Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.1, March : 2025

METAHEURISTICS CONSTRAINT HANDLING STRATEGIES IN MULTI-OBJECTIVE OPTIMIZATION: A COMPREHENSIVE REVIEW AND INNOVATIVE SOLUTION

Mr. Vishal T. Metkari Assistant Professor, Dept.of Electrical Engineering, Sanjeevan Engineering and Technology Institute, Tal-Panhala, Kolhapur, Maharashtra,
(Prof.) Dr. H. T. Jadhav, Dean, Faculty of Science & Technology, S.N.D.T. Women's university Mumbai, Maharashtra

ABSTRACT
Multi-objective optimisation algorithms (MOAs) solve complex engineering, design, and economics problems using computational search processes inspired by physical and biological phenomena. They were designed for unconstrained NP-complete problems, but constrained multi-objective optimisation problems (CMOPs) require special constraint management techniques. Modern constraint handling techniques (CHTs) integrated into MOAs are examined in this review to assess their performance and applicability. Key CHTs include penalty-based, feasibility rules, ε-constrained, stochastic ranking, repair-algorithm, and boundary-based strategies, each with pros and cons. Eight new feasibility rule and ε-constrained technique variants are proposed after reviewing over 60 key studies. High-Performance Optimisation Computing Platform (HP-OCP), these variants show superior adaptability and robustness. Adaptive penalty functions, ensemble CHTs, and hybrid approaches to balance exploration and exploitation are recommended in the review. We also discuss computational overhead and convergence speed-solution quality trade-offs. In dynamically constrained, multi-modal, and high-dimensional spaces, innovative approaches are needed. For designing algorithms that can solve real-world engineering problems with strict constraints, the review emphasises hybridisation and adaptive mechanisms. This review helps researchers and practitioners find effective CHTs in MOAs by consolidating existing knowledge and proposing new solutions. These findings may extend metaheuristic optimisation to more complex real-world problems.

Keywords
Constrained handling strategies, Multi-objective optimization, Metaheuristic algorithm, MATLAB computing platform

INTRODUCTION
Contemporary optimisation problems are notably nonlinear and involve variables that are constrained by various functions and bounds. These issues are referred to as constrained optimisation problems (COPs) or nonlinear programming (NLP) problems, and their formulation can be articulated in the following general form, as outlined by Deb (2000):

$$\begin{aligned} &Minimize f(x), \\ &Subject: \\ &g_j(x) \le 0,\ j = 1, 2, \ldots,\ J, \\ &h_k(x) = 0,\ \ k = 1, 2, \ldots, K, \\ &x_i^l \le x_i \le x_i^u,\ \ i = 1, 2, \ldots, n, \end{aligned} \qquad (1)$$

where $f(x)$ is the objective function of the vector variable x, $g_j(x)$ are inequality and $h_k(x)$ are equality constraints, both referred to also as performance constraints, and x^l_i , x^u_i are respectively the lower and upper limit values of component x_i in x. Equality constraints can be converted into inequality constraints as $g_k(x) = |h_k(x)| - e \le 0$, $k = 1, 2, ..., K$, where e is a small positive quantity (e.g. 1.0E-3), and hence the two groups of performance constraints in expression (1) can be unified into a single group $g_j(x) \le 0$, $j = 1, 2, ...,m$, where $m = J + K$.

Physical and biological phenomena inspire metaheuristic optimisation algorithms (MOAs), computational randomised search processes. The application field of MOAs has grown beyond physical and biological sciences to include engineering and economics. However, most MOAs were developed for unconstrained NP complete optimisation problems, making their application to COPs difficult due to constraint handling. MOAs are mostly used for COP issues. Since no constraint handling strategy (CHS) guarantees search algorithm convergence to the global optimum, overcoming the constraint handling obstacle has become an independent research field. Development of new CHSs and performance enhancement of existing ones have been research priorities. A number of authors have developed, improved, and evaluated CHSs implemented in MOAs for use in COPs in various fields.

Coello and Montes (2002) reviewed and evaluated the most popular CHSs in evolutionary algorithms (EAs), including penalty, separation of constraints and objectives, special operators, hybrid, and repair-algorithm-based CHSs. A genetic algorithm was used to evaluate several penalty-based techniques. Mezura-Montes and Coello (2011) reviewed CHSs in nature-inspired algorithms, such as EAs and swarm intelligence, including feasibility rules, ε-constrained, penalty, stochastic ranking, special operators, multi-objective concepts, and ensemble CHSs. Mallipeddi et al. (2012) assessed penalty and adaptive penalty, feasible solutions, ε-constraint method, stochastic ranking, and ensemble CHSs for optimal reactive power dispatch (ORPD) problems using DE algorithms. Jordehi (2015) evaluated penalty-based CHSs, separatists, hybrids of particle swarm optimisation (PSO) with other optimisation methods, and other methods not listed above. Miranda-Varela and Mezura-Montes (2018) compared four surrogate-assisted algorithms (SA-DECV) using feasibility rules, ε-constrained method, stochastic ranking, and diversity maintenance CHS to other algorithms using 24 well-known test functions. According to Ameca-Alducin et al. (2018), a DE algorithm with penalty function, feasibility rules, ε-constrained method, and stochastic ranking CHSs was evaluated using benchmark dynamic COPs. Lin et al. (2019) evaluated four dynamic multimodal population-based optimisation algorithms, utilising feasible solution, ε-constraint method, penalty function, dynamic penalty function, and stochastic selection and ranking CHSs for dynamic multimodal COPs. Caraffini et al. (2019) examined the behaviour of popular DE schemes with different CHSs, such as penalty function and saturation/toroidal correction techniques, focussing on mutation and crossover operators that introduce structural bias in DE algorithms and appropriate CHSs, DE control parameters, and population size to moderate the bias.

Despite the extensive research above, a systematic review of MOA CHSs has not been done. In this context, this paper provides a comprehensive state-of-the-art review of the most widely used and effective MOA CHSs. This paper introduces eight new variants of feasibility rules and ε-constrained CHSs, comparing their performance to existing ones using a swarm intelligence pity beetle algorithm (Kallioras et al. 2018) with 14 mathematical and 6 engineering COPs. All computations are done using the MATLAB variant of High-Performance Optimisation Computing Platform (HP-OCP), the evolution of OCP (Lagaros 2014), which is based on C# object-oriented general-purpose code for civil engineering structural design optimisation.

Section 2 presents a state-of-the-art review of nearly 60 CHS studies from the literature. Sect. 3 provides an overview of penalty methods, feasibility rules, ε-constrained, stochastic ranking, and ensemble CHSs, along with eight novel variants of each. After a complete outline and formulation of Metaheuristic algorithm constraint handling strategies, the study concluded with an emphasis on features and limitations and a comprehensive review of innovative multi-objective optimisation solutions.

Table1: Constraint handling Strategies in Literature Review

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.1, March : 2025

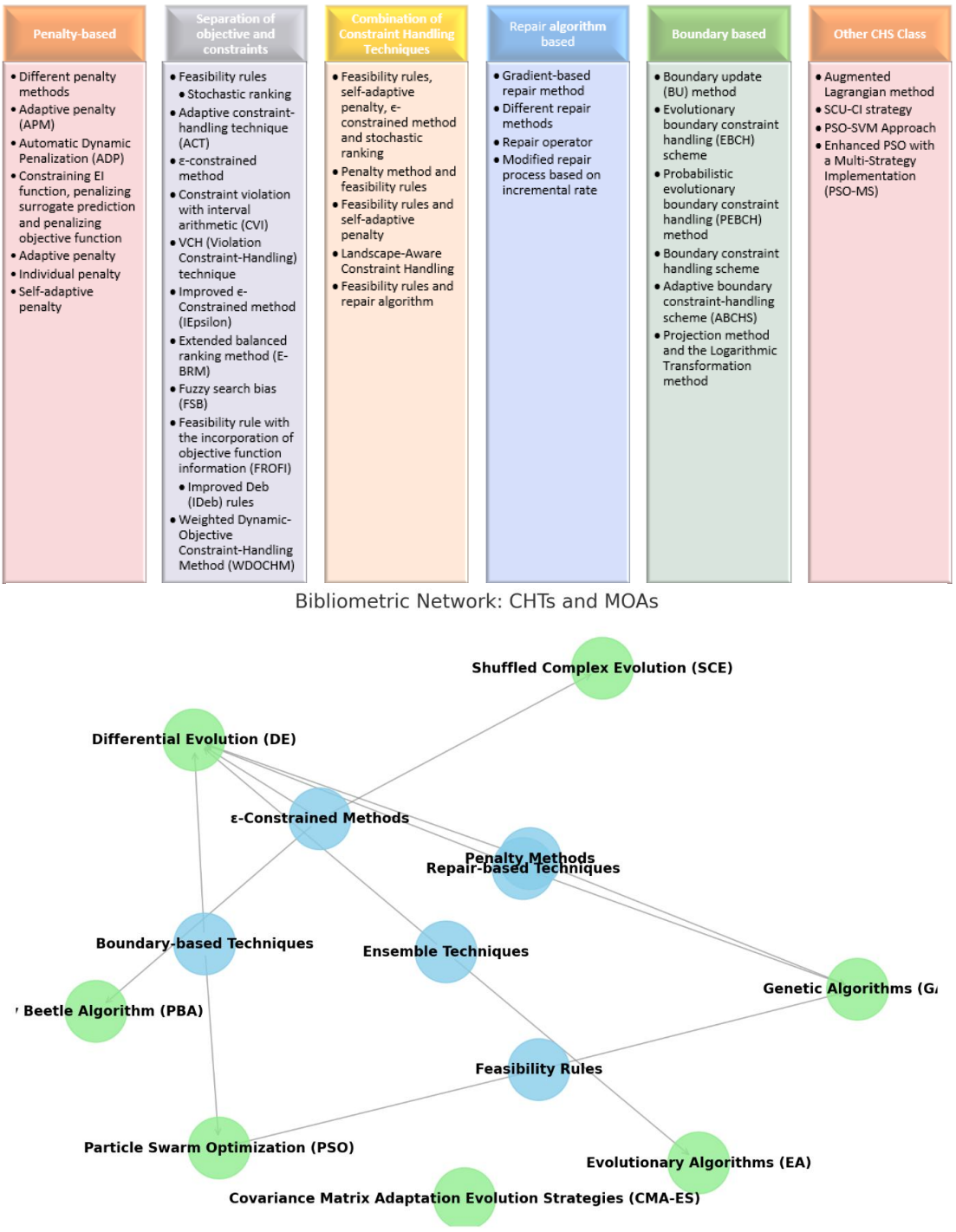

Fig. 1: Bibliometric network diagram showing CHT-MOA relationships.

2 COMPREHENSIVE REVIEW ON CONSTRAINT HANDLING STRATEGIES

This section reviews the most popular and effective MOA CHSs in six categories: penalty-based CHSs, separation-of-objective-and-constraint CHSs, combination CHSs, repair-algorithm-based CHSs, boundary-based CHSs, and others. Table 1 lists all reviewed CHSs, their MOAs, and their literature reference numbers.

2.1 Penalty-based constraint handling techniques

Objective function penalisation is the oldest and most popular CHS. Static, dynamic, adaptive, and death penalty methods have been developed over time. Miettinen et al. (2003) tested five penalty-based CHSs with genetic algorithms to solve 33 mathematical test problems of various types. The

adaptive penalty method (APM) was the most efficient and the parameter-free penalty method the most reliable. Nearly a decade later, Da Silva et al. (2011) proposed a parameter-free APM for use with DUVDE algorithm, a DE that uses Dynamic Use of Variants (DUV) to automatically select the best variant for structural and mechanical engineering applications. By using the feedback from the current candidate solution population to adaptively define the penalty factor for every constraint combined with DUV, this CHS made DE more efficient in solving COPs and achieved competitive results compared to GA and simple DE.

Montemurro et al. (2013) applied automatic dynamic penalisation (ADP), a parameter-free penalty-based CHS, to engineering design problems like composite laminate first buckling load maximisation. The proposed CHS is combined with BIANCA, a multi-population genetic algorithm that uses crossover and mutation operators to evolve different species simultaneously. ADP uses the information of infeasible individuals to automatically select and update penalty factors, expanding the search space. Li et al. (2019a) used a Kriging model in a dynamic surrogate-based optimisation (DSBO) of standard test functions and an engineering problem, using two sample selection criteria: maximisation of the expected improvement (EI) function and minimisation of surrogate model prediction. They handled constraint handling by constraining the EI function, penalising the surrogate model prediction, and penalising the objective function DSBO performed best in two tests and a beam optimisation problem using methods (a) and (b).

In 2019, Kawachi et al. introduced LSHADE, a DE algorithm with an APM CHS that considers both the objective function value and the constraint violation in the search region, to optimise benchmark functions from the Congress on Evolutionary Computation (CEC) 2017. In most cases, the proposed method outperformed a conventional APM CHS and other CHSs by balancing constraint violation and objective function values. In 2019, Datta et al. proposed the constraint handling with individual penalty (CHIP) technique, which combines a common penalty-based CHS with the bi-objective EA NSGA-II. Instead of using a global factor for the overall constraint violation, the method adaptively computes the penalty factors for each constraint using the overall constraint violation as an auxiliary objective function subject to minimisation. Constraint automatic normalisation and low computational demand are advantages of the proposed CHS. The EJADE-SP algorithm, developed by Li et al. (2020) for optimal power flow (OPF) problems, is an enhanced adaptive DE with a self-adaptive penalty (SP)CHS and includes a strategy for dynamic population reduction, crossover rate (CR) sorting mechanism, CR re-randomization, and scale factor F parameters. This algorithm with an SP CHS performed similarly to simple DE with different CHSs.

2.2 Techniques based on separation of objective and constraints

Deb (2000) created a penalty-based CHS called feasibility rules for population-based algorithms like genetic or other EAs that uses tournament selection feasibility rules to compare pairs of individuals without penalty parameters. The proposed method eliminates the need for user penalty factor selection in penalty-based CHSs to find the best solution.

Stochastic ranking (SR), a new CHS introduced by Runarsson and Yao (2000), uses a stochastic bubble-sort algorithm to balance objective and penalty functions and solves 13 benchmark problems using an evolution strategy. SR, like Deb (2000), eliminates user penalty factor selection. Coello and Mezura-Montes (2002) used the niched-pareto genetic algorithm (NPGA) for tournament selection in structural design optimisation problems. Nondomination is used in multi-objective problems. NPGA treat constraints as additional objective functions for single-objective COPs. The proposed method efficiently handles constraints without penalty functions and maintains population diversity without niching using the multiobjective concept.

Wang et al. (2009) used a hybrid EA with two mutation operators and a simplex crossover operator to generate new offspring in 13 high-profile benchmark test problems. The algorithm uses an adaptive CHS that adapts to population status based on infeasible, semi-feasible, and feasible conditions. Simple implementation, increased robustness, and effectiveness are among the method's benefits. In Takahama and Sakai (2010), a DE algorithm with archive and gradient-based mutation

operator and ε-constrained CHS was successfully applied to 18 CEC2010 test problems. The proposed CHS uses automatic control parameter adjustment to relax constraints and retain useful information from infeasible individuals during the search process. Mazhoud et al. (2013) used constraint violation with interval arithmetic (CVI) and a customised PSO algorithm in 24 benchmark mathematical and 3 engineering optimisation problems. CVI uses interval arithmetic for total violation normalisation, a simple lexicographic method for solving subsequent problems, and the total constraint violation as an objective function subject to minimisation.

Zhang et al. (2015) applied a new EA, backtracking search algorithm (BSA), to 13 benchmark functions and 4 engineering problems, incorporating feasibility rules and ε-constrained method with two ε value control methods. The ε-constrained CHS with self-adaptive ε control outperformed the other two CHSs in efficiency and convergence speed. Chehouri et al. (2016) implemented a parameter-free CHS for NLP problems using a GA. The approach uses the violation factor to divide the population into two families of feasible and infeasible solutions, with the latter containing individuals violating at least one constraint. For population evolution at each generation, the first family is sorted by fitness function value and the second by comparison rules. These rules compare two infeasible individuals by considering the number of violated constraints in addition to the constraints violation value.

Peng et al. (2018) applied EAs and a biassed dynamic weight CHS to 24 CEC2006 and 18 CEC2010 benchmark problems. Biassed weights select individuals with low objective and constraints violation values and dynamically adjust weights to focus on them. Since parents and children are closer to the feasible search space, the approach prioritises selecting infeasible individuals from them. When integrated into a DE algorithm, this CHS is stable, reliable, and competitive. Fan et al. (2018) applied IEpsilon, an improved ε-constrained CHS, to DE algorithm LSHADE44 for 28 CEC2017 benchmark problems. IEpsilon differs from the traditional ε-constrained handling method by adapting the ε value to the proportion of feasible individuals in the population, balancing the search for feasible and infeasible space during evolution.

In Rodrigues et al. (2018) proposed an alternative version of the balanced ranking CHS method, called Extended-BRM, and implemented it in evolutionary algorithms for application to CEC2006 and CEC2010 test functions and engineering problems. The proposed technique employs a self-adaptive mechanism to rank feasible and infeasible individuals in two separate groups, and later merges them during the search process with respect to the proportion of feasible and infeasible individuals in the current population. This CHS requires no parameter adjustment by the user, produces the most feasible solutions, and achieves superior performance compared to other CHS s, such as stochastic ranking, adaptive penalty method, etc. Li et al. (2019b) optimised 24 benchmark test functions using EAs and FSB, a fuzzy knowledge-based CHS guided by a membership function. Fuzzy knowledge works by searching for candidate solutions with large constraints violation values leading to smaller constraints violation values and better objective function values. The FSB CHS in various EAs (DE, ES, GA, PSO, EP) outperformed penalty function, stochastic ranking, ε-constrained, and feasibility rules CHSs in terms of robustness and efficiency.
Zhao et al. (2020) applied FROFI, a feasibility-rule-based CHS, to 2 test problems and 2 reservoir models using a radial basis function (RBF) surrogate-assisted DE algorithm. FROFI integrates objective function information through DE algorithm operators and a replacement procedure into the well-known Deb rules to improve objective function-constraint balance. Liu et al. (2020) integrated an improved Deb rule called IDeb into a Fruit fly optimisation algorithm (FOA) (a search strategy based on memory, inspired by fruit flies' foraging behaviour) to optimise truss structures. The method uses the classic Deb rule to identify feasible individuals in the solutions memory size and then evaluates the constraints violation only of those with better objective function values than the worst one stored in memory. Since constraints violation evaluations are more computationally expensive than objective function evaluations, the IDeb CHS can significantly reduce structural analysis computational demand.

Chu et al. (2020) used a DE algorithm with an ε-constraint CHS to optimise polyline-based core sandwich structures (PCSSs) topology. The dominant volume constraint is approximated by the

RBF surrogate model for low computational cost, then handled by the EC method to compare different infeasible solutions, resulting in more effective search space exploration and reduced RBF prediction model errors.

A metaheuristic search and rescue (SAR) optimisation algorithm by Shabani et al. (2020) simulates rescuer behaviour. Combining SAR with ε-constrained CHS led to high performance in benchmark test functions and 7 engineering problems. Stanovov et al. (2020) optimise benchmark functions and economic load dispatch problems using ε-constraint CHS variations and a DE algorithm. The combined fitness-ε vector length with individual penalty levels (ECIFL) variant outperformed classical ε-constraint CHS, proposed variants, and Iepsilon method when considering constraints violation, objective function values, and each constraint individually. For doubly fed induction generator design optimisation, Rodrigues et al. (2020) proposed a PSO-based algorithm with a weighted dynamic objective constraint handling method (WDOCHM-PSO). This method divides the constrained problem into two unconstrained objective functions: one considers the distance of infeasible individuals from the feasible region (violation); the other considers the optimisation problem without constraints and is used only for feasible individuals. Mao et al. (2019) used the SR CHS in shuffled complex evolution (SCE), an optimisation algorithm that combines the competitive complex evolution (CCE) algorithm with the simplex method to solve constrained reservoir scheduling problems. The SR CHS is efficient in identifying feasible solutions, even in the early stages of optimisation, and its integration into SCE creates a robust algorithm that can quickly find feasible regions.

2.3 Combination of constraint handling techniques

Recently, optimising with multiple CHSs has become popular. Mallipeddi and Suganthan (2010) combined CHSs (feasibility rules, self-adaptive penalty, ε-constraint method, stochastic ranking) with EP and DE algorithms for COPs. Each CHS represents a different population, and each parent is compared to all population children. A combination of CHSs outperformed individual ones. In a Unified Differential Evolution (UDE) algorithm, Trivedi et al. (2017) optimised 28 CEC 2017 test problems using the penalty method and feasibility rules CHS s. The proposed approach penalises the first half of maximum allowed function evaluations for efficient search space exploration and feasibility rules for the other half for efficient exploitation.

Biswas et al. (2018) used a DE algorithm, a self-adaptive penalty CHS, and a feasibility rules CHS to evaluate standard IEEE 30 57- and 118-bus systems for OPF objective functions like cost, emission, voltage stability, etc. Combinations of CHSs outperformed individual ones in most cases. Malan (2018) optimised 18 CEC2010 test problems using a DE algorithm and landscape-aware constraint handling. The landscape-aware approach switches between four CHSs based on the landscape characteristics gathered during the search process, with only one CHS active at a time, unlike the well-known ensemble of CHSs. Wang et al. (2019) optimise CEC2017 test problems using DEVNS, a DE algorithm variant with two mutation strategies and random crossover and mutation values. They also use feasibility rules and ε-constrained CHSs. In the proposed method, feasibility rules are used to compare trial and target individuals. Individuals with lower constraint violations than the specified ε level are considered feasible. Thus, constraints are relaxed, improving search efficiency.

Javed et al. (2019) optimised 24 CEC2006 benchmark problems using JADE and SADE algorithms, including feasibility rules, self-adaptive penalty, ε-constrained, and stochastic ranking CHSs. Both algorithms had a competitive feasibility rate but a lower success rate than other algorithms using a single CHS based on CEC2006 evaluation criteria. Kaucic et al. (2020) introduced IC-PSO, a PSO-based algorithm for numerical optimisation problems, and a hybrid CHS with Deb rules and a correction mechanism. The proposed CHS uses a suitable operator to repair infeasible individuals and applies feasibility rules to accelerate PSO algorithm convergence to feasible regions.

2.4 Repair algorithm-based constraint handling techniques

Repair algorithms (typically heuristics) are used to fix infeasible individuals and make them feasible in COP solutions. Repaired individuals can be used only for fitness function evaluation or to replace the initial individual in the next generation during evolution. The developer's decision to replace the original individual with the repaired one, a small percentage, or every infeasible individual in the population during evolution depends on the problem. Chootinan and Chen (2006) solved 11 optimisation test problems using a constraint gradient-based repair mechanism in a simple GA. Constraint gradient information is obtained via approximate finite differences or direct derivatives of constraints to guide infeasible individuals into the feasible search space. The user only needs to adjust the repair probability parameter, and the CHS performs similarly to stochastic ranking. Salcedo-Sanz (2009) surveys the main EA repair heuristics for COP performance. Zein et al. (2017) used a GA-integrated repair operator for composite structure preliminary design. While the inner loop solves the problem using the GA and surrogate models, the outer loop reconstructs the surrogate model using the optimal surrogate solution. The proposed method balances accuracy and computational demand.

Li et al. (2018) modified PSO and DE algorithms to handle equality constraints on Economic Dispatch problems using six test generators. The repair technique evaluates the objective function derivative and incrementally adjusts unit output by sharing the unbalanced system constraint violation. CHS is efficient, especially for large power systems. In 2020, Gandomi and Deb implemented the boundary update (BU) CHS in EAs and mathematical algorithms and tested it in engineering and mathematical optimisation problems. BU restricts and changes variable limits during optimisation to avoid constraints violation, generating solutions within the search space with "updated" limits. The BU method solves complex COPs well despite requiring constraint variables pre-categorizing.

2.5 Boundary-based constraint handling techniques

Expression (1) has function and bound constraints. Function constraints include inequality or equality functional forms of decision variables, while bound constraints enforce upper and lower bounds. Most research has focused on CHSs for function constraint handling, but a few has tackled boundary constraint handling methods. BCHMs reset infeasible variable solution vectors in the search space and make them feasible.

Gandomi and Yang (2012) compare an evolutionary boundary constraint handling (EBCH) scheme integrated into DE with other boundary-based methods using several test problems. In the proposed method, components that violate bounds are replaced by a random component between the bound and the current best solution. The proposed scheme outperforms absorbing, toroidal, reflecting, and random BCHMs in most cases. Gandomi and Kashani (2018) optimised 7 benchmark test functions using PEBCH and PSO. The boundaries violation quantity is considered in PEBCH, a probabilistic version of EBCH (Gandomi and Yang 2012). A probability distribution function is used, with higher density near the violated boundary. Mozaffari et al. (2019) examine the effects of chaotic maps on evolutionary algorithms (EAs). To handle boundary constraints, a controlling formula is used to correct a violated solution position into acceptable solution limits, enhancing global search and preventing early convergence.

Biedrzycki et al. (2019) used CEC2017 benchmark functions to evaluate seven DE algorithms and seventeen BCHMs, finding that DE algorithm efficiency depends on the BCHM. Juárez-Castillo et al. (2019) optimised sixty single-objective COPs using an adaptive BCH scheme in PSO and DE algorithms. Unless the population is entirely infeasible, the adaptive approach randomly selects a BCHM from a predefined set based on a dynamically updated probability. This adaptive scheme outperformed single BCHMs in COPs with moderate or large variables or constraints and in COPs where identifying feasible individuals is difficult. Arouri and Sayyafzadeh (2020) developed a gradient-based algorithm for production optimisation using simultaneous perturbation stochastic approximation (SPSA) and adaptive moment estimation. After testing the algorithm with projection and logarithmic transformation BCHMs, the former performed better.

In 2020, Biedrzycki et al. evaluated the covariance matrix adaptation evolution strategy (CMA-ES) algorithm for optimising unimodal and multimodal CEC2017 and Black-Box Optimisation Benchmarking (BBOB) problems using 22BCHMs categorised as repair, feasibility preserving, specialised, and penalty functions. Darwinian reflection and resampling performed best among BCHMs.

2.6 Other constraint handling techniques

In Evolution Strategies for optimising sphere and ellipsoid functions with linear constraints, Atamna et al. (2020) used an augmented Lagrangian approach as CHS to convert the initial constrained objective function into an unconstrained one and use its parameters during optimisation. In 2020, Qian et al. used a GA and surrogate model to solve 9 numerical and 2 engineering optimisation problems using the Kriging surrogate prediction model instead of the actual constraints, which is updated during optimisation to avoid infeasible solutions. It yields optimal feasible solutions and requires less computational power than other methods.
Rosso et al. (2021) introduced a non-penalty-based constraint handling approach for PSO algorithms using SVM, a supervised classification machine learning method. A simple bisection algorithm was implemented to preserve population feasibility while constraint handling with SVM appears more adaptive to nonlinear and discontinuous boundaries due to its generality. Rosso et al. (2022) developed Particle Swarm Optimisation (PSO) to solve constrained problems using a twist on the classical penalty function technique. Current implementation includes state-of-the-art improvements and suggestions (inertia weight, neighbourhood). To localise the feasible region in difficult optimisation problems, a new local search operator was added.This operator hybridises with the Evolutionary Strategy, another milestone meta-heuristic algorithm. The self-adaptive variant is used because it doesn't require tuning any other parameter.

Table 2: Comparative Performance of discussed CHS in literature review

Technique	Strength	Weakness	Best Suited For
Penalty-Based	Simple, adaptable to various problems	Requires parameter tuning	General-purpose problems
Feasibility Rules	Clear decision-making process	May bias towards feasibility prematurely	Problems with hard constraints
ε-Constrained	Balances exploration and exploitation	Parameter sensitivity	Dynamic or multi-modal problems
Stochastic Ranking	Avoids over-reliance on penalties	Computationally intensive	Complex constraint landscapes
Repair Algorithms	Directly addresses feasibility	Problem-specific	Engineering and structural optimization
Boundary-Based	Efficient for bound-constrained problems	Limited applicability	Simple bound-dominant problems
Hybrid and Ensemble Methods	Combines strengths of multiple techniques	High computational demand	Complex and multi-modal optimization

In given table 2 shows various CHSs comparative performance, Although penalty-based methods and feasibility rules are simple and easy to implement, ε-constrained and stochastic ranking techniques excel in dynamics and complexity. Hybrid methods combine the strengths of individual methods, offering promise. Repair and boundary-based methods provide domain-specific solutions for real-world applications.

3 CONSTRAINT HANDLING STRATEGY FORMULATION

This section discusses four commonly used CHSs in the literature: penalty methods, feasibility rules, ε-constrained method, and stochastic ranking. A brief description of ensemble-based methods is also provided. Additionally, Kawachi et al. (2019) introduced adaptive penalty, Deb (2000) feasibility

rules, Fan et al. (2018) improved ε-constrained, and Runarsson and Yao (2000) stochastic ranking innovations.

3.1 Penalty methods

A penalty term to the objective function makes a constrained problem unconstrained in penalty methods. Penalty factors and constraint function violations determine the penalty term, which can change during optimisation. General form of penalised objective function (Coello and Montes 2002):

$$F(x) = f(x) \pm \left[\sum_{j=1}^{J} r_j \cdot g_j(x) + \sum_{k=1}^{K} c_k \cdot h_k(x) \right], \qquad (2)$$

where $F(x)$ is the penalized objective function (also called fitness function), $g_j(x)$ and $h_k(x)$ are the inequality and equality constraints, respectively, and r_j and c_k are positive weight coefficients called penalty factors. Converting the equality into inequality constraints, as previously discussed in Sect. 1, Eq. (2) can be rewritten as:

$$F(x) = f(x) \pm \sum_{j=1}^{m} r_j \cdot g_j(x). \qquad (3)$$

Kawachi et al. (2019) proposed an adaptive penalty factor calculation method during evolution to avoid over- or under-penalization that could diverge the search process from the optimal. More specifically, a fitness function is created: where $\underline{v}(x)$ is the mean constraint violation and PF is the penalty factor, and subsequently three steps are implemented. First the penalty factor candidates (PFCs) are calculated by comparing two individuals as follows:

$$PFC_{k,l} = -\frac{f(x_k) - f(x_l)}{\overline{v}(x_k) - \overline{v}(x_l)}, \qquad (5)$$

where subscripts k and l denote two distinct individuals; PFC is calculated for all possible combinations in a given population. In the second step, the penalty factor (PF) is defined, as follows: if the percentage of negative $PFCs$ exceeds 50%, the PF remains the same with the one of the previous generations; otherwise PF is calculated as the average value of the positive $PFCs$. In the third step PF is updated. If the proportion of feasible individuals $r\,f$ in the population exceeds the p_{feas}, then the penalty factor of the next generation is obtained as follows in equation (6); where $p_{rate} \in [0, 1]$ and p_{feas} are user defined parameters.

$$PF_{G+1} = p_{rate} \cdot PF_G, \qquad (6)$$

3.2 Feasibility rules

In penalty methods, different penalty factor values must be tested to determine the best option because inappropriate values can diverge the search process from the optimal solution. Deb (2000) proposed a tournament selection operator-based method for comparing two solutions based on the following criteria: (i) any feasible solution is preferred to any infeasible solution; (ii) the feasible solution with the better objective function value is preferred; and (iii) the infeasible solution with the smaller constraint violation value is preferred.

3.2.1 Original feasibility rules technique

According to the original feasibility rules technique, the fitness function is defined as follows:

$$F(x) = \begin{cases} f(x) & \text{if } g_j(x) \le 0 \ \ \forall j = 1, 2, ..., m \\ f_{max} + \sum_{j=1}^{m} max\{0, g_j(x)\} & otherwise \end{cases}, \qquad (7)$$

where f_{max} is the objective function value for the worst feasible solution of the current population. If no feasible solution exists in a population, f_{max} is set to zero. According to this technique, all constraint

violations are summed, and this sum is compared as a single value. Therefore, in case of infeasible solutions, these are compared based solely on the level of constraint violation.

3.2.2 Innovative solution of the feasibility rules technique

This section presents four novel feasibility rules solutions. To explore the design space globally, the comparison rules for the original technique have been modified to bias the search towards better feasible solutions rather than just feasible ones. The proposed variants assume that a small constraint violation should not be the sole criterion for selecting one individual over another, as in existing techniques. In particular, when comparing two infeasible individuals, the objective function value, number of violated constraint functions, and maximum constraint violation value should be considered, and the product of these three entities will determine the winner: if an individual has a lower constraint violation than another, but its objective function value is much higher, maintaining the f. Naturally, the second individual will be penalised more if it violates constraint very strongly. The lower bound is the objective function value of the best feasible individual found so far to avoid very small values that could misdirect the search. The product of an individual's objective function value and constraint violation level can be used to choose either feasible or infeasible. Selecting the feasible solution may lead to a local optimum, so the infeasible individual should be preferred if its objective function value is very low. Finally, the total number of violated constraint functions and the maximum constraint violation are considered since it is unclear which is better between a solution with many small violations and one very large violation. After considering the above, the new variants are formulated. Check and verify. Confirm $P_{violation}$ factor is used to calculate infeasible individual fitness function. The individual's normalised maximum constraint violation is multiplied by a term related to the relative number of violated constraint functions for a solution over the total number of constraint functions, depending on the variant. Each of the four variants has a $P_{violation}$ factor:

$$p_{violation} = \left\| max\left\{ max\left\{0, \overline{g}_j(x)\right\}, \forall j = 1, 2 \ldots, m\right\}\right\| > 1, \quad (8a)$$

$$p_{violation} = \left\| max\left\{ max\left\{0, \overline{g}_j(x)\right\}, \forall j = 1, 2 \ldots, m\right\}\right\| \times \left(1 + \frac{n_{constviol}}{n_{const}}\right) > 1, \quad (8b)$$

$$p_{violation} = \left\| mean\left\{ \overline{g}_j(x), \forall j = 1, 2 \ldots, m | \overline{g}_j(x) > 0\right\}\right\| \times \left(1 + \frac{n_{constviol}}{n_{const}}\right) > 1, \quad (8c)$$

$$p_{violation} = \left\| median\left\{ \overline{g}_j(x), \forall j = 1, 2 \ldots, m | \overline{g}_j(x) > 0\right\}\right\| \times \left(1 + \frac{n_{constviol}}{n_{const}}\right) > 1, \quad (8d)$$

where $g_j(x)$ is the normalised value of the j^{th} constraint (an example of a normalised constraint is shown below in Eq. 9b), $n_{constviol}$ is the number of violated constraint functions and n_{const} is the total number of constraint functions. The fitness function of an individual is calculated as the product of $p_{violation}$ with the maximum among the objective function value of the individual and the best objective function value of the feasible individuals found so far, as stated in the following expression:

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.1, March : 2025

$$g_j(x) : \sigma(x) \leq \sigma_{all}$$
$$\overline{g}_j(x) : \frac{\sigma(x)}{\sigma_{all}} - 1 \leq 0 \quad (9a)$$

$$F(x) = \begin{cases} f(x) & \text{if } \overline{g}_j(x) \leq 0, \forall j = 1, 2, ..., m \\ max\left(f_{bestfeasible}, f(x)\right) \times p_{violation} & otherwise \end{cases}. \quad (9b)$$

where an example of a constraint function is shown in Eq. (9b), where $\sigma(x)$ is the principal stress developed into a specific structural element and σ_{all} is the corresponding allowable value. Three criteria are used to compare two solutions using expression (9b): (i) between two feasible solutions, the one with the better objective function value is preferred; (ii) between two infeasible solutions, the one with lower constraint violation and fewer violated constraint functions is preferred; and (iii) between one feasible and one infeasible solution, the feasible solution is selected based on its objective function value, constraint violation level, and number of violated constraints. In maximisation problems, criteria are adjusted.

3.3 ε-constrained method

A modified version of the feasibility rules technique was originally proposed by Takahama and Sakai (2005), with the aim of providing comparison rules on constrained optimization problems during the evolutionary process. According to this technique, the overall constraint violation function $\phi(x)$ is defined as either the maximum or the sum of all constraints, and is used in conjunction with the ε level comparison $\{< \varepsilon\}$; for any $\varepsilon \geq 0$, an individual (f_1, ϕ_1) is considered better than another individual (f_2, ϕ_2) according to the following comparison rules:

$$(f_1, \varphi_1) <_\varepsilon (f_2, \varphi_2) \iff \begin{cases} f_1 < f_2 \text{ if } \varphi_1, \varphi_2 \leq \varepsilon \\ f_1 < f_2 \quad \text{if } \varphi_1 = \varphi_2 \\ \varphi_1 < \varphi_2 \quad \text{otherwise} \end{cases}. \quad (10)$$

Therefore: (a) if both overall constraint violations $\phi(x)$ of two individuals are either less than or equal to ε, the one with better objective function value f is preferred; (b) If the overall constraint violations $\phi(x)$ of two individuals are equal, the one with better objective function value f is preferred; and (c) if either or both constraint violations $\phi(x)$ of two individuals are larger than ε, the one with lower constraint violation $\phi(x)$ is preferred. Building upon their previous work, Takahama and Sakai (2010) suggested that the ε level can be usually controlled according to the following expression:

$$\varepsilon(t) = \begin{cases} \varphi_\theta & \text{if } t = 0 \\ \varepsilon(0)\left(1 - \frac{t}{T_C}\right)^{c_p} & \text{if } 0 < t < T_C \\ 0 & \text{if } t \geq T_C \end{cases}, \quad (11)$$

where the initial ε level is equal to ϕ_θ , that is the constraint violation of the top θ^{th} individual in the initial population, t is the number of iterations, T_C is the control generation and c_p is a user defined parameter with value greater or equal to 3.

3.3.1 Improved ε-constrained method

In Fan et al. (2018) proposed the following improved ε setting approach, which applies the ε level comparison $\{< \varepsilon\}$ to establish comparison during the evolutionary process, and further balance the evolutionary search of the population between feasible and infeasible regions:

Industrial Engineering Journal
ISSN: 0970-2555
Volume : 54, Issue 3, No.1, March : 2025

$$\varepsilon(k) = \begin{cases} \varphi_\theta & \text{if } k = 0 \\ \varepsilon(k-1)\left(1 - \frac{FEs}{T_C}\right)^{c_p} & \text{if } r_k < \alpha \text{ and } FEs < T_C \\ (1+\tau)\varphi_{max} & \text{if } r_k \geq \alpha \text{ and } FEs < T_C \\ 0 & \text{if } FEs \geq T_C \end{cases}, \quad (12)$$

where r_k is the proportion of feasible solutions (PFS) in the generation; ϕ_θ is the θ^{th} largest overall constraint violation of all individuals in the initial population (sorted in an array by their overall constraint violation), where $\theta = \gamma^* N_{pop}$, N_{pop} being the population size and $\gamma \in [0.2, 0.8]$; ϕ_{max} is the largest overall constraint violation of all individuals; T_C is the termination function evaluations (FEs), where $T_C \in [0.1maxFEs, 0.8maxFEs]$; and $c_p \in [2, 10]$, $\tau \in [0, 1]$ and $\alpha \in [0, 1]$ are user-defined parameters.

3.3.2 New variants of the ε-Constrained method

Analogously to the new feasibility rules variants introduced previously in Sect. 3.2.2, new variants of the ε-constrained technique are also proposed herein. The ε level comparison $\{<\varepsilon\}$ in the new variants for two individuals 1, 2 is defined via the following four rules:

$$(f_1, p_1) <_\varepsilon (f_2, p_2) \Leftrightarrow \begin{cases} f_1 < f_2 & \text{if } p_1, p_2 \leq \varepsilon \\ f_1 < f_2 & \text{if } p_1 = p_2 \\ max(f_{bf}, f_1) \times p_1 < max(f_{bf}, f_2) \times p_2 & \text{if } p_1, p_2 \geq \varepsilon \\ f_1 < max(f_{bf}, f_2) \times p_2 & \text{otherwise} \end{cases}, \quad (13)$$

where f_{bf} stands for the objective function value of the best feasible individual found so far, and $p1$, $p2$ denote the value of the $p_{violation}$ factor calculated for individuals 1 and 2, based upon Eqs. (8a) to (8d). If two individuals have maximum constraint violations less than or equal to ε level, the one with better objective function value f is preferred. If they are equal, the one with better objective function value f is preferred. If both are greater than ε level, the one with better product of constraint violations is preferred. New feasibility rules techniques consider the maximum constraint violation value and the total number of violated constraint functions. Eq. (12) is used to control the ε level, following Fan et al. (2018) formulation.

3.4 Stochastic ranking method

Stochastic ranking, proposed by Runarsson and Yao (2000), is a popular CHS that avoids penalty techniques. Comparing adjacent individuals in a population by penalty and objective function dominance ranks them from best to worst in stochastic ranking. Designers must define a probability $P_f \in [0, 1]$ for using the objective function exclusively for adjacent comparisons to improve performance. $P_f = 1/2$ means the objective function and penalty function comparison probabilities are equal. A bubble-sort algorithm ranks individuals by comparing adjacent individuals in N sweeps (the least λ sweeps) in each generation. If both adjacent individuals are feasible or a randomly generated number u is less than the selected probability P_f, their objective function values are compared. If one or both are infeasible, the constraint violation value is used. If the ranking order does not change within a sweep, the algorithm stops.

3.5 Ensemble of constraint handling techniques

It can be difficult to determine which CHS is best for COP global optimisation. A CHS may be better for one stage of evolution but not another, so different CHSs may be more effective at different stages of the search. This depends on the problem's multimodality, the search region's feasibility, the algorithm, etc. Mallipeddi and Suganthan (2010) proposed ECHS, a scheme combining four CHSs (feasibility rules, self-adaptive penalty, ε-constraint method, and stochastic ranking) to explore constrained optimisation problems, each focussing on a distinct population. The population's parents and offspring compete with each other and all other populations. This means that if an offspring

from a population is not selected for the next generation based on its CHS, other populations that use a different CHS may select it.

4. CONCLUSIONS

A comprehensive state-of-the-art review of almost 60 CHS variants for MOAs to solve multi-objective COPs is presented in this paper. Metaheuristic optimisation algorithms (MOAs) for constrained optimisation problems (COPs) require constraint handling techniques (CHTs). This review extensively examined CHT methods, benefits, drawbacks, and applications. Techniques like penalty-based, feasibility rules, ε-constrained, repair-based, and boundary-based strategies were examined, along with hybrid and ensemble approaches. Due to their simplicity and adaptability, penalty-based methods are still popular. Dynamic penalty adjustments, like those by Kawachi et al. (2019), reduce over-penalization and improve convergence. Introduced by Deb (2000), feasibility rules and improved ε-constrained methods by Fan et al. (2018) eliminate penalty factor tuning. They optimise search efficiency by prioritising constraint violation and objective performance. Mallipeddi and Suganthan (2010)'s hybrid and ensemble approaches use multiple CHTs to dynamically adapt to problem characteristics, improving performance in diverse scenarios. Repair algorithms accurately turn infeasible engineering solutions into feasible ones. Boundary-based methods efficiently handle variable limits and incorporate probabilistic adjustments for robustness. However, these methods have limitations. Due to parameter tuning or integration complexities, penalty-based and hybrid methods have high computational overhead. Scalability is another issue because some CHTs are less effective in high-dimensional or constrained problem spaces. No universal CHT guarantees optimal performance across diverse applications, so a problem-specific approach is needed. CHT selection should match the problem and computational resources for effective application. Exploration and exploitation are best balanced by adaptive and ensemble methods. Dynamic updates improve adaptability of penalty methods, which are reliable for well-defined constraints. In strict feasibility scenarios like structural design optimisations, repair algorithms should be considered. Ensemble approaches that combine adaptive penalty methods, feasibility rules, and repair-based techniques are best for robust constraint handling in MOAs, according to the review. These methods use CHT strengths and mitigate weaknesses to adapt to the problem landscape. Ensembles provide versatile and effective COP solutions across domains by combining adaptability, robustness, and computational efficiency. This comprehensive CHT synthesis offers insights and actionable recommendations to help practitioners solve complex optimisation problems.

Table 2: List of Abbreviations used in proposed article

Abbreviation	**Full Form**
COP	Constraint optimization problem
NLP	Non-linear programming
CHS	Constraint handling technique
EA	Evolutionary algorithm
DE	Differential evolution
ORPD	Optimal reactive power dispatch
SA-DECV	Surrogate-assisted differential evolution with combined variants
DCOPs	Dynamic constrained optimization problems
PBA	Pity beetle algorithm
OCP	Optimization computing platform
HP-OCP	High-performance optimization computing platform
APM	Adaptive penalty method
ADP	Automatic dynamic penalization
DSBO	Dynamic surrogate-based optimization
EI	Expected improvement
CHIP	Constraint handling with individual penalty

OPF	Optimal power flow
SR	Stochastic ranking
NPGA	Niched-pareto genetic algorithm
CVI	Constraint violation with interval arithmetic
BSA	Backtracking search algorithm
FOA	Fruit fly optimization algorithm
SAR	Search and Rescue optimization algorithm
WDOCHM-PSO	PSO-based algorithm combined with dynamic objective constraint handling method
DFIG	Doubly fed induction generator
SCE	Shuffled complex evolution
CCE	Competitive complex evolution
EP	Evolutionary programming
FS	Feasible solutions
SPSA	Simultaneous perturbation stochastic approximation
CMA-ES	Covariance matrix adaptation evolution strategy
MOA	Metaheuristic optimization algorithm
RST	Random sampling technique

References

[1] Rosso MM, Cucuzza R, Aloisio A, Marano GC (2022). Enhanced multi-strategy particle swarm optimization for constrained problems with an evolutionary-strategies-based unfeasible local search operator. *Applied Sciences*, 12(5):2285. https://doi.org/10.3390/app12052285

[2] Lagaros ND, Plevris V, Kallioras NA (2022). The mosaic of metaheuristic algorithms in structural optimization. *Archives of Computational Methods in Engineering*, 29:5457–5492. https://doi.org/10.1007/s11831-022-09773-0

[3] Cucuzza R, Rosso MM, Aloisio A, Melchiorre J, Giudice M, Marano GC (2022). Size and shape optimization of a guyed mast structure under wind, ice, and seismic loading. *Applied Sciences*, 12(10):4875. https://doi.org/10.3390/app12104875

[4] Biedrzycki R (2020). Handling bound constraints in CMA-ES: An experimental study. *Swarm and Evolutionary Computation*, 52:100627.

[5] Arouri Y, Sayyafzadeh M (2020). Accelerated gradient algorithm for well control optimization. *Journal of Petroleum Science and Engineering*, 190:106872.

[6] Atamna A, Auger A, Hansen N (2020). Invariance and linear convergence of evolution strategies with augmented Lagrangian constraint handling. *Theoretical Computer Science*, 832:68–97.

[7] Stanovov V, Akhmedova S, Semenkin E (2020). Combined fitness-violation epsilon constraint handling for differential evolution. *Soft Computing*, 24(10):7063–7079.

[8] Shabani A, Asgarian B, Salido M, Asil Gharebaghi S (2020). Search and rescue optimization algorithm: A novel method for solving constrained engineering optimization problems. *Expert Systems with Applications*, 161:113698.

[9] Kaucic M, Barbini F, Camerota Verdù FJ (2020). Polynomial goal programming and particle swarm optimization for enhanced indexation. *Soft Computing*, 24(12):8535–8551.

[10] Tsipianitis A, Tsompanakis Y (2020). Improved cuckoo search algorithmic variants for constrained nonlinear optimization. *Advances in Engineering Software*, 149:102865.

[11] Javed H, Jan MA, Tairan N, Mashwani WK, Khanum RA, Sulaiman M, Khan HU, Shah H (2019). Efficacy of an ensemble of constraint handling techniques in self-adaptive differential evolution. *Mathematics*, 7(7):635.

[12] Mao J-Q, Tian M-M, Hu T-F, Ji K, Dai L-Q, Dai H-C (2019). Shuffled complex evolution coupled with stochastic ranking for reservoir scheduling problems. *Water Science and Engineering*, 12(4):307–318.

[13] Li S, Gong W, Wang L, Yan X, Hu C (2019). Optimal power flow using improved adaptive differential evolution. *Energy*, 198:117314.

[14] Caraffini F, Kononova AV, Corne D (2019). Infeasibility and structural bias in differential evolution. *Information Sciences*, 496:161–179.

[15] Lin X, Luo W, Qiao Y, Xu P, Zhu T (2019). Empirical study of population-based dynamic constrained multimodal optimization algorithms. In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 722–730.

[16] Juárez-Castillo E, Acosta-Mesa HG, Mezura-Montes E (2019). Adaptive boundary constraint-handling scheme for constrained optimization. *Soft Computing*, 23(17):8247–8280.

[17] Datta R, Deb K, Kim J-H (2019). CHIP: Constraint handling with individual penalty using a hybrid evolutionary algorithm. *Neural Computing and Applications*, 31(9):5255–5271.

[18] Rodrigues MDC, Guimarães S, de Lima BSLP (2018). E-BRM: A strategy for evolutionary optimization under constraints. *Applied Soft Computing*, 72:14–29.

[19] Fan Z, Fang Y, Li W, Yuan Y, Wang Z, Bian X (2018). LSHADE44 with improved ε-constraint handling for constrained optimization problems. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 8477943.

[20] Biswas PP, Suganthan PN, Mallipeddi R, Amaratunga GAJ (2018). Differential evolution with effective constraint handling for power flow optimization. *Engineering Applications of Artificial Intelligence*, 68:81–100.

[21] Miranda-Varela ME, Mezura-Montes E (2018). An empirical study of constraint-handling techniques in surrogate-assisted evolutionary optimization. *Applied Soft Computing*, 73:215–229.

[22] Ameca-Alducin MY, Hasani-Shoreh M, Blaikie W, Neumann F, Mezura-Montes E (2018). A comparative analysis of constraint handling techniques for dynamic constrained optimization problems. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 8477750.

[23] Trivedi A, Biswas N, Chakroborty S, Srinivasan D (2018). Extending unified differential evolution with an ensemble of constraint-handling techniques. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8.

[24] Peng C, Liu HL, Gu F (2018). A novel constraint-handling method using dynamic weights for constrained optimization. *Soft Computing*, 22(12):3919–3935.

[25] Gandomi AH, Kashani AR (2018). Probabilistic evolutionary boundary constraint handling for particle swarm optimization. *Operations Research*, 18(3):801–823.

[26] Montemurro M, Vincenti A, Vannucci P (2013). Automatic dynamic penalization method (ADP) for handling constraints in genetic algorithms. *Computational Methods in Applied Mechanics and Engineering*, 256:70–87.

[27] Mazhoud I, Hadj-Hamou K, Bigeon J, Joyeux P (2013). Constraint-handling in particle swarm optimization for solving engineering problems. *Engineering Applications of Artificial Intelligence*, 26(4):1263–1273.

[28] Wang Y, Cai Z, Zhou Y, Fan Z (2009). Constrained optimization using a hybrid evolutionary algorithm with adaptive constraint-handling. *Structural and Multidisciplinary Optimization*, 37(4):395–413.

[29] Salcedo-Sanz S (2009). A survey of repair methods for constraint handling in evolutionary algorithms. *Computer Science Review*, 3(3):175–192.

[30] Farshi B, Alinia-Ziazi A (2010). Sizing optimization of truss structures using the method of centers and force formulation. *International Journal of Solids and Structures*, 47(18–19):2508–2524.

[31] Runarsson TP, Yao X (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294.

[32] Deb K (2000). Efficient constraint-handling in genetic algorithms. *Computational Methods in Applied Mechanics and Engineering*, 186(2–4):311–338.

[33] Malan KM (2018). Landscape-aware constraint handling applied to differential evolution. *Lecture Notes in Computer Science*, 11324 LNCS, pp. 176–187.

[34] Mallipeddi R, Suganthan PN, Jeyadevi S, Baskar S (2012). Efficient constraint handling for optimal reactive power dispatch problems. *Swarm and Evolutionary Computation*, 5:28–36.

[35] da Silva EK, Barbosa HJC, Lemonge ACC (2011). Adaptive constraint handling in differential evolution with dynamic variant usage for engineering optimization. *Optimization Engineering*, 12(1–2):31–54.

[36] Takahama T, Sakai S (2010). Constrained optimization using ε-constrained differential evolution with gradient-based mutation. In: *2010 IEEE World Congress on Computational Intelligence (WCCI)*, pp. 5586484.

[37] Mezura-Montes E, Coello CAC (2011). Past, present, and future of constraint-handling in nature-inspired numerical optimization. *Swarm and Evolutionary Computation*, 1(4):173–194.

[38] Takahama T, Sakai S (2005). ε-constrained optimization by particle swarm optimization with ε-level control. In: *Advances in Soft Computing*, pp. 1019–1029.

[39] Chehouri A, Younes R, Perron J, Ilinca A (2016). A genetic algorithm constraint-handling technique based on violation factors. *Journal of Computational Science*, 12(7):350–362.

[40] Rodrigues LL, Sebastian Solis-Chaves J, Vilcanqui OAC, Filho AJS (2020). Predictive incremental vector control for DFIG optimization using dynamic objective constraint-handling PSO methods. *IEEE Access*, 8:114112–114122.

[41] Jordehi AR (2015). A comprehensive review of constraint-handling strategies in particle swarm optimization. *Neural Computing and Applications*, 26(6):1265–1275.

[42] Coello CAC, Montes EM (2002). Dominance-based tournament selection for constraint-handling in genetic algorithms. *Advanced Engineering Informatics*, 16(3):193–203.

[43] Chootinan P, Chen A (2006). Genetic algorithms with a gradient-based repair mechanism for constraints. *Computers & Operations Research*, 33(8):2263–2281.

[44] Lagaros ND (2014). A generalized computing platform for structural optimization. *Structural and Multidisciplinary Optimization*, 49:1047–1066.

[45] Miettinen K, Mäkelä MM, Toivanen J (2003). Comparative numerical study of penalty-based constraint-handling methods in genetic algorithms. *Journal of Global Optimization*, 27(4):427–446.

[46] Miranda-Varela ME, Mezura-Montes E (2018). Empirical studies on surrogate-assisted evolutionary optimization constraint-handling techniques. *Applied Soft Computing*, 73:215–229.

[47] Gandomi AH, Yang XS (2012). Evolutionary boundary constraint-handling mechanisms. *Neural Computing and Applications*, 21(6):1449–1462.

[48] Wang S, Kang J, Tasgetiren MF, Gao L, Kizilay D (2019). Differential evolution enhanced with variable neighborhood search for real-parameter optimization problems. In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 2252–2260.

[49] Mozaffari A, Emami M, Fathi A (2019). Chaos-enhanced evolutionary algorithms with boundary constraints: performance, robustness, and scalability analysis. *Artificial Intelligence Review*, 52(4):2319–2380.

[50] Rodrigues MDC, Guimarães S, de Lima BSLP (2018). Ensemble constraint handling for optimization problems using evolutionary algorithms. *Applied Soft Computing*, 72:14–29.

[51] Javed H, Jan MA, Tairan N, Mashwani WK, Khanum RA, Sulaiman M, Khan HU, Shah H (2019). Efficiency of constraint handling techniques in self-adaptive differential evolution. *Mathematics*, 7(7):635.

[52] Li X, Xiao C, Lu Z (2018). A constraint handling method for economic dispatch problems. *Journal of Electrical Engineering and Technology*, 13(3):1099–1109.

[53] Fan Z, Fang Y, Li W, Yuan Y, Wang Z, Bian X (2018). LSHADE44 integrated with improved ε-constraint handling for optimization. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 8477943.

[54] Biswas PP, Suganthan PN, Mallipeddi R, Amaratunga GAJ (2018). Differential evolution with robust constraint handling for optimal power flow. *Engineering Applications of Artificial Intelligence*, 68:81–100.

[55] Ameca-Alducin MY, Hasani-Shoreh M, Blaikie W, Neumann F, Mezura-Montes E (2018). Comparative analysis of constraint handling for dynamic optimization problems. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*, pp. 8477750.

[56] Trivedi A, Biswas N, Chakroborty S, Srinivasan D (2018). Unified differential evolution extended with ensemble constraint handling. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8.

[57] Mezura-Montes E, Coello CAC (2011). Review on constraint handling in numerical optimization. *Swarm and Evolutionary Computation*, 1(4):173–194.

[58] Montemurro M, Vincenti A, Vannucci P (2013). Automatic dynamic penalization for genetic algorithms in constrained problems. *Computational Methods in Applied Mechanics and Engineering*, 256:70–87.

[59] Gandomi AH, Kashani AR (2018). Probabilistic evolutionary boundary handling for PSO optimization. *Operations Research*, 18(3):801–823.

[60] Takahama T, Sakai S (2010). Constrained optimization by ε-constrained PSO with ε-level control. In: *Advances in Soft Computing*, pp. 1019–1029.

[61] Deb K (2000). A novel constraint handling method for genetic algorithms. *Computational Methods in Applied Mechanics and Engineering*, 186(2–4):311–338.