X}^c$. The behavior is selected by candidate c and not known in advance. The roulette wheel approach provides chance to every behavior in the cohort to get selected purely based on its quality. In addition, it also may increase the chances of any candidate to select the better behavior as the associated probability $p^c, c(c = 1, 2, \ldots, C)$ (refer Eq. 4.1) in the interval [0, 1] is

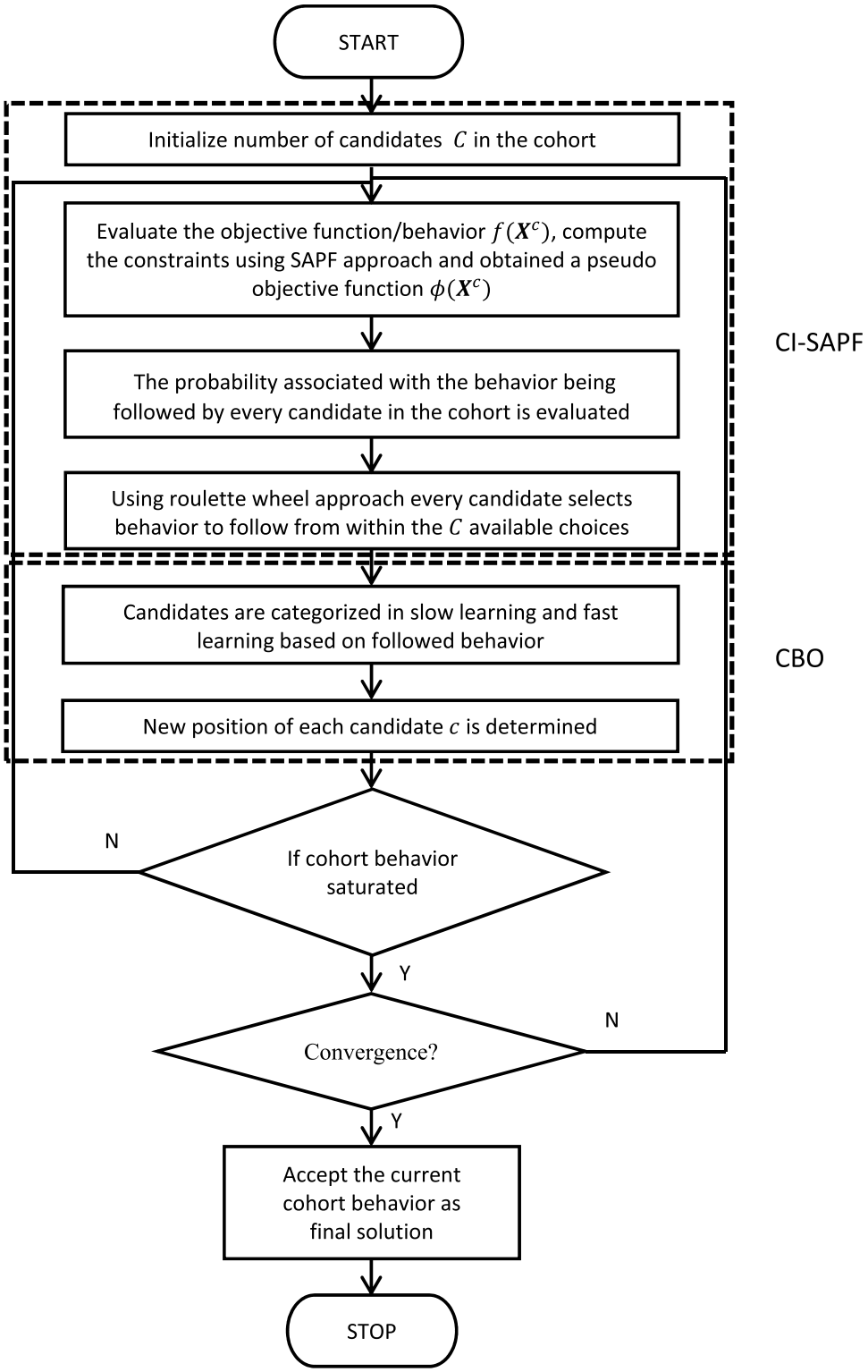

Fig. 3 CI-SAPF-CBO Flowchart

directly proportional to the quality of the behavior $\phi(\boldsymbol{X}^c)$. In other words, better the solution, higher is the probability of being followed by the candidates in the cohort.

Step 5: After following a suitable candidate's behavior in the cohort, all the candidates are arranged in descending order of the fitness value. In the context of CBO, the first half of the candidates (having greater fitness value) are referred to as stationary bodies and other half of the candidates (having lesser fitness value towards minimization) are referred to as moving bodies. For the sake of hybridization

of CI–SAPF–CBO, stationary bodies are considered as slow-learning candidates and moving bodies are considered as fast-learning candidates. The fast-learning candidates motivate slow-learning candidates to improve their learning ability in the cohort. The learning ability of every candidate is identified by determining the initial and final velocities of the colliding bodies (C candidates) after collision Kaveh and Mahadavi [38]. The learning ability of each candidate refers to the velocity m of colliding bodies.

The initial learning ability (initial velocity $m^c = 0$) of slow-learning candidates and fast-learning candidates is represented as follows:

$$m^c = 0, c = 1,2, \ldots, C/2 \tag{4.3}$$

$$m^c = \boldsymbol{X}^c - \boldsymbol{X}^{c-\frac{C}{2}}, c = C/2+1, = C/2+2, \ldots, \tag{4.4}$$

where m^c and $(\boldsymbol{X})^c$ are the learning ability and position of cth candidate, respectively; $(\boldsymbol{X})^{c-C/2}$ is the cth position of $(\boldsymbol{X})$. The final learning ability of candidates are evaluated by utilizing the initial learning ability of the candidates. The learning ability of slow-learning candidates is as follows:

$$m'^c = \frac{\left(p^{c+C/2} + \varepsilon p^{c+C/2}\right) m^{c+C/2}}{p^c + p^{c+C/2}}, c = 1,2, \ldots, = C/2, \tag{4.5}$$

where $m^{c+C}/2$ and m'^c are the initial learning ability of cth fast-learning candidate and final learning ability of cth slow-learning candidate, respectively;p^c is probability of cth candidate, $p^{c+^C/2}$ is the probability of cth fast-learning candidate. Furthermore, the learning ability of fast-learning candidate is represented as follows:

$$m'^c = \frac{\left(p^c - \varepsilon p^{c-C/2}\right) m^c}{p^c + p^{c-=C/2}}, \; c = C/2+1, C/2+2, \ldots, C \tag{4.6}$$

where m'^c is the final learning ability of cth fast-learning candidate, $p^{c-C/2}$ is the probability of cth slow-learning candidate pair and ε is the COR. It is introduced to evaluate the initial and final learning abilities of each of the candidates in order to control the exploration of search space and exploitation of the best solution [37]. More specifically, it controls the local and global searches. The index COR (ε) is calculated as follows:

$$\varepsilon = 1 - \frac{\text{iter}}{\text{iter}_{\max}}, \tag{4.7}$$

where m^c is initial learning ability of the candidates, m'^c is final learning ability of the candidates, iter and $\text{iter}_{\max}$ are the current iteration number and total number of iterations, respectively. In addition, the final learning abilities of fast-learning candidates is used to obtain the new position of candidates in the search space which fulfills the objective of removing the sampling space reduction factor R from CI-SAPF algorithm.

Step 6: The new position of attributes for every candidate in the search space are updated as follows:

$$(\boldsymbol{X})^{c_{\text{new}}} = (\boldsymbol{X})^c + \text{rand}.m'^c, c = 1,2, \ldots, = C/2 \tag{4.8}$$

$$(\boldsymbol{X})^{c_{\text{new}}} = (\boldsymbol{X})^{c-C/2} + \text{rand}.m'^c, \; c = C/2+1, = C/2+2, \ldots, C \tag{4.9}$$

where $(\boldsymbol{X})^{c_{new}}$, $(\boldsymbol{X})^c$ and m'^c are the new position of the attributes, previous position of attributes and final learning ability of fast-learning candidate, respectively. rand is the random vector uniformly distributed in the range [− 1,1]. The learning attempt is repeated from Step 2 until the termination criteria (number of iterations) is satisfied.

For the validation of proposed CI–SAPF and CI–SAPF–CBO techniques, the problems considered here are from design engineering domain, truss structure domain and linear and non-linear test problems. The CI–SAPF and CI–SAPF–CBO are coded in MATLAB 7.7.0 (R2013b) and the simulations are run on Windows platform using an Intel(R) Core (TM)2Duo, 2.93 GHz processor speed and 4 GB RAM. Furthermore, every individual problem is solved 30 times. The solutions obtained from proposed techniques and comparison with other contemporary algorithms are discussed in the following sections.

Test examples

The CI–SAPF and CI–SAPF–CBO algorithms are applied to solve 7 discrete variable truss structure problems, 11 mixed variable design engineering problems and 17 discrete variable linear and nonlinear test functions. To handle the discrete variables, a round off integer sampling approach [35] is employed. Also, the linear and nonlinear constraints involved with these problems are handled by proposed SAPF approach. To ensure the performance of the CBO algorithm, it is incorporated with static penalty function approach applied to solve all the other problems considered in the current work. In the previous studies, CBO is applied to solve 52-bar, 72-bar case 1 truss structure problems, pressure vessel and welded beam case 1 problem. In the current work, the MATLAB code for CBO is adopted form Kaveh and Mahdavi [38].

Truss structure problems

All the truss structure problems aimed to minimize overall weight by satisfying the constraints, such as maximum allowable stress $\sigma_{\max}$ in both tension and compression on

every member and maximum allowable displacement $u_{\max}$ at every node in both horizontal and vertical directions. In these problems, the number of variables is equal to the number of members of the truss. For symmetric truss structure problems, such as 25-bar, 45-bar, 52-bar and 72-bar, the variables are considered in a group as presented in respective comparison tables. For all truss structure problems, the design variable is cross-section area of each truss member and it is selected from within the set of discrete values. All the truss structure problems are successfully solved by CI–SAPF and CI–SAPF–CBO; however, CBO could not achieve the feasible solution for 6-bar, 10-bar cased 1 and 2, 25-bar case 1 and 2, 38-bar, 45-bar and 72-bar case 2. This is due to premature convergence of solution. In general, the problem definition for the truss structure is as follows:

$$\text{Minimize} f = W = \sum_{i=1}^{N} \rho A_i l_i \tag{5.1}$$

$$\text{subject to } \left|\sigma_i\right| \leq \sigma_{\max} i = 1,2,\ldots,N \tag{5.2}$$

$$\left|u_j\right| \leq u_{max} \quad j = 1,2,\ldots,M \tag{5.3}$$

where W Objective function (Weight). A_i Set of cross section area of every truss structure member $i, i = 1,2,\ldots,N$. ρ Weight density of the truss structure material. l_i Length of truss structure member $i, i = 1,2,\ldots,N$.$\sigma_{\max}$ Maximum allowable stress. $u_{\max}$ Maximum allowable displacement.

Test example-1: six-bar truss structure [65]

For six-bar truss structure problem CI-SAPF and CI-SAPF-CBO obtained same results as the GA [65] (refer Table 1. The best, mean and worst solutions obtained from 30 trials using CI-SAPF is 4962.09*lb* and those of CI–SAPF–CBO algorithm are 4962.09*lb*, 4963.86*lb* and 4965.39*lb* with standard deviation 0 (zero) and 1.72, respectively. The average number of function evaluations for both the proposed techniques are 2735 and 2449, respectively. The average computational time are 4.58 s and 3.32 s, respectively. The average function value reported using GA is 5250*lb* whereas, CI–SAPF and CI–SAPF–CBO are 4962.09*lb* and 4963.8558*lb*, respectively. It is noticed that CI-SAPF and CI–SAPF–CBO performed better than GA. The same problem is attempted using CBO,however, the algorithm unable to get the feasible solution due to premature convergence of the solution. The other computational details of CI–SAPF and CI–SAPF–CBO associated with this problem are presented in Tables 34 and 35, respectively.

Table 1 Comparison of results for solving 6-bar truss structure problem

Techniques	GA [65]	CI–SAPF	CI–SAPF–CBO
Truss Weight $W(lb)$	4962.09	4962.09	4962.09
Function evaluations	NA	2250	1740

[NA]Not available

Test example-2: ten-bar truss structure [28, 45, 59, 65, 78]

For case 1 and case 2 of the ten-bar truss structure problem, CI–SAPF and CI–SAPF–CBO algorithm solutions are compared with other contemporary algorithms (refer Tables 2 and 3. The best, mean and worst function values (*Wlb*) obtained for case 1 and case 2 using CI–SAPF and CI–SAPF–CBO with standard deviation and the average CPU time are presented in Table 4. The statistical details along with the parameters used for CI–SAPF associated with these problems are listed in Tables 34 and 35, respectively. For case 1, it is noticed that the solutions obtained using ABC, ADS, PC and CI–SAPF are similar,however, CI–SAPF and CI–SAPF–CBO algorithms performed computationally better than other two approaches, whereas, for solving case 2, PC yielded significantly better objective function value within a large number of function evaluations 2,363,380 and average CPU time 99 s. When compared with the ADS, the solution obtained here for case 1 are similar,however, ADS performance is better in terms of function evaluations. The Search Dimension Ratio (SDR) incorporated in ADS helped to explore and exploit the search space and yielded better solutions with fewer number of function evaluations. The CBO algorithm could not achieve the feasible solution.

Table 2 Comparison of results for case 1 solving 10-bar truss structure problem

Techniques	GA [65]	ABC [78]	ADS [28]	PC Kulkarni et al. [45]	CI-SPF [35]	CI–SAPF	CI–SAPF–CBO
Truss weight $W(lb)$	5499.35	5490.74	5490.74	5490.74	5490.74	5490.60	5490.60
Function evaluations	NA	25,800	1000	1,852,059	19,250	16,940	14,160

[NA]Not Available

Table 3 Comparison of results for case 2 solving 10-bar truss structure problem

Techniques	PSO [59]	PSOPC [59]	HPSO [59]	MBA [73]	PC Kulkarni et al. [45]	CI–SAPF	CI–SAPF–CBO
Truss weight $W(lb)$	5243.71	5133.16	5073.51	5067.33	4686.77	5061.76	5061.76
Function evaluations	NA	NA	NA	NA	2,363,380	9450	8400

[NA]Not Available

Table 4 Statistical results for 10-bar case 1 and 2 using CI–SAPF and CI–SAPF–CBO

Results	CI–SAPF		CI–SAPF–CBO	
	Case 1	Case 2	Case 1	Case 2
Best	5490.60	5061.76	5490.60	5061.76
Mean	5505.75	5062.22	5505.71	5061.81
Worst	5534.96	5067.33	5532.85	5062.12
Std. Dev	16.65	1.54	14.09	0.13
Avg. CPU time (sec)	10.64	12.70	10.80	33.42
Avg. function evaluations	22,654	22,874	16,845	24,325

Table 7 Statistical results for 25-bar case 1 and 2 using CI–SAPF and CI–SAPF–CBO

Results	CI–SAPF		CI–SAPF–CBO	
	Case 1	Case 2	Case 1	Case 2
Best	512.81	470.14	512.81	473.47
Mean	530.87	486.79	531.95	485.97
Worst	548.61	498.06	543.07	487.94
Std. Dev	10.16	6.80	8.80	2.82
Avg. CPU Time (*sec*)	12.21	15.95	11.71	22.18
Avg. Function Evaluations	22,588	26,796	21,652	28,513

Test example-3: spatial 25-bar truss structure (transmission tower) [39, 44, 55, 59]

For solving case 1 and 2 of 25-bar truss structure problem, CI–SAPF and CI–SAPF–CBO solutions are compared with other contemporary algorithms (refer Tables 5 and 6. The best, mean and worst function values with standard deviation, average function evaluations and average CPU time obtained from 30 independent trials are presented in Table 7. It is observed that both the techniques performed better as compared to PSO, PSOPC, HPSO and DHPSACO algorithm. The algorithm of PC obtained better solution as compared to CI–SAPF and CI–SAPF–CBO,however, the computational cost is significantly higher due to slower convergence. The other statistical details of CI–SAPF and CI–SAPF–CBO are presented in Tables 34 and 35.

Test example-4: planer 38-bar truss structure [45, 72]

For solving 38-bar truss structure problem, the best, mean and worst function values (Wlb) obtained using CI-SAPF are 5891.05lb, 5895.37lb and 5898.44lb, respectively, with standard deviation 2.19, average function evaluations

Table 5 Comparison of results solving 25-bar case 1 truss structure problem

Techniques	HS [55]	DHPSACO [39]	PSO [59]	PSOPC [59]	HPSO [59]	PC Kulkarni et al. [44]	CI-SPF [35]	CI–SAPF	CI–SAPF–CBO
Truss weight $W(lb)$	560.59	551.61	566.44	560.59	560.59	477.17	516.20	512.81	512.81
Function evaluations	NA	NA	NA	NA	NA	1,844,457	21,350	18,500	15,000

[NA]Not Available

Table 6 Comparison of results solving 25-bar case 2 truss structure problem

Techniques	PSO [59]	PSOPC [59]	HPSO [59]	DHPSACO [39]	PC Kulkarni et al. [44]	CI-SPF [35]	CI–SAPF	CI–SAPF–CBO
Truss weight $W(lb)$	567.49	556.9	551.14	551.14	464.15	500.04	470.14	473.47
Function evaluations	NA	NA	NA	NA	1,963,415	14,350	31,500	15,000

[NA]Not Available

69920 average and CPU time 87.8s. The best, mean and worst function values resulted using CI–SAPF–CBO are 5889.95 *lb*, 5903.25 *lb* and 5927.08 *lb*, respectively, with the standard deviation of 10.93, average function evaluations of 46900 and average CPU time of 33.30s. The results obtained using CI–SAPF and CI–SAPF–CBO algorithms are compared with other contemporary approaches (refer Table 8 and the associated parameters are listed in Tables 34 and 35. The solutions reported by both the algorithms are better as compared to PC. However, PC exhibited more robust performance with standard deviation of 0.37.

Test example-5: planer 45-bar truss structure [2, 43]

For 45-bar truss structure problem the results obtained using CI-SAPF and CI-SAPF-CBO algorithms along with the other approaches is presented in Table 9. Using CI-SAPF the best, mean and worst function values (*Wlb*) obtained from 30 trials are14322.29*lb*, 14476.49*lband*14667.10*lb*, respectively, with standard deviation91.77, average function evaluations 114525 and average CPU time64.69s. Also, the best, mean and worst function values obtained using CI-SAPF-CBO are14322.29, 14413.89 and14480.44, respectively, with standard deviation of40.18, average function evolutions of 103237 and average CPU time of 56.94 s. The other computational details and parameters associated with the both algorithms are listed in Tables 34 and 35, respectively. The CI–SAPF and CI–SAPF–CBO algorithm obtained better results as compared to other algorithms,however, CI–SAPF–CBO yielded better solutions with less computational cost (average function evolutions and average CPU time).

Test example-6: spatial 52-bar truss structure [38, 39, 55, 59, 73, 86]

The discrete 52-bar planer truss structure problem for weight minimization is successfully solved using CI-SAPF and CI–SAPF–CBO algorithms and the solutions are compared with other contemporary approaches presented in Table 10. The best, mean and worst CI–SAPF solutions obtained from 30 trials are 1894.48 kg, 1913.98 kg and 1934.94 kg, respectively, with standard deviation 11.99, average number of function evaluations 101751 and average CPU time 52.73 s. Also, for CI–SAPF–CBO, the best, mean and worst function values are 1891.44 kg, 1909.33 kg and 1920.11 kg, respectively, with standard deviation 7.18, average function evaluations 95645 and the average CPU time 61.16 *s*. The other associated details are listed in Tables 34 and 35. The reported solution using CI–SAPF is better as compare to other contemporary approaches. However, CBO algorithm exhibited faster convergence as compared to CI–SAPF and CI–SAPF–CBO algorithms due to its exploitation quality.

Test example-7: spatial 72-bar truss structure [39, 44, 55, 59, 86]

The CI–SAPF and CI–SAPF–CBO algorithms are successfully applied for solving case 1 and 2 of the 72-bar truss structure problem and the solutions are compared with

Table 8 Comparison of results solving 38-bar truss structure problem

Techniques	ISCSO (Rudolph and Schmidt, 2012)	PC Kulkarni et al. [45]	CI-SPF [35]	CI–SAPF	CI–SAPF–CBO
Truss weight $W(lb)$	5889.90	5893.02	5900.34	5891.05	5889.95
Function evaluations	4618	1,793,966	106,666	151,240	42,750

Table 9 Comparison of results solving 45-Bar truss structure problem

Techniques	ISCSO [2]	PC [43]	CI-SPF [35]	CI-SAPF	CI–SAPF–CBO
Truss weight $W(lb)$	14,341.21	14,377.92	14,354.27	14,322.29	14,322.29
Function evaluations	NA	4,494,278	163,483	95,625	51,750

[NA]Not Available

Table 10 Comparison of results for 52-bar truss structure problem

Techniques	GA [86]	HS [55]	HPSO [59]	DHPSACO [39]	MBA [73]	CBO [38]	CI–SAPF	CI–SAPF–CBO
Truss weight $W(kg)$	1970.14	1906.76	1905.49	1904.83	1902.61	1899.35	1894.48	1891.44
Function evaluations	60,000	NA	100,000	5300	NA	3840	175,240	42,432

[NA]Not Available

Table 11 Comparison of results solving 72-bar case 1 truss structure problem

Techniques	GA [86]	HPSO [59]	PC Kulkarni et al. [44]	CBO Kaveh and Mahadavi [38]	CI-SPF [35]	CI–SAPF	CI–SAPF–CBO
Truss weight $W(lb)$	400.66	388.9400	372.41	385.54	373.54	372.41	372.41
Function evaluations	NA	NA	8,843,207	5330	99,288	72,000	38,880

[NA]Not Available

Table 12 Comparison of results solving 72-bar case 2 truss structure problem

Techniques	GA [86]	HPSO [59]	DHPSACO [39]	PC Kulkarni et al. [44]	CI-SPF [35]	CI–SAPF	CI–SAPF–CBO
Truss weight $W(lb)$	427.20	933.09	393.38	379.91	381.34	380.18	379.06
Function evaluations	NA	NA	NA	8,730,598	63,000	48,240	60,912

[NA]Not Available

Table 13 Statistical results for 72-bar case 1 and 2 using CI–SAPF and CI–SAPF–CBO

Results	CI–SAPF		CI–SAPF–CBO	
	Case 1	Case 2	Case 1	Case 2
Best	372.41	380.18	372.41	379.06
Mean	381.53	383.24	398.08	387.81
Worst	384.63	385.96	418.90	397.39
Std. Dev	3.25	1.69	11.89	5.43
Avg. CPU time (sec)	78.61	118.52	53.88	80.39
Avg. Function evaluations	85,500	124,080	85,872	91,872

other algorithms (refer Tables 11 and 12. The CI–SAPF and CI–SAPF–CBO algorithm yielded a better solution with less computational time and function evaluations. The best, mean and worst function values, standard deviation, average function evaluations and average CPU time obtained using 30 trials are presented in Table 13. The associated parameters used to run CI–SAPF are illustrated in Table 34.

Design engineering problems

The proposed CI–SAPF and CI–SAPF–CBO algorithms are also validated by solving 11 problems from design engineering domain including stepped cantilever beam problem (minimization of volume), pressure vessel problem (cost minimization), speed reducer problem (minimization of weight), reinforced concrete beam problem (minimization of cost), welded beam design problem case 1 (minimization of cost), case 2 (minimization of overall fabrication cost), multiple disc clutch brake problem (minimization of mass), helical tension compression spring problem (minimization of volume), I-beam (minimization of vertical deflection), cantilever beam problem (minimization of weight) and compound gear problem (minimization of gear ratio). For the sake of comparison, CBO algorithm is also applied to solve these problems except welded beam case 2 (previously reported by [38]). These problems are consisted of mixed design variables. For the statistical analysis, the CI–SAPF, CI–SAPF–CBO and CBO are run 30 times for every problem.

Test example-8: stepped cantilever beam design problem [23, 35, 83]

The stepped cantilever beam design problem for volume minimization is proposed by Thanedar and Vanderplaats [83]. The design variables are discrete $(b_1, h_1, b_2, h_2, b_3, h_3)$ and continuous (b_4, h_4, b_5, h_5). This problem is initially solved using branch and bound and simulated annealing approach. The solution reported is 64558 [83]. The CI–SAPF, CI–SAPF–CBO and CBO are also applied to solve this problem and the results are compared with other algorithms (refer Table 14. The best, mean and worst function values along with the standard deviation, average function evaluations and average CPU time obtained from applied techniques are presented in Table 15. It is observed that CI–SAPF and CI–SAPF–CBO reported better solutions as compared to RNES and similar results as compare to FA and CI-SPF. The associated statistical results for CI–SAPF and CI–SAPF–CBO are presented in Tables 34 and 35.

Test problem-9: pressure vessel design problem [10, 40, 74]

The pressure vessel design problem for cost minimization is successfully solved using CI–SAPF and CI–SAPF–CBO algorithm. The CBO algorithm could not achieve feasible

Table 14 Comparison of results solving stepped cantilever beam design problem

Design variables	RNES [6]	FA [23]	CI-SPF [35]	CBO	CI–SAPF	CI–SAPF–CBO
Volume (cm^3)	64,269.59	63,893.52	63,893.49	80,329.37	63,893.45	63,893.47
Function Evaluations	NA	NA	19,740	4560	10,000	12,000

[NA]Not Available

Table 15 Statistical results for stepped cantilever beam problem using CI–SAPF, CI–SAPF–CBO and CBO

Results	CI–SAPF	CI–SAPF–CBO	CBO
Best	63,839.45	63,893.47	80,329.37
Mean	64,248.96	64,229.23	85,830.51
Worst	64,333.45	64,579.43	101,024.23
Std. Dev	175.29	207.14	6610.61
Avg. CPU time (sec)	5.36	7.45	1.14
Avg. function evaluations	21,552	30,437	4448

solution due to premature convergence. The results obtained using CI–SAPF and CI–SAPF–CBO algorithms compared with other contemporary algorithms are presented in Table 16. For CI-SAPF, the best, mean and worst function values obtained from 30 trials are 6051.48, 6065.25 and 6103.27 with standard deviation 11.91, average function evaluation 11974 average and CPU time 4.43 s. Also, for CI–SAPF–CBO, the best, mean and worst solution are 6059.72, 6066.11 and 6090.53 with standard deviation 9.22, average average function evaluation of 12, 113 and CPU time 4.29 s. From the statistical results, it is noticed that CI–SAPF obtained better solution as compared to other approaches with less computational cost. CI–SAPF–CBO obtained similar results as compared to LCA and OIO. It is due to the faster convergence rate of the CBO algorithm.

Table 18 Statistical results for speed reducer design problem using CI–SAPF, CI–SAPF–CBO and CBO

Results	CI–SAPF	CI–SAPF–CBO	CBO
Best	2817.09	2816.34	2840.44
Mean	2820.32	2817.65	3534.00
Worst	2825.87	2819.57	3582.75
Std. Dev	2.06	1.03	264.39
Avg. CPU time (*sec*)	2.25	1.02	0.30
Avg. function evaluations	14,420	7597	613

Test example-10: speed reducer design problem [3, 9, 20]

For the speed reducer design problem, the results obtained using CI-SAPF, CI-SAPF-CBO and CBO algorithms are presented in Table 17 in comparison with other contemporary approaches. The proposed algorithms yielded better results with less computational cost and function evaluations. The best, mean and worst function values with standard deviation, average computational time and average number of function evaluations are presented in Table 18. The other associated parameters are listed in Tables 34 and 35. The CI–SAPF–CBO algorithm obtained better solutions so far with very less computational cost (refer Table 17).

Table 16 Comparison of results for pressure vessel design problem

Techniques	NIDPM [74]	Augmented Lagrange [40]	GA [10]	CPSO [29]	LCA [36]	OIO [37]	CI–SAPF	CI–SAPF–CBO
Cost	7981.57	7198.04	6288.74	6061.08	6059.85	6059.71	6051.48	6059.72
Function evaluations	NA	NA	900,000	200,000	24,000	50,000	9744	9184

[NA]Not Available

Table 17 Comparison of results solving Speed Reducer Problem for weight optimization

Techniques	AIS-GA-C [3]	AIS-GA [9]	EA [20]	PC Kulkarni et al. [45]	CI-SPF [35]	CBO	CI–SAPF	CI–SAPF–CBO
Weight $W(lb)$	2994.47	2994.34	3025.01	2828.59	2817.56	2840.44	2817.09	2816.79
Function evaluations	36,000	150,000	36,000	1,132,700	16,513	1848	5145	8442

[NA]Not Available

Table 19 Comparison of results solving reinforced concrete beam design problem

Techniques	FA [23]	PC Kulkarni et al. [45]	CI-SPF [35]	CBO	CI–SAPF	CI–SAPF–CBO
Min. Cost	359.21	359.21	359.21	362.63	359.21	359.21
Function evaluations	30,000	563,490	22,050	252	4515	1710

[NA]Not Available

Table 20 Statistical results for reinforced concrete beam problem using CI–SAPF, CI–SAPF–CBO and CBO

Results	CI–SAPF	CI–SAPF–CBO	CBO
Best	359.21	359.21	362.63
Mean	359.24	359.32	379.02
Worst	359.47	359.95	386.40
Std. Dev	0.08	0.18	9.35
Avg. CPU time (sec)	0.73	0.47	0.03
Avg. function evaluations	3047	2255	182

Test example-11: reinforced concrete beam design [23, 35]

For reinforced concrete beam problem, CI–SAPF and CI-SAPF-CBO algorithm reported similar results with less computational cost as compared to FA, PC and CI-SPF (refer Table 19. The best, mean and worst function values with standard deviation, average function evaluation and average CPU time obtained from 30 trials are presented in Table 20.

Test problem-12: welded beam design case 1 [10, 11, 15, 29]

Test problem-13: welded beam design problem case 2 [14, 18, 41] For welded beam design problem (case 1 and case 2), the solutions obtained from CI–SAPF and CI–SAPF–CBO algorithms are compared with other contemporary techniques (refer to Tables 21 and 22. For case 1, the CBO solutions are taken from Kaveh and Mahdavi [38] and in the current work CBO is tested on case 2. For both the cases of welded beam problem, CI-SAPF and CI–SAPF–CBO performed significantly better. This is due to the probability-based roulette wheel approach which increased the chances to follow the better solution in the cohort. The statistical results for case 1 and case 2 obtained form 30 trials are presented in Table 23.

Test problem-14: multiple disc clutch brake [19, 70] The multiple disc clutch brake design problem is previously solved using NSGA [19], PSO and AIA [70]. The CI–SAPF and CI–SAPF–CBO algorithm are successfully validated by comparing the solutions with these algorithms (refer Table 24. The best, mean and worst function values obtained using CI–SAPF and CI–SAPF–CBO algorithm are very similar,however, significantly better as compared to other algorithms. The statistical results from 30 trials are presented in Table 25. The solutions obtained from CBO are marginally worse.

Table 21 Comparison of results solving welded beam design case 1 problem

Techniques	GA [15]	GA [10]	GA [11]	PSO [29]	CBO [38]	CI–SAPF	CI–SAPF–CBO
Min. Cost	2.43	1.75	1.73	1.73	1.72	1.55	1.55
Function evaluations	NA	NA	NA	NA	NA	11,786	9552

[NA]Not available

Table 22 Optimal solutions for the welding beam design case 2 problem

Techniques	GA [18]	PSO [40]	PSO [14]	CBO	CI–SAPF	CI–SAPF–CBO
Min. cost	1.94 (infeasible)	2.03	1.95	4.98	1.65	1.65
Function evaluations	NA	189,800	NA	1152	3000	7092

[NA]Not available

Table 23 Statistical results for welded beam problem case 1 and case 2 using CI–SAPF, CI–SAPF–CBO and CBO

Results	Case1		Case2		
	CI–SAPF	CI–SAPF–CBO	CI–SAPF	CI–SAPF –CBO	CBO
Best	1.55	1.55	1.65	1.65	4.98
Mean	1.56	1.55	1.88	1.93	7.36
Worst	1.57	1.56	2.08	2.15	10.49
Std. Dev	0.01	0.003	0.2013	0.2138	2.14
Avg. CPU time (sec)	1.57	2.24	2.08	2.15	0.27
Avg. function evaluations	11,786	8721	8740	6996	684

Table 24 Comparison of results solving multiple disc clutch brake design problem

Techniques	NSGA [19]	Rao et al. [70]		CBO	CI–SAPF	CI–SAPF–CBO
		PSO	AIA			
Mass (kg)	0.41	0.31	0.32	0.28	0.24	0.24
Function evaluations	NA	NA	NA	210	825	450

[NA]Not available

Table 25 Statistical results for multiple disc clutch brake problem using CI–SAPF, CI–SAPF–CBO and CBO

Results	CI–SAPF	CI–SAPF–CBO	CBO
Best	0.24	0.24	0.27
Mean	0.24	0.24	0.44
Worst	0.24	0.24	0.83
Std. Dev	8.62E-17	9.25E-18	0.23
Avg. CPU time (sec)	0.23	0.18	0.08
Avg. function evaluations	671	800	450

Table 27 Statistical results for tension–compression helical spring problem using CI–SAPF, CI–SAPF–CBO and CBO

Results	CI–SAPF	CI–SAPF–CBO	CBO
Best	2.6586	2.6586	3.2440
Mean	2.6601	2.6594	4.1223
Worst	2.6722	2.6630	5.5331
Std. Dev	0.0032	0.0014	0.8765
Avg. CPU time (sec)	2.05	2.02	0.04
Avg. function evaluations	3030	2991	65

Test problem-15: helical compression spring design [23, 74, 87]

A mixed variable tension–compression helical spring problem is previously solved in Yun [87], Gandomi et al. [23], Sandgren [74] and Kulkarni et al. [45]. The solutions obtained using CI–SAPF and CI–SAPF–CBO algorithms are successfully validated and compared with other contemporary algorithms presented in Table 26. It observed that the results obtained from CI–SAPF are very similar to FA and PC,however, computationally CI-SAPF performed better with lesser number of function evaluations. The best, mean and worst function values with standard deviation, average function evaluations and average CPU time are presented in Table 27. The other statistical details and associated parameters are presented in Tables 34 and 35.

Test problem-16: minimize I-section beam vertical deflection [8, 22] For I-section beam problem, CI–SAPF, CI–SAPF–CBO and CBO reported better function values as compared to CS [22] and SOS [8] (refer Table 28. The best, mean and worst function values with standard deviation, average CPU time and average function evaluations are presented in Table 29. For this problem, the stand-

Table 26 Performance comparison of various algorithms solving helical spring design problem

Techniques	Nonlinear B&B [74]	AHGA [87]	FA (Gandomi et, al. 2011)	PC (Kulakrni et al. (2016a))	CBO	CI–SAPF	CI–SAPF–CBO
Spring volume (in^3)	2.79	2.03	2.66	2.66	3.24	2.66	2.66
Function evaluations	NA	NA	NA	498,567	108	840	1000

[NA]Not available

Table 28 Comparison of results solving minimize I-beam vertical deflection

Techniques	CS [22]	SOS [8]	CBO	CI–SAPF	CI–SAPF–CBO
Min. deflection	0.0130747	0.0130741	82E-4	66E-4	66E-4
Function evaluations	5000	5000	432	3900	2400

[NA]Not available

Table 29 Statistical results for I-section beam problem using CI–SAPF, CI–SAPF–CBO and CBO

Results	CI–SAPF	CI–SAPF–CBO	CBO
Best	66E-4	66E-4	82E-4
Mean	66E-4	66E-4	0.04
Worst	66E-4	66E-4	0.13
Std. Dev	4.65E-07	9.65E-06	0.05
Avg. CPU Time (*sec*)	1.27	0.47	0.016
Avg. Function evaluations	7830	4553	123

Table 30 Comparison of results solving cantilever beam design

Techniques	CS [22]	SOS [8]	CBO	CI–SAPF	CI–SAPF–CBO
Min. weight	1.34	1.34	3.20	1.34	1.34
Function evaluations	NA	15,000	2190	13,750	3025

[NA]Not available

ard deviation obtained by CS and SOS are $1.3E-4$ and $4.0E-5$, respectively,however, CI–SAPF algorithm is more robust with standard deviation of $4.6472E-7$ with fewer number of function evaluations (refer Table 29. The other computational details and associated parameters are presented in Tables 34 and 35.

Test problem-17: cantilever beam [8, 22] The continuous variable cantilever beam design problem aims at the minimization of overall weight. The CI–SAPF approach is successfully validated by solving this problem and compared with other techniques presented in Table 30. The solutions reported from CI–SAPF–CBO and CBO are presented in Table 31. The function values using CI–SAPF and CI–SAPF–CBO for solving cantilever beam problem is very similar to CS [22] and SOS [8] and equally robust. The computational cost of the CI–SAPF is marginally better as compared to CI–SAPF–CBO, SOS and CBO (refer Table 31. The other computational details and associated parameters are presented in Tables 34 and 35.

Table 31 Statistical results for cantilever beam problem using CI–SAPF, CI–SAPF–CBO and CBO

Results	CI–SAPF	CI–SAPF–CBO	CBO
Best	1.34	1.34	3.20
Mean	1.34	1.34	6.28
Worst	1.34	1.34	7.58
Std. Dev	1.74E-07	6.51E-05	1.78
Avg. CPU time (sec)	0.91	1.83	0.06
Avg. function evaluations	9581	11,761	2320

Test problem-18: compound gear train [14, 40, 74] For the compound gear train design problem CI–SAPF, CI–SAPF–CBO and CBO are successfully validated by comparing the solutions with the methods such as nonlinear B&B Sandgren [74], Lagrange multiplier Kannan and Kramer [40] and PSO [14] as presented in Table 32. CI–SAPF and CI–SAPF–CBO algorithms obtained very similar results as PSO. The best, mean and worst solutions obtained using CI–SAPF, CI–SAPF–CBO and CBO with standard deviation, average CPU time and average function evaluations are illustrated in Table 33. The other computational details and associated parameters are presented in Tables 34 and 35.

Linear and nonlinear benchmark test problems

The CI–SAPF, CI–SAPF–CBO and CBO algorithms are also applied for solving several maximization and minimization test problems [82] such as dynamic variable problem, transportation problem, multistage problem, Rosen Suzuki convex test problems, knapsack problem and two cases of integer linear problem. Moreover, two cases of non-convex integer problem and global nonlinear mixed discrete programming problem [89], three-bar test problem [80] and six monotone functions [53] are also solved. These all problems consisted of discrete variables and linear-nonlinear type constraints. The discrete variables are handled using a round off integer sampling technique [35] and constraints are handled using SAPF approach, however, CBO could not achieve the feasible solution for transportation problem and four monotone functions. For all the solved problems, the result comparison is presented in Table 36.

Table 32 Comparison of results solving compound gear design problem

Techniques	Nonlinear B&B [74]	Lagrange Multiplier [40]	PSO [14]	CBO	CI–SAPF	CI–SAPF–CBO
$f(x)$	5.7e − 06	2.1246e-08	2.7e − 12	4.5033E-9	2.70e-12	2.70e-12
Gear ratio	0.1466	0.1441	0.1442	0.1443	0.1442	0.1442
Error %	1.65%	0.11%	0.0011%	0.0462%	0.0011%	0.0011%
Function evaluations	NA	NA	NA	420	1260	360

[NA]Not available

Table 33 Statistical results for compound gear train problem using CI–SAPF, CI–SAPF–CBO and CBO

Results	CI–SAPF	CI–SAPF–CBO	CBO
Best	2.7E−12	2.7E−12	4.50E−09
Mean	3.97E−11	1.73E−11	1.46E−05
Worst	6.60E−10	2.31E−11	7.06E−05
Std. Dev	5.26E−12	9.55E−12	3.13E−05
Avg. CPU time (sec)	3.12	2.17	0.03
Avg. function evaluations	8513	6358	102

Result analysis and discussion

The proposed CI–SAPF and CI–SAPF–CBO algorithms are successfully validated by solving discrete and mixed design variable problems involved with linear and nonlinear constraints. These problems are also solved using CBO algorithm to compare the performance of individual algorithm. The statistical results of all the problems considered in the work are presented in Tables 34, 35, 36. The Tables 34 and 35 present the best, mean and worst function values for CI–SAPF and CI–SAPF–CBO, respectively, including standard deviation of function values, average number of function evaluations, average computational time, closeness to the reported solution and the set of parameters required to run the CI-SAPF algorithm. Table 36 presents the comparison of solutions obtained from CI–SAPF, CI–SAPF–CBO, CBO algorithms and other contemporary techniques used to solve linear and nonlinear optimization test problems. The problems considered here are from truss structure domain (10 problems), mixed variable design engineering domain (11 problems) and linear and non-linear test problems with integer variables (17 problems). In CI–SAPF–CBO, the CBO algorithm enhanced the exploration of the search space which assisted the CI algorithm to reach towards the better solutions within substantially less computational cost. The SAPF approach served to handle the linear and nonlinear constraints and CI worked as a global optimizer. Most importantly, CI–SAPF–CBO came up with a generalized approach, which does not require fine tuning of parameters except number of candidates C. This makes the proposed algorithm easier to apply to wide range of applications.

The problems from truss structure domain and design engineering domain are previously solved using CI–SPF approach incorporated with static penalty function approach [35]. All the discrete variables truss structure domain problems considered in the present work are solved for minimization of weight subject to stress (in links) and deflection (in nodes). For 38 bar, 52 bar and 72 bar case 1, CI–SAPF–CBO obtained even better results as compared to CI–SAPF and CBO algorithms. For solving 52-bar and 72-bar truss structure problems and welded beam (case 1) design engineering problem, the physics inspired CBO algorithm is incorporated with a round off approach to handle the discrete variables and static penalty function approach to handle the inequality constraints. In the current work, CI–SAPF and CI–SAPF–CBO exhibited the superiority in dealing with such problems as compared to the CI–SPF [35], CBO [38] and other contemporary algorithms. The results obtained from CI–SAPF and CI–SAPF–CBO are similar as in both the approaches, a probabilistic roulette wheel approach provided the possible choices to follow the best candidate, wherein, the proposed SAPF approach handled the constraints and CBO is incorporated to refine the solution obtained from CI. The CBO algorithm is tested on all the problems considered here,however, it exhibited fast and premature convergence. However, in CI–SAPF–CBO, a probabilistic roulette approach associated with the CI assisted the algorithm to escape the solution from local minima.

An adaptive penalty function approach is incorporated in GA Nanakorn and Meesomklin [65] with the similar motivation to evade the setting of penalty parameter. For six-bar truss structure problem, both the proposed CI–SAPF and CI–SAPF–CBO algorithms obtained same results as compared to GA and those for ten-bar (case 1) truss structure problem performed better than GA. In Nanakorn and Meesomklin [65], the penalty parameter is set using a ratio between best infeasible fitness value and average feasible fitness value. This process requires a separate scaling factor which needs to be set to adjust the strength of penalty

Table 34 CI–SAPF results solving truss structure and engineering design problems

Test examples	Best mean worst	Standard deviation	Average number of function evaluations	Average computational time (sec)	% improvement over the best reported solution	Set of parameters C, R
Test Example-1 6-Bar	4962.09	9.25E−13	2735	4.58	0	5, 0.965
	4962.09					
	4962.09					
Test Example-2	5490.60	16.65	22,654	17.87	0.0024	5, 0.965
10-Bar Case 1	5505.75					
10-Bar Case 2	5534.96					
	5061.76	1.54	22,874	31.72	0.1099	5, 0.967
	5062.22					
	5067.33					
Test Example-3	512.81	10.16	22,588	12.21	0.6566	5, 0.955
25-Bar Case 1	530.88					
25-Bar Case 2	548.61					
	473.47	6.80	26,796	15.95	5.3139	5, 0.955
	486.74					
	498.06					
Test Example-4 38-Bar	5891.05	2.18	69,920	40.60	0.0019*	5, 0.967
	5895.38					
	5898.44					
Test Example-5 45-Bar	14,322.29	91.77	114,525	63.17	0.1309	5, 0.97
	14,455.68					
	14,667.10					
Test Example-6 52-Bar	1894.48	11.98	101,751	52.73	0.2563	5, 0.9567
	1913.97					
	1934.94					
Test Example-7 72-Bar	372.41	3.25	85,500	78.61	0.3033	5, 0.955
Case 1	381.53					
72-Bar Case 2	384.63					
	380.18	1.69	124,080	118.52	0.0718*	5, 0.955
	383.24					
	385.96					
Test Example-8 Steeped	63,893.45	175.29	21,552	5.36	5.57e-5	5, 0.955
Cantilever Beam	64,248.96					
	64,333.45					
Test Example-9 Pressure	5850.66	104.65	11,396	4.85	0.6663	5, 0.955
vessel	5960.49					
	6116.89					
Test Example-10 Speed	2817.09	2.06	14,420	2.25	0.0165	5, 0.955
Reducer	2820.32					
	2825.87					
Test Example-11 Concrete	359.21	0.08	3047	0.73	0	5, 0.955
Beam	359.24					
	359.47					
Test Example-12 Welded	1.55	77E−4	11,786	4.92	15.7316	5, 0.955
Beam 1	1.56					
	1.57					
Test Example-13 Welded	1.65	0.20	8740	2.08	10.0600	5, 0.955
Beam 2	1.88					
	2.08					

Table 34 (continued)

Test examples	Best mean worst	Standard deviation	Average number of function evaluations	Average computational time (sec)	% improvement over the best reported solution	Set of parameters C, R
Test Example-14 Multiple Disc Clutch	0.24	8.62E−17	671	0.23	25.0316	5, 0.955
	0.24					
	0.24					
Test Example-15 Tension Compression Spring	2.66	32E−4	3030	2.05	0	5, 0.955
	2.66					
	2.67					
Test Example-16 I Section Beam	66.26E-4	4.65E−07	7830	1.27	49.518	5, 0.955
	66.26E-4					
	66.28E-4					
Test Example-17 Cantilever Beam	1.34	1.74E−05	9581	0.91	0	5, 0.955
	1.34					
	1.34					
Test Example-18 Gear Train Design	2.7E-12	5.26E−12	8513	3.12	0	5, 0.955
	3.97E-11					
	6.60E-10					

*The solution obtained using CI–SAPF is worse than other algorithms

parameter. This is one of the limitations of the adaptive penalty function approach which may increase preliminary trials to set the scaling factor; however, in SAPF approach, the function value is used as a penalty parameter which is updated in every iteration of the algorithm. This function value keeps on improving as the iteration progresses which accelerates the convergence rate and obtain competent results with lesser computational cost. For 10-bar (case 2), 25-bar (Case 1 and 2), 52-bar and 72-bar 3-D spatial discrete truss structure, CI-SAPF and CI-SAPF-CBO performed better as compared to PSO and PSOPC. In PSO and PSOPC [59] the penalty function is incorporated which degenerated the function value. To overcome this limitation, fly-back mechanism is used in HPSO to handle the constraints which expedited the convergence rate. In CI–SAPF, all the candidates are keen to improve their behavior (function value) and a probability-based roulette wheel approach provides them possible choices to follow the better solution and CI further push the solution towards global minima. In CI–SAPF–CBO, the exploration quality of CBO controls the search space which drives the candidate to achieve the better behavior in the cohort.

The DHPSACO [39] and MBA [73] are employed with a modified feasibility based [17] constraint handling approach and obtained better solutions so far with least number of function evaluations. A harmony search strategy is adopted in DHPSACO to explore the search space which required an additional parameter referred to as pitch adjustment rate to select the neighborhood values. A SOS (Cheng and Prayogo 2014) is also incorporated with feasibility-based rule for solving design engineering problems such as steeped cantilever beam for weight minimization and design of I-section beam for minimum vertical deflection. A similar feasibility-based approach is also adopted by Datta and Figueira [14] in real-integer-discrete-coded PSO for solving compound gear train design problem. As compared to the feasibility-based rule, CI–SAPF and CI–SAPF–CBO observed to be more superior obtaining the same results for cantilever beam and compound gear train problems. For I-section beam design problem, CI–SAPF and CI–SAPF–CBO obtained better solutions than PSO and SOS. Moreover, an ADS [27] algorithm is also compared with CI–SAPF and CI–SAPF–CBO for 10-bar case 1 truss structure problem. The ADS achieved the best solution in 1000 function evaluations, however, the external penalty function is incorporated for constraint handling, in which an initial penalty coefficient required to be set and further updated by every stagnation escape period. This required several preliminary trials and may increase the computational cost.

A nonlinear integer and discrete programming method [74] is proposed to solve mechanical design engineering problems such as gear train design, tension compression spring and pressure vessel, in which a nonlinear branch and bound approach and exterior penalty function approach are incorporated to handle discrete and integer variables and to handle the constraints, respectively. A concept of augmented Lagrange multiplier [40] is incorporated with Powell's method and Fletcher and Reeves Conjugate Gradient method for solving mechanical design problems and it is noticed that the zeroth-order search (Powell's method) found

Table 35 CI–SAPF–CBO results solving truss structure and engineering design problems

Test examples	Best mean worst	Standard deviation	Average number of function evaluations	Average computational time (sec)	% Improvement over the best reported solution
Test Example-1 6-Bar	4962.09	1.37	2449	3.32	0
	4963.86				
	4965.39				
Test Example-2 10-Bar Case 1 10-Bar Case 2	5490.60	14.09	16,845	10.81	0.0024
	5505.71				
	5532.85				
	5061.76	1.88	24,325	34.53	0.1099
	5062.50				
	5067.33				
Test Example-3 25-Bar Case 1 25-Bar Case 2	512.81	8.80	21,652	11.71	0.6560
	531.95				
	543.07				
	473.47	2.82	28,513	22.18	5.3139
	485.97				
	487.94				
Test Example-4 38-Bar	5889.95	10.93	46,900	33.30	0
	5903.25				
	5927.08				
Test Example-5 45-Bar	14,322.29	40.18	103,237	56.95	0.1319
	14,413.89				
	14,480.44				
Test Example-6 52-Bar	1891.44	7.17	95,645	61.17	s
	1909.33				
	1920.11				
Test Example-7 72-Bar Case 1 72-Bar Case 2	372.41	11.89	85,872	53.89	0.3033
	398.09				
	418.90				
	379.07	5.43	91,872	80.39	0.2213
	387.81				
	397.39				
Test Example-8 Steeped Cantilever Beam	63,893.47	207.14	30,437	7.46	3.14e-5
	64,229.23				
	64,549.43				
Test Example-9 Pressure vessel	5850.65	97.05	11,033	4.69	0.6664
	5946.46				
	6095.05				
Test example-10 Speed Reducer	2816.34	1.03	7597	2.01	0.0273
	2817.65				
	2819.57				
Test example-11 Concrete Beam	359.21	0.18	2255	0.47	0
	359.31				
	359.95				
Test example-12 Welded Beam 1	1.55	35E−4	8721	2.24	15.7316
	1.55				
	1.56				
Test Example-13 Welded Beam 2	1.65	0.21	6996	2.15	10.1600
	1.93				
	2.16				

Table 35 (continued)

Test examples	Best mean worst	Standard deviation	Average number of function evaluations	Average computational time (sec)	% Improvement over the best reported solution
Test example-14 multiple disc clutch	0.24	8.62E−17	800	0.18	25.0136
	0.24				
	0.24				
Test example-15 Tension compression Spring	2.66	14E−4	2991	2.03	0
	2.666				
	2.66				
Test Example-16 I Section Beam	66.26E−4	9.65E−06	4553	0.47	49.5100
	66.26E−4				
	66.26E-4				
Test example-17 steeped beam 2	1.34	6.51E−05	11,761	1.8353	0
	1.34				
	1.34				
Test example-18 gear train design	2.70E−12	9.55E−12	6358	2.18	0
	1.73E−11				
	2.31E−11				

to be more efficient results than first order method. For constraint handling, a dynamic constraint handling approach is incorporated with augmented Lagrange multiplier-based method. In the current work, CI–SAPF and CI–SAPF–CBO reduced the effort of setting up the penalty parameter and also helped the algorithm to work smoothly with different penalty parameter for every learning attempt. This makes the algorithm more superior than other contemporary techniques and CI–SPF.

With the motivation of parameter-less penalty function approach, He and Wang [29] and Coello [10] proposed the constrained handling approach as co-evolutionary Particle Swarm Optimization (CPSO) and self-adaptive penalty function using GA, respectively. For solving pressure vessel problem, the penalty function approach is divided into two with separate incremental weighing factors [29]. From the results, it is noted that the function evaluations for GA [10] and CPSO [29] are 900,000 and for 200,000, respectively. The performance of the CI–SAPF–CBO is observed to be more efficient with quite less number of function evaluations (refer Table 12 with better function value as compared to GA, CPSO and CI–SAPF.

To validate the ability of CI–SAPF and CI–SAPF–CBO, the discrete variables linear problems (transportation problem, knapsack problem and linear integer programming) and nonlinear problems (dynamic problem, multistage problem and Rosen Suzuki convex programming problem) are adopted from Srivastava and Fahim (2001). In Srivastava and Fahim (2001), a two-phase optimization procedure is proposed, in which a gradient-based steepest descent method for feasible solution and hem-stitching approach for infeasible solution (when constraint violated) are distinctly incorporated. For the second phase, the integer combinations vector is obtained from the neighborhood of the first phase. The process adopted in this work could make the algorithm more efficient and driven towards promising results. The CI–SAPF and CI–SAPF–CBO algorithms are also applied for solving a Signomial Discrete Programming (SDP) problem Tsai et al. [89] (non-convex integer programming and global nonlinear mixed discrete programming problems). When compared to Floudas's approach of transformation of objective function and constraints (SDP problem) into convex problem, CI-SAPF and CI–SAPF–CBO are observed to have yielded superior solution.

Analysis of CI-SAPF approach

The proposed CI-SAPF algorithm is successfully tested on different domain problems and the results are discussed in previous section. CI–SAPF algorithm is run for 30 times for solving every problem. The best, mean and worst function values along with the standard deviations and function evaluations including the associated parameters C (number of candidates) and R (sampling space reduction factor) are presented in Table 34. The main advantage of the proposed SAPF approach is that it can be directly applicable to a variety of constrained optimization problems without preliminary trials. The CI–SAPF algorithm is reported to be more efficient which reduced the efforts of trial-and-error process of setting the suitable penalty parameter. In CI–SAPF algorithm, the penalty parameter is selected based on the available set of design variables which updated iteratively Fig. 4d). The

Table 36 CI–SAPF results solving linear and nonlinear problems

Sr. No	Test functions	Solver	Search space		Function Value	Optimum variables
			Lower limit	Upper limit		
1	Dynamic problem (Maximization)	Srivastava and Fahim [82]	[0, 0, 0]	[10, 10, 10]	16	[0, 1, 2]
		CI–SAPF			**16**	**[0, 1, 2]**
		CI–SAPF–CBO			**16**	**[0, 1, 2]**
		CBO			**15**	**[0, 2, 1]**
2	Transportation problem	Srivastava and Fahim [82]	[0,…,0]	[100,…,100]	40.5	[5, 15, 0, 0, 0, 15, 0, 5, 5]
		CI–SAPF			**40.8**	**[2,18, 0, 3, 0,12, 0, 2]**
		CI–SAPF–CBO			**40.8**	**[2,18, 0, 3, 0,12, 0, 2]**
3	Multistage oroblem (Maximization)	Srivastava and Fahim [82]	[0,0,0]	[100,100,100]	55.2	[3, 1, 0]
		CI–SAPF			**55.2**	**[3,1,0]**
		CI–SAPF–CBO			**55.2**	**[3,1,0]**
		CBO			**32.7**	**[0,1,1]**
4	Rosen-Suzuki test problem convex programming problem (Minimization)	Srivastava and Fahim [82]	[− 10,− 10,− 10,− 10]	[20, 20, 20, 20]	− 44	[0,1, 2,- 1]
		CI–SAPF			**− 44**	**[0,1, 2,− 1]**
		CI–SAPF–CBO			**− 44**	**[0, 1, 2, − 1]**
		CBO			**− 32**	**[− 1, 1, 2, 0]**
5	Knapsack problem (maximization)	Srivastava and Fahim [82]	[0,…,0]	[100,…,100]	19,979	[32, 2, 1, 0, 0, 0, 0]
		CI–SAPF			**20,059**	**[16, 18, 9, 0, 2, 8,0]**
		CI–SAPF–CBO			**20,240**	**[27, 3, 4, 4, 8, 1, 5]**
		CBO			**18,428**	**[16, 10, 8, 0, 19, 8, 38]**
6	Integer linear problem (a) (maximization)	Srivastava and Fahim [82]	[0,…,0]	[200,…,200]	316	[4, 87, 34, 149, 0]
		CI–SAPF			**1037**	**[200, 199, 67, 104, 0]**
		CI–SAPF–CBO			**1040**	**[200, 200, 67, 106, 0]**
		CBO			**393**	**[111, 138, 27, 0,136]**
7	(b) (Maximization)	Srivastava and Fahim [82]	[0, 0]	[100,1 00]	33	[3,6]
		CI–SAPF			**33**	**[3,6]**
		CI–SAPF–CBO			**33**	**[3,6]**
		CBO			**33**	**[3,6]**
8	Non-convex Integer problem (formulation 1)	Tsai et al. [89]	[1, 1, 1]	[5, 5, 5]	− 75.7579	[1, 2, 5]
		CI–SAPF			**− 75.7579**	**[1, 2, 5]**
		CI–SAPF–CBO			**− 75.7579**	**[1, 2, 5]**
		CBO			**− 75.7579**	**[1, 2, 5]**
	(formulation 2)	Tsai et al. [89]	[0, 0, 0]	[5, 5, 5]	− 125	[5, 4, 0]
		CI–SAPF			**− 328.3159**	**[0, 5, 5]**
		CI–SAPF–CBO			**− 328.3159**	**[0, 5, 5]**
		CBO			**− 131.3264**	**[0, 2, 5]**
9	Global nonlinear mixed discrete programming	Tsai et al. [89]	[3, 3]	[6, 5]	− 246	[5, 4]
		CI–SAPF			**− 275**	**[5, 5]**
		CI–SAPF–CBO			**− 275**	**[5, 5]**
		CBO			**− 275**	**[5, 5]**
10	Three-bar truss design problem	Shin et al. [80]	[0.1, 0.2, 0.3, 0.5, 0.8, 1.0, 1.2]		3.0414	[1. 2, 0.5, 0.1]
		CI–SAPF			**3.0414**	**[1.2, 0.5, 0.1]**
		CI–SAPF–CBO			**3.0414**	**[1.2, 0.5, 0.1]**
		CBO			**3.0414**	**[1.2, 0.5, 0.1]**

Table 36 (continued)

Sr. No	Test functions	Solver	Search space		Function Value	Optimum variables
			Lower limit	Upper limit		
11	Monotone functions	Lawler and Bell [53]	[0, 0, 0, 0, 0]	[3, 3, 3, 3, 3]	8	[1, 1, 1, 1, 2]
		CI–SAPF			**16**	**[1, 1, 2, 1, 3]**
		CI–SAPF–CBO			**16**	**[1, 1, 2, 1, 3]**
		CBO			**16**	**[1, 1, 2, 1, 3]**
12		Lawler and Bell [53]	[0, 0, 0, 0, 0, 0, 0]	[7, 7, 7, 15, 15, 7, 15]	16	[1, 4, 1, 0, 2, 1, 2]
		CI–SAPF			**14**	**[0, 2, 4, 0, 2, 1, 6]**
		CI–SAPF–CBO			**14**	**[0, 2, 4, 0, 2, 1, 6]**
		CBO			**22**	**[2, 3,1, 1, 2, 1, 2]**
13		Lawler and Bell [53]	[0,0,0,0,0,0,0]	[7,7,7,15,15,7,15]	10	[0,6,0,1,1,1,1]
		CI–SAPF			**11**	**[1,3,2,1,1,1,2]**
		CI–SAPF–CBO			**11**	**[2,4,0,1,1,1,2]**
14		Lawler and Bell [53]	[0, 0, 0, 0, 0, 0, 0]	[7, 7, 7, 15, 15, 7, 15]	46	[0, 7, 0, 0, 0, 2, 1]
		CI–SAPF			**92**	**[2,4, 0, 0, 1, 2, 2]**
		CI–SAPF–CBO			**92**	**[2,4, 0, 0, 1, 2, 2]**
15		Lawler and Bell [53]	[0, 0, 0, 0, 0, 0, 0]	[7, 7, 7, 15, 15, 7, 15]	25	[1, 4, 1, 1, 1, 1, 2]
		CI–SAPF			**21**	**[2,3, 1, 1, 1, 1, 2]**
		CI–SAPF–CBO			**21**	**[2, 3, 1, 1, 1, 1, 2]**
16		Lawler and Bell [53]	[0, 0, 0, 0, 0, 0, 0]	[7, 7, 7, 15, 15, 7, 15]	1000	[0, 7, 0 ,0, 0, 2,1]
		CI–SAPF			**1331**	**[1, 3, 2, 1, 1, 1, 2]**
		CI–SAPF–CBO			**1331**	**[1, 3, 2, 1, 1, 1, 2]**

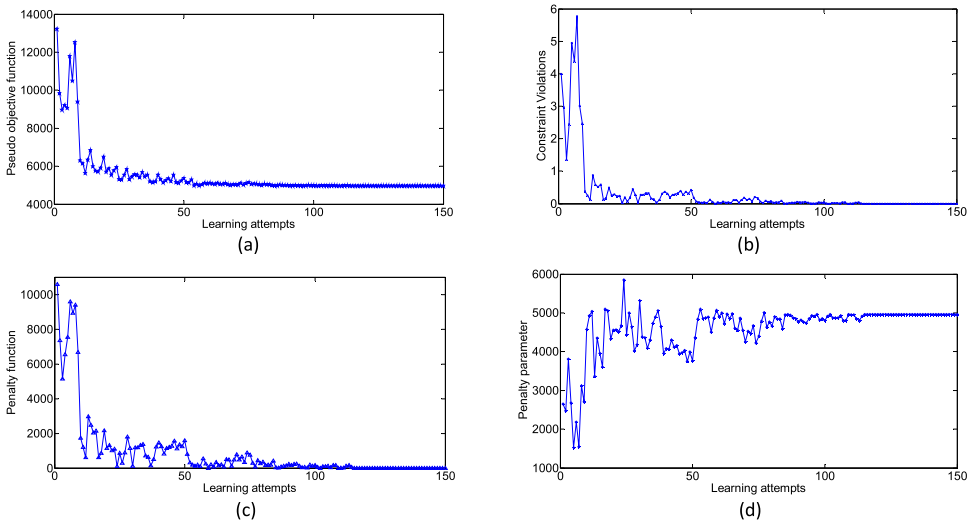

Fig. 4 Behavior of pseudo objective function (**a**), constraint violations (**b**), penalty function (**c**) and penalty parameter (**d**) solving discrete variable 6-bar truss structure problem using CI-SAPF

impact of this penalty parameter on the behavior of penalty function Fig. 4c), constraint violations Fig. 4b) and pseudo-objective function Fig. 4a) is observed to be precisely similar to calculated penalty parameter. The CI–SAPF is tested on four problems from different domains such as discrete variable six-bar truss structure problem, mixed variable pressure vessel design engineering problem, discrete variable linear problem and Rosen-Suzuki nonlinear problem. From the graphs Fig. 4, it is noted that the behavior of the overall cohort is based on the performance of SAPF which is ultimately keen towards the objective function (penalty parameter) value. From the graphs, the trends originated by the penalty parameters Fig. 4d) similar trends are followed by the constraint violations Fig. 4c), penalty functions Fig. 4b) and pseudo-objective functions Fig. 4a). As the penalty parameter value is iteratively updated, it behaves like a dynamic penalty function approach,

and similar trend obtained by the pseudo-objective function which represents the ability to jump out of the local minima and further helps to avoid the premature convergence of the solution.

Wilcoxon's rank sum test analysis

To examine the statistical significance of CI–SAPF and CI–SAPF–CBO, a non-parametric Wilcoxon's rank sum test (Mann–Whitney *U* Test) is performed for 30 independent trials (function values, CPU time and function evaluations). The null and alternative hypotheses are as follows:

H0: The distribution of results of CI-SAPF and CI-SAPF-CBO are identical.

H1: The distribution of results of CI-SAPF and CI-SAPF-CBO are not identical.

The lower tail test is performed at significance level $\alpha = 0.05$ (95%). The null hypothesis (H0) is rejected for *p*-value (calculated probability) less than α. The *p*-value is calculated based on the statistical test score. All 38 problems are independently tested including 10 discrete variable truss structure problems (refer Table 37, 11 mixed variable design engineering problems (refer Table 38 and 17 integer variable test functions (refer Tables 39 and 40. From the statistical results, the observations are as follows:

With function values: 31 times failed to reject the null hypothesis (H0),

With CPU time: 24 times failed to reject the null hypothesis (H0),

With function evaluations: 33 times failed to reject the null hypothesis (H0).

It means that there is no significant statistical difference between CI–SAPF and CI–SAPF–CBO when compared using function values and function evaluations; however, while comparing CPU time the CI–SAPF–CBO algorithm generated higher rank as compared to CI–SAPF. From this, it is concluded that CI–SAPF–CBO is marginally worse in computational time for solving some of these problems. In Tables 37, 38, 39, 40, the h-values 0 and 1 interpret the acceptance and rejection of the null hypothesis (H0), respectively. The main objective of hybridization of CI–SAPF–CBO is to make the generalized algorithm by removing the sampling space reduction factor *R* from CI–SAPF without degenerating the quality of solution. Here, the objective is achieved by overcoming the limitation of CI–SAPF algorithm with better quality of function value within lesser number of function evaluations and computational time.

Table 37 Comparison of CI–SAPF and CI–SAPF-CBO for truss structure problems using Wilcoxon's rank sum test at $\alpha = 0.05$

	Test Examples	6-Bar	10-Bar		25-Bar		38-Bar	45-Bar	52-Bar	72-Bar	
			Case 1	Case 2	Case 1	Case 2				Case 1	Case 2
Function Values	Test statistic	− 3.991794863	1.559756882	0.547023741	− 0.066529914	1.611502371	− 2.468999045	− 0.850104462	− 0.680083569	− 3.481732186	− 2.602058874
	P value	3.27875E-05	0.940591328	0.707818789	0.473477971	0.946464865	0.006774579	0.197633506	0.248225774	0.000249091	0.004633297
	h-value	1	0	0	0	0	1	0	0	1	1
CPU Time	Test statistic	3.932657161	4.050932565	0.243943019	0.650514718	− 1.951544155	− 1.463658116	1.604110158	− 1.374951564	3.208220316	0.532239315
	P value	0.999957994	0.999974493	0.596362525	0.742320101	0.025496174	0.071643692	0.945655113	0.084573231	0.999332204	0.702719872
	h-value	0	0	0	0	1	0	0	0	0	0
Function Evaluations	Test statistic	2.29158594	2.956885084	1.079263056	0.79096676	0.051745489	− 2.735118703	1.012733141	0.133059829	− 0.813143398	0.680083569
	P value	0.989035226	0.99844618	0.859764761	0.785518306	0.520634255	0.003117889	0.844406178	0.552926968	0.208067924	0.751774226
	h-value	0	0	0	0	0	1	0	0	0	0

Table 38 Comparison of CI–SAPF and CI–SAPF–CBO for engineering design problems using Wilcoxon's rank sum test at $\alpha = 0.05$

	Test examples	Steeped Beam	Pressure vessel	Speed Reducer	Concrete Beam	Welded Beam 1	Welded Beam 2	Multiple Disc Clutch	Tension Compression Spring	I Section Beam	Steeped Beam 2	Gear Train Design
Functionv-Values	Test statistic	− 1.175361821	− 0.199589743	3.740459631	− 0.51745489	5.10062677	− 0.798358973	0	0.118275403	− 6.1651054	0.554415953	− 4.124854692
	P value	0.119924996	0.420900725	0.999908158	0.302419326	0.999999831	0.212331102	0.5	0.547075277	3.52181E-10	0.710352895	1.85484E−05
	h-value	0	0	0	0	0	0	0	0	1	0	1
CPU time	Test statistic	− 3.474339973	0.088706553	− 3.903088311	6.652991439	3.577830951	− 3.223004741	0.443532763	− 0.140452041	5.056273493	− 2.542921172	1.508011393
	P value	0.000256056	0.535342438	4.74865E−05	1	0.999826771	0.000634267	0.671309786	0.444151421	0.999999786	0.005496501	0.934224194
	h-value	1	0	1	0	0	1	0	0	0	1	0
Function evaluations	Test statistic	− 1.995897432	− 0.539631528	0.842712249	− 0.591377017	6.652991439	− 6.652991439	0.443532763	− 0.384395061	3.829166184	− 1.478442542	1.700208923
	P value	0.022972544	0.294725585	0.800305302	0.277133918	1	1.43597E-11	0.671309786	0.35034283	0.999935711	0.069644683	0.955454183
	h-value	1	0	0	0	0	1	0	0	0	0	0

Table 39 Comparison of CI–SAPF and CI–SAPF–CBO for test problems using Wilcoxon's rank sum test at $\alpha = 0.05$

	Test examples	Dynamic problem (maximization)	Transportation problem	Multistage problem (maximization)	Rosen-suzuki test problem convex programming problem (minimization)	Knapsack problem (maximization)	Integer Linear problem (a) (maximization)	Integer Linear problem (b) (maximization)	Non-convex integer problem (formulation 1)	Non-convex integer problem (formulation 2)	Global nonlinear mixed discrete programming	Three-bar truss design problem
Function values	Test statistic	0	0.229158594	− 2.217663813	-0.236550807	-0.598769229	-0.827927823	0.221766381	1.108831906	1.552364669	0	− 0.221766381
	P value	0.5	0.590627173	0.013288882	0.406502644	0.274663392	0.203855688	0.587752124	0.866248648	0.939712504	0.5	0.412247876
	h-value	0	0	1	0	0	0	0	0	0	0	0
CPU time	Test statistic	− 6.608638162	-5.056273493	6.638207013	− 6.652991439	-4.169207968	-3.636968653	1.330598288	− 3.740459631	0.029568851	− 6.652991439	3.977010438
	P value	1.93936E−11	2.13764E−07	1	1.43597E−11	1.5283E−05	0.000137933	0.908339387	9.1842E-05	0.511794546	1.43597E−11	0.999965106
	h-value	1	1	0	1	1	1	0	1	0	1	0
Function evaluations	Test statistic	− 1.463658116	− 0.206981956	6.652991439	5.632866085	0.044353276	0.421356124	3.348672357	− 0.399179486	− .076398847	− 6.431225057	6.652991439
	P value	0.071643692	0.418011975	1	0.999999991	0.517688597	0.663252474	0.999594001	0.344880479	6.14557E-10	6.32898E−11	1
	h-value	0	0	0	0	0	0	0	0	1	1	0

Table 40 Comparison of CI–SAPF and CI–SAPF–CBO for monotone test functions using Wilcoxon's rank sum test at $\alpha = 0.05$

	Test examples	Monotone functions					
		1	2	3	4	5	6
Function values	Test statistic	0	0.384395061	− 0.221766381	1.818484327	− 1.108831906	0
	P value	0.5	0.64965717	0.412247876	0.96550493	0.133751352	0.5
	h-value	0	0	0	0	0	0
CPU time	Test statistic	− 6.372087356	− 4.538818604	4.361405499	-0.066529914	4.405758775	− 0.35482621
	P value	9.32363E-11	2.82851E-06	0.999993539	0.473477971	0.999994729	0.361359896
	h-value	1	1	0	0	0	0
Function evaluations	Test statistic	0.391787274	1.071870843	4.257914521	-0.066529914	4.213561244	3.836558396
	P value	0.6523923	0.858110976	0.999989683	0.473477971	0.999987431	0.999937615
	h-value	0	0	0	0	0	0

Applications

In the earlier sections, CI–SAPF and CI–SAPF–CBO algorithms are successfully applied to solve truss structure, design engineering and linear and nonlinear test problems having discrete and mixed design variable. To ensure the applicability of these techniques, two real-world applications from manufacturing engineering are also solved: multi-pass turning process problem for minimization of per unit production cost Gupta et al. [26] Chen 1996; Gayatri and Baskar [24] and multi-pass milling problem for maximization of production rate (minimization of time) [69, 79].

Multi-pass turning process problem

The multi-pass turning process problem aimed to minimize the unit production cost. The problem is previously solved using an integer programming model Gupta et al. [26]. In this, the problem is modified to solve in two phases. In first phase, the cost minimization of finish pass and rough pass is carried out and in second phase, optimal combination of depth of cut is identified. Then Chen and Tsai [7] proposed a nonlinear constrained optimization algorithm which is comprised of SA and Hooke-Jeeves pattern search (SA/PS). Onwobolu (2001) modified the problem by incorporating the surface finish constraints and adopted GA as solution methodology. This problem is also solved using GA, PSO, SA and Hybrid Genetic Simulated Swarm (HGSS) (Gayatri and Bhaskar [24] algorithms.

The multi-pass turning process problem is solved using GA, SA, PSO and HGGS in Gayatri and Bhaskar [24]. The constraint values are not illustrated in the literature, hence calculated in this work using same variables. It is noticed that, the cutting force (g_1) and chip-tool interface temperature constraints (g_4, g_6) are violated. It is evident that the solutions obtained using all the four techniques are not feasible (refer Table 41. For solving the same problem, both the proposed CI–SAPF and CI–SAPF–CBO algorithm obtained feasible solutions with best unit production cost of 2.59 $\$/piece$. The cost function obtained from CI–SAPF and CI–SAPF–CBO are marginally different. The best, mean and worst function values obtained from 30 trails are presented in Table 42 with other statistical results. It is noticed that, CI–SAPF found better solutions as compared to CI–SAPF–CBO with standard deviation of 0.07 and average function evaluations of 1110.

Multi-pass milling process problem

The multi-pass turning problem is adopted from Sonmez [79]. This problem is solved for the maximization of production rate (minimization of time). The problem contains mixed design variables, i.e. feed rate $\left(f_z \text{mm/tooth}\right)$ and spindle speed $(V\text{m/min})$ are continuous variables and depth of cut (a) is discrete variable. The total depth of cut is 5 mm. In the current work, the cutting strategy is adopted from [65], i.e. four passes are considered, first three are rough passes of $a = 1.5$ mm and for fourth finish cut $a = 0.5$ mm. All the constraints considered here are formulated in $\leq$ form. The problem is solved using CI–SPF, CI–SAPF, CI–SAPF–CBO and CBO. The total production time obtained using CI–SAPF and CI–SAPF–CBO is less as compared to ABC, PSO and SA (refer Table 43. It is observed that the feed rate obtained using proposed techniques is marginally higher than ABC, PSO and SA by satisfying all the constraints. From the result comparison, it is noticed that CI–SAPF–CBO yielded better solution as compared to other contemporary techniques. This is due to the use of CBO algorithm by which the exploration of search space makes the hybridization more powerful to obtain the best feasible solution. Moreover, the SAPF approach plays the key role to handle the practical constraints. The best production time is 2.15 min with average function evaluations of 1988 and average computational time of 12.26 s. Other statistical details are illustrated in Table 44.

Table 41 Comparison of results for multi-pass turning process problem

Variables	GA (Gayatri and Bhaskar [24]	SA (Gayatri and Bhaskar [24]	PSO (Gayatri and Bhaskar [24]	HGSS (Gayatri and Bhaskar [24]	CI–SAPF	CI–SAPF–CBO
V_r	389.15	447.94	499.99	499.9938	495.0761	496.0539
f_r	0.7209	0.7255	0.8999	0.8939	0.1163	0.107
d_r	2.03	2.21	2.50	2.50	2.4306	2.5186
V_s	102.78	117.17	89.55	93.8986	144.0232	167.1331
f_s	0.8788	0.5059	0.8999	0.8961	0.3289	0.299
d_s	1.94	1.58	1.00	1.00	1.0507	1.0644
g_1	115.5584	130.3326	188.2936	187.1010	0	− 0.0000
g_2	− 187.6149	− 184.4716	− 177.0964	− 177.2108	− 0.0195	− 0.0195
g_3	− 53,639.0414	− 65,729.5709	− 89,846.4004	-89,247.7831	− 1.1584	− 1.0325
g_4	400.4587	497.6636	691.2212	687.8356	− 0.0091	− 0.0105
g_5	− 9.9195	− 9.9733	− 9.9156	− 9.91635	− 0.001	− 0.0010
g_6	133.9725	50.0452	49.7859	49.4697	− 0.0001	-0.0002
g_7	− 196.3651	− 197.7465	− 198.2822	− 198.2045	-0.0199	-0.0199
g_8	− 17,282.8387	− 10,585.6043	− 7076.4803	− 7760.8662	− 0.7011	− 0.6552
g_9	− 131.5924	− 241.1098	− 227.9268	− 214.1411	− 0.0306	− 0.0311
Cost $/*piece*	2.42	3.07	2.23	2.22	2.59	2.59

Table 42 Statistical results for multi-pass turning process problem using CI–SAPF and CI–SAPF–CBO

Results	CI–SAPF	CI–SAPF–CBO
Best	2.59	2.59
Mean	2.69	2.71
Worst	2.84	2.84
Std. Dev	0.07	0.06
Avg. CPU time (*sec*)	0.26	0.18
Avg. Function evaluations	1110	1624

Conclusions and future directions

The ability of CI algorithm is exhibited to solve discrete and mixed variable constrained problems. The SAPF approach is developed for constraint handling, whereas CBO algorithm adopted for the local search. The ability of the CI–SAPF and CI–SAPF–CBO algorithms are examined for solving 10 discrete truss structure problems, 11 mixed variable design engineering problems and 17 discrete variable linear and nonlinear test problems. The penalty parameter required to run the SAPF approach is generated by CI-SAPF algorithm itself which iteratively updated based on the set of design variables. The SAPF approach eliminated the setting of penalty parameter which overcomes the limitation in SPF approach. From the results analysis and comparison, it is noticed that CI–SAPF algorithm performed exceptionally better in obtaining robust solutions with significantly less computational cost (i.e. computational time and function evaluations). The behavior of pseudo-objective function, penalty function and constraint violation are analyzed based on the obtained value of the penalty parameter. In addition, the CI-SAPF algorithm is hybridized by adopting the prominent qualities of CBO algorithm. The most important reason to use the CBO algorithm is to evade the setting of sampling space reduction factor R which is earlier used in all the versions of CI including CI–SAPF to narrow down the sampling space for better convergence of function value. The proposed CI–SAPF–CBO algorithm does not require parameter to run the algorithm. By adopting this, the preliminary computational efforts of parameter setting are eliminated. The results from CI–SAPF–CBO are significantly superior and robust as compared to the CI–SAPF and other contemporary algorithms. Furthermore, CI–SPF, CI–SAPF and CI–SAPF–CBO are also successfully applied for solving multi-pass turning and multi-pass milling process problems and exhibited superior results as compared to other contemporary techniques. Finally, a non-parametric Wilcoxon's rank sum test is conducted to check statistical significance of CI–SAPF–CBO over CI–SAPF for all 38 problems considered in Sect. 5. The test is performed based on the function values, function evaluations and CPU time obtained from 30 trials. Using function values and function evaluations, the performance of CI–SAPF and CI–SAPF–CBO is observed to be identical for 31 and 33 times, respectively. However, CPU time is observed to be identical only for 24 times. It shows that for 14 problems CI–SAPF–CBO is marginally worse as compare to CI–SAPF in terms of CPU time.

Table 43 Comparison of results for multi-pass milling process problem

Methods	Cutting strategy	f_z (mm/tooth)	V (m/min)	Arbor stress constraint	Arbor deflection constraint	Power constraint	T_{pr} (min)
ABC (Rao et al. 2010)	$a_{rough} = 1.5$	0.337	46.982	4.708	435.02	0.004	3.24
	$a_{rough} = 1.5$	0.337	46.982	4.708	435.02	0.004	
	$a_{rough} = 1.5$	0.337	46.982	4.708	435.02	0.004	
	$a_{finish} = 0.5$	0.432	64.41	271.97	1.131	1.400	
PSO (Rao et al. 2010)	$a_{rough} = 1.5$	0.34	46.61	1.5	431.9	0.01	3.24
	$a_{rough} = 1.5$	0.34	46.61	1.5	431.9	0.01	
	$a_{rough} = 1.5$	0.34	46.61	1.5	431.9	0.01	
	$a_{finish} = 0.5$	0.434	63.58	271.9	0.35	1.422	
SA (Rao et al. 2010)	$a_{rough} = 1.5$	0.336	44.633	1.5	436.1	0.204	3.26
	$a_{rough} = 1.5$	0.336	44.633	1.5	436.1	0.204	
	$a_{rough} = 1.5$	0.336	44.633	1.5	436.1	0.204	
	$a_{finish} = 0.5$	0.429	57.23	273.91	2.296	1.683	
CBO	$a_{rough} = 1.5$	2.1204	33.6372	− 3076.7856	− 7298.4319	− 3839.6497	2.23
	$a_{rough} = 1.5$	2.1204	33.6372	− 3076.7856	− 7298.4319	− 3839.6497	
	$a_{rough} = 1.5$	2.1204	33.6372	− 3076.7856	− 7298.4319	− 3839.6497	
	$a_{finish} = 0.5$	0.5144	31.4737	− 4695.8656	− 2031.3311	− 3848.6419	
CI–SAPF	$\boldsymbol{a_{rough}} = 1.5$	0.9396	30.0021	− 3911.8995	− 8133.5458	− 3844.8622	2.16
	$\boldsymbol{a_{rough}} = 1.5$	0.9396	30.0021	− 3911.8995	− 8133.5458	− 3844.8622	
	$\boldsymbol{a_{rough}} = 1.5$	0.9396	30.0021	− 3911.8995	− 8133.5458	− 3844.8622	
	$\boldsymbol{a_{finish}} = 0.5$	0.8222	30.0686	− 4589.8176	− 1925.2832	− 3848.1815	
CI–SAPF–CBO	$\boldsymbol{a_{rough}} = 1.5$	0.6686	30.0012	− 4139.5509	− 8361.1971	− 3845.9783	2.15
	$\boldsymbol{a_{rough}} = 1.5$	0.6686	30.0012	− 4139.5509	− 8361.1971	− 3845.9783	
	$\boldsymbol{a_{rough}} = 1.5$	0.6686	30.0012	− 4139.5509	− 8361.1971	− 3845.9783	
	$\boldsymbol{a_{finish}} = 0.5$	0.5350	30.0406	-4688.3005	− 2023.7660	− 3848.6666	

Table 44 Statistical results for multi-pass milling process problem using CI-SAPF, CI-SAPF-CBO and CBO

Results	CI-SAPF	CI-SAPF-CBO	CBO
Best	2.16	2.15	2.23
Mean	2.17	2.22	2.31
Worst	2.19	2.33	2.38
Std. Dev	0.82E-2	0.047	0.04
Avg. CPU time (sec)	17.61	12.26	1.39
Avg. function evaluations	2856	1988	227

In the near future, the CI–SAPF and CI–SAPF–CBO algorithms can be applied for solving real-world mechanical design engineering, transportation and healthcare domain. Based on the merits and demerits, other nature inspired algorithms can be hybridized with CI–SAPF for solving challenging and complex real-world problems from various domains.

Acknowledgements Authors would like to thank the anonymous reviewers for their valuable suggestions and comments which resulted in much improved manuscript quality.

Author contributions The detailed survey on constraint handling techniques specifically penalty function approach is presented in the paper. The Cohort Intelligence (CI) algorithm is incorporated with a novel self-adaptive penalty function (SAPF) approach which helped in avoiding preliminary trials of selecting penalty parameter. The approach is referred to as CI–SAPF. The CI–SAPF is further hybridized with Colliding Bodies Optimization (CBO) algorithm to promote a parameter-less metaheuristic algorithm. The approach is referred to as CI–SAPF–CBO. Several problems from discrete truss structure domain, mixed variable design engineering domain, and linear and nonlinear domain are solved to validate the CI–SAPF and CI–SAPF–CBO. The behavior of SAPF approach on pseudo objective function, constraint violations, penalty function and penalty parameter have been analyzed and discussed in very detail. The results obtained from CI–SAPF, CI–SAPF–CBO and CBO algorithms are analyzed and compared with other contemporary techniques. The CI–SPF, CI–SAPF and CI–SAPF–CBO are applied to solve a real-world manufacturing problems (multi-pass milling and turning processes).

Compliance with ethical standards

Ethical statement The authors declare that we do not have any conflict of interest. This article does not contain any studies with human participants performed by any of the authors. This article does not contain any studies with animals performed by any of the authors.

References

1. Aladeemy M, Tutun S, Khasawneh MT (2017) A new hybrid approach for feature selection and support vector machine model selection based on self-adaptive cohort intelligence. Expert Syst Appl 88:118–131
2. Arnout, S. (2011) 'International Student Competition in Structural Optimization' (ISCSO 2011), http://www.brightoptimizer.com
3. Bernardino HS, Barbosa HJC, Lemonge ACC (2007) A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. In: Proc. IEEE congress evolution computations, pp 646–653
4. Broyden CG, Attia NF (1984) A smooth sequential penalty function method for solving nonlinear programming problems. In: Thoft-Christensen P (eds) System modelling and optimization, lecture notes in control and information sciences, vol 59, Springer
5. Carlson SE, Shonkwiler R (1998) Annealing a genetic algorithm over constraints. In: SMC'98 conference proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218), San Diego, CA, USA, 1998, pp. 3931–3936 vol.4, https://doi.org/10.1109/ICSMC.1998.726702
6. Chen TY, Chen HC (2009) Mixed-discrete structural optimization using a rank-niche evolution strategy. Eng Opt 41(1):39–58
7. Chen MC, Tsai DM (1996) A simulated annealing approach for optimization of multi-pass turning operations. Int J Prod Res 34(10):2803–3282
8. Cheng MY, Prayogo D (2014) Symbiotic organisms search: a new metaheuristic optimization algorithm. Comput Struct 139:98–112
9. Coello CAC, Cortes NC (2004) Hybridizing a genetic algorithm with an artificial immune system for global optimization. Eng Opt 36(5):607–634
10. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41:113–127
11. Coello CAC, Montes EM (1992) Constraint-handling in genetic algorithms through the use of dominance-based tournament. IEEE Trans 41:576–582
12. Coath G, Halgamuge SK (2003) A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems. Evol Comput 4:2419–2425
13. Curtis FE, Nocedal J (2008) Flexible penalty functions for nonlinear constrained optimization. IMA J Numer Anal 28(4):749–776
14. Datta D, Figueira JR (2011) A real-integer-discrete-coded particle swarm optimization for design problems. Appl Soft Comput 11:3625–3633
15. Deb K (1991) Optimal design of a welded beam via genetic algorithms. AIAA J 29:2013–2015
16. Deb, K. and Agrawal, S. (1999) 'A niched-penalty approach for constraint handling in genetic algorithms', In: Proceedings of the international conference on artificial neural networks and genetic algorithms (ICANNGA-99), pp 235–243
17. Deb K (2000) An efficient constraint handling method for genetic algorithms. Comput Methods Appl Mech Eng 186(2–4):311–338
18. Deb K, Goyal M (1996) 'A combined genetic adaptive search (GeneAS) for engineering design. Comput Sci Inform 26:30–45
19. Deb K, Srinivasan A (2005) Innovization: innovation of design principles through optimization. KanGAL Report No. 2005007. Kanpur Genetic Algorithms Laboratory, Department of Mechanical Engineering, Indian Institute of Technology Kanpur, India
20. Efren M, Coello CAC, Ricardo L (2003) Engineering optimization using a simple evolutionary algorithm. In: Proc. 15th international conference on tools with artificial intelligence (ICTAI), pp 149–156
21. Emami H, Derakhshan F (2015) Election algorithm: a new socio-politically inspired strategy. AI Commun 28(3):591–603
22. Gandomi AH, Yang XS, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35
23. Gandomi AH, Yang X-S, Alavi AH (2011) Mixed variable structural optimization using firefly algorithm. Comput Struct 89(23–24):2325–2336
24. Gayatri R, Baskar N (2015) Evaluation process parameters of multi-pass turning process using hybrid genetic simulated swarm algorithm. J Adv Manuf Syst 14(4):215–233
25. Gen M, Cheng R (1996) A Survey of penalty techniques in genetic algorithms. In: Proceedings of the 1996 international conference on evolutionary computation, IEEE, pp 804–809
26. Gupta R, Batra JL, Lal GK (1995) Determination of optimal subdivision of depth of cut in multipass turning with constraints. Int J Prod Res 33(9):2555–2565
27. Hadj-Alouane AB, Bean JC (1997) A genetic algorithm for the multiple-choice integer program. Oper Res 45:92–101
28. Hasançebi O, Azad SK (2015) Adaptive dimensional search: a new metaheuristic algorithm for discrete truss sizing optimization. Comput Struct 154:1–16
29. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problem. Eng Appl Artif Intell 20(1): 89–99
30. Homaifar A, Lai SHY, Qi X (1994) Constrained optimization via genetic algorithms. Simulation 62(4):242–254
31. Jalili S, Kashan AH, Hosseinzadeh Y (2017) League championship algorithms for optimum design of pin-jointed structures. J Comput Civ Eng 31(2):04016048
32. Jalili S, Husseinzadeh Kashan A (2018) Optimum discrete design of steel tower structures using optics inspired optimization method. Struct Design Tall Special Build 27(9):e1466
33. Jalili S, Husseinzadeh Kashan A (2019) An optics inspired optimization method for optimal design of truss structures. Struct Des Tall Spec Build 28(6):e1598
34. Joines J, Houck C (1994) On the use of non-stationary penalty functions to solve non-linear constrained optimization problems with Gas. In: Proceedings of the first IEEE international conference on evolutionary computation, pp 579–584
35. Kale IR, Kulkarni AJ (2018) Cohort intelligence algorithm for discrete and mixed variable engineering problems. Int J Parallel Emergent Distrib Syst 33(6):627–662
36. Kashan AH (2011) An efficient algorithm for constrained global optimization and application to mechanical engineering design: league championship algorithm (LCA). Comput Aided Des 43(12):1769–1792
37. Kashan AH (2015) An effective algorithm for constrained optimization based on optics inspired optimization (OIO). Comput Aided Des 63:52–71

38. Kaveh A, Mahdavi VR (2015) Colliding Bodies Optimization Extensions and Applications. https://doi.org/10.1007/978-3-319-19659-6, Springer
39. Kaveh A, Talatahari S (2009) A particle swarm ant colony optimization for truss structures with discrete variables. J Constr Steel Res 65:1558–1568
40. Kannan BK, Kramer SN (1994) An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. J Mech Des 116(2):405–411
41. Kennedy J, Eberthart R (1997) A discrete binary version of the particle swarm optimizer. IEEE Conf Comput Cybernet Simul 5:4104–4108
42. Kulkarni AJ, Tai K (2010) Probability collectives: a multi-agent approach for solving combinatorial optimization problems. Applied Soft Computing 10(3):759–771
43. Kulkarni AJ, Kale IR, Tai K, Kazemzadeh He S (2012) Discrete optimization of truss structure using probability collectives. In: Proc. IEEE 12th international conference of hybrid intelligence system, pp 225–230
44. Kulkarni AJ, Kale IR, Tai K (2013) Probability collectives for solving truss structure problems. In: Proc. 10th world congress on structural and multidisciplinary optimization
45. Kulkarni AJ, Kale IR, Tai K (2016) Probability collectives for solving discrete and mixed variable problems. Int J Comput Aided Eng Technol 8(4):325–361
46. Kulkarni AJ, Durugkar IP, Kumar M (2013b) Cohort Intelligence: A Self Supervised Learning Behavior. Systems, man, and cybernetics (SMC), IEEE international conference, pp 1396–1400
47. Kulkarni AJ, Shabir H (2016) Solving 0–1 knapsack problem using cohort intelligence algorithm. Int J Mach Learn Cybernet 7(3):427–441
48. Kulkarni AJ, Baki MF, Chaouch BA (2016c) Application of the cohort-intelligence optimization method to three selected combinatorial optimization problems. Eur J Oper Res 250(2):427–447
49. Kulkarni O, Kulkarni N, Kulkarni AJ, Kakandikar G, (2016d) Constrained cohort intelligence using static and dynamic penalty function approach for mechanical components design. In: International journal of parallel, emergent and distributed systems, pp 1–19
50. Kulkarni AJ, Krishnasamy G, Abraham A (2017) Cohort intelligence: a socio-inspired optimization method, Intelligent Systems Reference Library, 114, Springer. https://doi.org/10.1007/978-3-319-44254-9), (ISBN: 978-3-319-44254-9)
51. Kulkarni AJ, Tai K (2011) A probability collectives approach with a feasibility-based rule for constrained optimization, Applied Computational Intelligence and Soft Computing, Article ID 980216
52. Kumar M, Kulkarni AJ, Satapathy SC (2018) Socio evolution & learning optimization algorithm: a socio-inspired optimization methodology. Fut Generation Comput Syst 81:252–272
53. Lawler EL, Bell MD (1966) A method for solving discrete optimization problem. Oper Res 14(6):1098–1112
54. Le Riche R, Knopf-Lenior C, Haftka RT (1995) A Segregated genetic algorithm for constrained structural optimization. In: Proceedings of the sixth international conference on genetic algorithms, Morgan Kaufmann, 558–565
55. Lee KS, Geem ZW, Lee SH, Bae KW (2005) The harmony search heuristic algorithm for discrete structural optimization. Eng Opt 37(7):663–684
56. Lemonge ACC, Barbosa HJC (2004) An adaptive penalty scheme for genetic algorithms in structural optimization. Int J Numer Methods Eng 59(5):703–736
57. Li B, Yu CJ, Teo KL, Duan GR (2011) An exact penalty function method for continuous inequality constrained optimal control problem. J Opt Theory Appl Springer 151:260–291
58. Liu ZZ, Chu DH, Song C, Xue X, Lu BY (2016) Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition. Inf Sci 326:315–333
59. Li LJ, Huang ZB, Liu F (2009) A heuristic particle swarm optimization method for truss structures with discrete variables. Comput Struct 87(7–8):435–443
60. Luenberger DG, Ye Y (2016) Penalty and barrier methods. In: Linear and Nonlinear Programming. International Series in Operations Research & Management Science', Vol. 228, Springe
61. Lv W, He C, Li D, Cheng S, Luo S, Zhang, X (2010) Election campaign optimization algorithm. Procedia Computer Sci 1(1):1377–1386
62. Michalewicz Z, Attia N (1994) Evolutionary optimization of constrained problems. In: Proceedings of the Third Annual Conference on Evolutionary Programming, World Scientific, pp. 98–108
63. Morales AK, Quezada CV (1998) 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
64. Moosavian N, Roodsari BK (2014) Soccer league competition algorithm: A novel metaheuristic algorithm for optimal design of water distribution networks. Swarm Evol Comput 17:14–24
65. Nanakorn P, Meesomklin K (2001) An adaptive penalty function in genetic algorithms for structural design optimization. Comput Struct 79:2527–2539
66. Nie PY (2006) A new penalty method for nonlinear programming. Comput Math Appl 52:883–896
67. Parsopoulos K, Vrahatis M (2002) Particle swarm optimization method for constrained optimization problems, In: Intelligent Technologies Theory and Applications: new trends in intelligent technologies. Vol. 16 of Frontiers in Artificial Intelligence and Applications, pp. 214–22
68. Rao RV, Savsani VJ, Vakharia DP (2011) Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 43(3):303–315
69. Rao RV, Pawar PJ (2010) Parameter optimization of a multi-pass milling process using non-traditional optimization algorithms. Appl Soft Comput 10(2):445–545
70. Rao RV, Savsani VJ (2012) Mechanical Design Optimization using advanced optimization techniques. Springer. https://doi.org/10.1007/978-1-4471-2748-2
71. Rajeev S, Krishnamoorthy CS (1992) Discrete optimization of structures using genetic algorithm. J Struct Eng ASCE 118(5):1123–1250
72. Rudolph S, Schmidt J International Student Competition in Structural Optimization' (ISCSO 2012), http://www.brightoptimizer.com
73. Sadollah A, Bahreininejad A, Eskandar H, Hamdi M (2012) Mine blast algorithm for optimization of truss structures with discrete variables. Comput Struct 49(63):102–110
74. Sandgren E (1990) Nonlinear integer and discrete programming in mechanical design optimization. J Mech Des 112(2):223–229
75. Satapathy S, Naik A (2016) Social group optimization (SGO): a new population evolutionary optimization technique. Complex Intell Syst 2(3):173–203
76. Shastri AS, Jadhav PS, Kulkarni AJ, Abraham A, (2016) Solution to constrained test problems using cohort intelligence algorithm. In: Innovations in Bio-Inspired Computing and Applications, pp 427–435
77. Shih CJ, Yang YC (2002) Generalized Hopfield network based structural optimization using sequential unconstrained minimization technique with additional penalty strategy. Adv Eng Softw 33(7–10):721–729
78. Sonmez M (2011) Artificial Bee Colony algorithm for optimization of truss structures. Appl Soft Comput 11(2):2406–2418

79. Sonmez AI, Baykasoglu A, Dereli T, Filiz IH (1999) Dynamic optimization of multipass milling operations via geometric programming. Int J Mach Tools Manuf 39:297–320
80. Shin DK, Gurdal Z, Griffin OH (1990) A penalty approach for nonlinear optimization with discrete design variables, Engineering Optimization, 16(1): 29–42
81. Smith A, Tate D (1993) Genetic optimization using a penalty function. In: Proceedings of the fifth international conference on genetic algorithms, Morgan Kaufmann, pp. 499–503
82. Srivastava VK, Fahim A (2001) A two-phase optimization procedure for integer programming problems. Comput Math Appl 42:1585–1595
83. Thanedar PB, Vanderplaats GN (1995) Survey of discrete variable optimization for structural design. J Struct Eng 121(2):301–330
84. Viswanathan J, Grossmann IE (1990) A combined penalty function and outer-approximation method for MINLP optimization. Comput Chem Eng 14(7):769–782
85. Wolpert DH, Tumer A (1999) Introduction to Collective Intelligence, Technical Report, NASA ARC-IC-99-63, NASA Ames Research Center
86. Wu SJ, Chow PT (1995) Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct 56(6):979–999
87. Yun YS (2005) Study on Adaptive Hybrid Genetic Algorithm and its Applications to Engineering Design Problems, MSc thesis, Waseda University
88. Teo TH, Kulkarni AJ, Kanesan J, Chuah JH, Abraham A (2016) Ideology algorithm: a socio-inspired optimization methodology. Neural Comput Appl 28(1):845–876
89. Tsai J-F, Li H-L, Hu N-Z (2002) Global optimization for signomial discrete pogramming problems in engineering design. Eng Opt 34(6):613–622
90. Yokota T, Gen M, Ida K, Taguchi T (1996) Optimal design of system reliability by an improved genetic algorithm, Electronics and Communications in Japan (Part III: Fundamental Electronic Science), Vol. 79, No. 2, pp. 41–51
91. Yu C, Teo KL, Zhang L, Bai Y (2010) A new exact penalty function method for continuous inequality constrained optimization problems. J Ind Manag Opt 6(4):895–910