Proceedings of the Federated Conference on Computer Science and Information Systems pp. 207–214
DOI: 10.15439/2020F81
ISSN 2300-5963 ACSIS, Vol. 21

An Effective Integrated Metaheuristic Algorithm For Solving Engineering Problems

Adis Alihodzic
University of Sarajevo, BiH
Department of Mathematics
ul. Zmaja od Bosne, 33-35, Sarajevo
Email: adis.alihodzic@pmf.unsa.ba

Sead Delalic
University of Sarajevo, BiH
Department of Mathematics
ul. Zmaja od Bosne, 33-35, Sarajevo
Email: delalic.sead@pmf.unsa.ba

Dzenan Gusic
University of Sarajevo, BiH
Department of Mathematics
ul. Zmaja od Bosne, 33-35, Sarajevo
Email: dzenang@pmf.unsa.ba

***Abstract*—To tackle a specific class of engineering problems, in this paper, we propose an effectively integrated bat algorithm with simulated annealing for solving constrained optimization problems. Our proposed method (I-BASA) involves simulated annealing, Gaussian distribution, and a new mutation operator into the simple Bat algorithm to accelerate the search performance as well as to additionally improve the diversification of the whole space. The proposed method performs balancing between the grave exploitation of the Bat algorithm and global exploration of the Simulated annealing. The standard engineering benchmark problems from the literature were considered in the competition between our integrated method and the latest swarm intelligence algorithms in the area of design optimization. The simulations results show that I-BASA produces high-quality solutions as well as a low number of function evaluations.**

I. Introduction

IN THE last fifteen years, it was shown that most design nonlinear constrained optimization problems are an essential class of issues in real-world applications, and almost all are characterized as NP-hard problems. For such design optimization problems, the finding of the best solution may require centuries, even with a supercomputer. These highly nonlinear and multimodal optimization problems are based on the optimization of objective functions with complex constraints which usually involve thousands of or even millions of elements, and they were written in the form of simple bounds or more often as nonlinear inequalities. Nonlinearly constrained optimization problems contain continuous and discrete design variables, nonlinear objective functions, and constraints, some of which may be active at the global optima. Due to the complex nature of an objective function, as well as the constraints that need to be met, it is challenging how to effectively and robustly explore overall search space. Therefore, practically solving engineering problems are come down to some efficient methods which are problem-specific [1]. Since optimization methods can not escape falling in into some of the local optima, metaheuristics as very modern and efficient global techniques are considered to overcome these type of problems [2]. Besides, those are capable of generating quality solutions in a reasonable amount of time. The creating of quality solutions is related to the establishment of the right balance between exploration and exploitation. [3]. Since a magic formula does not exist, which works for all types of problems [4], in this paper, several swarm intelligence algorithms [5] have been adopted for solving nonlinear engineering problems. Some of the most popular swarm intelligence optimization techniques are artificial bee colony(ABC) [6][7][8][9], firefly algorithm (FA) [10][1][11][12], cuckoo search (CS) [13][14], bat algorithm (BA) [15][16][17][18][19], flower pollination algorithm [20], and etc. In this article, we have combined the bat algorithm as a representative of swarm intelligent multi-agent algorithm with one agent simulated annealing method to produce as much as possible suboptimal solutions.

The Bat meta-heuristic algorithm (BA) has proposed by Xin-She Yang 2010 [15]. The primary mechanism of this swarm intelligence technique propagates echolocation of bats as agents. The agents seek for prey and avoid obstacles by using echolocation. In the paper [19], it has been shown that the BA very well performs local search, but at times it deviated into some local optima, and it can not reach the optimal solution while solving a hard problem. The original version of bat algorithm, as well as the other metaheuristic algorithms, were designed to address unconstrained problems. To tackle the constrained problems, bat algorithm (BA) uses a penalty approach as a constraint handling technique [16]. From the experiments presented in [16], it can be seen that the BA is almost always superior to other metaheuristics.

To promote the results obtained by the simple bat algorithm, in this article, we propose an integrated I-BASA approach to take on engineering problems. Unlike the original bat algorithm which is not capable to found satisfying balance between diversification and intensification, the proposed I-BASA approach based on simulated annealing (SA) [21], a new mutation operator, and Gaussian distribution achieves a right balance and raises overall search performance. The integrated I-BASA method was tested on the eight well-chosen benchmark problems, and the simulation results report that our approach almost always wins the state-of-the-art algorithms regarding the convergence and accuracy. In this paper, we have decided to exploit Deb's rules as a constraint handling process instead of a standard penalty method. Throughout the simulation results, it can be seen that introduced rules significantly improve the quality of the solutions.

The basic structure of the article looks like this. The basic definitions related to constrained optimization are described in

Section II, while the detailed description of the Bat Algorithm (BA) and Simulated Annealing (SA) is presented in Section III and Section IV, respectively. Details of our I-BASA approach are in Section V. The brief review of eight engineering optimization problems is there in Section VI. Parameter settings and comparative results of applying state-of-the-art algorithms for solving engineers problems are presented in Section VII. Ultimately, the article is concluded in the last Section VIII.

II. Constrained Optimization

The general form of most engineering problems is expressed over objective functions and constraints, which are usually nonlinear manner. These problems are considered as constrained optimization problems containing inequality and equality constraints. They become increasingly difficult or even impossible when the traditional techniques are employed for their solving. Generally, their solving can be reduced on the next nonlinear programming problem

$$\min_{\mathbf{x}\in\mathbf{F}\subset\mathbb{R}^n} \ f(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is a decision vector composed of n decision variables

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \tag{2}$$

The decision variables x_i may have continuous or discrete values, where each of them is limited by its lower bound L_i and upper bound U_i $(i = 1, \ldots n)$. The objective function f is defined on an n-dimensional hypercube $\mathbf{S}$ such that $S \subset \mathbb{R}^n$. It is used as a measure of effectiveness of a decision. The sets $\mathbf{F} \subseteq \mathbf{S}$ and $\mathbf{U} = \mathbf{S} \setminus \mathbf{F}$ denote feasible and infeasible search space, respectively. The feasible region can being presented as follows

$$\begin{aligned} \psi_k(\mathbf{x}) &\leq 0 \quad (k = 1 \ldots K) \\ \phi_j(\mathbf{x}) &= 0 \quad (j = 1 \ldots J), \end{aligned} \tag{3}$$

where letters K and J denote the number of inequality and equality, respectively. If a solution $\mathbf{x} \in F$, then all constraints defined by Eq. 3 must be satisfied. Otherwise, some of the constraints does not hold. For optimization algorithms, the participation of the equality constraints poses a problem in the sense of reducing available space F, so inequality ones usually replace those in this way

$$|\psi_k(\mathbf{x})| \leq \epsilon \quad (\forall k), \tag{4}$$

where $\epsilon \geq 0$ is a small violation tolerance.

It is well-known that swarm intelligence algorithms can not directly solve constrained engineering problems because they were designed for unconstrained ones. Therefore, the mapping of constrained problems into unconstrained ones is achieved by either using a penalty function or utilizing the fly-back mechanism. By using the penalty functions, a constrained issue is being addressed as an unconstrained in such way that infeasible solutions are punished or "penalized" so that the selection process favours feasible solutions. In this way, in the latest phases executing of algorithms, the search is directed towards the feasible regions of search space. The advantage of penalty functions lies in their simplicity and easy implementation, but the most challenging aspect of them lies in the finding appropriate penalty parameters in pursuit of constrained optimum. Their performance is not always satisfactory, and there is a need for more sophisticated penalty functions.

III. Bat Algorithm

In basic Bat algorithm (BA) offered by Yang [15], bats are being moved in a specific area thanks to the time delay between emission and reflection of the signal. Other words, bats produce a booming, but not long stroke and then monitor for the answers returned from the nearby objects. They have various rates of pulse emission and frequency. For experimental purposes, the pulse can be taken from $[0, 1]$, where 0 means that the emission does not exist and 1 means that the bats perform their maximum emitting. In the conventional bat algorithm, Yang has been proposed three idealized rules. The first rule states that each bat can determine distance by using echolocation, as well as it knows the background in some mysterious way. The second rule says that each bat flies entirely arbitrary when it hunts for prey. Also, any bat can adjust the wavelength of own emitted pulses and modify the vibration emission depending on the closeness of the victim. The last rule declares that loudness ranges of high positive value to some small constant value.

It is essential to highlight that the original Bat algorithm, besides standard control parameters, has a few relevant parameters. Those parameters are frequency tuning, climbing within a promising neighbourhood, shifting between exploration and exploitation. As we mentioned earlier, in order to deal with constrained design problems, Bat algorithm has to be modified. The use of penalty functions for reducing of constrained optimization to an unconstrained one, to which the pure Bat algorithm can be later applied, does not yield reliable results. Namely, the mentioned transformation demands much fine-tuning of the penalty elements that predict the quantity of penalization to be engaged. Since the shortage of punishment strategy does not commonly deliver satisfactory outcomes, we decided to employ the following three of Deb's rules in our I-BASA approach. The first Deb's rule tells that an algorithm chooses among two feasible solutions, the one with the better objective function value. Based on the second Deb's rule, a feasible solution beats infeasible one. In the last Deb's rule, if both solutions are infeasible, the one with the weakest amount of constraint violation was favoured. Some difficulties can appear in issues in which the global optima lies on frontier within feasible and infeasible parts.

Techniques for solving constrained design problems mainly begin with solutions which are not within the feasible area. Our proposed Bat algorithm for solving engineering problems also does not begin with the feasible initial population. During the running process, Deb's feasibility rules direct the solutions to the feasible region. Hence, slightly infeasible solutions are not discarded but kept in the population. They are utilized in

the generation in the next iteration with the hope of giving feasible solutions. In this strategy, initially larger error values are used, and this value is gradually reduced with each iteration until it reaches to whatever acceptable error value. The pseudo-code of the BA strategy for solving constrained engineering problems can be summarized in this way:

Step 1. *The building of beginning agents*: Build an initial group of N agents (bats) $(i = 1, 2, \ldots, N)$ which are randomly dispersed. Before beginning an iterative process, evaluate them, and by utilising Deb's rule, determine the fittest solution as $\mathbf{x}_{best}$.

Step 2. *Investigating of novel solutions*: Querying for a new promising solution $\mathbf{x}_i^t$ is done by Eq. 5 and Eq. 6.

$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} + \mathbf{v}_i^t, \tag{5}$$

$$\mathbf{v}_i^t = \mathbf{v}_i^{t-1} + (\mathbf{x}_{best} - \mathbf{x}_i^{t-1}) f_i, \tag{6}$$

In Eq. 5 and Eq. 6 letters $\mathbf{v}_i^t$ and $\mathbf{x}_{best}$ present agent quickness of change and current global most suitable solution, sequentially, while the alphabet f_i in Eq. 6 is the frequency which is being yielded as

$$f_i^t = f_{min} + (f_{max} - f_{min})\beta, \tag{7}$$

where β is a random quantity uniformly extracted from $[0, 1]$, while the letters f_{min} and f_{max} are constants which are usually initialized to 0 and 2, respectively.

It is worth noting here when a new vector $\mathbf{x}_i^t$ has been built by Eq. 5, then if its arbitrary element x_i^j is not inside the interval $[L_i,\ U_i]$, it will be substituted by the element $L_i + |x_i^j|\%(U_i - L_i)$.

Step 3. *Intensification and diversification*: In this step depending on the values $rand_1$ and pulse rate r_i^t, it is performed intensification and diversification by applying the innovative operator

$$\mathbf{x}_{new} = \begin{cases} \mathbf{x}_{best} + \epsilon A_t, & \text{if } rand_1 > r_i^t \\ \mathbf{x}_i^t, & \text{else} \end{cases} \tag{8}$$

where $A_t =< A_i^t >$ denotes noisiness on average in the iteration t of all agents $\mathbf{x}_i^t$, while the parameters ϵ and $rand_1$ are random numbers uniformly chosen from the intervals $[0, 1]$ and $[-1, 1]$, respectively. In Eq. 8, the pulse rate r_i^t was defined by

$$r_i^t = r_i^0(1 - e^{-\beta t}), \tag{9}$$

where $r_i^0 \in$ is an initial pulse rate of the ith agent, and β is a fixed number. Additionally, in this step, edge conditions have to be controlled as in Step 2.

Step 4. *The election of a different candidate in fly*: In this step, if the solution $\mathbf{x}_{new}$ in the sense of Deb's rules has better cost value than the past one $\mathbf{x}_i^{t-1}$ or holds $A_i^t > rand_2$, then the solution $\mathbf{x}_i^t$ and the cost value $f(\mathbf{x}_i^t)$ are modified to $\mathbf{x}_{new}$ and $f(\mathbf{x}_{new})$, respectively. Here, the letter $rand_2$ is a random number from $[0, 1]$, while the loudness A_i^t can be expressed as

$$A_i^t = \alpha A_i^{t-1}, \tag{10}$$

where the changeless factor α behaves likewise to the cooling constant in the SA algorithm.

Step 5. *Record the fittest solution*: According to Deb's rules, write down the fittest solution as $\mathbf{x}_{best}$.

Step 6. *The end criteria*: The algorithm is over if the finish criteria are reached or all iterations of the algorithm are consumed. Otherwise, revert to Step 2.

IV. Simulated Annealing

In this section, we explain the simulated annealing (SA) algorithm as one of the fundamental and often picked heuristic technique [21].Simulated annealing is a well-known heuristic algorithm, whose mechanism is relied on the annealing procedure through metal adaptation. If the convergence period is prolonged, this algorithm can almost always achieve a global convergence. The primary preference of simulated annealing is that it can control its transition probability by controlling temperature, which further implies that the algorithm principally escapes to being caught into some local optima. Since simulated annealing is one kind of Markov chain, the fundamental steps of this method for solving constrained design problems are:

Step 1. Select temperature T_0, generate randomly drawn components of a solution $\mathbf{x}_0$ and the counter of iterations t sets to one.

Step 2. Determine the stopping temperature T_{stop}, set n as a maximum number of iterations and define the cooling table as follows

$$T_t = \alpha T_{t-1}, \quad \alpha \in (0, 1), \tag{11}$$

where α is a cooling schedule factor.

Step 3. Randomly select a new solution $\mathbf{x}_{t+1}$ as follow

$$\mathbf{x}_{t+1} = \mathbf{x}_t + r_1, \tag{12}$$

where $r_1 \in (0,\ 1)$ denotes an uniform random number.

Step 4. Calculate the difference Δf between the fitness values of the solutions $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$ as follow

$$\Delta f = f(\mathbf{x}_{t+1}) - f(\mathbf{x}_t), \tag{13}$$

where $f(\mathbf{x})$ is the cost value of vector $\mathbf{x}$.

Step 5. According to Deb's rules, the new solution $\mathbf{x}_{t+1}$ generated by Eq. 12 is being accepted if it has more useful fitness value than $\mathbf{x}_t$. Otherwise, the solution $\mathbf{x}_{t+1}$ will be selected if the following condition is satisfied

$$p = e^{\frac{-\Delta f}{T}} > r, \tag{14}$$

where p denotes the transition probability, and $r \in (0, 1)$ is randomly chosen number.

Step 6. Memorize the current optimal solution $\mathbf{x}_*$ and the best cost value $f(\mathbf{x}_*)$. Reduce the temperature T due to the Eq. 11.

Step 6. If termination criterion $T > T_{stop}$ is not valid or holds $t \geq n$, then the iteration procedure is over. Oppositely, set $t \leftarrow t + 1$, and repeat above steps from Step 3 to Step 6.

There are many types of research on how to merge the simulated annealing and other optimization techniques to obtain hybrid methods [22]. In this paper for engineering optimization, for the first time, we integrate simulated annealing, Gaussian distribution, and a new mutation operator with the original BA to extra improve the searchability and also accelerate overall convergency.

V. An Integrated Bat Approach For Solving Constrained Engineering Problems: I-BASA

By analysing preliminary outcomes shown in the paper [16], we can infer that standard bat algorithm has succeeded at least once to produce near-optimal solutions during 30 independent runs. However, although it was able to generate acceptable solutions using a small number of evaluations, it can be perceived based on statistical results, how it is less stable contrast to other algorithms. The main disadvantages can be classified as a short seeking of search space and not well established an equilibrium between exploitation and exploration. To overcome discussed drawbacks, we incorporate some parts of the SA algorithm, a new mutation operator, and Gaussian perturbations into the original bat algorithm to improve its performance. As a result, we provide the I-BASA approach in solving engineering optimization problems. By applying this method, the overall stability will be increased because a better exploration disables algorithm being trapped in some local optimum. Also, as another consequence of that, the enhanced integrated bat algorithm will not iterate until all iterations are exhausted, and it only will require a few iterations for obtaining high-quality solutions. Hence, the proposed I-BASA consists of two significant parts similarly as it was done in the case of unconstrained optimization [23]. In the first part, as soon as the algorithm builds the first group of agents, fittest solutions are changed by novel solutions produced by employing SA, accompanied by the original updating formulas of the bat algorithm. In the second part of the mentioned approach, Gaussian distribution is utilised to scatter locations as much as possible. Also, a new mutation operator was introduced in order to raise the convergency of approach and establish an acceptable ratio between intensification and diversification. Also, the I-BASA includes Deb's rules to manage constraints instead of a penalty approach shown in paper [16]. Experimental analysis will show that our proposed I-BASA can efficiently perform intensification as well as diversification of the space compared to the rest algorithms. The details of our proposed I-BASA approach are given as follows:

Step 1. Our I-BASA method begins by randomly generating population P containing n agents $\mathbf{x}_i = (x_{i,j})_{j=1}^{d}$ $(i = 1, \cdots, n)$ of dimension d, where each vector $\mathbf{x}_i$ can be solution of an engineering problem. Also, in this step are initialized initial loudness A_i, pulse rates r_i and r_i^0 $(\forall i = 1, \cdots, n)$ as well as the annealing constant in SA algorithm. Before starting the iterative search process, for each solution, $\mathbf{x}_i$ fitness value is evaluated, and according to Deb's rules, the algorithm identifies both fittest solution $\mathbf{x}_{best}$ and the smallest violation g_{min}. After that, it determines the starting temeperature T_0 and the cycle counter t is reset to 0.

Step 2. Adaptation value of any agent $\mathbf{x}_i$ $(i = 1, \cdots, n)$ in the current temperature t can be depicted as:

$$Av(\mathbf{x}_i) = \frac{e^{-\frac{f(\mathbf{x}_i) - f(\mathbf{x}_{best})}{t}}}{\sum_{i=1}^{n} e^{-\frac{f(\mathbf{x}_i) - f(\mathbf{x}_{best})}{t}}} \tag{15}$$

According to the roulette selection strategy, the alternative solution $\mathbf{x}_{best}^{'}$ was picked up among all bats, while the new formula calculates the new velocity of movement v_i^t

$$\mathbf{v}_i^t = \mathbf{v}_i^{t-1} + (\mathbf{x}_{best}^{'} - \mathbf{x}_i^{t-1}) f_i, \tag{16}$$

where frequency f_i is selected by Eq. 7. To additionally boost the heterogeneity of agents into space, we introduced Gaussian operator δ by Eq. 5. Hence, the estimation of the solution $\mathbf{x}_i^t$ is accomplished by driving virtual agents $\mathbf{x}_i^{t-1}$ by the following equation

$$\mathbf{x}_i^t = \delta \mathbf{x}_i^{t-1} + \mathbf{v}_i^t, \tag{17}$$

where $\delta \in N(0, 1)$. In this step, it is necessary to scan the side conditions of the computed new solutions $\mathbf{x}_i^t$.

Step 3. For each solution $\mathbf{x}_i^t$, it should be checked the condition $r_i < rand_i$. If it is satisfied, then the local search is performed around the solution $\mathbf{x}_{best}$ as follows

$$\mathbf{x}_l^t = \mathbf{x}_{best} + a_1 \epsilon, \tag{18}$$

where $a_1 \in (0, 1)$ is a scaling factor, while $\epsilon \in (-1, 1)$ is a random number. As a result, the new solution $\mathbf{x}_l^t$ was generated. Then, as in Step 2, the boundary conditions have to be controlled for each coordinate of the vector $\mathbf{x}_l$. For the experimental purposes, we have fixed the parameter a_1 to value 0.1.

Step 4. In this step, the algorithm performs both computation sum of the violations and fitness value of the selected solution in Step 3. The generated solution from this stage will be

accepted as a new one if it is better than the previous one according to Deb's rules or it holds the condition $A_i^t > rand_i$. If one of these two conditions is satisfied, then it does perform the update process. It is based on the modifications of old solutions, fitness values, and violations with the new ones. Also, in this step, the rate r_i^t is increased by Eq. 9, while the loudness of signal A_i^t is decreased by Eq. 10. It will be demonstrated throughout the simulation that the most reliable results were found for $r_i^0 = 0.5$, $A_0 = 0.99$, and $\beta = 0.9$.

Step 5. In this step, we apply the new mutation operator $\mathbf{x}_{mut}$ to the previously calculated solutions $\mathbf{x}_i$ to additionally increase the search of the entire scope. The operator $\mathbf{x}_{mut}$ is defined by

$$\mathbf{x}_{mut} = \mathbf{x}_{r_3} + a_2(\mathbf{x}_{r_1} - \mathbf{x}_{r_2}), \tag{19}$$

where r_1, r_2, r_3 are three various randomly chosen numbers in the interval $(0, n)$, and $a_2 \in (0, 2)$ is a scaling factor. Then, we compare the quality of solutions before and after introducing the $\mathbf{x}_{mut}$ operator to attain the optimal solution $\mathbf{x}_{best}$ as well as fitness value $f(\mathbf{x}_{best})$.

Step 6. In this step we memorize the solution $\mathbf{x}_{best}$ and the highest fitness value according to Deb's rules. Also, the smallest violation g_{min} was determined. The value of temperature parameter T is updated from the cooling schedule, which is defined by Eq. 11, where $\alpha = 0.97$.

Step 7. The I-BASA method stops if the end criterion is reached or the counter t is equal max_no_cycles. In contrast, increment t by one and go to Step 2.

VI. Benchmark Problems

In this part, we will quickly outline eight non-linear design problems in order to assess the performance of our proposed I-BASA approach. Each of the eight problems P_i $(i = 1, 2, \cdots, 8)$ has discrete and continuous variables. Table I summarizes basic characteristics of mentioned problems, such as dimension d, number of linear L_e and non-linear N_e inequalities. Complete mathematical formulation and their description in detail can be found in papers [1][13].

P1. Pressure Vessel Design Problem

The basic task of the pressure vessel design problem is to develop a compressed air room with a pressure of 3×10^3 psi and a minimum volume of 750 ft^3. It is determined as a mixed discrete-continuos constrained problem because it has two discrete variables x_1 and x_2 and two continuous variables x_3 and x_4. These variables have the following meaning: x_1 is a shell thickness, x_2 is a thickness of the spherical head, x_3 is a radius of the cylindrical shell and x_4 is a shell length. The first two variables x_1, x_2 take the values inside interval $[0.0625,\ 6.1875]$, while values of the remaining two variables x_3, x_4 belong to interval $[10,\ 200]$. The primary goal is to reduce the total charge of the pressure vessel.

P2. Welded Beam Design Problem

The primary objective of the welded beam design problem is to reduce the construction cost of the welded beam subject to restrictions on shear stress τ, bending stress σ in the beam, end deflection δ of the beam and buckling load P_c on the bar. The length of the beam is equal to 14 in, while the force of size 6000 lb is enforced at the end of the shaft. The design variables related to this problem have the following meaning: x_1 is weld thickness h, x_2 present the clamping rod length l, x_3 denotes rod height t, and x_4 is rod thickness b. These variables are bounded by the following limits: $x_1 \in [0.125,\ 5]$, $x_2,\ x_3,\ x_4 \in [0.1,\ 10]$.

TABLE I
THE MAIN PROPERTIES FOR THE EIGHT BENCHMARK PROBLEMS

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
d	3	4	3	7	4	2	5	11
L_e	2	2	1	4	0	0	0	0
N_e	1	5	3	7	0	3	1	10

P3. Tension/compression spring design problem

The aim of this problem is to reduce the construction cost of the spring, which is limited by four nonlinear constraints. It can be described by three variables x_1, x_2 and x_3, where x_1 is a wire diameter d, x_2 is a mean diameter of the spring D and x_3 is a number of effective coils N. The ranges of those variables are: $x_1 \in [0.05,\ 1.0]$, $x_2 \in [0.25,\ 1.3]$, $x_3 \in [2,\ 15]$.

P4. Speed Reducer Design Problem

The speed deducer design problem is a mixed discrete-continuous optimization problem which describes how to design a simple gearbox. Its application can be exploited between the engine and a propeller of a light aeroplane to achieve a maximum speed of rotation. The primary goal is to perform reducing of the weight for speed reducer subject to restrictions on bending stress of the gear teeth, surface stress, transverse deflections of the shifts, and stress in the shafts. The variables participating in the construction of speed reducer have the following meaning: x_1 is a face width, x_2 is a module of teeth, x_3 is a number of teeth on the pinion, x_4 and x_5 respectively represent the length of the first and second shaft between the bearings. In contrast, x_6 and x_7 are the diameters of the first and the second shaft, respectively. For these seven variables hold: $2.6 \leq x_1 \leq 3.6,\ 0.7 \leq x_2 \leq 0.8,\ 17 \leq x_3 \leq 28,\ 7.3 \leq x_4,\ x_5 \leq 8.3,\ 2.9 \leq x_6 \leq 3.9,\ 5.0 \leq x_7 \leq 5.5$.

P5. Gear Train Design Problem

The gear train design problem is a discrete optimization problem. It represents a complex issue involving a highly non-linear design space. The determination of volume or centre-to-centre distance of gear is an important subject in the design of power transmission systems. The gear ratio for a reduction gear train can be defined as the ratio of the angular velocity between input and output shafts. The total gear train ratio can be defined as follows

$$Gear\ ratio = \frac{w_0}{w_i} = \frac{x_2 x_3}{x_1 x_4} \tag{20}$$

where the variables w_o and w_i present the angular velocities of the output and input shafts, respectively. At the same time,

variables x_1, x_2, x_3 and x_4 denote the numbers of teeth of the gears A, B, C and D, respectively. Those variables take values in the interval $[12,\ 60]$.

P6. Truss design problem with three-bar

The three-bar truss design problem is a continuous optimization problem in civil engineering first proposed by Nowicki in 1974. The purpose of this problem is to seek the optimum cross-section that decreases the weight of the truss. Two design variables x_1 and x_2 are used for its modelling which describe cross-sectional area. The values of mentioned variables are taken from the interval $[0,\ 1]$.

P7. Cantilever Beam Design Problem

The cantilever beam design problem presents a continuous optimization problem proposed by Fleury and Braibant. It can be described by using five connected square hollow blocks in order to make a beam. The beams are strictly braced at the one end, while a vertical force operates on the free end of the cantilever. The main objective of this problem was to minimize the weight of the cantilever. The design space includes five continuous variables x_j and one constraint g_1, where the range of variables x_j $(j = 1, 2, \cdots, 5)$ is the closed interval $[0.01,\ 100.0]$.

P8. Car Side Impact Design Problem

The car side impact design problem formulated by Gu is a mixed-continuous optimization problem. The overall number of elements in the model is approximately 90000, while the total number of nodes is close to 96000. For side-impact protection, two basic side-impact procedures are NHTSA and EEVC [1]. Based on these procedures, a car was exhibited to a side-impact. The prime goal is to reduce the weight using 11 design variables x_j $(j = 1, \cdots, 11)$ and 10 nonlinear constraints g_k $(k = 1, \cdots, 10)$. The bound conditions for these variables x_j are defined with $0.5 \leq x_j \leq 1.5$ $(j = 1, \cdots, 7)$, $x_8,\ x_9 \in \{0.192,\ 0.345\}$, $-30 \leq x_j \leq 30$ $(j = 10,\ 11)$.

VII. Experimental Analysis

In this experimental analysis,we have chosen 8 well-known constrained engineering problems to carry out a straightforward competition between our approach I-BASA and valuable algorithms such as SA, ABC, FA, CS, and BA. All algorithms participating in the simulation were carried out on the local machine which has the following performance:

- **Operating System:** Windows 10x64;
- **Type of processor:** Intel Core i7 3770K processor with a speed of 3.5 GHz;
- **Memory (RAM)**: 16GB;
- **Programming language:** C#;
- **Software:** Visual Studio 2019.

A. Parameter Settings

As metaheuristics have stochastic properties, each experiment was done in 30 series for each of the problems P_1, $P_2, \cdots, P_8$. The run of each algorithm is over when all its iterations are being consumed. For the experimental purposes, each algorithm allocates 2000 iterations. In this case analysis, except standard control parameters, each of the algorithms has extra control parameters which have a direct influence on their execution. The adjustments of algorithm parameters are given below:

- **SA** - The temperature T_0 at the beginning is set to 1.0, the stopping temperature T_{stop} was initialized to 1.0E-10, the beginning search period was set to 500, the annealing constant is equal to 0.5, the maximum number of rejections, acceptance and runs are set to 250, 150, and 50, respectively.
- **ABC** - The max. size of population $SP = 40$, the constant 'limit' is initialized to $SP \times d \times 5$, where d denotes the number of variables of the problem, while the modification rate MR and scout production period SPP were set to 0.9 and 400, respectively.
- **FA** - The max. size of firefly population is 40, the initial value of attractiveness β was set to 0.05, the randomization parameter α takes value from [0, 1]. Other parameters were set as $\beta_0 = 1$ and $\gamma = 1.0$;
- **CS** - The maximum population size SP is equal 40 for all benchmark problems. The parameter p_a of catching a cuckoo egg was set to 0.99;
- **BA** - The max. number of agents is 40, the initial values of the pulse rates and loudness are set to 0.5 and 0.99, respectively, the frequencies f_{min} and f_{max} respectively are set to 0 and 2.0, while both constants α and γ are initialized to 0.9;
- **I-BASA** - The size of bat population is 40, $f_{min} = 0$, $f_{max} = 2.0$, $\alpha = 0.9$, $\gamma = 0.99$, the values of parameters r_i^0 and loudness A_i^0 were initialized to 0.5 and 0.99, respectively. The annealing constant is fixed to 0.5.

B. Discussion of Experimental Results

The experimental results of the algorithms which participate in the competition are reported in Table II. The best feasible solutions demonstrate the capability of an algorithm to discover the optimal solution. At the same time, the statistical quantities such as mean and standard deviation determine the robustness of the algorithm. Also, the maximum number of iterations is closely related to the convergence of the algorithm. Best results are in bold, and those do not violate any of the constraints.

For the problem P_1, only our I-BASA approach obtained fittest solution in each run. On the other hand, the remaining algorithms are not equipped to gain the optimal solution except the ABC algorithm, which generated a slightly worse best result compared to the I-BASA method. Based on the statistical measures, it can be seen that our I-BASA is superior to other algorithms. For the problem P_2, only algorithms such as the I-BASA, CS and ABC have achieved the best optimum. The CS and I-BASA have utilized the smallest number of evaluations, wherein because of the more straightforward structure of the proposed I-BASA, the CS has consumed more than twice CPU time compared to the I-BASA algorithm as it can bee seen in Table III. Also, from the results shown in Table II, it can be observed that the I-BASA was slightly stable compared to the

TABLE II
COMPARISON OF RESULTS BETWEEN THE IBASA METHOD AND OTHER VALUABLE METAHEURISTICS FOR EIGHT DESIGN PROBLEMS OVER 30 INDEPENDENT RUNS

Problem	Statistics	SA	ABC	FA	CS	BA	I-BASA
P_1	*Best*	6099.738697241	6059.712773680	6059.712977959	6059.718470374	6059.796804678	**6059.712773616**
	Mean	75604.673833747	6059.713833747	6059.713518710	6059.711971580	6079.702933294	**6059.712773616**
	SD	4.54E+02	2.66E-06	3.24E-08	7.07E-09	1.18E+01	**6.27E-12**
	ANI	92649.3	1000	1000	2000	600	**1000**
	SP	1	25	40	13	40	**24**
P_2	*Best*	1.728322970	1.724852309	1.724852338	1.724852309	1.725381600	**1.704852309**
	Mean	1.739173713	1.724874566	1.724863780	1.724852309	2.255381896	**1.704852309**
	SD	5.92E-03	6.05E-06	4.02E-05	3.76E-12	5.85E-01	**3.65E-13**
	ANI	78545.8	1000	500	2000	1000	**800**
	SP	1	30	30	10	40	**26**
P_3	*Best*	0.012708232	0.012666879	0.012665383	0.012666450	0.012668443	**0.011215198**
	Mean	0.013009247	0.012797428	0.012694825	0.012695146	0.019492306	**0.011215198**
	SD	2.81E-04	9.96E-05	2.17E-05	2.89E-05	6.55E-03	**1.20E-12**
	ANI	252033.93	1000	600	1000	2000	**684.90**
	SP	1	25	30	20	40	**22**
P_4	*Best*	2994.842065360	**2993.542819303**	2993.542821348	**2993.542819303**	2993.668899790	2993.542819550
	Mean	2998.837062287	**2993.542819303**	2993.544598620	**2993.542819303**	3007.132712792	2993.542825755
	SD	2.53E+00	**2.48E-12**	5.50E-03	**2.65E-12**	6.53E+00	8.16E-06
	ANI	82264.43	**1000**	1000	**2000**	2000	1000
	SP	1	**12**	40	**7**	40	38
P_5	*Best*	3.38E-14	2.79E-13	2.55E-20	1.51E-15	5.71E-15	**1.11E-31**
	Mean	7.19E-10	1.60E-09	2.38E-13	2.79E-09	7.94E-09	**2.03E-16**
	SD	8.00E-10	3.38E-09	1.20E-12	3.24E-09	4.27E-08	**8.58E-16**
	ANI	163655.17	60	60	60	50	**60**
	SP	1	10	10	10	30	**10**
P_6	*Best*	263.896818396	263.895844535	263.895843378	263.895844333	263.895891445	**263.852843376**
	Mean	263.918269245	263.895913071	263.895843384	263.895875913	263.907303271	**263.852843376**
	SD	1.95E-02	7.28E-05	5.31E-09	2.85E-05	1.63E-02	**5.189E-14**
	ANI	87690.93	1000	500	2000	1000	**926.63**
	SP	1	40	35	10	40	**17**
P_7	*Best*	1.343702597	1.339912015	1.339911699	1.339912807	1.339919926	**1.339911698**
	Mean	1.349069456	1.339914165	1.339911722	1.339916864	1.433529924	**1.339911698**
	SD	3.65E-03	1.26E-06	2.73E-08	2.18E-06	2.80E-01	**5.44E-16**
	ANI	74087.9	2000	900	2000	1000	**811.07**
	SP	1	19	40	19	40	**24**
P_8	*Best*	22.851120887	22.843581559	22.842969358	**22.842824093**	22.846273985	**22.842824207**
	Mean	22.880749817	22.855576650	22.843086577	**22.844898883**	24.010514696	**22.843767285**
	SD	7.74E-02	1.57E-02	2.20E-04	**1.20E-03**	5.92E-01	**1.47E-03**
	ANI	70471.57	1000	600	**2000**	1500	**842.63**
	SP	1	40	40	**14**	40	**35**

ANI: *Average number of iterations*, **SP**: *Size of population*

CS algorithm. Further, the I-BASA method has achieved the best result for the problem P_3 as well as the best statistical values, such as mean value and standard deviation. Also, for this problem, the proposed I-BASA has consumed the least number of evaluations to generate the best optimum solution. By analyzing the outcomes in Table II, it can be seen that only CS and ABC have delivered the best optimum for the problem P_4, while the I-BASA and FA have generated slightly worse best results. Moreover, the ABC algorithm has used up the smallest number of evaluations as well as the least required CPU time as it reported in Table III. Since the problem P_5 is not a very hard problem, all algorithms were generated the best optimum. The least number of evaluations have consumed the algorithms ABC, FA, CS, and I-BASA. Our proposed I-BASA has produced the most precise and more stable solutions. For the problem P_6, the I-BASA method required both the smallest number of evaluations and least CPU time to build the robust solution as it can be seen in Table II and Table III. Further, the FA has generated slightly worse best solution than I-BASA, but much better optimal solution than other algorithms using only 17.500 evaluations. Therefore, for this problem, regarding convergence speed as well as robustness, the remaining algorithms are considerably inferior to the I-BASA algorithm. Considering the results of the problem P_7, we can observe that the I-BASA and FA algorithms were made the best results, where the I-BASA has a slight advantage in terms of precision, and drastically better statistical values such as mean value and standard deviation than the original FA. Also, for this problem, equally good results are obtained by ABC and CS algorithms. As it can be seen from Table II, the achieving global optima has cost 19465.68 evaluations by the proposed I-BASA, which is almost twice fewer evaluations than other methods. Finally, by analyzing the simulation results for the last problem P_8, we can conclude that I-BASA and CS algorithms achieve similar best solutions in each run. As it is shown in Table II, both FA and ABC algorithms were produced the acceptable solutions as well. The summary results confirm that the proposed I-BASA

TABLE III
AVERAGE TIME (IN SEC.) CONSUMED BY ALGORITHMS SA, ABC, FA, CS, BA AND I-BASA OVER 30 INDEPENDENT SERIES

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
SA	0.60	0.90	1.47	1.25	0.65	0.29	0.81	0.57
ABC	0.42	0.79	0.48	0.25	0.01	0.35	1.01	0.98
FA	5.91	3.96	2.80	13.58	0.03	1.47	8.92	5.24
CS	1.77	1.42	1.63	1.15	0.04	0.99	3.20	2.47
BA	0.22	0.54	0.65	0.95	0.01	0.19	0.53	0.82
I-BASA	0.41	0.64	0.28	1.00	0.01	0.18	0.59	1.01

approach can accomplish the best solutions from literature for engineering problems $P_1, P_2, \cdots, P_8$. Also, the proposed I-BASA works better than the other algorithms concerning the quality and robustness with noticeably enhanced convergence rate for the bulk of design problems.

VIII. CONCLUSION

The main job of the article was to design intelligent hybridization called I-BASA based on simulated annealing (SA), Bat algorithm (BA), Gaussian perturbations, novel mutation operator, which regulates the diversity of solutions in the population. It has been shown that I-BASA technique very successfully tackles design constrained problems while preserving a low convergence rate and generating accurate results. Also, it was demonstrated that the proposed I-BASA algorithm retains the standard BA's characteristics as well as improves its accuracy. Besides, the proposed I-BASA employs Deb's rules instead of a penalty approach which has been used in [16]. Additionally, our proposed I-BASA uses the geometric scheme as in the case of the SA to further improve the quality of solutions and speeds up the global convergence. Conducted experimental analysis of accuracy and performance on the eight benchmark problems state that our I-BASA model is robust, most accurate and stable as well as it has a rapid convergence rate. By examining the stated facts, it can be concluded that the I-BASA method in the future can be applied for practical solving of large-scale real-world engineering problems.

REFERENCES

[1] A. H. Gandomi, X. S. Yang, and A. H. Alavi, "Mixed variable structural optimization using Firefly Algorithm," *Computers and Structures*, vol. 89, no. 23-24, pp. 2325–2336, December 2011. doi: https://doi.org/10.1016/j.compstruc.2011.08.002

[2] X.-S. Yang, "Review of meta-heuristics and generalised evolutionary walk algorithm," *International Journal of Bio-Inspired Computation*, vol. 3, no. 2, pp. 77–84,, 2011. doi: https://doi.org/10.1504/IJBIC.2011.039907

[3] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 35:1–35:33, July 2013. doi: https://doi.org/10.1145/2480741.2480752

[4] X.-S. Yang, "Free lunch or no free lunch: That is not just a question?" *International Journal on Artificial Intelligence Tools*, vol. 21, no. 3, pp. 5360–5366, 2012. doi: https://doi.org/10.1142/S0218213012400106

[5] ——, "Efficiency analysis of swarm intelligence and randomization techniques," *Journal of Computational and Theoretical Nanoscience*, vol. 9, no. 2, pp. 189–198, 2012. doi: https://doi.org/10.1166/jctn.2012.2012

[6] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," *Technical Report - TR06*, pp. 1–10, 2005.

[7] M. Tuba and R. Jovanovic, "Improved ant colony optimization algorithm with pheromone correction strategy for the traveling salesman problem," *International Journal of Computers, Communications & Control*, vol. 8, no. 3, pp. 477–485, June 2013. doi: https://doi.org/10.15837/ijccc.2013.3.7

[8] N. Bacanin and M. Tuba, "Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators," *Studies in Informatics and Control*, vol. 21, no. 2, pp. 137–146, June 2012. doi: https://doi.org/10.24846/v21i2y201203

[9] I. Brajevic and M. Tuba, "An upgraded artificial bee colony algorithm (abc) for constrained optimization problems," *Journal of Intelligent Manufacturing*, vol. 24, no. 4, pp. 729–740, August 2013. doi: https://doi.org/10.1007/s10845-011-0621-6

[10] I. Fister, J. Fister, X. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, no. 1, pp. 34–46, 2013. doi: https://doi.org/10.1016/j.swevo.2013.06.001

[11] N. Bacanin and M. Tuba, "Firefly Algorithm for Cardinality Constrained Mean-Variance Portfolio Optimization Problem with Entropy Diversity Constraint," *The Scientific World Journal*, vol. 2014, pp. 115–139, April 2014. doi: https://doi.org/10.1155/2014/721521

[12] M. Tuba, N. Bacanin, and A. Alihodzic, "Firefly algorithm for multi-objective RFID network planning problem," *Telecommunications Forum Telfor (TELFOR)*, pp. 95–98, September 2014. doi: https://doi.org/10.1109/TELFOR.2014.7034365

[13] A. H. Gandomi, X. S. Yang, and A. H. Alavi, "Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems," *Engineering with Computers*, vol. 29, no. 1, pp. 17–35, January 2013. doi: https://doi.org/10.1007/s00366-011-0241-y

[14] W. Long, X. Liang, Y. Huang, and Y. Chen, "An effective hybrid cuckoo search algorithm for constrained global optimization," *Neural Computing and Applications*, vol. 25, no. 3-4, pp. 911–926, September 2014. doi: https://doi.org/10.1007/s00521-014-1577-1

[15] X.-S. Yang, "A new metaheurisitic bat-inspired algorithm," *Studies in Computational Intelligence*, vol. 284, pp. 65–74, 2010. doi: https://doi.org/10.1007/978-3-642-12538-6%5F6

[16] A. H. Gandomi, Yang, A. H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing and Applications*, vol. 22, no. 6, pp. 1239–1255, May 2013. doi: https://doi.org/10.1007/s00521-012-1028-9

[17] A. Alihodzic and M. Tuba, "Improved bat algorithm applied to multilevel image thresholding," *The Scientific World Journal*, vol. 2014, no. Article ID 176718, p. 16, July 2014. doi: https://doi.org/10.1155/2014/176718

[18] M. Tuba, A. Alihodzic, and N. Bacanin, "Cuckoo Search and Bat Algorithm Applied to Training Feed-Forward Neural Networks," vol. 585, pp. 139–162, 2014. doi: https://doi.org/10.1007/978-3-319-13826-8%5F8

[19] A. Alihodzic and M. Tuba, "Improved hybridized bat algorithm for global numerical optimization," *16th IEEE International Conference on Computer Modelling and Simulation, UKSim-AMSS 2014*, pp. 57–62, March 2014. doi: https://doi.org/10.1109/UKSim.2014.97

[20] S. M. Nigdeli, G. Bekdaş, and X.-S. Yang, "Application of the Flower Pollination Algorithm in Structural Engineering," *Modeling and Optimization in Science and Technologies*, vol. 7, pp. 25–42, December 2015. doi: https://doi.org/10.1007/978-3-319-26245-1%5F2

[21] J. M. P. V. S. Kirkpatrick, C. D. Gelatt, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983. doi: https://doi.org/10.1126/science.220.4598.671

[22] H. Yu, H. Fang, P. Yao, and Y. Yuan, "A combined genetic algorithm/simulated annealing algorithm for large scale system energy integration," *Computers & Chemical Engineering*, vol. 24, no. 8, pp. 2023–2035, September 2000. doi: https://doi.org/10.1016/S0098-1354(00)00601-3

[23] X. shi He, W.-J. Ding, and X.-S. Yang, "Bat algorithm based on simulated annealing and Gaussian perturbations," *Neural Computing & Applications*, vol. 25, no. 2, pp. 459–468, September 2013. doi: https://doi.org/10.1007/s00521-013-1518-4