CHAGSKODE Algorithm for Solving Real World 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 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 power. In particular, Evolutionary Algorithms (EAs) are a preferred tool for practitioners for solving these COPs within practicable time limits. We propose a novel hybrid Evolutionary Algorithm based on the Differential Evolution algorithm and Adaptive Parameter Gaining Sharing Knowledge-based algorithm to solve global real-world constrained parameter space. The proposed CHAGSKODE algorithm leverages the power of multiple adaptation strategies concerning the control parameters, search mechanisms, as well as uses knowledge sharing between junior and senior phases. We test our method on the benchmark functions taken from the CEC2020 special session & competition on real-world constrained optimization. Experimental results indicate that CHAGSKODE 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.**

***Keywords*—Differential Evolution; APGSK algorithm; Constrained Optimization; transformation; parameter adaptation; multi-operator; Evolutionary Algorithms**

I. Introduction

Constrained Optimization Problems (COP) arise in numerous fields, i ncluding n atural s ciences, e conomics, computer, and software engineering.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 (1). These constraints are utilized to discover feasible regions, which might be an ample continuous search space, a significant, u npredictable s pace, 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 1990. 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 1990, several methods for mutation, crossover and selection have been proposed resulting in modified versions of the algorithm [22]. 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. Notwithstanding, no single version of the DE algorithm (or boundary) 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-based DE (MODE) [23] is one that utilizes the strength of more than one DE operator, with more weight given to the better-one during the evolutionary process by progressively refreshing the size of each sub-population dependent on two markers, the nature of solutions and the variety of each sub-populations. We are proposing a hybrid algorithm that constitutes the division of population between a MODE and an Adaptive Parameter Gaining Sharing Knowledge (APGSK) [24] and later recombines the population in later stages of the evolutionary process. APGSK algorithm has recently been shown to perform well in unconstrained applications across various transformations. The inventiveness of this paper includes the pristine hybridization of a traditional DE algorithm and a Gaining Sharing Knowledge Algorithm for solving COP.

Towards the end of each generation, all sub-populations are assembled and afterward arbitrarily redivided depending on the new sub-population sizes. To be sure, to keep up with the variety of the population at the beginning phases of the optimization cycle and accelerate assembly at later ones, a linear decrease of the population size is performed in the DE as standard LSHADE literature [25]. However, we use a non-linear shrinkage of the population size in the sub-population allocated to APGSK [24]. Also, rather than utilizing all the constraints to ascertain the complete *violations*, the algorithm begins with a subset of the most violated constrained are figured out and slowly examines every one of them. The implementation of the proposed algorithm is decided 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 algorithms.

The rest of this paper is coordinated as follows: a brief literature survey on related works is introduced in Section II; the proposed algorithm and subtleties of its parts in Section III; the trial results and examinations in Section IV; and, at last, the conclusion and future work in Section V.

II. Literature Review

DE is a straightforward yet highly efficient 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 with the scaled contrasts of haphazardly chosen individuals and particular population individuals. Over the last couple of decades, DE has been applied to a plethora of real-world problems as well as traditional benchmark problems because of its straightforwardness and robust nature. Intrigued perusers are alluded to [29], [30] for a thorough survey on DE.

The performance of a DE algorithm heavily relies on the choice of control parameters, namely - scaling factor (F) and crossover rate (Cr). It was quite evidently noted that a single value for the control parameters was just not enough hence, researchers opted for self-adaptive versions of DE. [31], [32] Investigation on a multi-operator differential algorithm (MODE) is still at its initial stages. Although there is a large volume of works on MODE for the unconstrained optimization problems, equivalent arrangements with constrained optimization problems are still essentially desolate. Among a few, Elsayed et al. [33] proposed a self-flexible MODE to handle assorted constraint optimization issues. This self-adaptable MODE was furthermore improved by Elsayed et al. [34] with organizing a co-change structure change evolution system.

Another use of MODE can be found from crafted by Li et al. [35], while they had created three distinctive preliminary vectors dependent on three diverse mutation systems. Among those three mutation operators, two were utilized to expand the assembly pace of their proposed DE algorithm, and one operator was utilized to expand variety. With fruitful use of their MODE approach, they showed the matchless quality of MODE over conventional DEs in tackling constrained optimization issues. Additionally, Yu et al. [36] proposed a further developed MODE algorithm, while they had used two distinctive mutation operators to advance the entire population.

Aside from the central optimization engine, constraint handling strategy is the most significant part of the constrained optimization solver. A few constraint handling techniques have been introduced in the literature to stretch out EAs to constrained optimization issues [37]. The most well-known constraint handling methods depend on penalty functions, ϵ-constrained strategy, the superiority of feasible solutions approach, and so forth [37]. As of late, the strategy dependent on saving infeasible solutions in the population has additionally shown a considerable guarantee for handling constrained optimization issues [38].

They had additionally proposed a high-level procedure to change equality constraints into inequality constraints to take care of constrained optimization issues. Some eminent expansions of the MODE approach are unified DE methodology [39], constrained optimization evolutionary algorithm [40], and constrained optimization composite DE algorithm [41]. While every one of those high-level systems had its own benefits, the powerful selection of ideal operators in a DE is as yet a complex exploration setting.

Considering the conspicuous need of having a grounded system to choose best-performing evolutionary algorithm operators, Sallam et al. [42] proposed a landscape-based marker to pick the best-appropriate operator all through the evolutionary cycle of DE. Recently, Kumar et al. [43] proposed an upgraded rendition of the MODE algorithm, in which they had 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. [44] 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. More recently, Sallam et al. [23] proposed a novel EnMODE that uses three mutation operators, two crossover operators (binomial and exponential), as well as uses a probabilistic approach in selecting the operators. This EnMODE algorithm has won the CEC 2020 Real World Constrained Optimization competition.

Recently, Ali et al. [45] proposed a novel GSK algorithm which is a nature-motivated algorithm based on the conduct of people that has two essential stages for acquiring, and sharing knowledge: junior and senior. It is motivated from human behavior of knowledge acquisition from elementary stages (junior) to stages when they become highly competent (senior). These highly knowledged seniors tend to share their knowledge to the juniors to level their growth. Notwithstanding, they may not be capable or group the foundation and experience to arrange the individuals in their environmental factors. Under a similar idea, seniors attempt to work on their knowledge by collaborating with more extensive resources, including companions, collaborators, juniors, etc. Ali proposed an adaptive version of the GSK (AGSK) [24] where the control parameters for the original GSK [45] were dynamically adapted. AGSK also secured second position in CEC 2020 bound constrained optimization competition. The itemized depiction of the algorithm can be found in Algorithm **??**

In view of these pertinent writing studies, it very well may be guaranteed that the inquiry of discovering a proper selection of best-performing blend of operators and parameters for any evolutionary algorithm (say DE) is as 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: (i) how to progressively balance the size of sub-populations dependent on the nature of solutions and the variety of sub-populations and maintain overall population

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0314.v1

size, (ii) how to effectively utilize diverse operator's influence during the evolutionary interaction of a DE. (iii) are there any advantages of using a multi-level algorithm as opposed to one single algorithm Thinking about these as a difficult exploration setting, this paper proposes an upgraded MODE (CHAGSKODE) algorithm, which uses a hybrid of two different algorithms merged together only separated by population *division-recombination* strategy. It refreshes the population size progressively dependent on solution quality and variety of sub-populations and uses an information-sharing criteria from the senior to the junior solutions.

III. Proposed Algorithm

The CHAGSKODE algorithm is intended to address real-world constrained optimization issue. A preliminary solution to the issue is addressed as a segment vector $\vec{X} = [x_1, x_2, \ldots, x_D]^T$, where D is the problem dimensionality 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. Typically, an objective function $f(\vec{X})$, which estimates either the exhibition (or cost) of a framework, is made accessible. We are entrusted with looking for a solution $\vec{X}^*$ that minimizes the objective function, $f(\vec{X}^*) \leq f(\vec{X})\ \forall \vec{X} \in \Omega \subseteq \Re^D$ in feasible region, where Ω is a limited set addressing the pursuit area.

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

For infeasible solutions we will try to bring them into feasible regions using constraint handling by selectively incorporating treatment in an increasing order of the number of most violated constraints. Here we expect that we are given a greatest assessment budget of MAX_{FES} to address the optimization issue. In depicting the means of our algorithm up next, we use phrasing and the algorithmic documentation steady with the standard DE writing.

A. MODE Algorithm

As recently expressed, the general execution of the DE operator might change during the optimization stages; as such, the performance of one DE operator might be acceptable at the beginning stages of the optimization cycle and inefficient at the later ones, or the other way around. Additionally, an operator might perform well for specific kinds of problems, yet its performance might be inadequate for another. Therefore, the utilization of multi-operator DE is needed with putting more weight on the better-performing one at each evolutionary stage. The psuedo-code of the IMODE algorithm can be depicted in Algorithm **??**

1) DE Mutation: In CHAGSKODE, 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 [46], i.e., DE/current-to-ϕbest/1 mutation with (2) and without archive (3). 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 was introduced by Elsayed et al. [47] which is a DE/weighted-rand-to-ϕbest operator (4). The three mutation strategies can be listed as below:

- DE/current-to-ϕbest/1 + archive

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F_i \cdot (\vec{X}_{\phi,G} - \vec{X}_{i,G} + \vec{X}_{r_1^i,G} - \vec{X}_{r_3^i,G}), \tag{2}$$

- DE/current-to-ϕbest/1 − archive

$$\vec{V}_{i,G} = \vec{X}_{i,G} + F_i \cdot (\vec{X}_{\phi,G} - \vec{X}_{i,G} + \vec{X}_{r_1^i,G} - \vec{X}_{r_2^i,G}), \tag{3}$$

- DE/weighted-rand-to-ϕbest

$$\vec{V}_{i,G} = F_i \cdot \vec{X}_{r_1^i,G} + (\vec{X}_{\phi,G} - \vec{X}_{r_3^i,G}) \tag{4}$$

where $r_1^i \neq r_2^i \neq r_3^i$ are randomly selected integers, based on which $\vec{X}_{r_1^i,G}$, $\vec{X}_{r_2^i,G}$ are chosen from current population pool (P) and $\vec{X}_{\phi,G}$ is chosen from top $\phi\%$ solutions. The solutions that fail the selection test are stored in an external archive A, which is kept separate from the current population P. These substandard solutions aid in the preservation of the population variance in the environment. $\vec{X}_{r_3^i,G}$ is chosen from the pool consisting of the archive and current population $A \cup P$.

2) DE Control Parameters: The control parameter adaption is enlivened from the success history-based hyperparameter adaptation [25], [46], [48]. It keeps a memory archive of the effective DE hyperparameters, to be specific, scaling factor F in Eq (5) and crossover rate Cr in Eq (6). The memory archives $\mathbf{M}_F$ and $\mathbf{M}_{Cr}$ are initialized to values F_0 and Cr_0, respectively, i.e., $M_{F,i} = F_0, i = 1, \ldots, H$, and $M_{Cr,i} = Cr_0$, $i = 1, \ldots, H$. The memory archives are used to generate the hyperparameters for producing the trial vectors as follows:

$$F_i \sim randc_i\left(M_{F,r_i}, 0.1\right) \quad \text{and} \tag{5}$$

$$Cr_i \sim randc_i\left(M_{Cr,r_i}, 0.1\right). \tag{6}$$

The scaling factor F_i is sampled using Cauchy distribution to guarantee a wide spread of values. F_i is truncated to 1 in case $F_i > 1$, and is regenerated if $F_i < 0$ till a valid value of $F_i \in (0, 1]$ is produced. The crossover rate Cr is also sampled from the Cauchy distribution and truncated to lie in the range $[0, 1]$. The index r_i is uniformly randomly from the range $[1, H]$.

3) DE Crossover Rate: During crossover, to frame the donor vector $\vec{V}_{i,G}$ the mutation activity trades its parts with the objective vector $\vec{X}_{i,G}$ generates the trial vector $\vec{U}_{i,G}$. Crossover goes about as a variety safeguarding system by limiting the alterations in the trial vector. The CHAGSKODE algorithm uses both these crossover operations in a random order as depicted in Eq (7)

$$U_{i,j} = \begin{cases} \begin{cases} V_{i,j} & \text{for } j = \langle l \rangle_D, \langle l+1 \rangle, \ldots, \langle l+L+1 \rangle_D \\ X_{i.j} & \text{for all other } j \in [\,1, D] \end{cases} \\ \text{if } rand \leq 0.5 \\ \begin{cases} V_{i,j} & \text{if } (rand \leq Cr_i \text{or } j = j_{rand}) \\ X_{i.j} & \text{otherwise} \end{cases} \\ \text{otherwise} \end{cases} \tag{7}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0314.v1

B. APGSK Algorithm

The nature inspired APGSK Algorithm mimics human behavior of acquiring knowledge and offering it to others while communicating their own perspectives. They learn in their beginning phases through their small informal organization, which incorporates individuals from their neighbors, family members, families, etc. Then, at that point, they need to share their recently obtained knowledge and thoughts with individuals who are not really from their nearby associations, attributable to their anxiety of looking through others in the whole population.

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 pool is used to characterize the parameter's values made out of the accompanying two sets (k_f, k_r): [(0.1, 0.2), (0.5, 0.9), (1.0, 0.9) and (1.0, 0.1)] which is utilized for the first 50% of MAX_{FES}. While different sets: [(- 0.15, 0.2), (- 0.05, 0.9), (- 0.15, 0.9) and (- 0.05, 0.1)] applied in the leftover half of MAX_{FES} with a probability under 0.3. This is done to help the population diversity to diminish stagnation plausibility and to escape from neighborhood optima. Utilizing this plan, a probability parameter Kw_P that has a likelihood parameter p for each setting of the previously mentioned pool of settings which has been performed. Subsequently, every part in the whole population will have just one setting dependent on p. To set the worth of Kw_P, it begins after $0.1\times FES$ dependent on the performance of each setting utilizing Eq (8):

$$\omega_{ps} = \sum_{i=1}^{n} f(x_i^{new}) - f(x_i^{old}) \tag{8}$$

where ω_{ps} is summed difference between the new solution x_i^{new} and the old solution x_i^{old} fitness value for every n solutions in ps. Following that, the rate of improvement for each parametric setting is calculated as Eq (9):

$$\Delta_{ps} = \max(0.05, \frac{\omega_{ps}}{\text{sum}(\omega_{ps})}) \tag{9}$$

where 0.05 is the base probability allotted to each parameter set to ensure that all settings have a selection probability. At last, the rate of improvement (Δ_{ps}) for each parameter set is used to refresh Kw_P using Eq (10):

$$Kw_P^{G+1} = (1-c)Kw_P^G + c \cdot \Delta_{ps} \tag{10}$$

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 (11) as:

$$\text{if } rand \geq \frac{FES}{MAX_{FES}}, k = 0.5 \text{ else } k = 2 \tag{11}$$

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 (12)

$$NP_G = \left\lfloor NP_{\max} - (NP_{\max} - NP_{\min}) \frac{FES}{MAX_{FES}} \right\rceil, \tag{12}$$

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 (13):

$$NP_G = \lfloor NP_{init} + (NP_{\min} - NP_{init}) * \left(\frac{FES}{MAX_{FES}}\right)^{\left(1-\frac{FES}{MAX_{FES}}\right)} \rceil, \tag{13}$$

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

In this paper, we utilized the constrained handling method proposed by Elsayed in [44]. It begins by arranging the constraints concurring to the summed amount of the constraints violations of all solutions in the introductory population from the most un-violated to the most violated constraint or the other way around. Notably, the more constraints exist in any issue; the more perplexing will be that issue. Rather than utilizing all constraints, the algorithm begins with a subset of the constraints (Conn) for a predefined number of generations (W). After W generations, another subset is then added to the current one, and the technique attempts to accomplish the feasible regions of both the new and past generations of constraints. The interaction goes on until every one of the constraints are dealt with and the last feasible region is reached.

$$\psi(\vec{X}_i) = \sum_{k=1}^{K} \max\left(0, g_k(\vec{X}_i)\right) + \sum_{e=1}^{E} \max\left(0, |h_e(\vec{X}_i)| - \delta_e\right) \tag{14}$$

To choose between any solution and its parent, the technique created by Deb [49] is embraced, and it has three situations:

- from two feasible individuals, the one with the best function esteem is chosen;
- from two infeasible individuals, the one with the littlest amount of constraint violations (ψ) is picked, where ψ is figured utilizing Eq. (14) and
- a feasible solution is in every case better compared to an infeasible one.

E. Putting it all together

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0314.v1

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**

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**

IV. Experimental Results

A. Benchmark

To pass judgment on the presentation of the proposed CHAGSKODE 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 [43].

These issues are classified to six sets:

- Industrial Chemical Processes contains seven test problems (RC01 - RC07);
- Process Synthesis and Design Problems has seven test problems (RC08 - RC14);
- Mechanical Engineering Problem includes 19 test problems (RC15 - RC33);
- Power System Problems has 11 test problem (RC34 - RC44);
- Power Electronic Problems contains six test problems (RC45 - RC50);
- Livestock Feed Ration Optimization contains seven test problems (RC51 - RC57);

Algorithm 3 CHAGSKODEAlgorithm

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
1: Generate initial random population of size NP
Output: Evaluate $f(X)$, and calculate fitness evaluation
2: Calculate for mean constraint violations for all (g_n) equality and (h_n) in-equality constraints
3: Calculate mean violation selection $mean_{viol} \leftarrow \frac{\text{violation}}{g_n + h_n}$
4: Select counter for treatment of violation
5: Randomly allocate population into PS_1 and PS_2 with default ratio of 75% to 25% at the beginning
6: **while** $FES \leq MAX_{FES}$ **do**
7: $count \leftarrow count + 1$
8: **if** $count == CS$ **then**
9: Pr_1 and Pr_2 as stated in
10: **end if**
11: **if** $count == 2 \times CS$ **then**
12: Share the best solution to both subpopulation
13: Reset Pr_1, Pr_2 and $count$
14: **end if**
15: **if** $rand \in [0, 1] \leq Pr_1$ **then**
16: Apply IMODE and optimize the sub-population
17: Use the assigned DE mutation strategies to generate offsprings where every DE mutation strategy (operator) op optimizes determined solutions NP_{op}
18: Evaluate for fitness $f(X)$ and update fitness evaluation $FES \leftarrow FES + PS_1$
19: Update the solutions and return the best solutions.
20: **end if**
21: **if** $rand \in [0, 1] \leq Pr_2$ **then**
22: Apply APGSK and optimize the sub-population
23: Update fitness evaluation $FES \leftarrow FES + PS_2$
24: Update the solutions and return the best solutions.
25: **end if**
26: Recombine the solutions from the both the algorithms $PS \leftarrow PS_1 + PS_2$
27: Check for constraint violation and calculate mean violation following the evolution process
28: **end while**

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

 doi:10.20944/preprints202208.0314.v1

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

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 CHAGSKODE 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 CHAGSKODE 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. Also, we would like to investigate the performance of the algorithm across a wide range of transformation operations like bias, shift, rotation, etc.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0314.v1

TABLE I: Table of Outcomes for RC01-RC08

Criteria		RC01	RC02	RC03	RC04	RC05	RC06	RC07	RC08
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 II: Table of Outcomes for RC09-RC16

Criteria		RC09	RC10	RC11	RC12	RC13	RC14	RC15	RC16
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 III: Table of Outcomes for RC17-RC24

Criteria		RC17	RC18	RC19	RC20	RC21	RC22	RC23	RC24
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 IV: Table of Outcomes for RC25-RC32

Criteria		RC25	RC26	RC27	RC28	RC29	RC30	RC31	RC32
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									

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 August 2022 doi:10.20944/preprints202208.0314.v1

TABLE V: Table of Outcomes for RC33-RC40

Criteria		RC33	RC34	RC35	RC36	RC37	RC38	RC39	RC40
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 VI: Table of Outcomes for RC41-RC48

Criteria		RC41	RC42	RC43	RC44	RC45	RC46	RC47	RC48
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 VII: Table of Outcomes for RC49-RC57

Criteria		RC49	RC50	RC51	RC52	RC53	RC54	RC55	RC56	RC57
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										

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 multi-functional 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, "Multi-operator differential evolution algorithm for solving real-world constrained optimization problems," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, pp. 1–8.

[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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650219308946

[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. 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.

[32] 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.

[33] 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. [Online]. Available: https://doi.org/10.1016/j.cor.2011.03.003

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

[35] W. Li, S. Li, Z. Chen, L. Zhong, and C. Ouyang, "Self-feedback differential evolution adapting to fitness landscape characteristics," *Soft Comput.*, vol. 23, no. 4, p. 1151–1163, Feb. 2019. [Online]. Available: https://doi.org/10.1007/s00500-017-2833-y

[36] X. Yu, Y. Lu, X. Wang, X. Luo, and M. Cai, "An effective improved differential evolution algorithm to solve constrained optimization problems," *Soft Comput.*, vol. 23, no. 7, p. 2409–2427, Apr. 2019. [Online]. Available: https://doi.org/10.1007/s00500-017-2936-5

[37] "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm and Evolutionary Computation*, vol. 1, no. 4, pp. 173–194, 2011.

[38] 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.

[39] A. Trivedi, K. Sanyal, P. Verma, and D. Srinivasan, "A unified differential evolution algorithm for constrained optimization problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 1231–1238.

[40] B. Xu, X. Chen, and L. Tao, "Differential evolution with adaptive trial vector generation strategy and cluster-replacement-based feasibility rule for constrained optimization," *Information Sciences*, vol. 435, 04 2018.

[41] B.-C. Wang, H.-X. Li, J.-P. Li, and Y. Wang, "Composite differential evolution for constrained evolutionary optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 7, pp. 1482–1495, 2019.

[42] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741741930750X

[43] 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.

[44] S. Elsayed, R. Sarker, C. C. Coello, and T. Ray, "Adaptation of operators and continuous control parameters in differential evolution for constrained optimization," vol. 22, no. 19, 2018. [Online]. Available: https://doi.org/10.1007/s00500-017-2712-6

[45] A. Wagdy, 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.

[46] 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.

[47] S. Elsayed, N. Hamza, and R. Sarker, "Testing united multi-operator evolutionary algorithms-ii on single objective optimization problems," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 2966–2973.

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

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