Iranian Journal of Numerical Analysis and Optimization
Vol. 15, No. 4, 2025, pp 1538–1588
https://doi.org/10.22067/ijnao.2025.93363.1640
https://ijnao.um.ac.ir/

Research Article

Combining an interval approach with a heuristic to solve constrained and engineering design problems

D. Sharma*, and S.D. Jabeen

Abstract

Solving intricate constrained optimization problems with nonlinear constraints is usually difficult. To optimize the constraint and structure engineering design challenges, this work presents a novel hybrid method called SDDS-SABC, which is based on the split-detect-discard-shrink technique and the Sophisticated ABC algorithm inspired by the integration of branch-and-bound-like concepts of interval analysis with heuristics, and it differs from other methods in the literature. The advantage of the SDDS process is that it shrinks the entire search region through recursive breakdown and improves computational effort to focus on subregions covering potential solutions for further decomposition. In order to identify the most promising

*Corresponding author

Received 6 May 2025; revised 8 August 2025; accepted 4 September 2025

Dhirendra Sharma
Department of Mathematics, Birla Institute of Technology Mesra, Ranchi 835215, India, e-mail: dhirendrasharma428@gmail.com

Syeda Darakhshan Jabeen
Department of Mathematics, Birla Institute of Technology Mesra, Ranchi 835215, India, e-mail: syed_sdj@yahoo.co.in

How to cite this article
Sharma, D. and Jabeen, S.D., Combining an interval approach with a heuristic to solve constrained and engineering design problems. *Iran. J. Numer. Anal. Optim.*, 2025; 15(4): 1538-1588. https://doi.org/10.22067/ijnao.2025.93363.1640

subregion, SABC's values are crucial in assisting in the extraction of the best solutions from the subregions. Until the region shrinks to a nominal width that represents the global or nearly global solution(s) to the optimization problem, both SDDS and SABC are successively repeated. The selection and rating criteria are used to support positive decision-making, with the mindset of removing the subregion containing the unpromising solution(s). Simultaneously, the subregion exhibiting a viable solution is acknowledged as the present shrink region in anticipation of a subsequent split. We present a new initialization technique for food sources in the SABC algorithm, called the quasi-random sequence-based Halton set, which outperforms the current initialization procedure. Create a composite strategy that uses the employed bee phase to investigate their neighborhood while preserving their cooperative nature. In order to increase the optimization efficiency, we also present a new dynamic penalty approach that does not rely on any additional characteristics or factors like the majority of existing penalty methods. We test the statistical validity of SDDS-SABC by applying it to engineering design problems and benchmark functions (CEC 2006). The results demonstrate that SDDS-SABC performs better than its most studied competitors and proves its viability in resolving difficult real-life problems. Additionally, the SDDS-SABC approach is appropriate and numerically stable for the optimization problems. The main innovation of the approach being described is its capacity to perform a static and better optimal solution in the majority of runs, even when the problem is excessively complex.

AMS subject classifications (2020): 65K10, 90C26, 90C31, 90C59

Keywords: Constrained optimization, ABC algorithm, Dynamic penalty function, Engineering design problem

1 Introduction

Optimization is a numerical approach to solving practical problems in a variety of domains and is a difficult decision-making process that keeps getting difficult. The goal of decision-making is to, given the situation, determine the best set of variables to combine in order to maximize or decrease the objective function within the specified bounds. Sustainable restrictions mean that

it is usually more difficult to solve scientific and engineering problems when the search space structure is limited to regions that are both feasible and nonfeasible. This makes it challenging to extract the best feasible solution from a subset of the feasible space. Constrained optimization problems are what these problems are known as (COPs). In general, typical mathematical programming techniques found in literature cannot address nonconvex or discrete problems since they require gradient information in order to find optimal solutions. They are also vulnerable to starting points. When the optimization problems feature several or impulsive peaks, selecting the initial points incorrectly makes the search for the global optimum difficult and unstable. Furthermore, as the dimension of the choice variable increases and optimization problems become extremely nonlinear, conventional methods break down if the problem's complexity rises any further. Using sophisticated, effective algorithms that possess derivation-free formulations, simplicity, and flexibility is the best option. These algorithms can offer excellent results in various real-world optimization scenarios. Numerous nature-inspired intelligent optimization strategies and constraint-handling methodologies have been developed to address these kinds of problems [2]. Particle swarm optimization (PSO) [27], ant colony optimization (ACO) [13], artificial bee colony (ABC) [26], grey wolf optimizer (GWO) [41], and other swarm-based algorithms are popular. The Darwinian evolution theory underpins evolutionary algorithms such as genetic algorithm (GA) [42], differential evolution (DE) [48], and Memetic algorithm [15], among others. The simulated annealing (SA) [4], gravitational search algorithm (GSA) [50], big-bang big-crunch algorithm (BBBC) [52], and other chemistry- or physics-based methods are inspired by chemical or physical phenomena. The last type of algorithms is social or human-based ones, such as the arithmetic optimization algorithm (AOA) [1], teaching learning-based optimization (TLBO) [54], and brain storm optimization (BSO) [57]. These algorithms are inspired by human or social behaviors. These algorithms' drawbacks include their propensity for local convergence, need for parameter tweaking, and so on. They perform so poorly in the majority of optimization problems. Furthermore, the majority of these techniques is designed to address unconstrained optimization problems. Several techniques for handling constraints are integrated into these algorithms to

handle restricted optimization problems. During the algorithm's iterative process, these techniques direct the population of solutions towards the more feasible region. Furthermore, the aforementioned significant shortcomings of the heuristic algorithms prompted the necessity of creating a more sophisticated version of the algorithms in terms of computing efficiency and solution quality. Therefore, obtaining more robust algorithms, fusing local search methods combined with additional heuristic approaches, or combining multiple heuristic methods captured the researcher's interest to form hybrid algorithms (see [14, 2, 58, 20, 21]). Hybridization of PSO with GWO [55], GA with GSA [16], Cuckoo Search (CS) with DE [67], hybrid firefly algorithm with grouping attraction (HFA-GA) [8], an improved firefly algorithm (UFA) [6], an enhanced leadership-based GWO (GLF-GWO) [19], GGA with the gradient-descent (GD) [10], PSO with DE [32], and many more have been recently proposed hybrid algorithms. Additionally, other approaches to managing constraints have been proposed, such as (1) penalty function methods [64, 60] (2) handling the goal function and constraints independently [38, 45, 12, 53, 59] (3) Hybrid approaches [45, 38] and (4) multiobjective-optimization methods [62, 23]. Using penalty functions, such as the death penalty, static penalties, dynamic penalties, annealing penalties, adaptive penalties, and co-evolutionary penalties, is a common and simple method of handling constraints. These functions convert restricted problems into unconstrained ones. The majority of these constraint handling strategies have significant disadvantages. Certain methods may yield an unfeasible solution or necessitate numerous more factors with uncertain values; still others are situation-specific, meaning that a special approach must be developed for a given problem. By lowering their fitness values in proportion to the degrees of constraint violation, these techniques penalize impractical solutions. Furthermore, the majority of these penalty systems have parameters that need to be carefully experimented with in order to determine the proper values in order to produce workable solutions. Certain problems may respond well to the specified parameter values, while other problems may not respond well to them. In order to address this reliance of algorithm performance on penalty parameters, scholars have developed advanced penalty function methodologies.

It is worth questioning what the better sampling methods are that could be employed for generating initial solutions in evolutionary algorithms and what the appropriate penalty function value is to improve the advancement of solutions towards feasibility. It is usually more difficult to solve scientific and engineering problems when the search space structure is limited to regions that are both feasible and nonfeasible. This makes it challenging to extract the best feasible solution from a subset of the feasible space. The objective of this study is to present a hybrid optimization approach combining an interval approach with the ABC algorithm to solve the benchmark constrained optimization and engineering design problems. The aims of this work are to implement a new initialization method in the ABC algorithm and formulate a new dynamic penalty function formula to handle the constraints of the benchmark optimization problems.

The No Free Lunch idea is supported by the fact that, despite the fact that numerous heuristic algorithms have been created; they are all plagued by the inability to effectively address every optimization problem that is presented to us. Furthermore, research demonstrated that certain algorithms yield more optimal outcomes than others. Thus, creating an enhanced heuristic method for various optimization issues remains an unresolved matter and is greatly appreciated by scholars, provided that they provide a noteworthy contribution to the domain. This encourages us to present a novel, efficient heuristic algorithm for solving COPs that differs from the one seen in the literature.

We present a novel hybrid optimization technique in this work, termed "(SDDS-SABC)", that combines two phases: Phase 1 involves the split-detect-discard-shrink (SDDS) strategy, and Phase 2 involves the sophisticated artificial bee colony (SABC) algorithm being executed. The fundamental principle of the SDDS is to divide the whole of n-dimensional Euclidean space into two subregions of a specific form by first splitting near the first variable's midpoint on its axis. Every subregion has undergone the SABC phase in order to identify and eliminate any subregions that cover unpromising solutions. By analyzing the two solutions that SABC created in each subregion and selecting the subregion with the most promising solution, the interval arithmetic rule has been utilized to identify the subregions that

can be rejected. The hybrid algorithm's first cycle is now finished. Through repetitive switching of the SDDS and SABC phases, the approach explores the search for the region of promise. The ith variable's axis is split about the chosen subregion's midpoint in n-dimensional Euclidean space during the ith cycle SDDS phase. During each splitting, the selected subregion is split into two subregions, and ABC is implemented for every subregion. The procedure for splitting, detecting, discarding, and shrinking the chosen subregion is repeated several times using SDDS and SABC, concentrating computing effort on the promising subregion each time. When the reduced zone approaches negligible width, the cycle count comes to an end. This condition means that the global optimum, or something close to it, has been identified. This improves the search space's exploitation potential while enhancing the ability to explore high-quality solutions and combine algorithmic strength. A variety of numerical test issues and engineering design difficulties have been resolved in order to assess the effectiveness of the suggested methodology. The Friedman and Wilcoxon test was used to compare the suggested SDDS-SABC algorithm to other cutting-edge algorithms.

Our results demonstrate that for most benchmark functions in the domain of wide and restricted search spaces. We find that our hybrid methodology outperforms the majority of recently developed techniques.

2 Novelty and contributions of our proposed method

We propose a novel hybrid algorithm called SDDS-SABC to address the limitations of the existing heuristic algorithms and the penalty function. In the following paragraphs, we highlight our paper's novelty and contributions.

a. In this paper, we provide a novel hybrid algorithm that combines the ideas of interval analysis and heuristics to solve intricate restricted optimization problems. The goal of this integration is to increase algorithms' robustness, convergence rate, and solution quality while guiding them toward an efficient, effective, and robust search.

b. We present a quasi-random formulation for initialization after recognizing the drawbacks of random, chaotic, or logistic initialization of the food source. In this case, we create an initial population that is more uniformly dispersed

over the search area, which may lead to more reliable results.

c. To increase the algorithm's adaptability, the composite strategy is employed the bee stage to introduce a two-way search space exploration.

d. We present a new dynamic penalty approach that is straightforward in form and does not require any additional parameters or penalty factors in order to address the drawbacks of the current penalty factors. In order to increase the optimization efficiency, the burden of fine-tuning the penalty factors and parameters has been eliminated. It has not yet been documented in the literature that these three key features were used in the heuristic algorithm's creation.

e. Analyze our proposed hybrid algorithm's effectiveness through comprehensive numerical testing on benchmark CEC 2006 and some engineering design problems in MATLAB.

f. Compare our SDDS-SABC hybrid algorithm with other state-of-the-art algorithms using the Friedman and Wilcoxon test and demonstrate its statistical performance.

g. Our results demonstrate that our method works well for the majority of benchmark optimization problems in the domain of wide and narrow search spaces. The proposed approach performs more accurately than the most recent methods.

3 An overview of the hybrid ABC algorithm

Karaboga introduced the ABC algorithm, a nature-inspired stochastic optimization technique based on swarm intelligence, in 2005 (see[24]). The clever way that honey bees look for food and communicate that knowledge to other bees in their hive has served as inspiration for the algorithm. The technique was originally designed to solve unconstrained optimization problems. On the one hand, its capacity to resolve a wide range of multidimensional and multimodal real-world optimization problems has drawn a lot of interest since its inception. However, it also had some significant drawbacks, including a slow rate of convergence, inept exploration, limited exploitation, and a propensity to become trapped in local optima. The algorithm has been found to be better than other algorithms despite its drawbacks due of its adaptabil-

ity, simplicity, resilience, and requirement for a smaller number of training parameters. Therefore, it is easier to combine it with multiple algorithms. Considering its benefits and shortcomings, academics have been inspired to expand, alter, or combine ABC with different population-based algorithms or traditional techniques in order to improve its efficiency. The study expanded the scope of hybrid ABC algorithm development beyond numerical COPs to include a broad spectrum of application-based problem optimization.

For example, in order to deal with restrictions, Karaboga and Akay [25] devised an updated ABC to solve COPs, applying Deb's feasibility-based tournament selection operator criteria [25]. Changes have been made to ABC's scout bee operator and selection mechanism by Mezura-Montes and Cetina-Domínguez [40]. They dealt with using the search area confined by the equality and inequality criteria by using the dynamic tolerance property and tournament selection. In order to improve exploitation, Brajevic [5] suggested updating the ABC algorithm and changing the phases of the employed and scout bees. Deb's feasibility-based principles helped them keep the limitations under control. Once more, Li and Yin [28] have presented a self-adaptive constrained ABC algorithm (SACABC) based on the feasible rule approach and multiobjective optimization technique. The employed bees produced better results in their method by using the new search scheme that adheres to the feasible rule. To further investigate the new search area, the observer bees employed an improved search approach based on the multiobjective optimization problem technique. Brajevic and Tuba [7] proposed an upgraded ABC algorithm and modified employed and scout bee's phases for better exploitation. They used Deb's feasibility-based rules to manage the constraints. Furthermore, Brajevic [5] has presented a new version of the crossover-based ABC method, called CBABC, to solve constrained optimization issues.

Two distinct formulas for inequality and equality constraints were established in order to address border restrictions and dynamic tolerance. Conversely, Deb's feasibility-based rules have been loosened in the improved ABC (IABC) algorithm suggested [30] by approximating feasible solutions to a better objective function value with a slight violation. Inspired by the gbest-guided ABC (GABC) method, they have also developed a new search

technique to improve exploitation, utilizing the most optimal solution's data. It has been pointed out by Liu et al. [33] that Deb's feasibility-based rules may result in premature convergence, especially for the problems with an equality constraint. In order to address limitations, Long et al. [37] have developed a unique constrained optimization technique called IABC-MAL. This method combines the advantages of the modified augmented Lagrangian (MAL) method with IABC algorithm capability for achieving the global optimum. The first attempt to combine the augmented Lagrangian approach and the ABC algorithm is presented in this publication. For restricted optimization problems, Bansal, Joshi, and Sharma [3] suggested modifying GABC (MGABC). In their work, GABC [68] is adjusted by introducing the idea of fitness probability-based individual mobility in both the employed and onlooker bee phases. This inspired them to suggest a brand-new dynamic penalty function and an ABC-based Levy flight algorithm (DPLABC) for resolving the COPs.

To expedite the local search, they have used a dynamic logistic map in conjunction with the Levy flying technique with the used bee phase. Wang and Kong [63] have discussed the enhanced artificial bee colony (EABC) algorithm and its application in solving optimization problems. The algorithm is compared to other variants of the ABC algorithm on various test functions and engineering optimization problems. Phoemphon [46] has introduced grouping and reflection of the artificial bee colony, a distinctive adaptation of the traditional ABC algorithm meticulously tailored to meet the specific demands of high-dimensional numerical optimization problems by balancing exploration and exploitation processes. Patra et al. [44] have presented an efficient multi-objective optimization approach utilizing the ABC algorithm for minimizing generation fuel cost and transmission loss through the optimal placement and sizing of flexible AC transmission system (FACTS) controllers. Liu et al. [36] have developed a learning-based ABC algorithm by integrating deep reinforcement learning for operation optimization in gas pipelines. In addition to the aforementioned algorithms, some hybrid ABC algorithms were specially created by researchers to address real-world application problems in the fields of economics (ABC with CMA-ES [66]), industrial engineering, electrical engineering [39], and mechanical engineering (ABC with LS-SVM

[18]); inventory model (ABC with GA [47]; DE with ABC, [9]; sheduling (HABC [17]); cluster analysis (PSO-ABC [49]); routing problem [51, 22]; and wireless network [61, 65, 43].

4 Sophisticated artificial bee colony (SABC) algorithm

Based on the fundamentals of Karaboga's ABC algorithm, we provide a SABC algorithm that includes changes to the initialization procedure, used, and scout bees search approach, all of which we will cover in the upcoming subsections. According to the idea, one potential solution to the optimization problem is to locate the food sources. The associated solution's fitness and the objective function are represented by the amount of nectar present in the food supply. Finding the food source with the most nectar is the goal (optimal solution). The employed bee, spectator bee, and scout bee are the three groups into which the SABC algorithm divides the bees in order to do this. There are an equal number of food sources, working bees, and bystander bees. Each bee group's participation is crucial for producing higher-quality honey. In order to reach the optimum, the mathematical formula used by the bees in a new food location update must be sufficiently competitive. In order to draw in other interested bees, employed bees search for food sources and disseminate information about them. Assuming a probability related to the quality of the food sources, observer bees follow and utilize the food sources found by all working bees. The hired bees, known as scouts, abandon a food source and look for other sources if a solution matching that food supply is not improved by a particular number of limits. Each step of the SABC process is explained in depth in the algorithm below:

Algorithm-1:
begin
Define it, $MaxIt$ and MNC as current iteration numbers, maximum iteration number and maximum number of cycle, respectively.
Compute initial population of probable solutions $x_{i,j}$ of popsize SN ($i = 1, 2, \ldots, SN, j = 1, 2, \ldots, n$) using our proposed quasi-random method (see Sec 4.2) and calculate their fitness value.

repeat

begin Employed bee's stage:

The Employed bee's, produce new solutions $v_{i,j}$ using our proposed strategy search techniques. (see Sec. 4.3).

Apply greedy selection

Memorise the best solution achieved so far

end

begin Onlooker bee's stage:

Compute the probability value for the new found ith solution using the formula:

$$p_i = \frac{fit_i(x_i)}{\sum_{i=1}^{SN} fit_i(x_i)},$$

where

$$fit_i(x_i) = \begin{cases} 1 + |f(x_i)| & \text{if } f(x_i) \leq 0, \\ \frac{1}{1+f(x_i)} & \text{if } f(x_i) \geq 0. \end{cases}$$

is the fitness value of the ith solution and $f(x_i)$ is the objective function value of the solution x_i.

Using Roulette wheel selection to produce a new solution:

$$v_{i,j} = x_{i,j} + \sin(i - \frac{it}{MaxIt}) * (x_{i,j} - x_{k,j}) \qquad (i \neq j)$$

Compute the fitness value and apply greedy selection

end

begin Scout bee's stage:

if $x_{i,j}$ remains same till max limit is reached abandon $x_{i,j}$ using our proposed scout bees formula (see sec. 4.4)

end

Compute the fitness of new found solutions

Memorize the best found solution achieved so far

end

Update the best found solution.

Until

predefined MNC is reached.

4.1 ABC parameters and the effect of the algorithm's initial solutions

We are well aware that ABC conducts excellent exploration but subpar exploitation. Furthermore, for certain complicated functions, it can become caught in local optima and has a rapid convergence rate. The ABC Algorithm's convergence depends heavily on the parameters needed to run it; hence, they must be carefully chosen. These include the population size or popsize (SN), the maximum number of cycles (MNC), the limit value (L) for giving up the food source, and the first potential solution (s) or location of food sources (n). Selecting these parameters incorrectly can lead to pre-convergence or the convergence to an optimal solution at a higher computational cost. ABC is also a black-box optimizer. As a result, when optimizing complex functions, it is impossible to pinpoint the ideal solution's location within the problem's search space. Consequently, these solutions will be improved iteratively by the ABC optimization process's steps until a stopping condition is satisfied, regardless of how well the initial population guess turned out. Generally speaking, accurate first guesses can facilitate the algorithm's search for the optima. Conversely, if poor predictions are made at the beginning, then the algorithm might not be able to discover the global optima. In these circumstances, scientists can decide to employ a sophisticated initialization procedure to produce a diversified initial population that spreads widely and covers interesting areas of the search space that may include good local optima or potential global optima. Furthermore, by improving the methods used by employed bees, observer bees, and scout bees to produce new food sources, one can overcome the negative effects of parameters and initial solutions. In each iteration, the structured approach

needs to provide sufficient force to move those suboptimal solutions towards the optimal region.

As a result, the ABC study has made avoiding local optima and quickening the rate of convergence attractive objectives. We change the search equation of the fundamental ABC algorithm and provide a new initialization technique to address these. This improved ABC algorithm is known as the SABC algorithm.

4.2 Quasi-random sequence based food sources initialization

To enhance the quality of the current initialization procedure, we are driven to implement a new initialization method in the ABC algorithm. One of the common techniques employed by researchers is random initialization, in which food sources and/or beginning solutions are randomly chosen from a uniform distribution between the lower and upper bounds of the decision variables. On the other hand, there is little likelihood that a randomly generated population will encompass interesting areas of the search space for tiny populations. Consequently, it reduces the likelihood of discovering global optima. Conversely, if the population size is extended to encompass the whole search space region, the computing cost goes up. Another possibility is that the population becomes concentrated in a certain location as a result of repeated generation and overlap of identical solutions. In order to obtain a decent distribution of initial solutions regardless of population size, some writers focused on substituting the chaotic initialization approach based on a chaotic or logistic map for the random initialization. With less calculation time, this mapping strategy explores a superior solution and performs considerably better than the random one. However, their disadvantage is that the mapping that is employed is dependent on a chaotic/bifurcation parameter, the values of which must be carefully chosen by the user and vary depending on the situation.

Using a stratified-random approach called quasi-random sequence, we create a parameter-free and more equally distributed beginning population in

our algorithms, which could yield more accurate results than the methods mentioned previously. To determine if using quasi-random sequences with the Halton set in the beginning population would result in a better value for the objective function at the end is our goal. A GA's starting population is typically referred to as random. However, it is a well-known fact that algorithms cannot produce random numbers. Commonly used algorithmically produced numbers simply attempt to mimic random numbers. More precisely, they are known as pseudorandom numbers. We refer to numbers that are genuinely independent as "genuine random numbers" in order to distinguish them from pseudorandom numbers. Quasi-random sequences are another type. A quasi-random sequence's points are arranged to keep as far away from one another as possible. Stated differently, the points produced by quasi-random sequences attempt to mimic points with a "perfect uniform distribution," whereas the points produced by pseudorandom numbers attempt to mimic actual random points. The former is unachievable, whereas the latter is highly challenging if not impossible. Large partitions that are not needed, however, will raise the cost of computing.

The Halton set initialization method based on quasi-random sequences is computed and discussed in *Algorithm-2* below.

Algorithm-2:
begin

define the n-dimensional decision variable $x_i \in [x_{\min}, x_{\max}]$ $(i = 1, 2, \ldots, n)$ & $x_{\min} < x_{\max}$, $x_{\min}$ *and* $x_{\max}$ *may or may not be the same for each* x_i

Set $i = 1$

Set SN = The number of food sources that will be accessible to bees

repeat

(i) define halton set object Q that contains n-dimensional decision variable x_i *points*

(ii) each $P(i,:)$ *is a point in a Halton sequence, the* j*th coordinate of the point,*

$P(i,j)$ *is equal to* $\sum_{k=1}^{\infty} a_{ij}(k) b_j^{-k-1}$*, where* b_j *is the* j*th prime number*

(iii) $a_{ij}(k)$ *coefficients are nonnegative integers less than* b_j *such that*

$i-1=\sum_{k=0}^{\infty} a_{ij}(k)b_j^k$

i.e. the $a_{ij}(k)$ *values are the base* b_j *digits of the integer* $i-1$.

(iv) generate random variable Hal_i^r $r=1,2,\ldots,SN$ *using the formula* $Hal_i^r = x_{\min} + Q.(x_{\max} - x_{\min})$, *where* Q *is Halton point set in* $(0,1)$

(v) set $i=i+1$

until $(i=n)$

represent $Hal_i^1, Hal_i^2, Hal_i^3, \ldots, Hal_i^{SN}$ *as the first, second, third,..., SNth randomly selected food source* $(i=1,2,\ldots,n)$

end

Using quasi-random initialization, we display the widely dispersed locations of sources of food in two dimensions in Figure 1 and the deviation from the uniform random distribution in Figure 2.
From Figures 1 and 2, we can see that the quasi-random method covers the space better than all the other methods. Additionally, the quasi-random set initialization method increases the distance between the generated points, and this is also a good indicator for covering a large area in the space. On the other hand, with the uniform random method, which is the most commonly used sampling method, the generated points do not cover the whole space, and there are many gaps.

4.3 Composite-strategy for employed bees stage

Upon initializing the food sources or solutions $x_{i,j}$
$(i=1,\ldots,SN, j=1,2,\ldots,n)$ of size popsize, the ABC algorithm directs the employed bees to the search zone that corresponds to the food source's position. Every bee travels to a single food source $x_{i,j}$ and learns where it is by heart. Then, in search of new food sources, hired bees start to search the area around the food sources they have committed to memory. Of course, not every bee behaves the same way when it comes to foraging, and

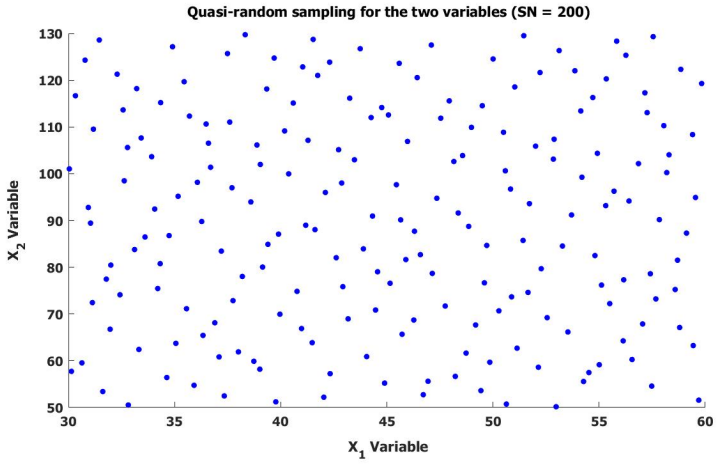

Figure 1: Quasi random sampling

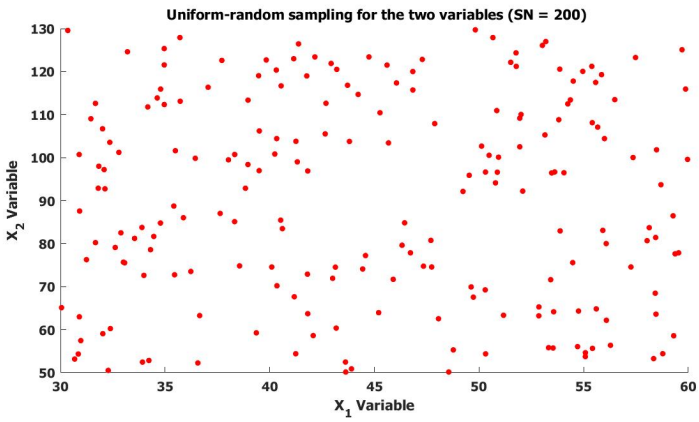

Figure 2: Uniform random sampling

individual bee behavior may vary throughout. Occasionally, their distinctly distinct behaviors serve as the foundation for developing new strategy equations for the neighborhood search process in (1). This allows them to explore their neighborhood and generate a new solution $v_{i,j}$ while

preserving their cooperative contribution.

$$v_{i,j} = \frac{x_{i,j} + x_{i,k}}{2} + \sin(\exp(i - \frac{it}{MaxIt})) * (x_{i,j} - x_{i,k}), \qquad \text{for } 1 \leq i \leq SN. \tag{1}$$

where $j, k \in \{1, 2, \ldots, n\}$ are selected at random, and k and j are distinct from one another.

The bees in (1) swap out their previously learned food places, x_{ij}, for a randomly selected food position, $x_{i,k}(\neq x_{i,j})$. They then explore locally, rotating 360 degrees to create a new position, $v_{i,j}$, by moving left to right or above to downward. When deciding whether to keep the old location $x_{i,j}$ or consider the new one $v_{i,j}$, a greedy selection method is used to assess the fitness value at each step.

4.4 Scout bee's phase

We have put out a new formula for scout bees that will enhance the performance of the SABC algorithm. Here, using our suggested formula, the bees haphazardly create a new food source or solution to replace the abandoned one.

$$x_{i,j} = x_j^{\min} + P.(x_j^{\max} - x_j^{\min}), \tag{2}$$

where $P = \phi_{i,j} * (\frac{1}{1+e^{(\frac{j}{maxIt})}})$, $\phi_{i,j} \in rand(0, 1)$, $i = 1, 2, \ldots, SN$, $j = 1, 2, \ldots, n$ and $MaxIt$ is the maximum number of iteration.

It is important to note that this formula aids in exploring the whole solution space, gradually shifting to the upper bound as the number of iterations increases from the lower bound.

5 Problem definition

We take a look at the COP, which is defined as follows:

Optimize $f(x) = f(x_1, x_2, \ldots, x_n)$,

$$\text{subject to } S = \{G_l(x) \leq 0, H_k(x) = 0; l = 1, 2, \ldots, p; k = 1, 2, \ldots, q\}, \quad (3)$$
$$\text{for all } x \in R^n.$$

The feasible region specified by a set of $p+q$ constraints is denoted by $S \subseteq D$, and the objective function $f(x)$ is defined on the search space $D \subseteq R^n$. The domain of the decision variables of the problem are defined by the n-dimensional interval vector in Euclidean space $R^n = \underline{x_j} \leq x_j \leq \overline{x_j}; j = 1, 2, \ldots, n$, where x_j is the jth variable whose upper and lower bounds are $\underline{x_j}$ and $\overline{x_j}$, respectively. The function $f(x)$ is not necessarily differentiable, but it might be linear, nonlinear, convex, nonconvex, and differentiable. The p equality and q inequality constraints are $G_l(x)$, $H_k(x)$, and they might be linear, nonlinear, convex, or nonconvex. In practice, inequality constraints $G_k(x) = |H_k(x)| - \varepsilon \leq 0, (k = 1, 2, \ldots, q)$ are used in place of equality constraints $H_k(x) = 0, (k = 1, 2, \ldots, q)$. Here, ε is a very tiny positive value. As a result, $(m+p)$ inequality restrictions replace all of the previously mentioned constraints.

In order to solve the inequality COPs using our proposed SDDS-SABC approach, we now employ the penalty function method, which is a widely used constraint handling technique appropriate for population-based optimization. The process of converting into an unconstrained one from a constrained optimization problem is the main characteristic of this approach.

When their related solutions defy the constraints, penalize the objective function by a certain amount. This allows for the preservation of workable solutions while rejecting unworkable ones. However, as neither over-nor under-penalization is desirable, determining a suitable penalty amount is a matter of interest. Many penalty methods have been proposed in the literature; each has its advantages and disadvantages. The functions for the death penalty, static penalty, and dynamic penalty approach (Joines and Houck 1994a), adaptive penalty (Yen 2009), exact penalty (Yu et al. 2010), and so on are a few examples of these techniques. The most sophisticated and widely used approach is the technique of the dynamic penalty function, which we will talk about in Section 5.1. This section introduces many variants of dynamic penalty function techniques that Liu et al. have recently developed

in the following years. In Section 5.2, we also present a novel approach using the dynamic penalty function.

5.1 Existing methods for dynamic penalty function

While there are other approaches to penalize the nonfeasible function, such as dynamic, adaptive, static, and so on, a dynamic penalty works better. Here, nonstationary values are applied at various iterations as a penalty to the unfeasible individuals. As near the feasible area inside the search space, the penalty parameter progressively increases with the number of iterations. They frequently rely on other, difficult-to-determine characteristics. We will now talk about the recently suggested dynamic penalty systems, pointing out their shortcomings and suggesting a new one to get around them.

(a) The dynamic penalty function was developed by Liu et al. [35] in 2015. It involves changing the values of the penalty parameters based on the generation number (gen). According to definitions, the penalized function is

$$\begin{aligned} F(x) &= f(x) + H(\beta, x), \\ &= f(x) + \sum_{j=1}^{q} p_j P_j^{\beta}(x) + \sum_{j=q+1}^{m} p_j P_j(x), \end{aligned} \tag{4}$$

where

$$P_j(x) = \begin{cases} 0 & \text{if} \quad g_j(x) \leq 0, \\ |g_j(x)| & \text{otherwise.} \end{cases}$$

and

$$P_j(x) = \begin{cases} 0 & \text{if} \quad -\epsilon \leq g_j(x) \leq \epsilon, \\ |g_j(x)| & \text{else.} \end{cases}$$

Here, p_j changes with generation number in the following way, and β is a constant that is selected as either $\beta = 1$ or $\beta = 2$:

$$p_j(gen) = \begin{cases} 10^{\theta_1}.(1 + e^{(\frac{\theta_2(\frac{G_{\max}}{2} - gen)}{G_{\max}})}, & \text{if} \quad v_j > \epsilon, \\ 0, & \text{otherwise,} \end{cases}$$

where the tolerance for the constraint violation is ϵ, the maximum iteration number is $G_{\max}$, and the jth constraint violation is v_j. They made the assumptions that $gen = 500$, $\theta_1 = 3$, and $\theta_2 = 2, 4$, and 6, respectively, in their study. They demonstrated that the value of p_j rose exponentially with generation and that it could be used for optimization purposes, both for exploration and exploitation.

(b) Subsequently, in 2016, Liu et al. [34] changed their suggested dynamic penalty function (a) and reformulated the penalized function as follows:

$$F(x) = f(x) + H(\beta, x), \tag{5}$$

where
$p(gen) = 10^{\frac{\theta_2-\theta_1}{1+e^{\frac{20(-gen+\frac{G}{4})}{G}}}+\theta_1}$.

Depending on the generation number, the dynamic penalty factor (S-type function) changes. The restricted range for this component is $[10^{\theta_1}, 10^{\theta_2}]$, where the penalty parameter's scope is indirectly defined by θ_1 and θ_2. A smaller p will diverge in the search space early in the algorithm generation, increasing the variety of the population. With rise in generation, a more substantial p will boost the algorithm's convergence to the global optimum. They made the assumptions $\theta_1 = 2$ and $\theta_1 = 6$ in their work.

(c) Liu et al. [31] have further revised the penalty approach specified in (b). The penalized functions were expressed as follows:

$$F(x) = f(x) + P(x), \tag{6}$$

where
$P(x) = \sum_{j=1}^{q} \mu_j H_j(g_j(x)).g_j(x) + \sum_{j=q+1}^{m} \gamma_j H_j(h_j(x))|h_j(x)|$
$H_j(g_j(x))$ and $H_j(h_j(x))$ are denoted as

$$H_j(g_j(x)) = \begin{cases} 1, & \text{if} \quad g_j(x) > 0, \\ 0 & \text{otherwise}, \end{cases}$$

$$H_j(h_j(x)) = \begin{cases} 1, & \text{if} \quad |h_j(x)| > 0, \\ 0 & \text{otherwise}. \end{cases}$$

Liu et al. also modified the dynamic penalty factor (S-type function) as
$\mu(g) = 10^{\frac{\theta_2-\theta_1}{1+e^{\frac{20(-g+\frac{MCN}{4})}{MCN}}}+\theta_1}$,

where g is the iteration number and μ is constrained to lie in $[10^{\theta_1}, 10^{\theta_2}]$ by penalty parameters θ_1 and θ_2. Similar to the penalty technique mentioned above, the search space will diverge at the beginning of algorithm development for a smaller μ, increasing population diversity. A larger μ will improve the algorithm's convergence to the global optimum as generation increases. They have assumed $\theta_1 = 4$ and $\theta_2 = 6$ in this function.

5.2 Proposed dynamic penalty method

(d) It is generally known that the control parameters β, θ_1, and θ_2 have been the primary basis for the construction of all the aforementioned penalty techniques. The penalized objective function is greatly impacted by the values of these parameters, which must be adjusted suitably based on the algorithm's initial testing. It shows trouble biasing the search towards the viable region if these parameter values are not adequate. Unlike the previously stated penalty function approach, we present a novel dynamic penalty method in this work that has a straightforward form and does not require any additional parameters or punishment factors. Therefore, in order to increase the optimization efficiency, the burden of fine-tuning the penalty factors/parameters has been avoided here. In order to penalize the infeasible solutions and favor feasible solutions, we include in our penalty formula the maximum iteration ($MaxIt$), the current generation number (it), and the number of constrained violations ($nconv(it, i)$). This allows us to quickly and easily guide the population to the feasible region. The information of the objective function and restrictions violation that is clubbed by the penalized function has been specified as follows:

$$F(x) = f(x) + \hat{P} \sum_{l=1}^{p+q} G_l(x), \tag{7}$$

where

$$\hat{P} = \begin{cases} 0, & \text{if} \quad G_l(x) \leq 0, \\ 10^{\{e^{2*(ncov(it,i)*\frac{(\frac{MaxIt}{4}-it)}{(MaxIt)^4})}\}}, & \text{if} \quad G_l(x) > 0. \end{cases}$$

Our suggested dynamic penalty differs significantly in another way: It gradually reduces as solutions go towards the feasible solution area, and it increases for nonfeasible solutions the further they are from the viable zone. At each algorithm iteration, we illustrate in Figure 3 the fluctuation in the arbitrary nonfeasible solution's penalty coefficient ($\hat{P}$) when the population approaches the feasible solution space.

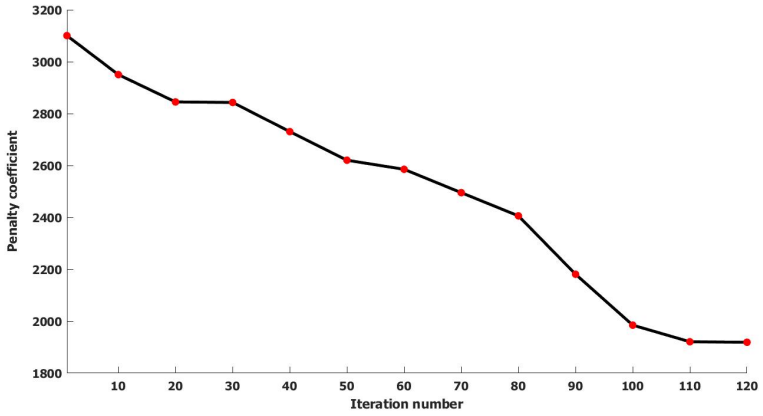

Figure 3: Variation in $\hat{P}$ of arbitrary nonfeasible solution in terms of iteration number

6 Proposed hybrid SDDS-SABC algorithm

In order to create an improved algorithm that could effectively solve the COP with higher reliability, two techniques SDDS and SABC have been combined to create a novel hybrid optimization algorithm (SDDS-SABC). Through recursive decomposition, the SDDS approach reduces the size of the entire search region and concentrates computing effort on the subregion that has viable answers for additional decomposition. Comparatively, SABC is essential in identifying the most promising subregion by removing the best

solution to date from the subregion it is being implemented over. These procedures are carried out one after the other until the region shrinks to a nominal width. Below is a detailed discussion of these approaches' most notable feature.

6.1 Method of split-detect-discard-shrink (SDDS)

The basic motivation behind the SDDS is to initially split at the midpoint of the axis of the first variable of n-dimensional Euclidean space to partition the entire Euclidean space into two subregions of a particular shape. In the proposed SDDS approach, a series of stages involving SDDS is systematically undertaken to sequentially partition the Euclidean space where COP is defined into smaller and smaller subregions then solved recursively until no further division is conceivable. The following is how these steps are handled: (1) Using recursive splitting, we divided the search space D into two discrete subspaces, D_1 and D_2, along the x_j, $(j = 1, 2, \ldots, n)$ axis, one variable at a time. (2) Determine which subspaces correspond to a better solution that represents a promising area in the search space by evaluating the function value at every feasible point in the two subspaces, D_1 and D_2. (3) Discard any subspace without a promising solution, based on the matching solutions of D_1 and D_2. (4) Reduce the initial search space D to the appropriate subspace of D_1 or D_2, depending on which one contains the most promising solutions. Now either D_1 or D_2 becomes D. Until the search space D is limited to an area of nominal width containing the global optimal solution, all of these steps are repeatedly performed.

We perform the recursive splitting of D into D_1 and D_2 in the following manner. The first variable's range, $x_1 \in [\underline{x}_1, \overline{x}_1]$, should first be divided into two equal and disjoint sub-intervals: $[\underline{x}_1, m_1]$ and $[m_1, \overline{x}_1]$. Selected along the x_1-axis, the point $m_1 = \frac{(\underline{x}_1 + \overline{x}_1)}{2}$ represents the first variable of n-dimensional Euclidean space. The promising region among D_1 and D_2 is replaced by D. Then, separating D into D_1 and D_2 is done by dividing the range of the second variable $x_2 \in [\underline{x}_2, \overline{x}_2]$ into two equal and disjoint subintervals $[\underline{x}_2, m_2]$ and $[m_2, \overline{x}_2]$ with a point $m_2 = (\underline{x}_2 + \overline{x}_2)/2$. This point is located along the

x_2-axis at the center of the n-dimensional Euclidean space's second variable. Thus, dividing the whole Euclidean space into the two subregions, D_1 and D_2, continues in this manner until the nth variable is reached. Actually, to continue the splitting process, each axis of variable x_j is taken in turn, starting with $j = 1$, going up to $j = n$. The expression for splitting D into two subregions of a certain form is as follows:

$$D_1 = \{x \in R^n : \underline{x}_i \leq x_i \leq m_i = (\tfrac{\underline{x}_i + \overline{x}_i}{2}), \underline{x}_j \leq x_j \leq \overline{x}_j, j = 1, 2, \ldots, i-1, i+1, \ldots, n\},$$

$$D_2 = \{x \in R^n : m_i = (\tfrac{\underline{x}_i + \overline{x}_i}{2}) \leq x_i \leq \overline{x_i}, \underline{x}_j \leq x_j \leq \overline{x}_j, j = 1, 2, \ldots, i-1, i+1, \ldots, n\}.$$

If D is not reduced to a region of nominal width after all the n-axis have been progressively separated, then we repeat the full sequential partition process.

The stages involved in SDDS have been demonstrated in Figure 4 below [56].

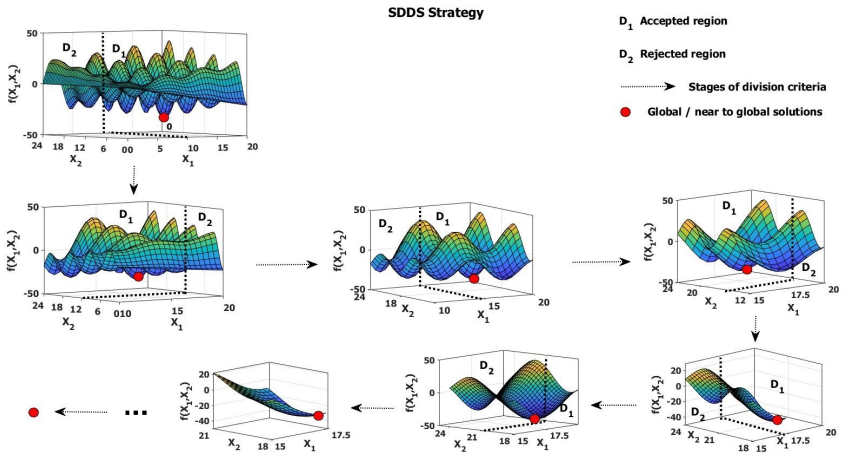

Figure 4: Displaying the SDDS strategy's steps

6.2 Using SABC to identify the promising subregion

View of both drawbacks and merits, we have been motivated to extend, modify, or hybridize ABC with variants of population-based algorithms or classical methods to boost its performance. The research did not just restrict the development of hybrid ABC algorithms to numerical COPs but also optimized a wide range of application-based problems. SABC phase has been applied to each subregion to detect and discard the subregion covering nonpromising solution. The interval arithmetic rule has been used to indicate the subregions which can be discarded by comparing the two solutions, SABC produced in the subregions, and choosing the subregion holding a promising solution. This completes the first cycle of the hybrid algorithm. The method proceeds to explore the search for the promising region by repeatedly alternating the SDDS and SABC phases. Understanding if one subregion, represented by D_1, covers a more promising solution(s) than the other subregion, D_2, and vice versa, and then choosing the most promising subregion based on that understanding is a critical challenge in this black box optimization process. In order to accomplish this, we have included the SABC algorithm into the SDDS technique, which may be applied to optimization issues that are specified in both D_1 and D_2. The subregion covering the nonpromising solution(s) was then discarded using a ranking and selection rule, and the subregion with the promising solution could be named the current shrink region D for further splitting. This process was done from the perspective of optimistic decision makers, who compared the best solutions obtained from both regions. The ranking and selection rule utilized for the minimization problem has been detailed below:

Let $F_1, Con_1 \in D_1$ *and* $F_2, Con_2 \in D_2$ be such that

$F_1 = f(x_1^*)$, $Con_1 = \sum_1^{p+q} G_l(x_1^*)$; x_1^*=best solution obtained by SABC in D_1 and

$F_2 = f(x_2^*)$, $Con_2 = \sum_1^{p+q} G_l(x_2^*)$; x_2^*=best solution obtained by SABC in D_2.

1. If $Con_1 = Con_2 = 0$ & $F_1 < F_2$, then choose D_1 and discard D_2

2. If $Con_1 = Con_2 = 0$ & $F_2 < F_1$, then choose D_2 and discard D_1
3. If $Con_1 < Con_2$ & whether $F_1 < F_2$ or $F_1 > F_2$, then choose D_1 and discard D_2
4. If $Con_2 < Con_1$ & whether $F_1 < F_2$ or $F_1 > F_2$, then choose D_2 and discard D_1
5. If $Con_1 = 0$, $Con_2 = \gamma(\neq 0)$ & whether $F_1 < F_2$ or $F_1 > F_2$, then choose D_1 and discard D_2
6. If $Con_1 = \gamma\ (\neq 0)$, $Con_2 = 0$, & whether $F_1 < F_2$ or $F_1 > F_2$, then choose D_2 and discard D_1

6.3 Computational Complexity

Using the fundamental ABC method, we can evaluate the computational complexity of our proposed method. It should be noted that as it varies depending on the problem, we disregard the time required to compute the objective function here. We assume that T_{MaxIt} is the maximum number of iterations. For ABC, its computational complexity is $O(T_{MaxIt} \times \text{SN})$. As for the proposed SDDS-SABC, the solutions need to be sorted at each iteration, so the computational complexity of SDDS-SABC is $O(T_{MaxIt} \times \text{SN} \times \log(\text{SN}))$. Although, the computational complexity of SDDS-SABC is higher than the basic ABC at the same maximum iteration, SDDS-SABC can achieve much better results than ABC. In addition, the time complexity of SDDS-SABC is similar to that of other ABC variants based on elite populations, but SDDS-SABC performs better than them.

7 Numerical results and discussion

Here, we demonstrate the validity of our proposed SDDS-SABC method through tests on well-known typical benchmark functions CEC 2006 [29] and some engineering design problems (EDPs) (see [19]). These test functions include diverse features like linear/nonlinear, low dimension/high di-

mension, continuous/discrete, separable/nonseparable, convex/nonconvex, unimodal/multi-modal varying feasible region (see Table 1). In this table,

Table 1: Test functions

Benchmark functions

Problem	n	Type of function	ρ	LI	NI	LE	NE	a
g01	13	quadratic	0.0111%	9	0	0	0	6
g02	20	nonlinear	99.99971%	0	2	0	0	1
g03	10	polynomial	0.0000%	0	0	0	1	1
g04	5	quadratic	52.1230%	0	6	0	0	2
g05	4	cubic	0.0000 %	2	0	0	3	3
g06	2	cubic	0.0066%	0	2	0	0	2
g07	10	quadratic	0.0003%	3	5	0	0	6
g08	2	nonlinear	0.8560 %	0	2	0	0	0
g09	7	polynomial	0.5121%	0	4	0	0	2
g10	8	linear	0.0010%	3	3	0	0	6
g11	2	quadratic	0.0000%	0	0	0	1	1
g12	3	quadratic	4.7713%	0	1	0	0	0
g13	5	nonlinear	0.0000%	0	0	0	3	3
g14	10	nonlinear	0.0000%	0	0	3	0	3
g15	3	quadratic	0.0000%	0	0	1	1	2
g16	5	nonlinear	0.0204%	4	34	0	0	4
g17	6	nonlinear	0.0000%	0	0	0	4	4
g18	9	quadratic	0.0000%	0	13	0	0	6
g19	15	nonlinear	33.4761%	0	5	0	0	0
g20	24	linear	0.0000%	0	6	2	12	16
g21	7	linear	0.0000%	0	1	0	5	6
g22	22	linear	0.0000%	0	1	8	11	19
g23	9	linear	0.0000%	0	2	3	1	6
g24	2	linear	79.6556 %	0	2	0	0	2

"n" is the dimension of the problem, $\rho = |F|/|S|$ represents the proportion between the feasible region and the search space. Also, LI, NI, LE, NE and "a" represent the numbers of linear inequality constraints, nonlinear inequality constraints, linear equality constraints, nonlinear equality constraints, and active constraints, respectively. We study the robustness of our proposed dynamic penalty-based constraint handling techniques integrated into the SDDS-SABC method on selected problems of CEC 2006 (see Table 2). Furthermore, we compare these results with the present dynamic penalty methods. The overall best-found results have been displayed in Table 2.

Table 2: Comparative analysis between the proposed dynamic penalty and existing penalty methods

Problems	(a)	(b)	(c)	(d)
g01	-15.9472628	-15.8610643	-15.6103179	-15.4618784
g05	5126.8837241	5125.046215	5124.1968423	5124.0049727
g15	952.8593772	951.9805434	951.6104782	951.5300084
g24	-6.239074165	-6.403721373	-6.53271104	-6.632598318
	(a) [35];	(b): [34];	(c): [33] ;	(d): Proposed

The results show that our proposed penalty method explores feasible solution space efficiently to provide promising results. We have coded the said optimization method in MATLAB and executed it in an HP Pavilion Laptop with Intel (R) 11th Gen Core i5-512GB SSD @ 2.40 GHz. The basic parameters used in the SDDS-SABC method include; colony size $SN = 100$ (equal to the number of employed and onlooker bees), the Scout bee's food source abandonment parameter $= round(SN \times n \times p_{val})$, where p_{val} is a small probability value in the range (0.05 - 0.08). The maximum cycle number (MCN) is 50, which serves as the termination criterion. The values of the control parameters of the SABC algorithm used in our simulation studies and the values assumed by the authors in their respective state-of-the-art algorithms, which we have used for comparison purposes, have been displayed in Table 3. Firstly, we study the robustness of SDDS-SABC method implemented over our proposed dynamic penalty-based constraint handling techniques through 24 benchmark functions, and secondly, on the engineering design problems comparing results obtained using different penalty methods. Using the following indices—exactness, consistency, efficacy, and statistical analysis—we assess resilience in several ways. The following definitions apply to these: a) Accuracy: the degree of the best-found solution's quality and its separation from the global solution. In a similar vein, the degree to which the worst-found solution deviates from the global answer and its quality, tested on 25 independent runs of the best and worst identified solutions.

(b) Consistency: comprehend the resilience and stability of the optimization technique on the problem that leads to the best possible solution. Test the following: the average and standard deviation of the solutions from the 25

Table 3: **Control parameter values of different algorithms**

(a) **Based on ABC**

Algorithms	Popsize	MaxIt	Limit
I-ABC	20	6000	SN×n
CB-ABC	90	500	SN×n
IABC-MAL	30	500	0.5×SN×n
MG-ABC	50	1000	SN×n
SDDS-SABC	100	50	round(p_{val}× n× SN)

(b) **Based on nonABC**

Algorithms	Popsize	MaxIt	Limit
UFA	50	40	30
GLF-GWO	3×n	3000	$10^4 \times n$
GGA	50	50	25
JaQA	50	50	25
SDDS-SABC	100	50	round(p_{val}×n×SN)

runs.

c) Statistical analysis: compare the significant difference in the performance of our proposed SDDS-SABC method with other existing algorithms. Test on algorithms through the Friedman test and Wilcoxon signed ranks test, which are standard nonparametric statistical tests.

7.1 Sensitivity analysis of some key parameters in SABC algorithm

In addition to improving algorithmic efficiency, appropriate input values for the algorithm's parameters are crucial for its robustness, stability, and best-found objective value. Sensitivity analysis has therefore been performed to examine the impact of the input values of the important SABC parameters, such as MaxIt and SN, on the performance of our algorithm SDDS-SABC towards achieving the optimal solution and its stability in each algorithm run. We have set MaxIt to 50 and the algorithm's iteration numbers range from 1 to MaxIt. Also, we have taken the population size SN is 100. We display this study graphically on a specific benchmark test function, g07. The evolution

of the best-found solutions at various iterations when MaxIt=50 is shown in Figure 5(i). Additionally, it demonstrates that when SN=100, the algorithm converges to the best-found value of 24.34849. The algorithm converges to the best-found value of 24.34849 at MaxIt=50 for SN=100, as seen in Figure 5(ii). Once more, the stable solution 24.34849 is reached with values higher than 50. For other test functions, we see a comparable effect. We cannot obtain the best or nearly best solution to the problems if we set the value of MaxIt and SN to be less than 50 and 100, respectively.

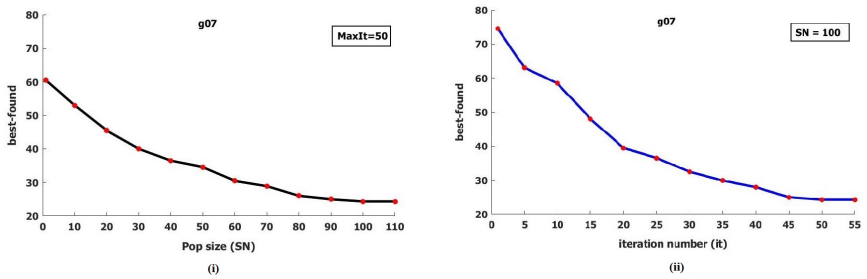

Figure 5: Evolution of best-found value with respect to iteration number and colony size

7.2 Study on algorithm performance based on exactness, consistency and effectiveness

We present in Table 4, the best-found, worst-found, mean and standard deviation (std) of the result obtained from each benchmark function in 25 independent runs of the SABC-SDDS algorithm. The results found are encouraging.

We also compare our results with other state-of-the-art hybrid algorithms viz., I-ABC, CB-ABC, IABC-MAL and MG-ABC (see Table 5) in terms of best-found, worst-found and mean values. These said hybrid algorithms had been developed by combining other heuristics or traditional methods with ABC. Table 5 shows that our SDDS-SABC algorithm outperforms other algorithms in some problems and works equally well for other problems. However,

Table 4: Statistical results of SDDS-SABC on benchmark functions, averaged over 25 independent runs

Functions	optimal value	best-found	worst-found	mean	std
g01	-15.0000000000	-15.4618784341837	-14.7361050072335	-15.071895226323	5.99789658392
g02	-0.8036191042	-0.875454224809	-0.77323254632	-0.741139419121	0.31517588432024
g03	-1.0005001000	-1.098598548944	-0.964899232223	-1.01285024552	0.041820449166
g04	-30665.5386717834	-30665.8337336955	-30698.7685959581	-30653.100331767	0.2687651065913
g05	5126.4967140071	5124.0049727086	5176.27733899147	5134.7200444372	0.4966102760171
g06	-6961.8138755802	-6983.37054536156	-6698.63125675940	-6946.0112672204	0.59174789701531
g07	24.302090681	24.348490589297	27.8909837113848	25.70325545062	1.051962320689
g08	-0.0958250415	-0.095722476579	-0.0813461879519	-0.0907741721133	0.0038036427493
g09	680.6300573745	684.560836511063	695.96431708531	689.024312257864	2.0631346433681
g10	7049.2480205286	7047.3431791133	7124.63089605634	7066.0444725057	19.190512293263
g11	0.749900000	0.7505189211405	0.76423525277597	0.7562800118598	0.00344183933935
g12	-1.00000000	-0.999954717347	-0.9398378640313	-0.990291881794	0.01317550762524
g13	0.0539415140	0.0505131549885	0.0560012746738	0.0549429543562	0.00109084362246
g14	-47.7648884595	-	-	-	-
g15	961.7150222899	951.530008446544	960.601193735686	955.3401698407	2.3820716382008
g16	-1.9051552586	-	-	-	-
g17	8853.5396748064	-	-	-	-
g18	-0.8660254038	-0.942906584997	-0.87048513538499	-0.90112279347847	0.0179637098296
g19	32.6555929502	-22410.82795313840	6013.94217031911	-14859.743320706	4321.0628338304
g20	0.2049794002	-	-	-	-
g21	193.7245100700	193.508264477410	194.783450798573	193.62084757633	0.0148378378312
g22	236.4309755040	-	-	-	-
g23	-400.0551000000	-4156.949391713	-4087.08875546532	-4109.4664146789	11.035324578671
g24	-5.5080132716	-6.6325983187262	-5.34276622737366	-5.921377563442	0.38492496179489

one difficulty noted in our algorithm is that it could not reach a reasonably good solution for problems g14, g16, g17, g20 and g22. Likewise, I-ABC, CB-ABC, IABC-MAL too were unable to locate the optimal solution for the problems g20 and g22. Also, MG-ABC could not solve problems g21, g22, g23 and g24, where our SDDS-SABC algorithm worked well except for g22. For g21, SDDS-SABC produced a better best-found value than the I-ABC and CB-ABC and equally well for IABC-MAL. In terms of mean, SDDS-SABC provides a much better result than the I-ABC. We also noted that no hybrid algorithms with ABC could find the reasonably good optimal/near to optimal solution for the problems g14, g17, g20, and g22 because the ratio between their feasible region and search space are minimal (ρ=0.0000%),. A similar situation arises for problem g16 occupying ρ=0.0204%, which is very small in percentage value (see Table 1). So, finding the global or near-to-global solution for those problems is challenging. In addition, problems g14, g16, and g17 have highly nonlinear type of functions. Although g20 and g22 are linear, due to their high dimension, that is, $n = 24$ and $n = 22$, respectively, their solutions could not be traced due to their small feasible space. Our SDDS-SABC algorithm could find better results for the nonlinear type of function (g02, g08 and g19) when the proportion between the feasible region

and search space is either large, small or midway (99.99971%, 0.8560% and 33.4761%) even for high dimensions problems (n=20, n=2 and n=15). An exciting observation noted through problem g13 is that for the low dimension problem ($n = 5$), even though it is highly nonlinear, we could find a better result from a tiny space of 0.0000% originating between the feasible region and search space. So, performance of SDDS-SABC depends on the nature of the function, the problem's dimension, and the percentage between the feasible region and search space.

We have also compared our SDDS-SABC results with the other popular non-ABC based hybrid algorithms like; UFA, GLF-GWO, GGA, and JaQa [11]. We can see from Table 6 that our proposed SDDS-SABC performs better than UFA, GLF-GWO, GGA, and JaQa algorithms in terms of best-found value for problem g03. SDDS-SABC performs better than GGA in terms of best-found value for problem g04. However, GLF-GWO could not find any feasible solution for g05, whereas our SDDS-SABC could provide a better result. For the problem g06, our proposed SDDS-SABC algorithm gives better solution than UFA, GLF-GWO, GGA, and JaQa algorithms in terms of best-found value. Similarly, for problem g15, SDDS-SABC works better than UFA, GLF-GWO, GGA, and JaQA in terms of best-found, worst-found, and mean values. In g21, SDDS-SABC outperforms UFA for best-found value and gives better value than GGA and JaQa in term of mean value. Also, GLF-GWO could not provide any feasible solution for the problem g21. In g24, we can see that our SDDS-SABC and GGA give approximately equal best-found values. Moreover, UFA and GLF-GWO could not find optimal solutions to problems g20 and g22. Compared with the rest of the algorithms, only GGA could solve problem g20 and find a better result. On the other hand, GGA could not work much well for problems g14 and g22. Finally, except for test problems g14, g16, g17, g20, and g22, our SDDS-SABC algorithm provides significantly better results than other algorithms.

Table 5: The comparison of best-found, worst-found and mean results with ABC based algorithms

Functions		I-ABC (2015)	CB-ABC (2015)	IABC-MAL (2017)	MG-ABC (2018)	SDDS-SABC
g01	best-found	-15.000	-15.000	-15.000000	-15.000000	-15.4618784341837
	worst-found	-15.000	-15.000	-15.000000	-9.000000	-14.7361050072335
	mean	-15.000	-15.000	-15.000000	-13.553540	-15.071895226323
g02	best-found	-0.803619	-0.803619	-0.803619	-0.8036108	-0.875454224809
	worst-found	-0.778278	-0.777844	-0.785568	-0.7604863	-0.77323254632
	mean	-0.800094	-0.794522	-0.799460	-0.7890629	-0.741139419121
g03	best-found	-1.000	-1.0005	-1.000500	-1.000400	-1.098598548944
	worst-found	-0.999	-1.0005	-1.000500	-1.000258	-0.964899232223
	mean	-1.0004	-1.0005	-1.000500	-1000383	-1.01285024552
g04	best-found	-30665.539	-30665.539	-30665.539	-30665.540	-30665.8337336955
	worst-found	-30665.539	-30665.539	-30665.539	-30665.540	-30698.7685959581
	mean	-30665.539	-30665.539	-30665.539	-30665.540	-30653.100331767
g05	best-found	5126.498	5126.197	5126.498	5126.4970	5124.0049727086
	worst-found	5126.944	5126.497	5126.498	6112.169	5176.27733899147
	mean	5131.861	5126.497	5126.498	5467.7560	5134.7200444372
g06	best-found	-6961.814	-6961.814	-6961.814	-6961.8030	-6983.37054536156
	worst-found	-6961.814	-6961.814	-6961.814	-6957.1230	-6698.63125675940
	mean	-6961.814	-6961.814	-6961.814	-6959.4890	-6946.0112672204
g07	best-found	24.311	24.3062	24.3064	24.326530	24.348490589297
	worst-found	24.677	24.3062	24.3062	25.099270	27.8909837113848
	mean	24.366	24.3062	24.3062	24.780640	25.70325545062
g08	best-found	-0.095825	-0.095825	-0.095825	-0.095825	-0.095722476579
	worst-found	-0.095825	-0.095825	-0.095825	-0.095825	-0.0813461879519
	mean	-0.095825	-0.095825	-0.095825	-0.095825	-0.0907741721133
g09	best-found	680.631	680.630	680.630	680.6302	684.560836511063
	worst-found	680.637	680.630	680.630	680.6322	695.96431708531
	mean	680.633	680.630	680.630	680.6309	689.024312257864
g10	best-found	7049.321	7049.248	7049.248	7104.006	7047.3431791133
	worst-found	7049.343	7049.248	7049.248	7504.944	7124.63089605634
	mean	7124.042	7049.248	7049.248	7357.461	7066.0444725057
g11	best-found	0.7499	0.7499	0.749900	0.749995	0.7505189211405
	worst-found	0.7499	0.7499	0.749900	0.750127	0.76423525277597
	mean	0.7499	0.7499	0.749900	0.750025	0.7562800118598
g12	best-found	-1.000	-1.000	-1.000000	-1.000000	-0.999954717347
	worst-found	-1.000	-1.000	-1.000000	-1.000000	-0.9398378640313
	mean	-1.000	-1.000	-1.000000	-1.000000	-0.990291881794
g13	best-found	0.053958	0.053942	0.0539498	0.05394861	0.0505131549885
	worst-found	0.055130	0.43880	0.0539498	0.4377867	0.0560012746738
	mean	0.054144	0.066770	0.0539498	0.171074	0.0549429543562

Table 5: (continued)

g14	best-found	-47.665	-47.765	-47.765	-47.675860	-
	worst-found	-47.830	-47.765	-47.765	-46.465260	-
	mean	-47.201	-47.765	-47.765	-47.246220	-
g15	best-found	961.715	961.715	961.715	961.715100	951.530008446544
	worst-found	961.720	961.715	961.715	965.208600	960.601193735686
	mean	961.716	961.715	961.715	962.173700	955.34016984070
g16	best-found	-1.905	-1.905	-1.905	-1.905155	-
	worst-found	-1.905	-1.905	-1.905	-1.905155	-
	mean	-1.905	-1.905	-1.905	-1.905155	-
g17	best-found	8860.864	8853.533875	8853.533875	8853.53	-
	worst-found	8983.359	8941.940741	8927.597785	9241.820	-
	mean	8909.994	8902.869928	8883.163028	8915.998	-
g18	best-found	-0.866025	-0.672216	-0.866025	-0.8660253	-0.94290658499
	worst-found	-0.856622	-0.866025	-0.866025	-0.8648695	-0.87048513538499
	mean	-0.865310	-0.866025	-0.866025	-0.8657735	-0.90112279347847
g19	best-found	32.784	35.746	32.6556	-5.508013	-22410.82795313840
	worst-found	34.856	32.6557	32.6556	-5.508013	6013.94217031911
	mean	33.344	32.6556	32.6556	-5.508013	-14859.743320706
g20	best-found	-	-	-	1.393571	-
	worst-found	-	-	-	1.399163	-
	mean	-	-	-	1.394359	-
g21	best-found	193.725	257.156	193.725	-	193.725
	worst-found	964.030	193.725	193.725	-	194.783450798573
	mean	622.678	193.725	193.725	-	193.62084757633
g22	best-found	-	-	-	-	-
	worst-found	-	-	-	-	-
	mean	-	-	-	-	-
g23	best-found	-358.183	-400.055	-400.055	-	-4156.94939171
	worst-found	899.881	-400.055	-400.055	-	-4087.08875546532
	mean	169.021	-400.055	-400.055	-	-4109.4664146789
g24	best-found	-5.508	-5.508	-5.508	-	-6.6325983187262
	worst-found	-5.508	-5.508	-5.508	-	-5.34276622737366
	mean	-5.508	-5.508	-5.508	-	-5.921377563442

Table 6: The comparison of best-found, worst-found and mean results with different nonABC algorithms

Functions		UFA (2019)	GLF-GWO (2020)	GGA (2021)	JaQA (2022)	SDDS-SABC
g01	best-found	-15.000000	15.00000	-15.0000000000	-15.00000	-15.4618784341837
	worst-found	-15.000000	-14.9999	-15.0000000000	-15.00000	-14.7361050072335
	mean	-15.000000	-15.000000	-15.0000000000	-15.00000	-15.071895226323
g02	best-found	-0.8033	-0.803619	-0.8030191042	-0.803605	-0.875454224809
	worst-found	-0.5205742	-0.6275	-0.8010191042	-0.800272	-0.77323254632
	mean	-0.7458475	-0.7249	-0.8020191042	-0.80111	-0.741139419121
g03	best-found	-1.0005	-1.0005	-1.0004181146	-1.0005	-1.098598548944
	worst-found	-1.0005	-0.0006	-0.9993067321	-0.9987	-0.964899232223
	mean	-1.0005	-0.7386	-1.0000114315	-1.00031	-1.01285024552
g04	best-found	-30665.5203	-30665.539	-30678.4386717834	-30665.5387	-30665.8337336955
	worst-found	-30665.539	-30665.0825	-30667.0386717834	-30665.5387	-30698.7685959581
	mean	-30665.539	-30665.3389	-30667.6386717834	-30665.5387	-30653.100331767
g05	best-found	5126.49671	-	5126.4967135571	5126.484	5124.0049727086
	worst-found	5126.49671	-	5126.4967135601	5126.611	5176.27733899147
	mean	5126.49671	-	5126.4967135581	5126.504	5134.7200444372
g06	best-found	-6961.83884	-6961.4784	-6961.8130705802	-6961.814	-6983.37054536156
	worst-found	-6961.81388	-6961.4886	-6961.8130665802	-6961.814	-6698.63125675940
	mean	-6961.81388	-6961.7341	-6961.8130685802	-6961.814	-6946.0112672204
g07	best-found	24.306209	24.3851	24.3934806327	24.0012	24.348490589297
	worst-found	24.306209	25.6385	27.7034023081	24.6781	27.8909837113848
	mean	24.306209	24.7221	26.5452127381	24.2121	25.70325545062
g08	best-found	-0.09582504	-0.0958	-0.0958233590	-0.095825	-0.095722476579
	worst-found	-0.09582504	-0.0958	-0.0951989658	-0.095825	-0.0813461879519
	mean	-0.09582504	-0.0958	-0.0955852752	-0.095825	-0.0907741721133
g09	best-found	680.630057	680.6538	680.6301199745	680.631	684.560836511063
	worst-found	680.630057	680.3862	680.6313973745	680.631	695.96431708531
	mean	680.630057	681.0990	680.6306403745	680.631	689.024312257864
g10	best-found	7049.24802	7729.9603	7049.2479999286	7006.52	7047.3431791133
	worst-found	7049.24802	8554.3989	7049.2480002286	7121.83	7124.63089605634
	mean	7049.24802	8276.2365	7049.2480000286	7086.136	7066.0444725057
g11	best-found	0.7499	0.7499	0.7493788256	0.7499	0.7505189211405
	worst-found	0.7499	0.9998	0.7498827597	0.7499	0.76423525277597
	mean	0.7499	0.7669	0.7496174447	0.7499	0.7562800118598
g12	best-found	-1.000000	-1.000	-1.000000000	-1.00	-0.999954717347
	worst-found	-1.000000	-1.000	-1.000000000	-1.00	-0.9398378640313
	mean	-1.000000	-1.000	-1.000000000	-1.00	-0.990291881794
g13	best-found	0.0539415	0.9527	0.0539181140	0.00174	0.0505131549885
	worst-found	0.0539415	2.3466	0.0539417680	0.00174	0.0560012746738
	mean	0.0539415	1.1903	0.0539299140	0.00174	0.0549429543562

Table 6: (continued)

g14	best-found	-47.764879	-46.7926	1172.2351115405	-48.0111	-
	worst-found	-47.764879	-38.1941	1312.2351115405	-46.1022	-
	mean	-47.764879	-41.8264	1252.2351115405	-46.8843	-
g15	best-found	961.7150223	961.7157	952.4957146499	961.6758	951.530008446544
	worst-found	961.7150223	971.8903	960.1071304699	961.6758	960.601193735686
	mean	961.7150223	965.7727	955.0510816799	961.6758	955.34016984070
g16	best-found	-1.90515526	-1.9045	-1.9051549586	-1.9052	-
	worst-found	-1.90515526	-1.6776	-1.9051543336	-1.9052	-
	mean	-1.90515526	-1.8346	-1.9051546316	-1.9052	-
g17	best-found	8853.533875	-	8892.5396953064	8853.5396	-
	worst-found	8853.533875	-	8962.5396748064	8902.223	-
	mean	8853.533875	-	8918.3396748064	8872.5142	-
g18	best-found	-0.8660254	-0.8660	-0.8619563027	-0.86603	-0.94290658499
	worst-found	-0.8660254	-0.6569	-0.8461369131	-0.86601	-0.87048513538499
	mean	-0.8660254	-0.8233	-0.8541324790	-0.86602	-0.90112279347847
g19	best-found	32.655593	32.2874	-66409.2048070498	- 32.6699	-22410.82795313840
	worst-found	32.655593	82.7696	-364.8315650498	- 32.7872	6013.94217031911
	mean	32.655593	43.0767	30.3617957802	- 32.6551	-14859.743320706
g20	best-found	-	-	0.3929794002	0.24072	-
	worst-found	-	-	0.5209794002	0.24794	-
	mean	-	-	0.4389794002	0.24381	-
g21	best-found	193.724520	-	193.7245100700	193.4011	193.7245100700
	worst-found	520.165650	-	193.7245100700	203.9120	194.783450798573
	mean	255.559033	-	193.7245100700	193.7302	193.62084757633
g22	best-found	-	-	-	5.08E+02	-
	worst-found	-	-	-	3.03E+07	-
	mean	-	-	-	2.14E+03	-
g23	best-found	-400.0551	-0.0651	-397.7451000000	-412.520	-4156.94939171
	worst-found	-400.0551	809.3461	-392.4451000000	-388.2426	-4087.08875546532
	mean	-400.0551	269.7458	-395.8651000000	-399.3486	-4109.4664146789
g24	best-found	-5.50801327	-5.5080	-6.7054079016	-5.5094	-6.6325983187262
	worst-found	-5.50801327	-3.0000	-6.0978333186	-5.5094	-5.34276622737366
	mean	-5.50801327	-5.2834	-6.3582253626	-5.5094	-5.921377563442

7.3 Study on algorithm performance using statistical analysis

On benchmark functions, nonparametric statistical tests on the best-found values have been carried out in order to rank the performance of the suggested and current algorithms. To determine whether there is a difference between the estimated outcomes produced by different algorithms, the nonparametric Friedmans test is employed. Furthermore, at a significance level of 5%, Wilcoxon's signed rank test has been applied independently to two groups of algorithms as a nonparametric test. It is expected that all algorithms function similarly under the null hypothesis. Reject the null hypothesis if the provided p-value is less than 0.05. The null hypothesis is rejected, indicating that all of the algorithms under investigation perform significantly

differently. From Table 7 of the Friedmans mean rank test, we see that the mean rank of SDDS-SABC attains the lowest value 2.26 and hence ranked 1. It means that our SDDS-SABC is better than the other hybrid ABC-based algorithms CBABC with rank 2, IABC-MAL with rank 3, SACABC with rank 4 and the last rank is 5 of IABC.

In Wilcoxon signed rank test results (see Table 8) comparing pairwise SDDS-SABC with IABC, CB-ABC, IABC-MAL and MG-ABC respectively, the results show that in every pairwise comparison, the sum of SDDS-SABC's positive ranks is significantly greater than the sum of its negative ranks. Furthermore, their p-value is less than 0.05, as can be shown. This suggests that SDDS-SABC performs better than other available methods.

Table 7: On ABC based hybrid algorithms, the mean ranking attained by Friedman's mean rank test at a significance level of 5%.

Algorithms	Mean rank	Rank
IABC(Liang et al., 2015)	3.66	5
CBABC (Brajevic, 2015)	2.75	2
IABC-MAL (Long et al., 2017)	3.00	3
MGABC (Bansal, 2018)	3.34	4
SDDS-SABC	**2.25**	**1**

Table 8: Results of the Wilcoxon signed rank test on hybrid algorithms that are ABC based, with a significance threshold of 5%.

Comparison	Observations	No. of test functions	Sum of positive rank	Sum of negative rank	p-value
SDDS-SABC with I-ABC	SDDS-SABC<I-ABC	14	165.00	25.00	0.005
	SDDS-SABC>I-ABC	5			
SDDS-SABC with CB-ABC	SDDS-SABC<CB-ABC	14	164.00	26.00	0.005
	SDDS-SABC>CB-ABC	5			
SDDS-SABC with IABC-MAL	SDDS-SABC<IAB-CMAL	14	164.00	26.00	0.005
	SDDS-SABC>IABC-MAL	5			
SDDS-SABC with MG-ABC	SDDS-SABC<MG-ABC	11	113.00	23.00	0.020
	SDDS-SABC>MG-ABC	5			

Again, comparing the result of SDDS-SABC with the different types of nonABC hybrid algorithms (see Table 9) using Friedman's mean rank test, we can see that the mean rank of SDDS-SABC is lowest with a value of 1.88 and therefore ranked as 1. The other lowest values are 2.37, 2.41, 2.53, and 3.18 for JaQA, GGA, UFA, and GLF-GFO, respectively, giving ranks 2, 3, 4, and 5 per their performance. The first rank of SDDS-SABC indicates that this works better than the other algorithms. For the same hybrid algo-

rithms, the Wilcoxon signed rank test results have been displayed in Table 10. The pairwise comparison of SDDS-SABC with UFA, GLF-GWO, GGA, and JaQa, respectively, demonstrates that for every pairwise comparison, the total of the positive rank of SDDS-SABC is significantly larger than the negative rank. Furthermore, the p-values of SDDS-SABC are less than 0.05, indicating that it performs better than the other algorithms in the comparison. IBM SPSS statistics has been used to conduct these Friedman and Wilcoxon tests.

Table 9: On nonABC based hybrid algorithms, the mean ranking attained by Friedman's mean rank test at a significance level of 5%.

Algorithms	Mean rank	Rank
UFA (Brajevic et al., 2019)	2.53	4
GLF-GFO (Gupta et al., 2020)	3.18	5
GGA (D'Angelo et al., 2021)	2.41	3
JaQA (Das et al., 2022)	2.37	2
SDDS-SABC	**1.88**	**1**

Table 10: Results of the Wilcoxon signed rank test on hybrid algorithms that are non-ABC based, with a significance threshold of 5%.

Comparison	Observations	No. of test functions	Sum of positive rank	Sum of negative rank	p-value
SDDS-SABC with UFA	SDDS-SABC<UFA	14	165.00	25.00	0.005
	SDDS-SABC>UFA	5			
SDDS-SABC with GLF-GWO	SDDS-SABC<GLF-GWO	13	136.00	17.00	0.005
	SDDS-SABC>GLF-GWO	4			
SDDS-SABC with GGA	SDDS-SABC<GGA	13	144.00	46.00	0.049
	SDDSSABC>GGA	6			
SDDS-SABC with JaQA	SDDS-SABC<JaQA	13	146.00	47.00	0.051
	SDDSSABC>JaQA	6			

8 Applications of algorithm on real-life engineering design problems:

In this section, we present the challenging five real-life engineering design problems; see Table 11. These problems have been solved by our proposed SDDS-SABC algorithm and further compared with state-of-art hybrid algorithms (see Table 12).

Table 11: Engineering design problems

Sr. no.	Problem	n	LI	NI	LE	NE
EDP1	Three-bar truss design	2	0	3	0	0
EDP2	Compression spring design	3	0	4	0	0
EDP3	Cantilever beam design	5	0	1	0	0
EDP4	Pressure vessel design	4	2	2	0	0
EDP5	Heat exchanger design	8	3	3	0	0

8.1 Three-bar truss design problem

A three-bar planar truss structure (see [19]) has been taken into account in this case study. Initially, Nowacki developed this problem to reduce the volume of a statically loaded three-bar truss. Each truss element's stress is subject to limitations. The problem has been defined mathematically as follows:

$$\begin{aligned}
\text{Min} \quad & f_1(x_1, x_2) = L * (2\sqrt{(}2x_1) + x_2), \\
\text{s.t.} \quad & g_1(x_1, x_2) = \frac{\sqrt{(}2x_1) + x_2}{\sqrt{(}2x_1^2) + 2x_1x_2} R \leq \sigma, \\
& g_2(x_1, x_2) = \frac{x_2}{\sqrt{(}2x_1^2) + 2x_1x_2} R \leq \sigma, \\
& g_3(x_1, x_2) = \frac{1}{x_1 + \sqrt{(}2)x_2} R \leq \sigma, \\
& 0 \leq x_1, x_2 \leq 1,
\end{aligned}$$

where $L = 100\ c.m.,\ R = 2\ KN/cm^2\ and\ \sigma = 2KN/cm^2.$

Numerous studies have been published in the literature in an effort to solve this real-life problem. The results of this problem using SDDS-SABC have been compared with other algorithms including SC-GWO, PSO, wPSO, CS, Ray & Saini, Tsai, mGWO, wGWO, m-SCA, OBSCA, SSA, MFO, WOA, ISCA, Chaotic SSA and shown in Table 12.

8.2 Compression spring design problem

To get a minimum weight for a compression spring (see [19]), one needs to find the best values for the variables representing wire diameter (d), mean coil diameter (D), and the number of active coils (N). The following is the mathematical formulation of the problem where the limitations imposed on the objective function are stress, spike frequency, and defection.

$$
\begin{aligned}
Min \quad & f_2(x) = (x_3 + 2)x_2x_1^2, \\
s.t. \quad & g_1(x) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leq 0, \\
g_2(x) = & \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \leq 0, \\
& g_3(x) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0, \\
& g_4(x) = \frac{x_1 + x_2}{1.5} - 1 \leq 0, \\
0.05 \leq x_1 \leq 2, & 0.25 \leq x_2 \leq 1.30, 2 \leq x_3 \leq 15.
\end{aligned}
$$

This problem has been solved by several authors using their proposed algorithms such as; SC-GWO, GWO, PSO, PSO (He & Wang), GSA, SCA, GA, mGWO, wGWO, mSCA, OBSCA, MFO, WOA, SSA, ISCA, and Chaotic SSA. We compare their results with ones obtained by our SDDS-SABC algorithm in Table 12. The results are self-explanatory.

8.3 Cantilever beam design problem

This problem aims to reduce the cantilever beam's overall weight by optimizing the five hollow square cross-section specifications. The thickness of all the cross-sections is the same but has a different length. The problem includes five estimated parameters. One side of the liver is connected to a rigid body, and a load is attached to the other end. The formulation of this has been mathematically expressed as follows:

$$
\begin{aligned}
Min \quad & f_3(x) = 0.6224(x_1 + x_2 + x_3 + x_4 + x_5), \\
s.t. \quad & g(x) = \frac{60}{x_1^3} + \frac{27}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0, \\
& 0.01 \leq x_1, x_2, x_3, x_4, x_5 \leq 100.
\end{aligned}
$$

The solution to this problem using our proposed SDDS-SABC algorithm has been displayed in Table 12. The results obtained by other algorithms such as BASZNN, BAS, BAS-WPT, BSAS, ZNN, ALO, GCA I, GCA II, CS, SOS, and EPO ,have been shown in the same table for better comparison.

8.4 Pressure vessel design problem

The objective of this problem is to minimize the overall cost of the cylindrical pressure vessel in terms of material, forming, and welding under the nonlinear constraints of stresses and yield criteria. The decision parameters involve the thickness of the shell (T_{SH}), the thickness of the head (T_{HD}), inner radius (R), and the length of the cylindrical shell (L). Mathematically the problem has been expressed as follows: (see [19])

$$
\begin{aligned}
Min \quad & f_4(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 19.84x_1^2x_3 + 3.1661x_1^2x_4, \\
& x = (x_1, x_2, x_3, x_4) = (T_{SH}, T_{HD}, R, L), \\
s.t. \quad & g_1(x) = 0.0193x_3 - x_1 \leq 0, \\
& g_2(x) = 0.00954x_3 - x_1 \leq 0, \\
& g_3(x) = 1296000 - \frac{4}{3}\pi x_3^3 - \pi x_3^2 x_4 \leq 0, \\
& g_4(x) = x_4 - 240 \leq 0, \\
& 1 \times 0.0625 \leq x_1, x_2 \leq 99 \times 0.0625, \\
& 10 \leq x_3, x_4 \leq 200.
\end{aligned}
$$

The solution to this problem by our proposed SDDS-SABC algorithm has been shown in Table 12. We compare our results with state-of-the-art algorithms such as SC-GWO, GWO, PSO, SCA, GASA, GA, DE, Branch and Bound, Lagrangian Multiplier, ACO, ES, mGWO, wGWO, mSCA, OBSCA,

MFO, WOA, SSA, ISCA and Chaotic SSA (see Table 12). It has been observed that the proposed SDDS-SABC algorithm outperforms others.

8.5 Heat exchanger design problem

This is a challenging benchmark minimization problem because all the constraints are strictly enforced (Xin-She Yang and Amir H. Gandomi, 2012). It has eight design variables and six inequality restrictions, three linear and three nonlinear. The problem has been stated as follows:

$$\begin{aligned} Min \quad & f_5(x) = x_1 + x_2 + x_3, \\ s.t. \quad & g_1(x) = 0.0025(x_1 + x_6) - 1 \leq 0, \\ & g_2(x) = 0.0025(x_5 + x_7 - x_4) - 1 \leq 0, \\ & g_3(x) = 0.01(x_8 - x_5) - 1 \leq 0, \\ & g_4(x) = 833.33252x_4 + 100x_1 - x_1x_6 - 8333.333 \leq 0, \\ & g_5(x) = 1250x_5 + x_2x_4 - x_2x_7 - 125x_4 \leq 0, \\ & g_6(x) = x_3x_5 - 2500x_5 - x_3x_8 + 125 \times 10^4 \leq 0. \end{aligned}$$

As this is a challenging problem, several authors have not considered it under test except for the BAT algorithm. However, our proposed SDDS-SABC algorithm could solve this tough problem. The results of this problem obtained by both BAT and SDDS-SABC have been presented in Table 12.
Therefore, the results show that SDDS-SABC algorithm is efficient enough to extract the optimal values.
We study the robustness of our proposed dynamic penalty-based constraint handling techniques integrated into the SDDS-SABC method on EDPs (see Table 13).

9 Conclusion

A new and effective hybrid SDDS-SABC method for handling challenging restricted optimization issues was presented in this paper. As far as we are

Table 12: Result's comparison obtained from various algorithms for engineering design problems

EDP1		EDP2		EDP3		EDP4		EDP5	
Algorithm	best-found	Algorithm	best-found	Algorithm	best-found	Algorithm	best-found	Algorithm	best-found
SC-GWO	263.8963	SC-GWO	0.012672	SC-GWO	6059.7179	BASZNN	1.3301	BASZNN	-
SCA	263.9506	GWO	0.012675	GWO	6136.6600	BAS	1.3331	BAS	-
PSO	263.8986	PSO	0.012675	PSO	6061.0777	BAS-WPT	1.3011	BAS-WPT	-
wPSO	263.8994	SCA	0.012678	SCA	6076.3651	BSAS	1.3000	BSAS	-
GWO	263.9497	GSA	0.012702	GSA	8538.8360	ZNN	1.3400	ZNN	-
CS	263.9716	RW-GWO	0.012674	PSO	6061.0780	ALO	1.3300	ALO	-
Ray & Saini	264.3000	GA	0.012705	GA	6288.7450	GCA I	1.3400	GCA I	-
Tsai	263.68	PSO (He & Wang)	0.012675	GA	6410.3810	GCA II	1.3400	GCA II	-
mGWO	263.8967(IF)	ES	0.012681	DE	6059.7340	CS	1.3399	CS	-
wGWO	263.8964	RO	0.0126788	ACO	6059.0888	SOS	1.3300	SOS	-
m-SCA	263.9481	CC	0.012833	ES	6059.7456	EPO	1.1900	EPO	-
OBSCA	263.9463	MO	0.012730	BB	8129.1040	SDDS-SABC	**1.1383**	BAT	7049.2480
SSA	263.8958	mGWO	0.012676	LM	7198.043			SDDS-SABC	**6948.2644**
MFO	267.1922	wGWO	0.012672	mGWO	6059.7359				
WOA	263.9858	m-SCA	0.012725	wGWO	6059.7207				
ISCA	263.9002	OBSCA	0.012874	m-SCA	0.012725				
Chaotic SSA	267.192	MFO	0.012758	OBSCA	0.012874				
SDDS-SABC	**263.605322**	WOA	0.012676	MFO	6059.7143				
		SSA	0.012676	WOA	6059.7410				
		ISCA	0.01270	SSA	6059.7254				
		Chaotic SSA	0.012668	ISCA	0.01270				
		SDDS-SABC	**0.018326**	Chaotic SSA	0.012668				
				SDDS-SABC	**6031.8439**				

CC - Constraint Correction; MO - Mathematical Optimization; BB - Branch and Bound; LM - Lagrangian Multiplier

Table 13: Result's comparison on performance of different penalty methods

Problems	(a)	(b)	(c)	(d)
EDP1	263.6083840317	263.7490760091	263.6152947853	263.6053225360152
EDP2	Infeasible	Infeasible	Infeasible	0.018326244566
EDP3	5.906791130523017	3.46209183752	2.880236483910	1.138339974684857
EDP4	6095.623409850	6113.738986468	6032.2416268761	6031.843954691072
EDP5	7913.60005519897	8613.847481208	8527.930681880	6948.26443981868
	(a) [35];	(b): [34];	(c): [33] ;	(d): Proposed

aware, the ABC algorithm performed better at exploration than exploitation due to an imbalance between its exploration and exploitation capabilities. We have modified the ABC algorithm's startup step to maximize exploitation by producing the initial solution using a quasi-random sequence based on the Halton set. Additionally, we have enhanced the scout bee's phase using the sigmoid function and implemented a new search strategy scheme for the bees in use. To meet the constraints, we have designed a new penalty function that is something akin to dynamic penalty logic. The performance of our suggested SDDS-SABC approach is confirmed by the numerical results shown here. Our technique can be applied to other real-life restricted optimization problems in engineering and management, as demonstrated by the exciting experimental findings of real-life engineering design problems.

Acknowledgements

The first author is thankful to DST (Department of Science & Technology, New Delhi, India) for granting DST INSPIRE Research Fellowship (IF 190027) for pursuing Ph.D degree.

Declaration of Competing Interest

The authors declare that they have no known competing nancial interests or personal relationships that could have appeared to inuence the work reported in this paper.

References

[1] Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M. and Gandomi, A. *The arithmetic optimization algorithm.* Comput. Methods Appl. Mech. Eng., 376 (2021) 113609.

[2] Akashah, F. *A review of optimization techniques application for building performance analysis.* Civ. Eng. J., 8(4) (2022) 823–842.

[3] Bansal, J., Joshi, S. and Sharma, H. *Modified global best artificial bee colony for constrained optimization problems.* Comput. Electr. Eng., 67 (2018) 365–382.

[4] Bertsimas, D. and Tsitsiklis, J. *Simulated annealing.* Stat. Sci., 8 (1) (1993) 10–15.

[5] Brajevic, I. *Crossover-based artificial bee colony algorithm for constrained optimization problems.* Neural Comput. Appl., 26 (6) (2015) 1587–1601.

[6] Brajević, I. and Ignjatović, J. *An upgraded firefly algorithm with feasibility-based rules for constrained engineering optimization problems.* J. Intell. Manuf., 30 (7) (2019) 2545–2574.

[7] Brajevic, I. and Tuba, M. *An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems.* J. Intell. Manuf., 24 (4) (2013) 729–740.

[8] Cheng, Z., Song, H., Wang, J., Zhang, H., Chang, T. and Zhang, M. *Hybrid firefly algorithm with grouping attraction for constrained optimization problem.* Knowl.-Based Syst., 220 (2021) 106937.

[9] Cui, L., Deng, J., Zhang, Y., Tang, G. and Xu, M. *Hybrid differential artificial bee colony algorithm for multi-item replenishment-distribution problem with stochastic lead-time and demands.* J. Clean. Prod., 254 (2020) 119873.

[10] D'Angelo, G. and Palmieri, F. *GGA: A modified genetic algorithm with gradient-based local search for solving constrained optimization problems.* Inf. Sci., 547 (2021) 136–162.

[11] Das, R., Das, K. and Mallik, S. *An improved quadratic approximation-based Jaya algorithm for two-echelon fixed-cost transportation problem under uncertain environment.* Soft Comput., 26 (2022) 10301–10320.

[12] Deb, K. *An efficient constraint handling method for genetic algorithms.* Comput. Methods Appl. Mech. Eng., 186 (2000) 311–338.

[13] Dorigo, M. and Di Caro, G. *Ant colony optimization: a new meta-heuristic.* Proc. Congr. Evol. Comput. (CEC99), 2 (1999) 1470–1477.

[14] Duong, H., Nguyen, Q., Nguyen, D. and Van Nguyen, L. *PSO based hybrid PID-FLC Sugeno control for excitation system of large synchronous motor.* Emerg. Sci. J., 6(2) (2022) 201–216.

[15] Fu, X., Pace, P., Aloi, G., Yang, L. and Fortino, G. *Topology optimization against cascading failures on wireless sensor networks using a memetic algorithm.* Comput. Netw., 177 (2020) 107327.

[16] Garg, H. *A hybrid GSA-GA algorithm for constrained optimization problems.* Inf. Sci., 478 (2019) 499–523.

[17] Gu, X. *Application research for multiobjective low-carbon flexible job-shop scheduling problem based on hybrid artificial bee colony algorithm.* IEEE Access, 9 (2021) 135899–135914.

[18] Guermoui, M., Gairaa, K., Boland, J. and Arrif, T. *A novel hybrid model for solar radiation forecasting using support vector machine and bee colony optimization algorithm: review and case study.* J. Sol. Energy Eng., 143 (2021) 020801.

[19] Gupta, S. and Deep, K. *Enhanced leadership-inspired grey wolf optimizer for global optimization problems.* Eng. Comput., 36 (2020) 1777–1800.

[20] Jabeen, S.D. *Vibration optimization of a passive suspension system via genetic algorithm.* Int. J. Model. Simul. Sci. Comput., 4 (2013) 1250022.

[21] Jabeen, S.D. *Vehicle vibration and passengers comfort.* Int. Conf. Comput. Intell., vol 509. Springer, Singapore (2015) 357–372.

[22] Javaheri, D., Gilani, A. and Ghaffari, A. *Energy-efficient routing in IoT networks with ABC optimization and machine learning for smart city infrastructure.* Front. Collaborative Res., 2 (2024) 1–13.

[23] Jiao, L., Li, L., Shang, R., Liu, F. and Stolkin, R. *A novel selection evolutionary strategy for constrained optimization.* Inf. Sci., 239 (2013) 122–141.

[24] Karaboga, D. *An idea based on honey bee swarm for numerical optimization.* Tech. Rep. TR06, Erciyes Univ., Fac. Eng. Comput., 2005.

[25] Karaboga, D. and Akay, B. *A modified artificial bee colony (ABC) algorithm for constrained optimization problems.* Appl. Soft Comput., 11 (2011) 3021–3031.

[26] Karaboga, D. and Basturk, B. *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm.* J. Global Optim., 39 (2007) 459–471.

[27] Kennedy, J. and Eberhart, R. *Particle swarm optimization.* Proc. Int. Conf. Neural Netw. (ICNN'95), 4 (1995) 1942–1948.

[28] Li, X. and Yin, M. *Self-adaptive constrained artificial bee colony for constrained numerical optimization.* Neural Comput. Appl., 24 (2014) 723–734.

[29] Liang, J., Runarsson, T., Mezura-Montes, E., Clerc, M., Suganthan, P., Coello, C. and Deb, K. *Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization.* J. Appl. Mech., 41 (2006) 8–31.

[30] Liang, R., Wu, C., Chen, Y. and Tseng, W. *Multi-objective dynamic optimal power flow using improved artificial bee colony algorithm based on Pareto optimization.* Int. Trans. Electr. Energy Syst., 26 (2016) 692–712.

[31] Liu, F., Sun, Y., Wang, G. and Wu, T. *An artificial bee colony algorithm based on dynamic penalty and Lévy flight for constrained optimization problems.* Arab. J. Sci. Eng., 43 (2018) 7189–7208

[32] Liu, H., Cai, Z. and Wang, Y. *Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization.* Appl. Soft Comput., 10 (2010) 629–640.

[33] Liu, H., Xu, B., Lu, D. and Zhang, G. *A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm.* Appl. Soft Comput., 68 (2018) 360–376.

.

[34] Liu, J., Teo, K., Wang, X. and Wu, C. *An exact penalty function-based differential search algorithm for constrained global optimization.* Soft Comput., 20 (2016) 1305–1313.

[35] Liu, J., Wu, C., Wu, G. and Wang, X. *A novel differential search algorithm and applications for structure design.* Appl. Math. Comput., 268 (2015) 246–269.

[36] Liu, M., Yuan, Y., Xu, A., Deng, T. and Jian, L. *A learning-based artificial bee colony algorithm for operation optimization in gas pipelines.* Inf. Sci., 690 (2025) 121593.

[37] Long, W., Liang, X., Cai, S., Jiao, J. and Zhang, W. *An improved artificial bee colony with modified augmented Lagrangian for constrained optimization.* Soft Comput., 22 (2018) 4789–4810.

[38] Mani, A. and Patvardhan, C. *A novel hybrid constraint handling technique for evolutionary optimization.* Proc. IEEE Congr. Evol. Comput., (2009) 2577–2583.

[39] M'Dioud, M., Bannari, A., Er-Rays, Y., Bannari, R. and El Kafazi, I. *A Modified ABC Algorithm For The Best Placement Of DG Units.* Proc. Glob. Power, Energy Commun. Conf. (GPECOM), (2025) 392–399.

[40] Mezura-Montes, E. and Cetina-Domínguez, O. *Empirical analysis of a modified artificial bee colony for constrained numerical optimization.* Appl. Math. Comput., 218 (2012) 10943–10973.

[41] Mirjalili, S., Mirjalili, S. and Lewis, A. *Grey wolf optimizer.* Adv. Eng. Softw., 69 (2014) 46–61.

[42] Mitchell, M., Holland, J. and Forrest, S. *When will a genetic algorithm outperform hill climbing.* Adv. Neural Inf. Process. Syst., 6 (1993) 51–58.

[43] Oubbati, O., Khan, A. and Liyanage, M. *Blockchain-enhanced secure routing in FANETs: Integrating ABC algorithms and neural networks for attack mitigation.* Synthesis, 2 (2024) 1–11.

[44] Patra, J., Yadav, A., Verma, R., Pal, N., Samantaray, S., Sahu, K., Singh, P., Parihar, R. and Panda, A. *Efficient multi-objective approach using ABC algorithm for minimizing generation fuel cost and transmission loss through FACTS controllers placement and sizing.* Iran. J. Sci. Technol. Trans. Electr. Eng., 49 (2025) 1313–1335.

[45] Peng, C., Liu, H. and Gu, F. *A novel constraint-handling technique based on dynamic weights for constrained optimization problems.* Soft Comput., 22 (2018) 3919–3935.

[46] Phoemphon, S. *Grouping and reflection of the artificial bee colony algorithm for high-dimensional numerical optimization problems.* IEEE Access, 12 (2024) 91426–91446.

[47] Pramanik, P. and Maiti, M. *An inventory model for deteriorating items with inflation induced variable demand under two level partial trade credit: A hybrid ABC-GA approach.* Eng. Appl. Artif. Intell., 85 (2019) 194–207.

[48] Price, K. *Differential evolution vs. the functions of the 2/sup nd/ICEO.* Proc. IEEE Int. Conf. Evol. Comput., (1997) 153–157.

[49] Pu, Q., Xu, C., Wang, H. and Zhao, L. *A novel artificial bee colony clustering algorithm with comprehensive improvement.* Vis. Comput., 38 (2022) 1395–1410.

[50] Rashedi, E., Nezamabadi-Pour, H. and Saryazdi, S. *GSA: a gravitational search algorithm.* Inf. Sci., 179 (2009) 2232–2248.

[51] Rathod, V., Gumaste, S., Guttula, R., Zade, S. and Singh, R. *Optimization of energy consumption in mobile Ad-Hoc networks with a swarm intelligence-based ABC algorithm.* Discov. Appl. Sci., 7 (2025) 805.

[52] Rezaee Jordehi, A. *A chaotic-based big bang–big crunch algorithm for solving global optimisation problems.* Neural Comput. Appl., 25 (2014) 1329–1335.

[53] Runarsson, T. and Yao, X. *Stochastic ranking for constrained evolutionary optimization.* IEEE Trans. Evol. Comput., 4 (2000) 284–294.

[54] Satapathy, S. and Naik, A. *Data clustering based on teaching-learning-based optimization.* Proc. Int. Conf. Swarm, Evol. Memetic Comput., (2011) 148–156.

[55] Şenel, F., Gökçe, F., Yüksel, A. and Yiğit, T. *A novel hybrid PSO–GWO algorithm for optimization problems.* Eng. Comput., 35 (2019) 1359–1373.

[56] Sharma, D. and Jabeen, S. *Hybridizing interval method with a heuristic for solving real-world constrained engineering optimization problems.* Struct., 56 (2023) 104993.

[57] Shi, Y. *Brain storm optimization algorithm.* Proc. Int. Conf. Swarm Intell., (2011) 303–309.

[58] Surono, S., Goh, K., Onn, C., Nurraihan, A., Siregar, N., Saeid, A. and Wijaya, T. *Optimization of Markov weighted fuzzy time series forecasting using genetic algorithm (GA) and particle swarm optimization (PSO).* Emerg. Sci. J., 6 (2022) 1375–1393.

[59] Takahama, T. and Sakai, S. *Efficient constrained optimization by the ε constrained adaptive differential evolution.* Proc. IEEE Congr. Evol. Comput., (2010) 1–8.

[60] Tessema, B. and Yen, G. *An adaptive penalty formulation for constrained evolutionary optimization.* IEEE Trans. Syst. Man Cybern. A, 39 (2009) 565–578.

[61] Tran, S., Vu, H., Pham, T. and Hoang, D. *Constrained Pareto-Based Weighted-Sum ABC Algorithm for Efficient Sensor Networks Deployment.* Proc. Int. Conf. Green Technol. Sustain. Dev., (2024) 310–321.

[62] Wang, Y., Cai, Z., Guo, G. and Zhou, Y. *Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems.* IEEE Trans. Syst. Man Cybern. B, 37 (2007) 560–575.

[63] Wang, Z. and Kong, X. *An enhanced artificial bee colony algorithm for constraint optimization.* Eng. Lett., 32 (2024) 276.

[64] Yeniay, O. *Penalty function methods for constrained optimization with genetic algorithms.* Math. Comput. Appl., 10 (2005) 45–56.

[65] Yesodha, K., Krishnamurthy, M., Selvi, M. and Kannan, A. *Intrusion detection system extended CNN and artificial bee colony optimization in wireless sensor networks.* Peer-to-Peer Netw. Appl., 17 (2024) 1237–1262.

[66] Zhang, X., Lou, Y., Yuen, S., Wu, Z., He, Y. and Zhang, X. *Hybrid artificial bee colony with covariance matrix adaptation evolution strategy for economic load dispatch.* Proc. IEEE Congr. Evol. Comput. (CEC), (2019) 204–209.

[67] Zhang, Z., Ding, S. and Jia, W. *A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems.* Eng. Appl. Artif. Intell., 85 (2019) 254–268.

[68] Zhu, G. and Kwong, S. *Gbest-guided artificial bee colony algorithm for numerical function optimization.* Appl. Math. Comput., 217 (2010) 3166–3173.