Framework of Ensemble Parmeter Adapted Evolutionary Algorithm for Solving Constrained Optimization Problems

Debanjan Saha,* Karam M. Sallam† Shuvodeep De‡ Ali W. Mohamed§
†College of Engineering, Northeastern University, Boston, MA, USA saha.deb@northeastern.edu
†Faculty of Science and Technology, University of Canberra, Australia karam.sallam@canberra.edu.au
‡Postdoctoral Researcher, University of Alabama shuvode@vt.edu; sde@ua.edu
§Operations Research, Faculty of Statistical Research, Cairo University, Egypt aliwagdy@gmail.com

***Abstract*—Real-world optimization problems are often governed by one or more constraints. Over the last few decades, extensive research has been performed in Constrained Optimization Problems (COPs) fueled by advances in computational intelligence. In particular, Evolutionary Algorithms (EAs) are a preferred tool for practitioners for solving these COPs within practicable time limits. We propose an ensemble of multi-method hybrid EA framework with four mutation operators, two crossover operators, multi-search [Differential Evolution (DE) & Gaining Sharing Knowledge (GSK)] optimization algorithm, and ensemble of constraint handling techniques to solve global real-world constrained optimization problem. The proposed framework FEPEA has an ascendancy of multiple adaptation strategies concerning the control parameters, search mechanisms, two sub-populations as well as uses knowledge sharing mechanism between junior and senior phases. The algorithm also combines the power of four popular constraint handling techniques (CHT) and uses a voting mechanism to select any particular CHT. On top of that, this algorithm also uses both linear and non-linear population size reduction in every step of the evolutionary process. We test our method on 57 real-world problems provided as part of the CEC 2020 special session & competition on real-world constrained optimization benchmark suite. Experimental results indicate that FEPEA is able to achieve state-of-the-art performance on real-world constrained global optimization when compared against other well-known real-world constrained optimizers.**

***Index Terms*—constrained optimization; multi-operator; multi-parameter adaptation; nsemble constraint handling techniques; Evolutionary Algorithms**

I. Introduction

Constrained Optimization Problems (COP) arise in numerous fields, including natural sciences, economics, computers, and engineering design. Constraints are of two types: equality and inequality and while minimizing/maximizing the objective function, all relevant constraints must be satisfied. A COP can be depicted using Eq (7). These constraints are utilized to discover feasible regions, which might be an ample continuous search space, a significant, unpredictable space, a small space, or various disconnected regions. Based on the type of constraints and objective function, COP can be classified into different sub-categories like linear or non-linear, continuous or discontinuous, uni-modal or multi-modal. Of course, a certain COP can belong to more than one category.

Computational Intelligence (CI) based-techniques, like Evolutionary Algorithms (EA), are broadly employed to solve COP, because they enjoy many fundamental upper hands over customary numerical programming strategies [1]–[18], [18]–[21] . Nonetheless, there is no assurance that they will find ideal (global) solutions, and the nature of their solutions depends on the topography of the problem, the set of constraints, and its parametric settings. Among existing evolutionary algorithms, one of the most popular is Differential Evolution (DE) which was proposed by Storm and Price in 1996. This population-based method uses strategies consisting of mutation, crossover, and selection to arrive at new solutions and propel them toward the optimal solution. Since 1996, several methods for mutation, crossover and selection have been proposed resulting in modified versions of the algorithm. DE being simple and easy to implement yet powerful found applications in several fields and has acquired prominence for taking care of issues in continuous areas, and has demonstrated its predominance over other notable algorithms for taking care of complex optimization issues with various properties [22]. Notwithstanding, no single version of the DE algorithm (or EA) performs the best for a wide range of test cases. This roused researchers to present variants that utilizes the qualities of various operators.

A Multi-operator DE (MODE) algorithm leverages the power of more than one DE operator, for instance, mutation operator, crossover operator, with a preference towards the better performing solutions based on objective function evaluation value and mean constraint violation value during the evolutionary process. It also progressively refreshes each sub-population size based on two major markers: the nature of solutions and the variety of each sub-population. The proposed hybrid algorithm framework constitutes the division of population between a Multi-Operator DE (MODE) [23] and an Adaptive Parameter Gaining Sharing Knowledge (APGSK) [24]. Later in the evolutionary process, it recombines these sub-populations to form the current generation population. The contributions of this paper includes a hybrid framework consisting of multiple EA (here we have used two search optimization algorithms but it generalizes to multiple evolutionary algorithms), a weighted mutation operator for the top ϕbest solutions in the sub-population assigned to MODE which

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0307.v1

explores promising search spaces in close proximity to the best solutions, parametric adaptation of various control parameters like crossover operator, knowledge parameter and a voting mechanism based selection of ensemble constraint handling techniques.

Towards the end of each generation, both the sub-populations are assembled together to form one whole population and successively reallocated in the subsequent generation based on the success rate, functional evaluation value and mean constraint violation. In addition, a linear decrease of the population size is performed in the sub-population allocated to MODE as standard LSHADE literature [25]. However, we use a nonlinear shrinkage of the population size in the sub-population allocated to APGSK [24]. The underlying constraints are handled using voting mechanism which probabilistically selects between four most popular constraint handling techniques namely: Superiority of Feasible solutions (SF), Stochastic Ranking (SR), ϵ-Constrained method (ϵC) and Self-Adaptive Penalty method (SP). The proposed algorithmic framework is extensively tested by tackling 57 real-world constrained issues taken from the CEC2020 [26] competition on non-convex constrained optimization issues from the real world. The outcomes show that this algorithm measurably outperforms existing state-of-the-art constrained optimization algorithms.

The rest of this paper is structured as follows: a brief literature survey on a simple DE, various multi-operator DE, and various most widely used constraint handling techniques has been discussed in Section II; the proposed algorithm framework and subtleties of its parts are discussed in Section III; the benchmark, algorithm hyper-parameters, experimental results and examinations are mentioned in Section IV; and, at last, the conclusion and scope for future work is presented in Section V. Finally, more avid users are elluded to check out the Appendix section for more detailed guide on accurate experimental results.

II. Literature Review

A. Differential Evolution (DE) Algorithm

DE is a straightforward yet highly efficient population based Evolutionary Algorithm (EA) proposed by Price and Storn [27], [28] in 1996 for real parameter optimization. DE is not the same as conventional EAs in that it perturbs the current generation population wherein it undergoes an evolutionary process, evolving the current population using various operations like mutation, crossover and selection. Over the last couple of decades, DE has been applied to a plethora of real-world constrained optimization problems as well as numerical optimization problems because of its straightforwardness, simplicity in implementation and robust nature. Intrigued perusers are alluded to [29], [30] for a thorough survey on DE.

A DE algorithm generally starts by generating an initial population vector $\vec{X}_i = [x_{i,1}, x_{i,2}, \ldots, x_{i,D}]^T$, where D refers to the dimensionality of the problem and every solution segment, $x_i, i = 1, 2, \ldots, D$, is a real number. From this time forward, the terms *solution* and *vector* will be utilized reciprocally. This population is randomly initialized between a lower and upper bound ($x_{j,min}$ and $x_{j,max}$) given by Eq (1)

$$x_{i,j} = x_{j,min} + \text{rand}_{i,j}[0,1] \times (x_{j,max} - x_{j,min}) \quad (1)$$

where $i = 1, 2, \ldots, NP$ and $j = 1, 2, \ldots, D$. A simple DE algorithm comprises of three major steps: mutation, cross-over and selection.

1) Mutation: This is the first step in a simple DE (DE/rand/1) evolutionary process where it tries to create a mutant vector $\vec{V}_i$ by randomly selecting three candidate solutions from the current population. It multiplies the scaling factor (F) with the difference between two of the candidates and adds it to the remaining candidate to generate a *trail mutant* vector given by Eq (2)

$$\vec{V}_i = \vec{X}_{r_1^i} + F \times (\vec{X}_{r_2^i} - \vec{X}_{r_3^i}) \quad (2)$$

where $r_1^i \neq r_2^i \neq r_3^i$ and $r_1^i, r_1^i, r_1^i \in [1, NP]$ and control parameter $F > 0$

2) Crossover: Once a mutant vector is generated, a crossover operation is performed to generate an offspring solution. In a simple DE setup, two main types of crossover operations are binomial and exponential crossover. In binomial crossover is applied when a random number generated is less than the control parameter Crossover Rate (Cr) given by Eq (3)

$$U_{i,j} = \begin{cases} V_{i,j} & \text{if } (rand \leq Cr_i \text{ or } j = j_{rand}) \\ X_{i.j} & \text{otherwise} \end{cases} \quad (3)$$

In exponential crossover, an integer is randomly selected $c \in [1, D]$ in the search space. This acts as the lower bound of the decision space in which another integer is selected $C \in [c, D]$. This represents the number of elements where the donor or parent vector actually participates in offspring generation. Once, these two parameters are selected the offspring is generated using Eq (4)

$$U_{i,j} = \begin{cases} V_{i,j} & \text{for } j = \langle c \rangle_D, \langle c+1 \rangle, \ldots, \langle c+C+1 \rangle_D \\ X_{i.j} & \text{for all other } j \in [\ 1, D] \end{cases} \quad (4)$$

3) Selection: Once the offspring is generated, the final step is to perform a greedy based selection as to whether the parent or the offspring solution should survive till the next generation. Most of the times, this decision is taken based on the objective function value of the parent and offspring also known as *fitness* value given by Eq (5).

$$\vec{X}_{i,G+1} = \begin{cases} \vec{U}_{i,G} & \text{if } f(\vec{U}_{i,G}) \leq f(\vec{X}_{i,G}) \\ \vec{X}_{i.j} & \text{otherwise} \end{cases} \quad (5)$$

where $f(\vec{U}_{i,G})$ and $f(\vec{X}_{i,G})$ are objective function evaluation or fitness values of the parent and offspring solutions, and G is the current generation of the population.

The performance of a DE algorithm heavily relies on the choice of control parameters, namely - scaling factor (F) and crossover rate (Cr). Liu et. al [31] proposed a fuzzy adaptive DE (FADE) with an adaptation in the mutation and crossover operators. This variant performed better than a standard DE. It was quite evidently noted that a single value for the control

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0307.v1

parameters was just not enough hence, researchers opted for self-adaptive versions of DE. [32], [33] One such variant called JADE proposed by Zhang et al. [34] introduced an external archive and sampling the scaling factor (μF) from a normal distribution within [0,1] and the crossover factor (μCR) from a cauchy distribution between [0,1]. Subsequently, Tanabe et al. [35] proposed SHADE where a memory based archive of scaling factor (S_F) and crossover rate (S_{CR}) were used instead of a single μF and μCR as in JADE. Later Tanabe improved his algorithm by adding a dynamic linear population size reduction (LSPR) during the evolutionry phase called LSHADE [25].

B. Multi Operator DE (MODE) Algorithm

Following the success of various control parameter adaptations in DE, researchers started looking out for possibilities of using multiple mutation and crossover operators rather than any single operator. Das et al. [36] proposed a neighbourhood-based mutation operator (DE/target-to-best/1). Wang et al. [37] proposed an orthogonal crossover (OXDE) to counteract the limitations of binomial and exponential crossovers. Guo et al [38] introduced eigenvector-based crossover operator which helped crossovers to be rotationally invariant. Elsayed et al. [39] proposed a self-adaptive multi-operator DE (SAMO-DE) which was designed to tackle constraint optimization issues by assigning different mutation operators to sub-populations. These sub-populations were assigned a success rate (SR) based on offspring quality determined by feasibility and constraint violations. Zamuda et al, [40] proposed jDE with two mutation operators and a population reduction based on number of functional evaluation. Wang et al. [41] proposed composite DE (CoDE) which used three mutation operators alongwith three fixed control parameters to generate three trail solutions, wherein the one with the best fitness values survived to the next generation. Following the success of these MODE algorithms Qin et al. [33] proposed self-adaptive DE (SADE) wherein the probability of operator as well as control parameter was dynamically updated based on the success rate. Individual archives were maintained for solutions which survived till the next generation as well as the ones which were removed from subsequent generations. The probability of selection of each mutation operator was adjusted as the number of solutions which survived till the next round over all the solutions which participated in the evolutionary process for the particular mutation operator.

Another DE variant proposed by Da Silva et al. [42] for unconstrained optimization used four mutation strategies and a technique for operator selection which was slightly biased towards the best individual in the current population. Mallipeddi et al. [43] proposed an ensemble of mutation strategies and control parameters with DE (EPSDE) which used a pool of mutation strategies (JADE [34] and DE/current-to-rand/1) and crossover operators (binomial and exponential), with control parameters $F \in (0.5, 0.9)$ and $Cr \in (0.1, 0.5, 0.9)$. Mallipeddi later proposed another parameter ensemble DE which also used Harmony Search [44] and two popular mutation operators (DE/current-to-pbest/bin [34] and DE/current-to-gr_best/bin [45]). This self-adaptable MODE was furthermore improved by Elsayed et al. [46] with organizing a co-change structure change evolution system. Cui et al. [47] proposed MPADE where the population was divided into three sub-populations and three mutation operators (DE/current-to-rbest/2, DE/current-to-nbest/2, and DE/current-to-pbest/2) were applied on each sub-population. Similarly, Wu et al. [48] proposed Multi-Population Ensemble DE (MPEDE) which used three mutation strategies (DE/current-to-rand/1, DE/current-to-pbest/1 and DE/rand/1) on three equally sized populations and a larger reward sub-population. After a determined number of fitness evaluations the best mutation strategy is awarded the reward sub-population.

Liu et al. [49] recently proposed Historical and Heuristic-Based Adaptive Differential Evolution (HHDE) which divides the population into three states: Superior (S), Medium (M) and Inferior (I) and uses past historical heuristic information of each individual based on fitness value to select from three commonly used mutation strategies (DE/current-to-rand/1, DE/current-to-pbest/1 and DE/rand/1). The top one-third population (S) are in close proximity to the best solution are assigned for exploration of promising areas using DE/current-to-rand/1. The middle one-third population (M) has the most scope of learning from better solutions (S) are assigned DE/current-to-pbest/1 for exploitation. The bottom one-third (I) can learn from good solutions (M) to obtain better solutions than I and are assigned exploration strategy of DE/rand/1. In addition it also uses multiple parameter adaptations for other control parameters as well.

Considering the conspicuous need of having a grounded system to choose best-performing evolutionary algorithm operators, Sallam et al. [50] proposed a landscape-based marker to pick the best-appropriate operator all through the evolutionary cycle of DE. Recently, Kumar et al. [51] proposed an upgraded rendition of the MODE algorithm, which applied a unique population size decrease method to diminish the size of the population for any ensuing cycles. Subsequent to tackling numerous CEC19 test issues, they have demonstrated the matchless quality of their proposed algorithm. To enhance the search capability of DE, Elsayed et al. [52] proposed another component to consequently choosing the best-fitted operators, for example, intensification factor, crossover rate, and population size. Diverse constrained optimization test issues were additionally utilized to survey their algorithm. Sallam et al. [53] proposed a ensemble multi-operator DE (EnMODE) that uses a probabilistic selection amongst three mutation operators and two crossover operators (binomial and exponential). EnMODE outperformed other state-of-the-art algorithms on the 57 problems presented as part of the CEC 2020 Real World Constrained Optimization benchmark suite. More recently, Biswas et al. [54] proposed a variant of MODE with adaptive hyperparameter optimization.

Ali et al. [55] proposed a nature-inspired gaining sharing knowledge (GSK) based algorithm wherein heuristic information is acquired and shared in two phases namely, the *junior* and the *senior* phase. The analogy used in this algorithm is same as how a human being acquires knowledge in his early phase of schooling termed as *juniors*. Later on, as the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0307.v1

human being becomes more proficient in their understanding of the concepts, they are transformed into *seniors*. These individuals then impart their knowledge with the then juniors in order to enhance their understanding. Initially, all the solutions are juniors which interact with other solutions during the evolutionary cycle. The algorithm uses two main control parameters - the knowledge ratio (k_r) and the knowledge factor (k_f). Consequently, Ali proposed an adaptive parameter version of the GSK (APGSK) [56] where the knowledge rate control parameter is dynamically adapted. In addition, a non-linear population size reduction is also introduced with this algorithm. The itemized depiction of the algorithm can be found in Algorithm 2. Most recently, Ali proposed another hybrid algorithm [57] for solving unconstrained problems whereby the population was divided into two parts and assigned to IMODE [23] and APGSK [56]. This algorithm outperformed all other state-of-the-art single-objective numerical optimization algorithm on the CEC 2021 benchmark suite.

C. Constrained Handling Technique

Aside from the central optimization engine, constraint handling strategy is the most significant part of the constrained optimization solver. According to a survey [58], [59], few of the most common constraint handling techniques due to their popularity in recent years like:

- Superiority of Feasible solutions (SF)
- Stochastic Ranking (SR)
- $epsilon$-Constrained methods (ϵC)
- Self-Adaptive Penalty methods (SP)
- Novel Special Operators
- Multi-Objective concepts (MO)
- Ensemble of Constrained-Handling Techniques (ECHTs)

1) Superiority of Feasible solutions (SF): Superiority of feasible solutions, as the name goes tend to favor feasible solutions over the infeasible. However, it also faces some limitations in very discreet and insatiable constraints search spaces. This technique introduced by Deb [60] has three situations:

- when the trial solution is feasible but the target solution is infeasible
- when both trial and target are both infeasible solutions
- when both trial and target are feasible solutions

Thus, it can be noted from SF that a feasible solution is always preferred over an infeasible solution. However, when both the trial and target tend to be infeasible, the one with the lower sum of constraint violations (ψ) is picked, where ψ is figured utilizing Eq. (6). Also, when both trial and target is feasible, the one with the least fitness value is preferred.

$$\psi(\vec{X}_i) = \sum_{k=1}^{K} \max\left(0, g_k(\vec{X}_i)\right) + \sum_{e=1}^{E} \max\left(0, |h_e(\vec{X}_i)| - \delta_e\right) \tag{6}$$

Mezura-Montes [61] introduced a DE-based method with Deb's comparative criteria as above, which proposed that the newly selected individual may be re-inserted into the current population (and not only inserted in the population for the next generation like in the original DE algorithm). The goal was to speed up convergence by selecting newly improved solutions as random parents for other children in the current generation. This was different from Deb's method because it allowed for preservation of trial vectors in current population even if the violations are equal. Indeed, this method produced good results, but only it was laced with inconsistency. Surprisingly, as of late, the strategy dependent on saving infeasible solutions in the population has additionally shown a considerable guarantee for handling constrained optimization issues [62].

2) Stochastic Ranking (SR): In order to decide whether the target or trial solution to be selected Runarsson and Yao [63] introduced a stochastic ranking technique which compared solutions based on violation or fitness value with a probability factor p_f. Owing to limitations of either choosing fitness or violation value, they later improved their ranking algorithm by introducing a adaption of linearly decreasing the probability factor from a value of 0.475 to 0.025. Although SR was initially designed for Evolutionary Strategy (ES) algorithms, but it has been later adapted to various EA [64], [65] and genetic algorithms [66].

3) $epsilon$-Constrained methods (ϵC): Takahama and Sakai [67] introduced an $epsilon$-based constraint handling technique which is a lexicographical ordering mechanism whereby the sum of constraint violation is minimized before the minimization of the objective function, with the goal of utilising its objective function value as a comparison criteria. The authors improved their algorithm using various techniques like using a DE and Gradient Based Mutation operator [67], and later the use of an archive to store solutions and ability to generate multiple solutions [68]. This was one of the best performing techniques which was later adopted by Brest et al. to propose a self-adaptive stochastic DE called ϵ-jDE [69] which has demonstrated great performance in CEC 2006 RW benchmark suite.

4) Self-Adaptive Penalty methods (SP): The most popular constraint handling methods is static penalty method where the fitness is penalized using a penalty for providing infeasible solutions. While this method is rather simpler and easier to implement, it has many short-comings including tuning hyperparameters according to problem topology as well as premature convergence into infeasible solutions. Subsequently, a two-part adaptive penalty function method was proposed in [70] whereby the diversity improved by increasing the fitness value of infeasible solution towards the best solution which could be a feasible solution in the current population with the best fitness value or the solution with the least constraint violation if there are no viable feasible solutions. Tessema et al. [71] used a similar two phase penalty function but the penalty value was decided using the number of feasible solutions in the current population. In a population with more infeasible solutions a greater penalty is imposed on solutions with larger constraint violation. If the population contains more feasible solutions, the ones with better fitness values receive the least penalties. This was a parameter less penalty function with a slight bias towards infeasible solution with better fitness values. However, both of these adaptive penalty function variants required large number (in millions) of functional evaluations.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0307.v1

5) Novel Special Operators: Researchers have investigated the use of various specialized operators like boundary based operations, for example, a binary division between feasible and infeasible solutions [72]. Another approach proposed by Huang [73] divided the population into two halves, wherein the first population uses DE to evolve the whole population irrespective of feasibility, while the second population is used to store only feasible solutions. New trial solutions are then generated using both the populations. In addition, it also uses Nelder-Mead local search operation on best feasible solutions. Other interesting approaches include Constraint Quadratic Approximation (CQA) [74] which used a surrogate of feasible search space to generate quadratic conjecture of objective function and other special equality operators.

6) Multi-Objective concepts (MO): Various multi-objective optimization concepts have been applied to solve constrained optimization problems. For instance, a ranking based on reducing constraint violation proposed by Ray [75] wherein the population needs to sustain a certain section of the infeasible population. After trail generation, the parent and offspring are separated based on feasibility and successive independent ranking is performed. Another interesting approach proposed by Reynoso-Meza [76] divides the optimization problem into three sub-problems of optimizing the objective function, the inequality constraints and the equality constraints respectively. Similarly, Wang et al [77] divided the problem into two sub-problems of optimizing the objective function and optimizing the sum of constraint violations. Consequently, Wang also proposed Adaptive Trade-off Model Evolutionary Strategy (ATMES) [78], where the population was sub-divided into three parts based only infeasible solution, feasible and infeasible solutions and only feasible solutions.

7) Ensemble of Constrained-Handling Techniques (ECHTs): Owing to the varied discrete feasible search spaces in constrained optimization problems, researchers have tend to use an ensemble of various constraint handling techniques which have notably performed well in various scenarios. Mallipeddi et al. [79] proposed the use of four ECHTs (superiority of feasible solution, self-adaptive penalty function, ϵ-constrained method and stochastic ranking) in four sub-populations. Elsayed et al. [80] used two ECHTs (superiority of feasible solution and ϵ-constrained method). Qu [81] proposed use of multi-objective DE based on feasibility of solutions and three ECHTs (superiority of feasible solution, self-adaptive penalty function and ϵ-constrained method). Tasgetiren et al. [82] proposed use of two DE variants, and a Variable Neighborhood Search (VNS) with three ECHTs (superiority of feasible solutions, ϵ-constrained method, and an adaptive penalty function). Similarly, Paldrak [83] suggested use of ensemble DE based on VNS which also used an opposition based learning and preferable good solution injection for creating trial solutions. Trivedi et al. [84] proposed an ensemble of three mutation operator as in CoDE [41] where a static based penalty function is applied on the first half of the functional evaluations and superiority of feasible solutions is applied on the rest. Xu [85] proposed use of ensemble DE variants and a two-level ϵ-constrained method using generation and population level comparison Wen et al. [86] proposed a voting mechanism for selection among four ECHTs (superiority of feasible solutions, ϵ-constrained method, stochastic ranking and self-adaptive penalty function)

In view of these pertinent literature review, it very well may be guaranteed that the choice of selection of best-performing blend of operators and parameters for any evolutionary algorithm (say DE) is yet considered as a drawn-out task, despite a couple of prior commitments from numerous researchers. Some conspicuous bearings originating from this exploration problem can be addressing the questions:

- using a framework comprising of an ensemble of search operators within a single algorithm.
- how to effectively handle various diverse set of mutation operators and their influence on exploration and exploitation of the search space in each generation of the population.
- how to distribute any particular set of operators or parameters among the current population, and various ways to segregate the population based on fitness, mean constrained violation or functional evaluation.
- the usage of multi-method and heterogeneous optimization algorithms (using different optimization algorithms with varied behavior) instead of a single optimization algorithm.
- using an ensemble of different constraint handling techniques and specialized operators.
- algorithmic performance (speed, accuracy and convergence) constraints.

Considering all the aforementioned concerns, this paper proposes an upgraded multi-method algorithm framework (**FEPEA**), which uses a hybrid of two different EA with different behaviour distributed into two sub-populations which is dynamically updated based on hyper-heuristics. It also reduces the population size in each algorithm based on success-rate using both linear and non-linear population size reduction.

D. Applications

Constrained EAs and GAs have showed tremendous potential in handling a variety of real world as well as multimodal optimization problems. Many of these algorithms is gaining widespread acceptance in different domains of machine learning and artificial inteligence, such as the use of such algorithms in neuro-evolution in Reinforcement Learning (RL) settings [87] or usage of evolutionary control parameters in Automated Machine Learning [87]. In particular, we have observed widespread use of AGSK in transportation problems [88], [89], path planning [90], knapsack problems [91], [92], electrochemical systems such as photo-voltaic cells [93], [94], engineering problems like fault diagnostics in power systems [95], as well as adoption in machine learning [96] and RL techniques [97]. As a result, it is reasonable to claim that constrained evolutionary algorithms offer enormous potential in the application of many engineering design challenges [1], [98]–[100].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0307.v1

III. The **FEPEA** Algorithm

Real-World Constrained Optimization Evolutionary Algorithm refers to addressing the optimization problem using evaluation of the objective function $f(\vec{X})$ only in a black-box setting - where the functional landscape information is not used. It is entrusted with looking for a solution $\vec{X}^*$ that minimizes the objective function: $f(\vec{X}^*) \leq f(\vec{X})\ \forall \vec{X} \in \mathfrak{F} \subseteq \mathfrak{S}^D$, where $\mathfrak{F}$ is in the feasible region, a limited set satisfying the various constraints imposed on the search space $\mathfrak{S}$. The optimization problem can be enacted using Eq (7) where $g(\vec{X})$ corresponds to gn inequality constraints and $h(\vec{X})$ corresponds to hn equality constraints.

$$\begin{aligned} &\text{minimize } f(\vec{X}) \text{ subjected to} \\ &g_i(\vec{X}) \leq 0,\ \forall\ i = 1, 2, \ldots, gn \\ &h_j(\vec{X}) = 0,\ \forall\ j = 1, 2, \ldots, hn \end{aligned} \tag{7}$$

In this paper we propose an hybrid ensemble evolutionary algorithm framework **FEPEA** which uses multiple different evolutionary algorithm with difference in characteristic behaviour. It uses an ensemble of various operators (mutation, crossover) and control parameters. Alongside, it also uses an ensemble of population size reduction techniques. As for constraint handling, we shall be an ensemble of various commonly used constraint handling techniques ECHTs such as SF, SR, eC, SP. These techniques are selected based on a voting mechanism within each sub-population. It must be noted that the following framework can be swiftly modified to include more algorithms and more wider choice of control parameters and constraint handling techniques. Henceforth, we shall adopt the usage of standard EA-ES notation and mnemonics as presented in literature.

A. IMODE Algorithm

As recently expressed, the general execution of the DE operator might change during the optimization phases; this largely depends on the initial seeds which are provided to the algorithm as well as the optimization topography as well. Additionally, an operator might perform well for specific kinds of problems, yet its performance might be inadequate for another. Hence, the utilization of multi-operator DE which determines the fate of the mutation strategy in a probabilistic manner is of the essence, as it is able to incorporate the knowledge of which solution performed well in comparison to others.The psuedo-code of the IMODE algorithm can be depicted in Algorithm 1

1) DE Mutation: In **FEPEA**, we utilize three mutation methodologies that are chosen in a probabilistic way as examined later in II-E. Two of these methodologies depend on JADE [34], i.e., DE/current-to-ϕbest/1 mutation with (8) and without archive (9). The previous one is a weaker variation of the DE/current-to-best/1 mutation, while the last one has shown promising execution in real parameter continuous optimization issues. The third mutation is a DE/weighted-rand-to-ϕbest operator (10). The three mutation strategies can be listed as below:

- DE/current-to-ϕbest/1 + archive

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F_i \cdot (\vec{X}_{\phi,G} - \vec{X}_{i,G} + \vec{X}_{r_1^i,G} - \vec{X}_{r_3^i,G}), \tag{8}$$

- DE/current-to-ϕbest/1 − archive

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F_i \cdot (\vec{X}_{\phi,G} - \vec{X}_{i,G} + \vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}), \tag{9}$$

- DE/weighted-rand-to-ϕbest

$$\vec{V}_{i,G} = F_i \cdot \vec{X}_{r_1^i,G} + (\vec{X}_{\phi,G} - \vec{X}_{r_3^i,G}) \tag{10}$$

where $r_1^i \neq r_2^i \neq r_3^i$ are randomly chosen unique integers from current population pool (NP) such that $r_1^i, r_2^i, r_3^i \in [1, NP]$ and $\vec{X}_{\phi,G}$ is chosen from top $\phi\%$ solutions based on objective function value and mean constrained violation. The solutions which survive till the next generation are kept in the current population P and the ones which succumb to the selection process are preserved in an external archive A. These spurned solutions are not completely abandoned and they do get another chance to participate in the mutation process later in the evolutionary cycle as $\vec{X}_{r_3^i,G}$ is chosen from the pool consisting of this archive and current population $A \cup P$. This preserves the diversity of the population.

2) DE Crossover Rate: **FEPEA** uses an ensemble of crossover operators namely binary and exponential crossover in a random order to produce trial vector $\vec{V}_{i,G}$. With an equal probability, a crossover operator randomly selected using a random number between [0,1] (if the random number between [0,1] selected is less than 0.5, exponential crossover takes place, otherwise, binary crossover takes place). This is can be envisaged in Eq (11). Within each crossover mechanism standard crossover operation is conducted as discussed in Sec II-A2. The crossover operation acts as a diversity safeguarding mechanism by limiting the alterations in the trial vector.

$$U_{i,j} = \begin{cases} \begin{cases} V_{i,j} & \text{for } j = \langle c \rangle_D, \langle c+1 \rangle, \ldots, \langle c+C+1 \rangle_D \\ X_{i.j} & \text{for all other } j \in [\ 1, D] \end{cases} \\ \text{if } rand \leq 0.5 \\ \begin{cases} V_{i,j} & \text{if } (rand \leq Cr_i \text{or } j = j_{rand}) \\ X_{i.j} & \text{otherwise} \end{cases} \\ \text{otherwise} \end{cases} \tag{11}$$

3) DE Control Parameters: The control parameter adaption is enlivened from the success history-based hyperparameter adaptation [25], [34], [35]. It keeps a memory archive of the effective DE hyperparameters, to be specific, scaling factor F in Eq (12) and crossover rate Cr in Eq (13). Success history based memory archives are allotted as in standard SHADE [35] to both the scaling factor F_i as $S_{F,i} = F_i, i = 1, \ldots, H$, and the crossover rate Cr_i as $S_{Cr,i} = Cr_i,\ i = 1, \ldots, H$ particularly. Subsequently, the hyperparameters for constructing the trial vectors are generated using these archives as in Eq (12,13):

$$F_i \sim randc_i\left(S_{F,r_i}, 0.1\right) \quad \text{and} \tag{12}$$

$$Cr_i \sim randc_i\left(S_{Cr,r_i}, 0.1\right) \tag{13}$$

The scaling factor F_i is sampled using Cauchy distribution to guarantee a wide spread of values. F_i is limited to a region of

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0307.v1

$F_i \in (0, 1]$, and set to a value of 1 in case $F_i > 1$, otherwise is re-sampled if $F_i < 0$ till a valid value of is produced. Similarly, the crossover rate Cr is also sampled from the Cauchy distribution and pruned to lie within the range of $[0, 1]$. An uniform random index r_i is selected from the range $[1, H]$.

At the end of each generation, the successful control parameters are archived in S_F and S_{CR}. The memory archive is updated at a random position $k \in [1, H]$. Initially k is set to 1, and subsequently increased with every successful solutions generated. If $k > H$ then k is reset to 1. If a generation could not generate any successful solution, the memory archive is not updated. The archive S_{CR} has a terminal value $\perp$ which is assigned when all elements of S_{CR} = 0. If this value is assigned to the memory it remains fixed at this value as the CR gets locked at 0 till end of the search. This slows down the overall convergence.

B. APGSK Algorithm

The nature inspired APGSK Algorithm intuitively reflects human behavior of acquiring and propagating knowledge within their eco-system. This basically occurs in two major phases: firstly, when the solutions starts imbibing the knowledge they are called *juniors*. These solutions consequently interact with the other solutions in the neighbourhood and acquire passive meta-heuristic information. Steadily these solutions become adept in their trade which is when they are transformed as *seniors*.

1) APGSK Control Parameters: In this paper, the control parameters for APGSK k_f and k_r are naturally dictated by choosing a pool for the two parameters and Kw_P probability. The parameter pool is made out of four sets of tuple consisting of the control parameters (k_f, k_r): [(0.1, 0.2), (1.0, 0.9), (0.5, 0.9) and (1.0, 0.1)] which is utilized for the first half of maximum functional evaluations, while another different set: [(-0.15, 0.2), (-0.05, 0.1), (-0.05, 0.9), and (-0.15, 0.9)] is applied to the residual half with a probability under 0.3 which ensures population diversity and prevents the algorithm from getting stuck at any local optima in the functional landscape.

The knowledge probability Kw_P that has a probability parameter p for each setting of the previously aforementioned parameter pool. This ensures that one tuple shall be selected for each solution based on its probability parameter p. The knowledge probability adaptation Kw_P begins after 10% of maximum functional evaluations and is determined by ω_{ps} which is the difference in the sum total objective function value of the current $x_i^{current}$ and previous solution $x_i^{previous}$ for each setting. The following can be depicted in Eq (14):

$$\omega_{ps} = \sum_{i=1}^{n} f(x_i^{current}) - f(x_i^{previous}) \tag{14}$$

where n is the total number of solutions in parameter tuple ps.

Following that, the rate of improvement for each parametric setting is calculated as Eq (15):

$$\Delta_{ps} = \max(0.05, \frac{\omega_{ps}}{\text{sum}(\omega_{ps})}) \tag{15}$$

which is normalized between 0.05 and maximum ω_{ps}. This kind of normalization ensures that none of the parametric setting should be set to a probability below 0.05 thereby preventing concentration of any selected parametric setting.

Finally, the rate of improvement (Δ_{ps}) for each parameteric setting is used to refresh the knowledge probability of next generation Kw_P_{G+1} using the previous knowledge probability Kw_P_G depicted in Eq (16):

$$Kw\_P_{G+1} = (1 - c)Kw\_P_G + c \cdot \Delta_{ps} \tag{16}$$

where c is the learning rate.

2) Knowledge Rate Adaptation: The heterogeneous idea of any population ought to be taken into the record to copy the cycle of knowledge acquiring and sharing for a given population through its life cycle. In this way, k needs to consider two situations. The first when $k \in (0, 1)$, while the second is when $k \leq 1$ with probability of $(\frac{FES}{MAX_{FES}})$ in Eq (17) as:

$$\text{if } rand \geq \frac{FES}{MAX_{FES}}, k = 0.5 \text{ else } k = 2 \tag{17}$$

C. Population Update

While MODE uses a Linear Population Size Reduction (LPSR) [25], AGSK, on the contrary, uses Non-Linear Population Size Reduction (NLPSR). The linear population size reduction at each generation G, $G = 1, 2, \ldots$ in MODE is updated as Eq (18)

$$NP_G = \left\lfloor NP_{\max} - (NP_{\max} - NP_{\min}) \frac{FES}{MAX_{FES}} \right\rceil, \tag{18}$$

At whatever point the condition $NP_G < NP_{G-1}$ happens, the most exceedingly terrible $NP_{G-1} - NP_G$ individuals are taken out from the population

Whereas the non-linear population size reduction (NLPSR) in APGSK is implemented as Eq (19)

$$NP_G = \left\lfloor (NP_{\min} - NP_{init}) * \left(\frac{FES}{MAX_{FES}} \right)^{\left(1 - \frac{FES}{MAX_{FES}}\right)} \right\rceil + NP_{init}, \tag{19}$$

where FES denotes the current of functional evaluation counter toward the finish of the given generation, MAX_{FES} denotes the maximum allowable functional evaluations, $NP_{\min}$ is the minimum population size (which is set to 4 in the case of MODE and 12 incase of APGSK), $NP_{\max}$ is the maximum population size, NP_{init} is the initial population size, and $\lfloor . \rceil$ is the closest whole number function.

D. Constraint Handling

Amongst various constraint handling methods superiority of feasible solutions has been a common choice for researchers. There are many variants of these. The constrained handling method used in **FEPEA** comprises of an ensemble of various popular constraint handling methods (ECHT), such as SF, SR, ϵC, and SP strategy by using a voting based mechanism [86] to select the best-performing constraint handling operator which minimizes the constraint violation mean. Experimental

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0307.v1

results illustrates that this method is able to successfully locate and generate better offsprings in the feasible region. This technique allows the users to use an ensemble of CHTs, and the framework **FEPEA** allows the users to add more customized novel constrained handling techniques to suit the problem which it is trying to resolve.

The equality constraints are transformed into inequality constraints by subtracting a minimal tolerance (10^7) from the equality constraints as $|h_j(\vec{x})| - \varepsilon \leqslant 0$. This is essential because, for any equality constraints, the feasible region can be represented as a line-segment between the bounds. If all the constraints happen to be equality constraints, the feasible region would be the intersection of all these regions, giving a point, which degrades the evolutionary process. Thus, in order to incorporate flexibility all the constraints are transformed using Eq (20).

$$cv_j(\vec{x}) = \begin{cases} \max\left(0, g_j(\vec{x})\right), & j = 1, \ldots, q \\ \max\left(0, |h_j(\vec{x})| - \varepsilon\right), & j = q+1, \ldots, m \end{cases} \tag{20}$$

The constraint violations are evaluated sequentially for the entire population using all the constraint handling techniques. The fitness value is calculated after apply a penalty of their respective denormalized mean violation for each function as in Eq. (21)

$$fitness = f_{eval}(i) + cv_{mean}(i) \tag{21}$$

where, $f_{eval}(i)$ is the functional evaluation value and $cv_{mean}(i)$ is the mean violation across all the independent executions. For each CHT a separate counter is maintained to denote which technique produced a better offspring. This counter is used in a voting manner to decide on the CHT to be used. Details of the technique can be found in Algorithm (4).

E. Difference Vector Computation

Following constraint violation evaluation, a difference vector is computed based on these computed violation mean is taken for various occurrences of feasible to feasible, infeasible to feasible and infeasible to infeasible solution. A simple difference vector of the form $\vec{\Delta}$ can be represented using Eq (22) which successfully generates offspring $\vec{U}_{i,G}$ such that $f(\vec{U}_{i,G}) \leq f(\vec{X}_{i,G})$ always hold true.

$$\vec{\Delta} = \vec{X}_{r1,G} - \vec{X}_{r2,G} \tag{22}$$

Finally, a greedy based selection operation as mentioned in Sec II-A3 is performed where, the solutions with a minimum fitness value is selected for the next iteration.

F. Putting it all together

From an initial population, it is divided into two-divisions (with 75% allocated to MODE and the rest allocated to APGSK) at the beginning and it gradually changed. Each population is fed to the designated algorithms and checked for their efficacy. A greedy selection is then performed to evolve the next generation. Finally, both the populations are recombined and re-partitioned again until it provides a better offspring.

The pseudo-code of the MODE algorithm is depicted in Algorithm 1.

Algorithm 1 IMODE Algorithm

Require: Define nop, $Prob_s \leftarrow 0.1$, MAX_{FES}, $prob_1$, $prob_2$, NP, $G \leftarrow 1$, $FES \leftarrow 0$
Input: Initial Population X of size NP
Output: Evaluate $f(X)$, and calculate fitness evaluation
1: **Initialization** each operator op is assigned same number of solutions NP_{op}
2: **while** $FES \leq MAX_{FES}$ **do**
3: Update Generation Number $G \leftarrow G + 1$
4: Use the assigned DE mutation strategies to generate offsprings where every DE mutation strategy (operator) op optimizes determined solutions NP_{op}
5: Evaluate for fitness $f(X)$
6: Update $FES, FES \leftarrow FES + NP$, Update NP
7: Compute the difference vector
8: Perform greedy selection among the solutions
9: Update each of the optimized solutions
10: **end while**

The pseudo-code of the AGSK algorithm is depicted in Algorithm 2.

Algorithm 2 AGSK Algorithm

Require: mentioned in parameteric setting and Kw_P
Input: Initial Population X of size NP
Output: Evaluate $f(X)$, and calculate fitness evaluation
1: **Initialization** Define initial parameter pool settings
2: **while** $FES \leq MAX_{FES}$ **do**
3: **if** $FES \leq 0.1 \times MAX_{FES}$ **then**
4: Update the value of Kw_P
5: **end if**
6: Set one setting to every solution based on Kw_P
7: Evaluate for fitness $f(X)$ and update $FES, FES \leftarrow FES + NP$, Update NP
8: Compute the improvement value of every setting
9: **end while**

The pseudo-code of the **FEPEA**can be depicted from Algorithm 3.

The pseudo-code of the Voting algorithm is depicted in Algorithm 4.

IV. Experimental Results

A. Benchmark

To pass judgment on the presentation of the proposed **FEPEA** algorithm, a few trials were led on 57 constrained optimization issues with various measurements goes from 2 factors to 158 factors and distinctive number of equality also, inequality constraints goes from 2 constraints to 148 constraints. Subtleties of these issues can be found in [51].

These issues are classified to six sets:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0307.v1

Algorithm 3 FEPEA Algorithm

Require: Define $N = PS_1 + PS_2$, nop, $Prob_s \leftarrow 0.1$, MAX_{FES}, $prob_1 \leftarrow 1$, $prob_2 \leftarrow 1$, NP, $G \leftarrow 1$, $FES \leftarrow 0$ and all other hyper-parameters
Input: **Initialization** IMODE and AGSK Parameters
Generate initial random population of size NP
Output: Evaluate $f(X)$, and calculate fitness evaluation
Calculate for mean constraint violations for all (g_n) equality and (h_n) in-equality constraints
Calculate mean violation selection $mean_{viol} \leftarrow \frac{\text{violation}}{g_n+h_n}$
Perform violation treatment using Voting Algorithm 4
Randomly allocate population into PS_1 and PS_2 with default ratio of 75% to 25% at the beginning
while $FES \leq MAX_{FES}$ **do**
  $count \leftarrow count + 1$
  if $count == CS$ **then**
    Pr_1 and Pr_2 as stated in
  end if
  if $count == 2 \times CS$ **then**
    Share the best solution to both subpopulation
    Reset Pr_1, Pr_2 and $count$
  end if
  if $rand \in [0, 1] \leq Pr_1$ **then**
    Apply IMODE and optimize the sub-population
    Use the assigned DE mutation strategies to generate offsprings where every DE mutation strategy (operator) op optimizes determined solutions NP_{op}
    Evaluate for fitness $f(X)$ and update fitness evaluation $FES \leftarrow FES + PS_1$
    Update the solutions and return the best solutions.
  end if
  if $rand \in [0, 1] \leq Pr_2$ **then**
    Apply APGSK and optimize the sub-population
    Update fitness evaluation $FES \leftarrow FES + PS_2$
    Update the solutions and return the best solutions.
  end if
  Recombine the solutions from the both the algorithms $PS \leftarrow PS_1 + PS_2$
  Check for constraint violation and calculate mean violation using the Voting Algorithm 4
end while

Algorithm 4 Voting Algorithm

Require: known set of constraint handling techniques H
Input: Parent Population $P_{i,G}$, Offspring Population $P_{i,G+1}$
Output: Calculated fitness violation evaluation and updated Population $P_{o,G+1}$
Initialization Initialize violation terms V_o, V_p
for $h \leq H$ **do**
  if $P_{i,G+1} \leq P_{i,G}$ according to rules of h **then**
    Update $V_o = V_o + 1$
  else
    Update $V_p = V_p + 1$
  end if
end for
if $V_o \geq V_p$ **then**
  Update $P_{i,G+1}$ in $P_{o,G+1}$
else
  Update $P_{i,G}$ in $P_{o,G+1}$
end if

- Industrial Chemical Processes contains seven test problems ($\mathcal{F}01 - \mathcal{F}07$);
- Process Synthesis and Design Problems has seven test problems ($\mathcal{F}08 - \mathcal{F}14$);
- Mechanical Engineering Problem includes 19 test problems ($\mathcal{F}15 - \mathcal{F}33$);
- Power System Problems has 11 test problem ($\mathcal{F}34 - \mathcal{F}44$);
- Power Electronic Problems contains six test problems ($\mathcal{F}45 - \mathcal{F}50$);
- Livestock Feed Ration Optimization contains seven test problems ($\mathcal{F}51 - \mathcal{F}57$);

All the experiments were executed on the following system:

- OS: Windows 10
- CPU: AMD Ryzen 7 3700X (3.59 GHz)
- RAM: 16 GB
- Language: MATLAB 2018a
- Compiler: MinGW-w64 C/C++ Compiler

B. Parameter Settings

1) IMODE Parameters:

- Number of Operators (N_{op}) = 3
- Archive Factor = 1.4
- Scaling Factor = 0.5
- Crossover Factor = 0.5
- Minimum Population Size = 4

2) APGSK Parameters: (k_f, k_r):

- First 50%: [(0.1, 0.2), (0.5, 0.9), (1.0, 0.9) and (1.0, 0.1)]
- Last 50%:[(- 0.15, 0.2), (- 0.05, 0.9), (- 0.15, 0.9) and (- 0.05, 0.1)]
- Minimum Population Size = 12

C. Results Obtained

We have tested the variability in performance of the proposed **FEPEA** along with the individual algorithms IMODE and AGSK. Although, the APGSK algorithm is able to obtain good results, the ECDF plots using Performance Profiles in Fig(1) of these three algorithms pitted against each other indicate that that the hybridization of both the algorithm is able to obtain better results than either of the algorithm ran individually.

In addition we perform non-parametric tests of the proposed **FEPEA** along with other state-of-the-art known to perform well in recent RW Constrained Optimization like IUDE, Eps-MAgES, iLSHADE44, COLSHADE, EnMODE and AGSK. The results presented in the table indicate that the proposed algorithm is able to obtain state-of-the-art results.

We also use the scoring mechanism provided by the benchmark suite to confer on the results, wherein, the proposed **FEPEA** is able to beat the competitors comprehensively.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0307.v1

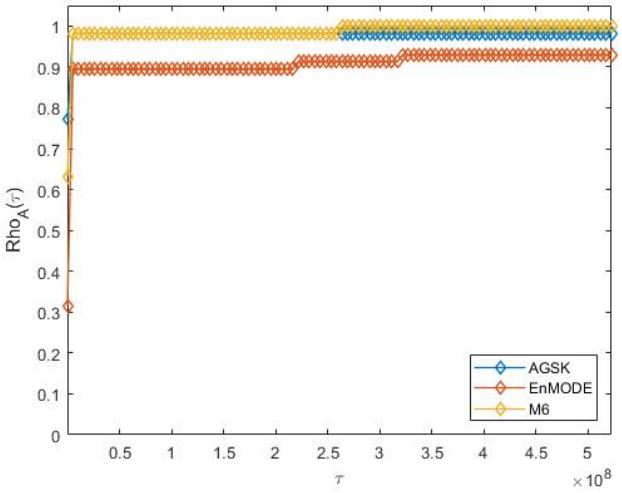

Fig. 1: ECDF plot for EnMODE vs APGSK vs cAGSKODE

TABLE I: Friedman Test Results

Algo	IUDE	EpsMAgES	iLSHADE44	COLSHADE	EnMODE	AGSK	**FEPEA**
Rank	23.47	21.59	21.05	16.69	33.98	33.85	**50.00**

The algorithm has been tested on the CEC 2020 RW Benchmark and it has proven to achieve great results as depicted in the Tables.

V. Conclusion and Future Works

In this paper a hybrid **FEPEA** is proposed to solve real world constrained optimization problems from the CEC 2020 Benchmark Suite. This algorithm mainly merges the two algorithm IMODE and AGSK which have individually proven to be the top contenders in prior CEC competitions. This algorithm not only leverages multiple self adaptation techniques and multiple operators originally introduced in each algorithms, but it also unites both linear and non-linear population size reduction from both the algorithms. The performance of **FEPEA** reveals that it is able to attain state-of-the-art performance and hunt for solutions in the feasible space. This paper also shows that hybridization of multiple algorithms tends to be mutually beneficial when solving for multiple optimization problems across a varied topography.

As part of future developments we will try to look for even more state-of-the-art constraint handling procedures with higher efficacy across a discrete feasible space. Consequently, we are also researching in the use of this constrained optimization algorithm for solving mechanical systems problems like thermal buckling load [99] utilising isogeometric finite element analysis of stiffened laminated composite plates [100]. Also, we would like to investigate the performance of the algorithm across a wide range of transformation operations like bias,

TABLE II: Results obtained using Scoring function as part of the CEC2020 benchmark suite

Algo	IUDE	EpsMAgES	iLSHADE44	COLSHADE	EnMODE	AGSK	**FEPEA**
Score	23.47	21.59	21.05	16.69	33.98	33.85	**50.00**

TABLE III: Table of Outcomes for $\mathcal{F}01$-$\mathcal{F}08$

Criteria		$\mathcal{F}01$	$\mathcal{F}02$	$\mathcal{F}03$	$\mathcal{F}04$	$\mathcal{F}05$	$\mathcal{F}06$	$\mathcal{F}07$	$\mathcal{F}08$
Best	f	0.0000e+00	4.2089e+03	-1.7989e+04	-9.9986e-01	-3.6466e+03	9.9790e-01	6.9962e-01	-1.0000e+00
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Median	f	0.0000e+00	6.2822e+03	-8.7264e+03	-9.9262e-01	-3.2793e+03	1.6035e+00	1.1227e+00	-2.2630e-01
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	4.4616e-01	0.0000e+00
Mean	f	5.1768e+00	6.1573e+03	-9.5337e+03	-9.9042e-01	-3.2717e+03	1.5354e+00	1.3067e+00	-4.1083e-01
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	1.2211e-02	6.1918e-01	0.0000e+00
Worst	f	4.1790e+01	6.6728e+03	-3.8688e+03	-9.6503e-01	-2.7061e+03	2.2681e+00	2.3718e+00	6.1899e-02
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	7.2799e-02	1.8102e+00	0.0000e+00
Std	f	1.1940e+01	5.6421e+02	3.8138e+03	8.8730e-03	2.1472e+02	4.1420e-01	4.7718e-01	4.5155e-01
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	1.8026e-02	6.5119e-01	0.0000e+00
FR c									

TABLE IV: Table of Outcomes for $\mathcal{F}09$-$\mathcal{F}16$

Criteria		$\mathcal{F}09$	$\mathcal{F}10$	$\mathcal{F}11$	$\mathcal{F}12$	$\mathcal{F}13$	$\mathcal{F}14$	$\mathcal{F}15$	$\mathcal{F}16$
Best	f	5.3716e-01	1.0000e-01	-1.3000e+01	1.4782e+00	2.2303e+04	2.4666e+04	2.4630e+03	1.1720e-02
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Median	f	7.1534e-01	1.0000e-01	5.5000e+00	2.9248e+00	2.2311e+04	4.3727e+04	2.8338e+03	1.1756e-02
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Mean	f	7.4605e-01	1.0001e-01	5.1650e+00	2.5176e+00	2.2333e+04	4.4398e+04	2.7687e+03	1.5230e-02
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Worst	f	1.0194e+00	1.0001e-01	2.5359e+01	2.9248e+00	2.2474e+04	6.3315e+04	2.9117e+03	9.3064e-02
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Std	f	1.5561e-01	1.3136e-05	1.0870e+01	5.3610e-01	4.5170e+01	9.7984e+03	1.2283e+02	1.6230e-02
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
FR c									

TABLE V: Table of Outcomes for $\mathcal{F}17$-$\mathcal{F}24$

Criteria		$\mathcal{F}17$	$\mathcal{F}18$	$\mathcal{F}19$	$\mathcal{F}20$	$\mathcal{F}21$	$\mathcal{F}22$	$\mathcal{F}23$	$\mathcal{F}24$
Best	f	2.5000e-03	1.5902e+01	4.4079e-02	0.0000e+00	1.2498e-01	5.2325e-01	0.0000e+00	0.0000e+00
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Median	f	2.5000e-03	6.9090e+02	2.7365e-01	0.0000e+00	1.2555e-01	5.2883e-01	1.9380e-01	3.7666e+00
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Mean	f	2.5000e-03	7.9146e+02	3.1193e-01	0.0000e+00	1.2951e-01	5.3127e-01	1.2245e+00	3.4398e+00
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Worst	f	2.5000e-03	1.8843e+03	6.0141e-01	0.0000e+00	1.8938e-01	5.4981e-01	7.4791e+00	4.3945e+00
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Std	f	4.4262e-19	5.5281e+02	1.6515e-01	0.0000e+00	1.2938e-02	6.5552e-03	2.1986e+00	1.1048e+00
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
FR c									

TABLE VI: Table of Outcomes for $\mathcal{F}25$-$\mathcal{F}32$

Criteria		$\mathcal{F}25$	$\mathcal{F}26$	$\mathcal{F}27$	$\mathcal{F}28$	$\mathcal{F}29$	$\mathcal{F}30$	$\mathcal{F}31$	$\mathcal{F}32$
Best	f	-3.5453e+18	2.4811e+01	1.2123e+02	4.9896e+03	1.1493e+06	3.5975e-04	0.0000e+00	-3.2217e+04
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Median	f	-4.5838e+13	5.9894e+01	3.4911e+02	5.4819e+03	1.6227e+06	1.1425e-03	4.3450e-19	-3.2217e+04
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Mean	f	-1.5007e+17	5.8173e+01	3.3100e+02	5.9970e+03	1.6213e+06	1.3582e-03	7.8479e-11	-3.2217e+04
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Worst	f	-1.6765e+11	9.6575e+01	4.5297e+02	1.4182e+04	2.1506e+06	6.0318e-03	1.5603e-09	-3.2217e+04
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Std	f	7.0843e+17	2.0606e+01	9.3922e+01	1.7919e+03	2.6015e+05	1.1419e-03	3.1251e-10	7.4260e-12
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
FR c									

TABLE VII: Table of Outcomes for $\mathcal{F}33$-$\mathcal{F}40$

Criteria		$\mathcal{F}33$	$\mathcal{F}34$	$\mathcal{F}35$	$\mathcal{F}36$	$\mathcal{F}37$	$\mathcal{F}38$	$\mathcal{F}39$	$\mathcal{F}40$
Best	f	2.6393e+00	2.0285e+00	9.6867e+01	1.3004e+02	-6.7958e+00	-1.0859e+01	-2.0524e+01	1.0305e+03
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Median	f	2.6393e+00	1.0600e+01	3.2930e+02	2.5908e+02	-2.4633e+00	-5.4786e-01	1.8928e+00	2.3486e+04
	v	0.0000e+00	1.4860e-01	0.0000e+00	6.1115e-01	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Mean	f	2.6393e+00	1.0431e+01	3.2499e+02	2.7357e+02	-1.9698e+00	9.7238e-01	7.6360e-01	4.9706e+04
	v	0.0000e+00	1.5048e-01	5.8108e-01	1.1077e+00	1.2659e-02	1.2363e-02	1.8745e-02	2.9733e+00
Worst	f	2.6394e+00	2.2741e+01	6.8588e+02	4.5050e+02	5.2081e+00	1.4463e+01	2.1526e+01	4.8057e+05
	v	0.0000e+00	3.4354e-01	3.8987e+00	3.9145e+00	1.2565e-01	1.0981e-01	1.3226e-01	9.7348e+00
Std	f	1.2508e-06	6.2414e+00	1.2826e+02	7.8276e+01	2.9765e+00	7.5524e+00	9.7206e+00	9.4272e+04
	v	0.0000e+00	1.3037e-01	1.0963e+00	1.3378e+00	3.3610e-02	2.7123e-02	4.0669e-02	3.5560e+00
FR c									

TABLE VIII: Table of Outcomes for $\mathcal{F}41$-$\mathcal{F}48$

Criteria		$\mathcal{F}41$	$\mathcal{F}42$	$\mathcal{F}43$	$\mathcal{F}44$	$\mathcal{F}45$	$\mathcal{F}46$	$\mathcal{F}47$	$\mathcal{F}48$
Best	f	2.4067e+03	-1.5122e+03	-7.1221e+02	-6.0519e+03	4.1428e-01	3.0069e-01	1.5997e-01	8.9752e-02
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Median	f	1.0822e+04	-8.5192e+02	-4.0033e+02	-5.8830e+03	1.6642e+00	9.0235e-01	6.9454e-01	4.8018e-01
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	1.2000e-07	0.0000e+00	7.4516e-03
Mean	f	1.5604e+04	-8.5301e+02	-4.0156e+02	-5.8718e+03	2.2788e+00	1.3208e+00	8.5780e-01	6.9715e-01
	v	2.8400e+00	3.1441e-01	1.0847e-01	0.0000e+00	2.4488e-02	2.5060e-02	2.8297e-02	7.5673e-02
Worst	f	1.3470e+05	2.0610e+01	-4.1687e+01	-5.6930e+03	5.5520e+00	3.1915e+00	2.0522e+00	2.0462e+00
	v	1.2588e+01	3.8312e+00	1.5443e+00	0.0000e+00	1.5013e-01	2.9334e-01	2.2166e-01	4.3372e-01
Std	f	2.5420e+04	3.6722e+02	1.6109e+02	9.7141e+01	1.7147e+00	1.0743e+00	5.8715e-01	5.3454e-01
	v	4.2509e+00	9.2583e-01	3.7936e-01	0.0000e+00	4.2907e-02	6.2438e-02	5.4517e-02	1.1090e-01
FR c									

TABLE IX: Table of Outcomes for $\mathcal{F}49$-$\mathcal{F}57$

Criteria		$\mathcal{F}49$	$\mathcal{F}50$	$\mathcal{F}51$	$\mathcal{F}52$	$\mathcal{F}53$	$\mathcal{F}54$	$\mathcal{F}55$	$\mathcal{F}56$	$\mathcal{F}57$
Best	f	8.9536e-02	8.1703e-02	1.9402e+03	1.0086e+03	1.9159e+03	3.3261e+02	0.0000e+00	0.0000e+00	0.0000e+00
	v	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Median	f	4.0093e-01	2.8774e-01	4.7870e+03	4.3282e+03	5.0558e+03	3.5933e+03	0.0000e+00	1.9857e+01	0.0000e+00
	v	9.3333e-07	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00	0.0000e+00
Mean	f	5.5587e-01	4.4447e-01	5.5060e+03	4.1844e+03	5.0044e+03	4.0343e+03	8.7865e+00	4.7390e+01	1.6094e-01
	v	4.6614e-02	3.9396e-02	2.6448e-03	0.0000e+00	0.0000e+00	2.4291e-03	0.0000e+00	0.0000e+00	0.0000e+00
Worst	f	2.1419e+00	1.2800e+00	1.4714e+04	6.4224e+03	1.1503e+04	9.1097e+03	1.4634e+02	1.9717e+02	4.0234e+00
	v	2.5449e-01	3.3982e-01	3.9307e-02	0.0000e+00	0.0000e+00	2.2804e-02	0.0000e+00	0.0000e+00	0.0000e+00
Std	f	4.6180e-01	3.5667e-01	2.8061e+03	1.1448e+03	1.8417e+03	2.0272e+03	3.1473e+01	5.9556e+01	8.0468e-01
	v	7.3491e-02	7.7423e-02	8.1914e-03	0.0000e+00	0.0000e+00	5.4948e-03	0.0000e+00	0.0000e+00	0.0000e+00
FR c										

shift, rotation, etc. Lastly, we would like to use hyperparameter optimization techniques [54] for finding good initial control parameter guesses for our proposed algorithm.

REFERENCES

[1] B. Devarajan, "Free vibration analysis of curvilinearly stiffened composite plates with an arbitrarily shaped cutout using isogeometric analysis," *arXiv preprint arXiv:2104.12856*, 2021.

[2] B. Devarajan and R. K. Kapania, "Thermal buckling of curvilinearly stiffened laminated composite plates with cutouts using isogeometric analysis," *Composite Structures*, vol. 238, p. 111881, 2020.

[3] J. Miglani, B. Devarajan, and R. K. Kapania, "Thermal buckling analysis of periodically supported composite beams using isogeometric analysis," in *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2018, p. 1224.

[4] Q. Li, B. Devarajan, X. Zhang, R. Burgos, D. Boroyevich, and P. Raj, "Conceptual design and weight optimization of aircraft power systems with high-peak pulsed power loads," SAE Technical Paper, Tech. Rep., 2016.

[5] B. Devarajan and R. K. Kapania, "Analyzing thermal buckling in curvilinearly stiffened composite plates with arbitrary shaped cutouts using isogeometric level set method," *Aerospace Science and Technology*, p. 107350, 2022.

[6] B. Devarajan, "Thermomechanical and vibration analysis of stiffened unitized structures and threaded fasteners," Ph.D. dissertation, Virginia Tech, 2019.

[7] B. Devarajan, D. Locatelli, R. K. Kapania, and R. J. Meritt, "Thermomechanical analysis and design of threaded fasteners," in *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2016, p. 0579.

[8] J. Miglani, B. Devarajan, and R. K. Kapania, "Isogeometric thermal buckling and sensitivity analysis of periodically supported laminated composite beams," *AIAA Journal*, pp. 1–10, 2021.

[9] A. Karakoti, V. R. Kar, and B. Devarajan, "Hygrothermoelastic responses of sinusoidally corrugated fiber-reinforced laminated composite structures," in *Advanced Composite Materials and Structures*. CRC Press, pp. 201–216.

[10] S. De, K. Singh, J. Seo, R. K. Kapania, E. Ostergaard, N. Angelini, and R. Aguero, "Structural design and optimization of commercial vehicles chassis under multiple load cases and constraints," in *AIAA Scitech 2019 Forum*, 2019, p. 0705.

[11] M. Jrad, S. De, and R. K. Kapania, "Global-local aeroelastic optimization of internal structure of transport aircraft wing," in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017, p. 4321.

[12] J. H. Robinson, S. Doyle, G. Ogawa, M. Baker, S. De, M. Jrad, and R. K. Kapania, "Aeroelastic optimization of wing structure using curvilinear spars and ribs (sparibs)," in *17th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2016, p. 3994.

[13] S. De, K. Singh, J. Seo, R. K. Kapania, E. Ostergaard, N. Angelini, and R. Aguero, "Lightweight chassis design of hybrid trucks considering multiple road conditions and constraints," *World Electric Vehicle Journal*, vol. 12, no. 1, p. 3, 2021.

[14] S. De, M. Jrad, D. Locatelli, R. K. Kapania, and M. Baker, "Sparibs geometry parameterization for wings with multiple sections using single design space," in *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2017, p. 0570.

[15] S. De, K. Singh, J. Seo, R. Kapania, R. Aguero, E. Ostergaard, and N. Angelini, "Unconventional truck chassis design with multifunctional cross members," SAE Technical Paper, Tech. Rep., 2019.

[16] S. De, "Structural modeling and optimization of aircraft wings having curvilinear spars and ribs (sparibs)," Ph.D. dissertation, Virginia Tech, 2017.

[17] S. De, K. Singh, B. Alanbay, R. K. Kapania, and R. Aguero, "Structural optimization of truck front-frame under multiple load cases," in *ASME International Mechanical Engineering Congress and Exposition*, vol. 52187. American Society of Mechanical Engineers, 2018, p. V013T05A039.

[18] S. De, "Mathematical modeling of cyclic voltammogram curves of copper deposition," 2021.

[19] D. Saha and S. De, "Practical self-driving cars: Survey of the state-of-the-art," 2022.

[20] S. De and Q. Huang, "Estimation of kinetic parameters of additives by semi-analytical solution," in *ECS Meeting Abstracts*, no. 17. IOP Publishing, 2020, p. 1483.

[21] S. De and R. K. Kapania, "Algorithms for 2d mesh decomposition in distributed design optimization," *arXiv preprint arXiv:2002.00525*, 2020.

[22] S. Biswas, D. Saha, S. De, A. D. Cobb, S. Das, and B. A. Jalaian, "Improving differential evolution through bayesian hyperparameter optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 832–840.

[23] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Improved multi-operator differential evolution algorithm for solving unconstrained problems," *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.

[24] A. W. Mohamed, A. A. Hadi, A. K. Mohamed, and N. H. Awad, "Evaluating the performance of adaptive gainingsharing knowledge based algorithm on cec 2020 benchmark problems," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–8.

[25] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, pp. 1658–1665.

[26] A. Kumar, G. Wu, M. Z. Ali, R. Mallipeddi, P. N. Suganthan, and S. Das, "A test-suite of non-convex constrained optimization problems from the real-world and some baseline results," *Swarm and Evolutionary Computation*, vol. 56, p. 100693, 2020.

[27] R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by Differential Evolution," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 842–844.

[28] R. Storn and K. Price, "Differential Evolution–A simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[29] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[30] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[31] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, pp. 448–462, 2005.

[32] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[33] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[34] J. Zhang and A. C. Sanderson, "JADE: Adaptive Differential Evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[35] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for Differential Evolution," in *2013 IEEE Congress on Evolutionary Computation*, 2013, pp. 71–78.

[36] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 526–553, 2009.

[37] Y. Wang, Z. Cai, and Q. Zhang, "Enhancing the search ability of differential evolution through orthogonal crossover," *Inf. Sci.*, vol. 185, pp. 153–177, 2012.

[38] S.-M. Guo and C.-C. Yang, "Enhancing differential evolution utilizing eigenvector-based crossover operator," *IEEE Transactions on Evolutionary Computation*, vol. 19, pp. 31–49, 2015.

[39] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Multi-operator based evolutionary algorithms for solving constrained optimization

problems," *Comput. Oper. Res.*, vol. 38, no. 12, p. 1877–1896, Dec. 2011.

[40] A. Zamuda and J. Brest, "Population reduction differential evolution with multiple mutation strategies in real world industry challenges," in *ICAISC*, 2012.

[41] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Transactions on Evolutionary Computation*, vol. 15, pp. 55–66, 2011.

[42] E. K. da Silva, H. J. C. Barbosa, and A. C. C. Lemonge, "An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization," *Optimization and Engineering*, vol. 12, pp. 31–54, 2011.

[43] R. Mallipeddi, P. N. Suganthan, Q. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Appl. Soft Comput.*, vol. 11, pp. 1679–1696, 2011.

[44] R. Mallipeddi, "Harmony search based parameter ensemble adaptation for differential evolution," *J. Appl. Math.*, vol. 2013, pp. 750 819:1–750 819:12, 2013.

[45] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, "An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, pp. 482–500, 2012.

[46] S. Elsayed, R. Sarker, and T. Ray, "Differential evolution with automatic parameter configuration for solving the cec2013 competition on real-parameter optimization," 06 2013.

[47] L. Cui, G. Li, Q. Lin, J. Chen, and N. Lu, "Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations," *Comput. Oper. Res.*, vol. 67, pp. 155–173, 2016.

[48] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Inf. Sci.*, vol. 329, pp. 329–345, 2016.

[49] X.-F. Liu, Z. hui Zhan, Y. Lin, W. neng Chen, Y. jiao Gong, T. Gu, H. Yuan, and J. Zhang, "Historical and heuristic-based adaptive differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, pp. 2623–2635, 2019.

[50] K. M. Sallam, S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Landscape-assisted multi-operator differential evolution for solving constrained optimization problems," *Expert Systems with Applications*, vol. 162, p. 113033, 2020.

[51] A. Kumar, R. K. Misra, D. Singh, and S. Das, "Testing a multi-operator based differential evolution algorithm on the 100-digit challenge for single objective numerical optimization," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 34–40.

[52] S. Elsayed, R. Sarker, C. A. C. Coello, and T. Ray, "Adaptation of operators and continuous control parameters in differential evolution for constrained optimization," vol. 22, no. 19, 2018.

[53] K. M. Sallam, S. M. Elsayed, R. K. Chakrabortty, and M. J. Ryan, "Multi-operator differential evolution algorithm for solving real-world constrained optimization problems," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–8.

[54] S. Biswas, D. Saha, S. De, A. D. Cobb, S. Das, and B. A. Jalaian, "Improving differential evolution through bayesian hyperparameter optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 832–840.

[55] A. W. Mohamed, A. Hadi, and A. Khater, "Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm," *International Journal of Machine Learning and Cybernetics*, vol. 11, 07 2020.

[56] A. W. Mohamed, H. F. Abutarboush, A. A. Hadi, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm with adaptive parameters for engineering optimization," *IEEE Access*, vol. 9, pp. 65 934–65 946, 2021.

[57] A. W. Mohamed, A. A. Hadi, P. Agrawal, K. M. Sallam, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm with adaptive parameters hybrid with imode algorithm for solving cec 2021 benchmark problems," *2021 IEEE Congress on Evolutionary Computation (CEC)*, pp. 841–848, 2021.

[58] C. A. C. Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, pp. 1245–1287, 2002.

[59] E. Mezura-Montes and C. A. C. Coello, "Constraint-handling in nature-inspired numerical optimization: past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[60] K. Deb, "An efficient constraint handling method for genetic algorithms," in *Computer Methods in Applied Mechanics and Engineering*, 1998, pp. 311–338.

[61] E. Mezura-Montes, C. A. C. Coello, and E. I. Tun-Morales, "Simple feasibility rules and differential evolution for constrained optimization," in *MICAI*, 2004.

[62] H. K. Singh, K. Alam, and T. Ray, "Use of infeasible solutions during constrained evolutionary search: A short survey," in *Artificial Life and Computational Intelligence*, T. Ray, R. Sarker, and X. Li, Eds. Cham: Springer International Publishing, 2016, pp. 193–205.

[63] T. P. Runarsson and X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 4, pp. 284–294, 2000.

[64] J. Liu, Z. Fan, and E. D. Goodman, "Srade: an adaptive differential evolution based on stochastic ranking," *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, 2009.

[65] R. Liu, Y. Li, W. Zhang, and L. Jiao, "Stochastic ranking based differential evolution algorithm for constrained optimization problem," in *GEC '09*, 2009.

[66] G. Leguizamón and C. A. C. Coello, "A boundary search based aco algorithm coupled with stochastic ranking," *2007 IEEE Congress on Evolutionary Computation*, pp. 165–172, 2007.

[67] T. Takahama and S. Sakai, "Constrained differential evolution with gradient-based mutation and feasible elites," *2006 IEEE International Conference on Evolutionary Computation*, pp. 1–8, 2006.

[68] ——, "Constrained optimization by the ϵ constrained differential evolution with an archive and gradient-based mutation," *IEEE Congress on Evolutionary Computation*, pp. 1–9, 2010.

[69] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 646–657, 2006.

[70] R. Farmani and J. A. Wright, "Self-adaptive fitness formulation for constrained optimization," *IEEE Trans. Evol. Comput.*, vol. 7, pp. 445–455, 2003.

[71] B. G. Tessema and G. G. Yen, "An adaptive penalty formulation for constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 39, pp. 565–578, 2009.

[72] G. Leguizamón and C. A. C. Coello, "Boundary search for constrained numerical optimization problems with an algorithm inspired by the ant colony metaphor," *IEEE Transactions on Evolutionary Computation*, vol. 13, pp. 350–368, 2009.

[73] F. Huang, L. Wang, and Q. He, "A hybrid differential evolution with double populations for constrained optimization," *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 18–25, 2008.

[74] E. F. Wanner, F. G. Guimarães, R. R. Saldanha, R. H. C. Takahashi, and P. J. Fleming, "Constraint quadratic approximation operator for treating equality constraints with genetic algorithms," *2005 IEEE Congress on Evolutionary Computation*, vol. 3, pp. 2255–2262 Vol. 3, 2005.

[75] T. Ray, H. K. Singh, A. Isaacs, and W. F. Smith, "Infeasibility driven evolutionary algorithm for constrained optimization," 2009.

[76] G. Reynoso-Meza, X. B. Ferragud, J. Sanchis, and M. A. Martínez, "Multiobjective optimization algorithm for solving constrained single objective problems," *IEEE Congress on Evolutionary Computation*, pp. 1–7, 2010.

[77] Y. Wang, Z. Cai, G. Guo, and Y. Zhou, "Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, pp. 560–575, 2007.

[78] Y. Wang, Z. Cai, Y. Zhou, and W. Zeng, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, pp. 80–92, 2008.

[79] R. Mallipeddi and P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 561–579, 2010.

[80] S. M. Elsayed, R. A. Sarker, and D. L. Essam, "Integrated strategies differential evolution algorithm with a local search for constrained optimization," *2011 IEEE Congress of Evolutionary Computation (CEC)*, pp. 2618–2625, 2011.

[81] B. Qu and P. N. Suganthan, "Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods," *Engineering Optimization*, vol. 43, pp. 403 – 416, 2011.

[82] M. F. Tasgetiren, P. N. Suganthan, Q. Pan, R. Mallipeddi, and S. Sarman, "An ensemble of differential evolution algorithms for constrained

function optimization," *IEEE Congress on Evolutionary Computation*, pp. 1–8, 2010.

[83] M. Paldrak, M. F. Tasgetiren, P. N. Suganthan, and Q. Pan, "An ensemble of differential evolution algorithms with variable neighborhood search for constrained function optimization," *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2610–2617, 2016.

[84] A. Trivedi, N. Biswas, S. Chakraborty, and D. Srinivasan, "Extending unified differential evolution with a new ensemble of constraint handling techniques," *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8, 2017.

[85] B. Xu and Z. Zhang, "Constrained optimization based on ensemble differential evolution and two-level-based epsilon method," *IEEE Access*, vol. 8, pp. 213 981–213 997, 2020.

[86] X. Wen, G. Wu, M. Fan, R. Wang, and P. N. Suganthan, "Voting-mechanism based ensemble constraint handling technique for real-world single-objective constrained optimization," *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.

[87] T. Faycal and C. Zito, "Direct mutation and crossover in genetic algorithms applied to reinforcement learning tasks," *ArXiv*, vol. abs/2201.04815, 2022.

[88] S. A. Hassan, Y. M. Ayman, K. A. Alnowibet, P. Agrawal, and A. W. Mohamed, "Stochastic travelling advisor problem simulation with a case study: A novel binary gaining-sharing knowledge-based optimization algorithm," *Complex.*, vol. 2020, pp. 6 692 978:1–6 692 978:15, 2020.

[89] P. Agrawal, T. Ganesh, and A. W. Mohamed, "Solution of uncertain solid transportation problem by integer gaining sharing knowledge based optimization algorithm," *2020 International Conference on Computational Performance Evaluation (ComPE)*, pp. 158–162, 2020.

[90] Y. Chen, D. Pi, B. Wang, A. W. Mohamed, and J. Chen, "Equilibrium optimizer with generalized opposition-based learning for multiple unmanned aerial vehicles path planning," 2021.

[91] P. Agrawal, T. Ganesh, and A. W. Mohamed, "Solving knapsack problems using a binary gaining sharing knowledge-based optimization algorithm," *Complex & Intelligent Systems*, pp. 1–21, 2021.

[92] H. A. A. Nomer, K. A. Alnowibet, A. Elsayed, and A. W. Mohamed, "Neural knapsack: A neural network based solver for the knapsack problem," *IEEE Access*, vol. 8, pp. 224 200–224 210, 2020.

[93] Y. Song, D. Wu, A. Mohamed, X. Zhou, B. Zhang, and W. Deng, "Enhanced success history adaptive de for parameter optimization of photovoltaic models," *Complex.*, vol. 2021, pp. 6 660 115:1–6 660 115:22, 2021.

[94] G. Xiong, L. Li, A. W. Mohamed, X. Yuan, and J. Zhang, "A new method for parameter extraction of solar photovoltaic models using gaining–sharing knowledge based algorithm," *Energy Reports*, vol. 7, pp. 3286–3301, 2021.

[95] G. Xiong, X. Yuan, A. W. Mohamed, and J. Zhang, "Fault section diagnosis of power systems with logical operation binary gaining-sharing knowledge-based algorithm," *International Journal of Intelligent Systems*, vol. 37, pp. 1057 – 1080, 2022.

[96] R. Nadakinamani, A. Reyana, S. Kautish, A. S. Vibith, Y. Gupta, S. F. Abdelwahab, and A. W. Mohamed, "Clinical data analysis for prediction of cardiovascular disease using machine learning techniques," *Computational Intelligence and Neuroscience*, vol. 2022, 2022.

[97] H. A. A. Nomer, A. W. Mohamed, and A. H. Yousef, "Gsk-rl: Adaptive gaining-sharing knowledge algorithm using reinforcement learning," *2021 3rd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pp. 169–174, 2021.

[98] Y. Ong, P. B. Nair, A. J. Keane, and K. W. Wong, "Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems," 2005.

[99] B. Devarajan and R. K. Kapania, "Thermal buckling of curvilinearly stiffened laminated composite plates with cutouts using isogeometric analysis," *Composite Structures*, vol. 238, p. 111881, 2020.

[100] B. Devarajan, "Analyzing thermal buckling in curvilinearly stiffened composite plates with arbitrary shaped cutouts using isogeometric level set method," *ArXiv*, vol. abs/2104.05132, 2022.