A modified bats echolocation-based algorithm for solving constrained optimisation problems

Nafrizuan Mat Yahya*

Faculty of Manufacturing Engineering,
Universiti Malaysia Pahang,
Pahang, Malaysia
Email: nafrizuanmy@ump.edu.my
*Corresponding author

M. Osman Tokhi

Department of Automatic Control and Systems Engineering,
University of Sheffield,
Sheffield, UK
Email: o.tokhi@sheffield.ac.uk

Abstract: A modified adaptive bats sonar algorithm (MABSA) is presented that utilises the concept of echolocation of a colony of bats to find prey. The proposed algorithm is applied to solve the constrained optimisation problems coupled with penalty function method as constraint handling technique. The performance of the algorithm is verified through rigorous tests with four constrained optimisation benchmark test functions. The acquired results show that the proposed algorithm performs better to find optimum solution in terms of accuracy and convergence speed. The statistical results of MABSA to solve all the test functions also has been compared with the results from several existing algorithms taken from literature on similar test functions. The comparative study has shown that MABSA outperforms other establish algorithms, and thus, it can be an efficient alternative method in the solving constrained optimisation problems.

Keywords: modified adaptive bats sonar algorithm; MABSA; bats echolocation; constrained optimisation problems.

Reference to this paper should be made as follows: Yahya, N.M. and Tokhi, M.O. (2017) 'A modified bats echolocation-based algorithm for solving constrained optimisation problems', *Int. J. Bio-Inspired Computation*, Vol. 10, No. 1, pp.12–23.

Biographical notes: Nafrizuan Mat Yahya received his PhD in Automatic Control and Systems Engineering at the University of Sheffield, UK. He is a Senior Lecturer at Faculty of Manufacturing Engineering, Universiti Malaysia Pahang, Malaysia. His research interests include intelligent control system optimisation, intelligent manufacturing automation and ergonomic for industrial application.

M. Osman Tokhi received his PhD in Electrical Engineering at the Heriot-Watt University, UK. He is a Reader at Department of Automatic Control and Systems Engineering, University of Sheffield, UK. He has extensive research and modelling experience in the area of control and systems including noise and vibration control, intelligent/adaptive control, high-performance and soft-computing modelling and control of dynamic systems and assistive robotics.

1 Introduction

Constrained optimisation problems normally come with lack of explicit mathematical formulation but have discrete definition domains, mixed of continuous and discrete design variables and also strong nonlinear objective functions with multiple complex constraints (Garg, 2014). However, due to computational drawbacks and the requirement of substantial gradient information traditional numerical programming strategies are incapable to solve constrained optimisation problems consistently (Sadollah et al., 2013). The alternative prospect to attain constrained optimisation problems is by metaheuristic methods (Hsieh, 2014).

Among most popular metaheuristic methods are swarm intelligence algorithms. These algorithms are inspired from the collective behaviour of swarms through complex interaction between individuals and its neighbourhood with nature (Hsieh, 2014). The most remarkable parts in any swarm intelligence algorithms are that the algorithm has advantages of memory, diverse multi-characters capability, rapid solution improvement mechanism and adaptable to internal and external changes (Garg, 2014). Particle swarm optimisation (PSO), artificial bee colony (ABC), ant colony

optimisation (ACO) and bat algorithm (BA) for instance are some examples of swarm intelligence algorithms that have already captured attention of researchers.

This paper introduces a new swarm intelligence algorithm, the so called modified adaptive bats sonar algorithm (MABSA) which is an improved version of the original adaptive bats sonar algorithm (ABSA) by Yahya et al. (2016). The algorithm is inspired from bat sonar used in bats echolocation to find prey. In this paper, the MABSA is proposed and applied to several constrained optimisation benchmark test functions.

The remainder of this paper is organised as follows. Section 2 details the constrained optimisation problems, the penalty function method as a constraint handling technique and several previous researches using swarm intelligence algorithms to solve constrained optimisation problems. Section 3 describes the bats echolocation concepts and the proposed algorithm. Section 4 discusses computer simulation results and finally, conclusions are drawn in Section 5.

2 Constrained optimisation problems

2.1 Background

A constrained optimisation comprises an objective function together with some equality and inequality constraints subject to lower bound and upper bound of variables as:

$$\begin{aligned} &\text{Optimise } F(x), \quad x = (x_1, x_2, ..., x_N) \\ &\text{subject to} \\ &g_j(x) \geq 0, \quad j = 1, 2, ..., J \\ &h_k(x) = 0, \quad k = 1, 2, ..., K \\ &\text{where} \\ &x_i^{(L)} \leq x_i \leq x_i^{(u)}, \quad i = 1, 2, ..., n \end{aligned} \tag{1}$$

Here, $g_j(x)$ represents inequality constraint functions with J inequality constraints. $h_k(x)$ represents equality constraints functions with K equality constraints. $x_i^{(L)}$ represents the lower bounds and $x_i^{(U)}$ the upper bounds of variable x_i with n variables, respectively.

2.2 A penalty function

In the penalty function, the original objective function $F(x)$ is replaced by a substituted function $C(x)$ which considers the original objective function $F(x)$ with added penalty function $P(x)$ that introduces a tendency term to penalise constraint violations produced by x. Therefore, considering the constrained optimisation problem defined previously, the substituted function is defined as follows:

$$\begin{aligned} &C(x) = F(x) + P(x) \\ &\text{where} \\ &P(x) = \mu \cdot \sum_{j=1}^{J} g_j^2(x) + v \cdot \sum_{k=1}^{K} h_k^2(x) \end{aligned} \tag{2}$$

where μ and v represent penalty coefficients which weigh the relative importance of each $g_j(x)$ (inequality constraint) and $h_k(x)$ (equality constraint) respectively. In this work, values of μ and v are problem-dependant.

2.3 Previous approaches of swarm intelligence algorithms

Various research works have been reported over the past two decades on dealing with constrained optimisation problems by using swarm intelligence algorithms.

The PSO has been the most favourable technique among them. Parsopoulos and Vrahatis (2005) have proposed a variant of PSO scheme, a unified particle swarm optimisation (UPSO) method with a penalty function approach. The proposed algorithm has abilities to explore and exploit the search process without needing extra requirements of function evaluations and also preserves feasibility of the encountered solutions. Cagnina et al. (2008) investigated simple constrained particle swarm optimiser (SiC-PSO) coupled with a constrained-handling technique. The algorithm is faster, more reliable and efficient after combining local best (*lbest*) and global best (*gbest*) models to update the velocity as well as adding *gbest* to the best position of the particles and to its neighbourhood.

Another popular swarm intelligence algorithm used is the ABC algorithm. For instance, Garg (2014) introduced a penalty function guided ABC algorithm to solve several structural engineering design problems. Earlier, Karaboga and Basturk (2007) adopted Deb's rule for the selection of mechanism to deal with constraints of the ABC algorithm to solve a set of constrained numerical optimisation problems. The rest of the techniques shall be categorised under swarm intelligence algorithms such as, BA (Yang and Hossein, 2012) which is based on the level and loudness of the pulse emitted in bats echolocation and a bacterial gene recombination algorithm (BGRA) that was inspired from virus resistance process in real bacteria (Hsieh, 2014).

Up to this time, there is no evidence of a developed algorithm based on the principles of bat sonar used in echolocation to tackle constrained optimisation problems. So, it is believed that the algorithm proposed in this paper will be the first as well as bring in the well-known advantages of swarm intelligence characteristics to solve specific problems.

3 The proposed algorithm

3.1 Bats echolocation in brief

Bats habitually live in large colonies around 700 to 1,000 individuals under sharing roosts (Altringham et al., 1996). There are two sets of acoustic communication used by a colony of bats. These are echolocation calls for foraging and positioning purposes and social calls for socialising or communicating between bats (Altringham et al., 1996). A colony of bats is able to create worthy communication and

sharing information about roost site or foraging area among one another.

Echolocation is the ability of bat to produce sound with echoes beyond the frequency range of human hearing and use for general orientation and finding prey (Altringham et al., 1996). With echolocation, a bat emits ultrasonic pulses through mouth or nose in short bursts. The sound reflects back as echoes bump into an object in the bat's path. The bat is able to recognise the object and its distance by calculating the time of reflection of the modulated echoes (Altringham et al., 1996).

The echolocation process of bats that leads to catching the prey comprises of three phase; search phase, approach phase and terminal phase (Altringham et al., 1996). When the bats begin to chase for prey in the search phase, they emit the pulse at low rate at around a frequency of 10 Hz (Altringham et al., 1996). When the bat senses and gets closer to the prey in the approach phase, the pulses have to get shorter to avoid overlap (Altringham et al., 1996). This is because the time between the pulse and echo is dwindled (Altringham et al., 1996). Now, pulse emission rate gets progressively increased up to 200 per second as the bat keeps updating the location of the prey (Altringham et al., 1996). The pulse emission rate increases because the bats need to emit more signals to trace the prey precisely as the angular position of the prey changes more swiftly due to the closer distance between a bat and the object. In the terminal phase, the frequency of emitted pulses upswings to more than 200 Hz and the pulse emission rate becomes faster at only fraction of millisecond long just before the prey is seized (Altringham et al., 1996).

According to Altringham et al. (1996), a colony of bats also includes the concept of reciprocal altruism of food sharing during echolocation process. This social behaviour of bats group is based on animals returning favours to their mutual benefit. The reciprocal altruism behaviour growth in individual survivorship such that the fitness of the recipient is elevated to a non-recipient (Altringham et al., 1996).

3.2 Modified adaptive bats sonar algorithm

ABSA was introduced by Yahya et al. (2016) to solve unconstrained single objective optimisation problems. But, to deal with constrained single objective optimisation problems, a crucial problem on how to incorporate the inequality constraints as well as equality constraints with the objective function must be tackled appropriately. ABSA does not function well on this kind of problems as it is based on a direct approach. A direct approach is often difficult to find the solution in feasible regions enclosed by constraints.

A new algorithm named the MABSA is proposed here by redefining as well as reformulating some elements in ABSA to compensate for this problem. The MABSA will be able to generate a potential solution that satisfy all constraints. The purpose of MABSA is to solve constrained optimisation problems.

The MABSA is formulated after modifying three searching procedures of the original ABSA and adding a new component to it. The three procedures are the ways to setting up the *beam length* (L), determining *starting angle* (θ_m) and *angle between beams* (θ_i) and also calculating *end point position* (pos_i). On the other hand, the bounce back strategy is a new component that has been included in the MABSA, which was not considered in ABSA formerly. This section will elaborate solely of these three elements. The other components of MABSA will not be further discussed here as they are similar to the ABSA as presented in Yahya et al. (2016).

In the MABSA, the new L is setup as:

$$L = Rand \times \left(\frac{SS_{size}}{10\% \times Bats} \right) \tag{3}$$

where the *solution range* (SS_{Size}) is the value between the *upper search space* (SS_{Max}) limit and the *lower search space* (SS_{Min}) limit. Every *dimension* (*Dim*) has its specific or known as *Dim* constraints. The solution range is divided into micron scale, such as 10% of the overall population of bats in the search space. The percentage is marked as possible search space size of each bat to emit sound without colliding with one another. The random value of L is offered to make real variation of beam lengths of each *number of beams* (*NBeam*) at every *Dim* (but stay within the *Dim* constraints) at every iteration. This fixation pushes every bat at each dimension to search for larger perimeter each time with the opportunity to diversify the search tactic during iterations and thus may find the global best solution that may be near to them.

Each *NBeam* with L is emitted from specific angle location. In the ABSA, the θ_m and θ_i are determined randomly in every iteration. So all bats will emit the *NBeam* from a set of similar angle location in each iteration. To add another randomisation character inside MABSA, θ_m and θ_i will be determined in random and separately for every bat at every iteration. So at each iteration, every bat will emit the *NBeam* from a different set of angle location. Therefore, this randomisation will also add on to diversify the searching process in MABSA.

In the MABSA, the way to calculate the pos_i is redefined. The pos_i for each transmitted beam in MABSA is calculated as:

$$pos_i = \alpha \times pos_{SP} + \beta \times L\left(\cos\left[\theta_m + (i-1)\theta\right]\right)^{\omega}$$

where

$i = 1, ..., NBeam;$ (4)

$NBeam$ is *number of beams*

pos_{SP} is beam's *starting position*

In the above equation, there are two random variables and one constant. The first random variable is called *position adaptability factor* (α). The value for α is chosen randomly within the range between 0 and 1. This factor is included to make sure that every bat is able to adapt to the new pos_{SP} faster as derived from the previous pos_{SP}, pos_{LB}, pos_{RB} and pos_{GB}. This factor has the same characteristic as random walk method. The second random variable is *collision*

avoidance factor (β). The value for β is also chosen randomly within the range between 0 and 1. The factor is essential to avoid the beams from overlapping or incidentally colliding with other bats' beam as every bat produces a number of beams from new pos_{SP} simultaneously.

The only constant in this equation is *beam-tuning constant* (ω) which is equal to 2. This constant can also be considered as acceleration constant. The function of this constant is to strengthen β so that ω will divert the angle of transmitted beam to a new angle in the designated search space. The value 2 is selected because it will give a good balance. If a very high value is selected, it will destroy the influence of the beam angle such that the orientation of new bat position will be catastrophic. A smaller value, on the other hand, will not make any significant change to the angle of transmitted beam.

The MABSA is also equipped with bounce back strategy. This will confirm that every pos_i achieved by each bat during the iterations is worth considering as possible optimum pos_{GB} for the algorithm. When each beam is transmitted from every bat, it will be verified to ensure that the pos_i of the transmitted beam does not fall beyond SS_{Max} or below SS_{Min}. If the pos_i reaches outside SS_{Size}, the transmitted beam will be diverted automatically to new location inside the labelled SS_{Size} using one of the following equations:

$$pos_i = SS_{Max} - \tau, \quad i = 1, ..., N \tag{5a}$$

$$pos_i = SS_{Max} + \tau, \quad i = 1, ..., N \tag{5b}$$

These equations contain *bounce back repositioning factor* (τ) where the value is $0 < \tau < 1$. This factor is to help the bats to relocate a beam transmission to a new beams' end point from the maximum or minimum search space. This factor will avoid overwriting other bats' beam end points. The *bounce back repositioning factor* is the fastest contingency action of bats to swing to newly transmitted beam's end point after hitting the designated search space boundaries. This strategy helps to significantly reduce the time spent in considering the previous factors (which are: *position adaptability factor*, *collision avoidance factor* or *beam-tuning constant*) as normal bats do. Algorithm 1 represents the pseudo code of MABSA.

Algorithm 1:

Objective function $F(x)$, $x = (x_1, \ldots, x_d)^T$
Initialise: Bats, *MaxIter*, *Dim*, SS_{Size}, $NBeam_{MAX}$ and $NBeam_{MIN}$
for $n \leftarrow 1$ ***to*** *Bats* ***do***
    for $d \leftarrow 1$ ***to*** *Dim* ***do***
        Generate random pos_{SP}
        Evaluate F_{SP} *value for* $F(pos_{SP})$
    end for
end for
Assign the most optimum value as F_{GB} *and its position as* pos_{GB}
while $t \leq MaxIter$ ***do***
    Define NBeam to transmit by using BNI
    for $n \leftarrow 1$ ***to*** *Bats* ***do***
        for $N \leftarrow 1$ ***to*** *NBeam* ***do***
            for $d \leftarrow 1$ ***to*** *Dim* ***do***
                Set L and limit μ ***(equation 3)***
            end for
        end for
        Generate random θ_m *and* θ
        Transmit NBeam starting from pos_{SP}
        for $N \leftarrow 1$ ***to*** *NBeam* ***do***
            for $d \leftarrow 1$ ***to*** *Dim* ***do***
                Determine pos_i *for each transmitted beam* ***(Equation 4)***
                Verify pos_i *for each transmitted beam within* SS_{Size}
                if $pos_i \leq SS_{Max}$ ***then***
                    Update pos_i ***(Equation 5a)***
                end if
                if $pos_i \leq SS_{Min}$ ***then***
                    Update pos_i ***(Equation 5b)***
                end if
            end for
            Evaluate F_i *value for* $F(pos_i)$
            Assign the optimum value of F_i *as* F_{LB} *and its position as* pos_{LB}
            if $F_{LB} \leq F_{SP}$ ***then***
                Assign F_{LB} *as* F_{RB} *and* pos_{LB} *as* pos_{RB}
            else
                Assign F_{SP} *as* F_{RB} *and* pos_{SP} *as* pos_{RB}
            end if
        end for
    end for
    Select the optimum value among F_{RB} *as current* F_{GB} *and its* pos_{RB} *as current* pos_{GB}
    if *current* $F_{GB} \leq$ *previous* F_{GB} ***then***
        Update current F_{GB} *as new* F_{GB} *and current* pos_{GB} *as new* pos_{GB}
    else
        Retain previous F_{GB} *and* pos_{GB}
    end if
    for $n \leftarrow 1$ ***to*** *Bats* ***do***
        Determine new pos_{SP}
        Evaluate new F_{SP} *value for* $F(pos_{SP})$
    end for
end while
Declare F_{GB} *as optimum fitness evaluated and* pos_{GB} *as its optimum value(s)*

4 Computer simulation and discussion

In order to show the superiority of MABSA in solving constrained optimisation problems, four constrained benchmark test functions from *CEC 2006* by Liang et al. (2006) were examined and tested. The results are compared against other established algorithms based on results recorded in the specific literature (no re-simulation exercises using the established algorithms were conducted).

The algorithms are; changing range genetic algorithm (CRGA) (Amirjanov, 2006), self-adaptive penalty function (SAPF) (Tessema and Yen, 2006), cultured differential evolution (CULDE) (Becerra and Coello, 2006), simple multimembered evolution strategy (SMES) (Mezura-Montes and Coello, 2005), adaptive segregational constraint handling evolutionary algorithm (ASCHEA) (Hamida and Schoenauer, 2002), particle swarm optimisation with differential evolution (PSO-DE) (Liu et al., 2007), stochastic ranking (SR) (Runarsson and Yao, 2000), differential evolution with level comparison (DELC) (Wang and Li, 2010), differential evolution with dynamic stochastic selection (DEDS) (Zhang et al., 2008), hybrid evolutionary algorithm and adaptive constraint handling technique (HEA-ACT) (Wang et al., 2009), improved stochastic ranking (ISR) (Runarsson and Yao, 2005), α constrained with nonlinear simplex method with mutation (α simplex) (Takahama and Sakai, 2005), Nelder-Mead simplex method and particle swarm optimisation (NM-PSO) (Zahara and Kao, 2009), artificial bee colony 2 (ABC2) (Karaboga and Basturk, 2007) and mine blast algorithm (MBA) (Sadollah et al., 2013).

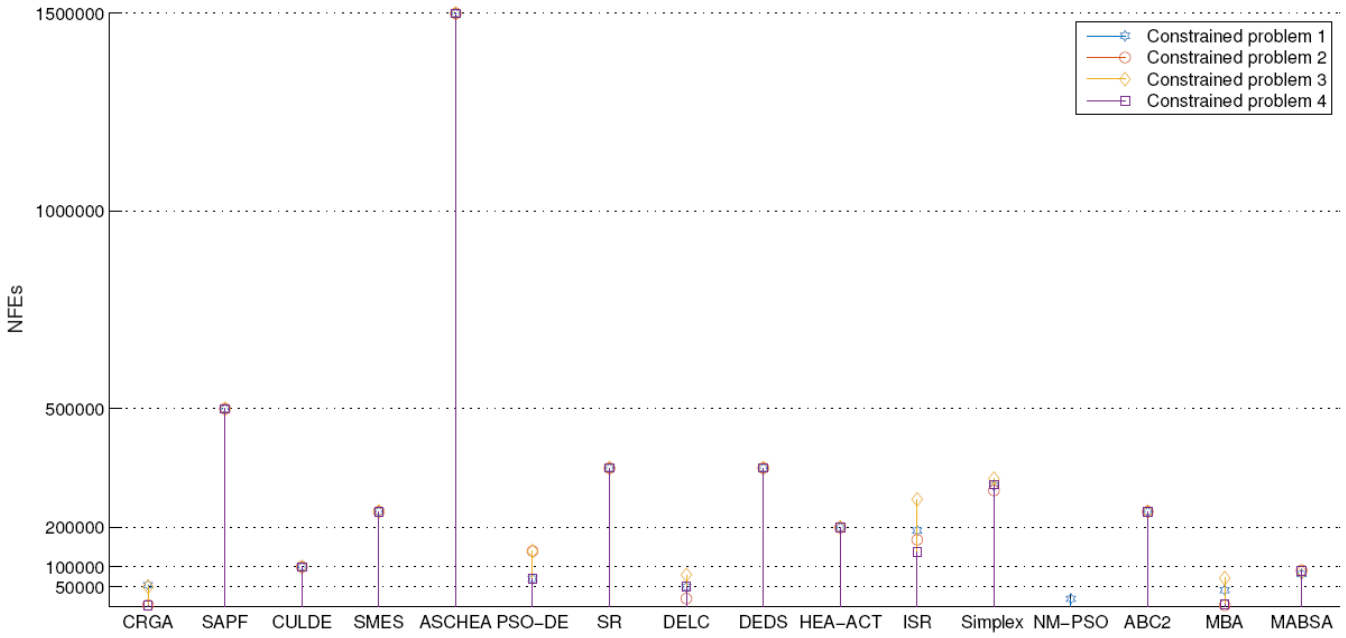
The quality of obtained optimisation results is compared in terms of statistical results (better *best*, *mean*, *median* and *worst* solution found), the robustness of the MABSA (the *standard deviation* values) and the *number of function evaluations* (*NFEs*).

The results of the *best* solution obtained from MABSA for constrained optimisation benchmark test functions are summarised in Table 1. The MABSA was capable of finding the *best* solution (minimum value) better than the optimum value suggested from *CEC 2006* for all constrained benchmark test functions. The time to converge to the *best* solution was recorded under 22 *seconds* for all four test functions, and this shows that the algorithm is able to reach to the *best* solution faster than ordinary methods. So it is worth mentioning that MABSA is very effective and efficient to solve constrained optimisation problems.

Table 1 Results of the *best* solution obtained from MABSA for constrained benchmark test functions

Items	*Test function 1*	*Test function 2*	*Test function 3*	*Test function 4*
Run no.	23	2	21	5
No. of bats	1,000	700	1,000	700
NFEs	3,200	2,100	3,600	2,000
Time to converge (seconds)	9.7244	20.9769	14.2320	0.3656
Iteration to converge	31	89	34	3
$F(x)$	–30,994.6595	–7,091.3568	662.4557	0.7500
Optimum value of $F(x)$	–30,665.5390	–6,961.8139	680.6301	0.7500

Figure 1 Comparison of *NFEs* used by considered algorithms for all constrained benchmark problems (see online version for colours)



In terms of *NFEs*, MABSA showed good potential to be popular algorithm in future as it converges fast to the optimum solution. For instance, by considering the *NFEs* from the best solution obtained in all constrained benchmark test functions tested, MABSA started to settle down to the optimum solution after approximately 2,000 to 4,000 *NFEs*.

Figure 1 compares the average *NFEs* used by all algorithms to solve four constrained benchmark test functions. When comparing the average *NFEs* used by MABSA on all constrained benchmark test functions with other established algorithms, the value between 70,000 and 100,000 is reasonable and more productive. The small value of *NFEs* will force the algorithm to settle down earlier as possible without a chance to explore more but may end up with the algorithm trapped in local optimum such as in CRGA, NM-PSO or DELC. On the other hand, if too many *NFEs* are used such as in ASCHEA or even SAPF, the algorithm may waste the time to find good solution but the solution which was already encountered earlier than the last set of *NFEs* is examined.

Figure 2 Bar plot of statistical results obtained using different algorithms for test function 1

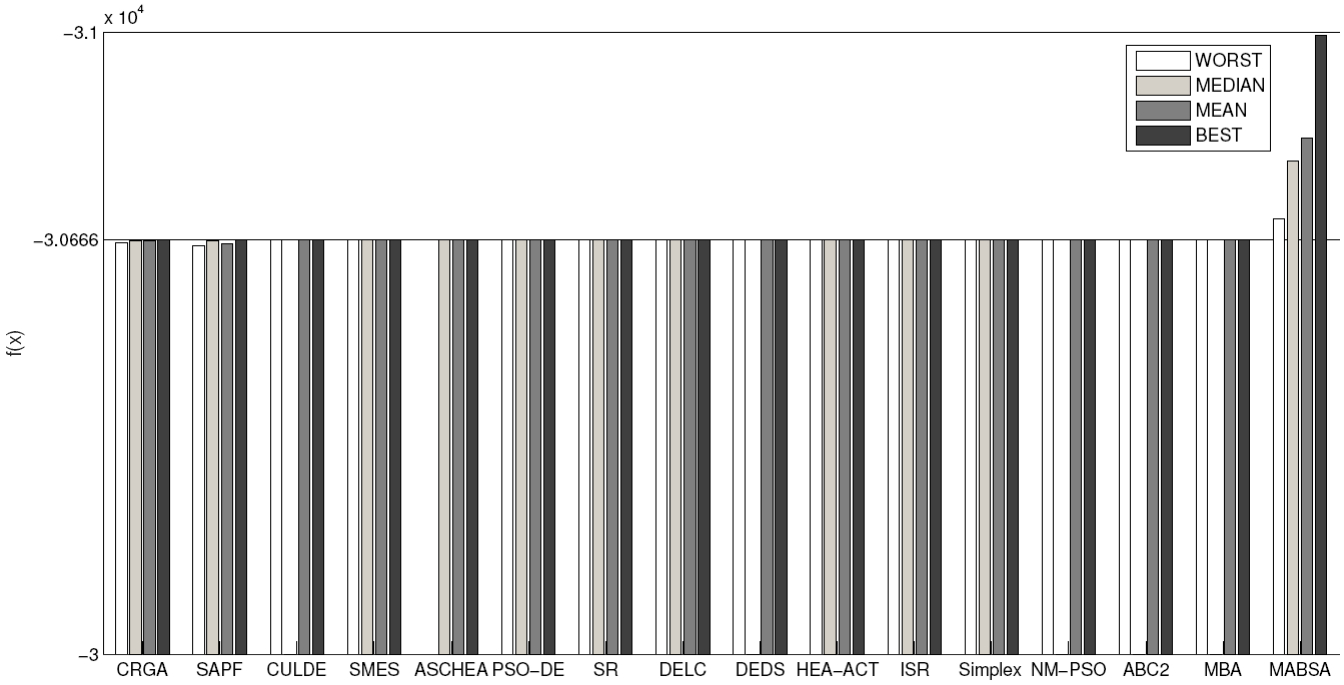


Table 2 Comparison of statistical results obtained using different algorithms for test function 1

Method	*Worst*	*Median*	*Mean*	*Best*	*Std. dev.*	*NFEs*
CRGA	–30,660.3130	–30,665.2520	–30,664.3980	–30,665.5200	1.6000	54,400
SAPF	–30,656.4710	–30,663.9210	–30,659.2210	–30,665.4010	2.0430	500,000
CULDE	–30,665.5387	n/a	–30,665.5387	–30,665.5387	0.0000	100,100
SMES	–30,665.5390	–30,665.5390	–30,665.5390	–30,665.5390	0.0000	240,000
ASCHEA	n/a	–30,665.5000	–30,665.5000	–30,665.5000	n/a	1500,000
PSO-DE	–30,665.5387	–30,665.5387	–30,665.5387	–30,665.5387	$8.3000e^{-10}$	70,100
SR	–30,665.5390	–30,665.5390	–30,665.5390	–30,665.5390	$2.0000e^{-05}$	350,000
DELC	–30,665.5390	–30,665.5390	–30,665.5390	–30,665.5390	$1.0000e^{-11}$	50,000
DEDS	–30,665.5390	n/a	–30,665.5390	–30,665.5390	$2.7000e^{-11}$	350,000
HEA-ACT	–30,665.5390	–30,665.5390	–30,665.5390	–30,665.5390	$7.4000e^{-12}$	200,000
ISR	–30,665.5390	–30,665.5390	–30,665.5390	–30,665.5390	$1.1000e^{-11}$	192,000
α-simplex	–30,665.5387	–30,665.5387	–30,665.5387	–30,665.5387	$4.2000e^{-11}$	305,343
NM-PSO	–30,665.5390	n/a	–30,665.5390	–30,665.5390	$1.4000e^{-05}$	19,658
ABC2	–30,665.5390	n/a	–30,665.5390	–30,665.5390	0.0000	240,000
MBA	–30,665.3300	n/a	–30,665.5182	–30,665.5386	$5.0800e^{-02}$	41,750
MABSA	–30,700.2654	–30,793.4331	–30,829.8768	–30,994.6595	110.3421	82,090

Note: ‘n/a’ means not available.

Figure 3 Bar plot of statistical results obtained using different algorithms for test function 2

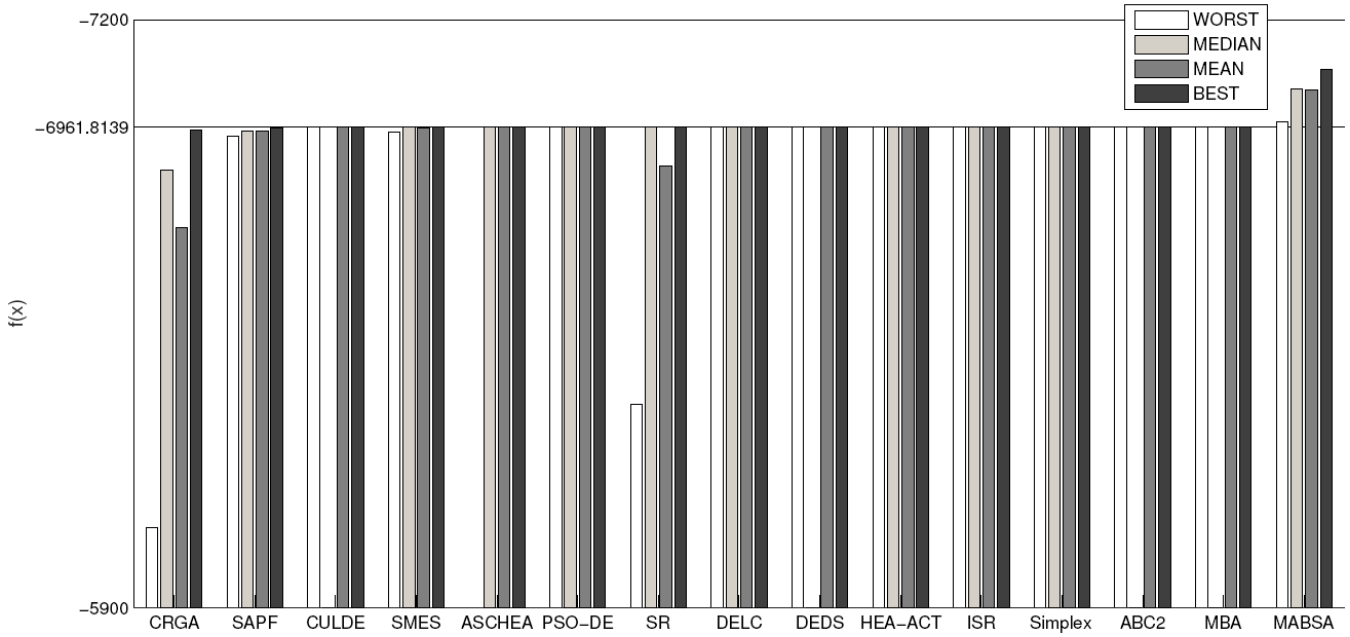


Table 3 Comparison of statistical results obtained using different algorithms for test function 2

Method	*Worst*	*Median*	*Mean*	*Best*	*Std. dev.*	*NFEs*
CRGA	–6,077.1230	–6,867.4610	–6,740.2880	–6,956.2510	270.0000	3,700
SAPF	–6,943.3040	–6,953.8230	–6,953.0610	–6,961.0460	5.8760	500,000
CULDE	–6,961.8139	n/a	–6,961.8139	–6,961.8139	0.0000	100,100
SMES	–6,952.4820	–6,961.8140	–6,961.2840	–6,961.8140	1.8500	240,000
ASCHEA	n/a	–6,961.8100	–6,961.8100	–6,961.8100	n/a	1,500,000
PSO-DE	–6,961.8139	–6,961.8139	–6,961.8139	–6,961.8139	$2.3000e^{-09}$	140,100
SR	–6,350.2620	–6,961.8140	–6,875.9400	–6,961.8140	160.0000	350,000
DELC	–6,961.8140	–6,961.8140	–6,961.8140	–6,961.8140	$7.3000e^{-10}$	20,000
DEDS	–6,961.8140	n/a	–6,961.8140	–6,961.8140	0.0000	350,000
HEA-ACT	–6,961.8140	–6,961.8140	–6,961.8140	–6,961.8140	$4.6000e^{-12}$	200,000
ISR	–6,961.8140	–6,961.8140	–6,961.8140	–6,961.8140	$1.9000e^{-12}$	168,800
α-simplex	–6,961.8139	–6,961.8139	–6,961.8139	–6,961.8139	$1.3000e^{-10}$	293,367
ABC2	–6,961.8050	n/a	–6,961.8130	–6,961.8140	$2.0000e^{-03}$	240,000
MBA	–6,961.8139	n/a	–6,961.8139	–6,961.8139	0.0000	2,835
MABSA	–6,973.2374	–7,047.2779	–7,043.7395	–7,091.3568	34.227384	91,530

Note: n/a' means not available.

4.1 Test function 1

For test function 1, there are 15 different algorithms from literature that have been chosen to compare with MABSA. These include CRGA, SAPF, CULDE, SMES, ASCHEA, PSO-DE, SR, DELC, DEDS, HEA-ACT, ISR, α simplex, NM-PSO, ABC2 and MBA. Table 2 shows a comparison between MABSA and other algorithms in terms of statistical results obtained for solving test function 1.

Overall, MABSA lead other algorithms to all criteria (*worst*, *median*, *mean* and *best* value) which demonstrate the quality of algorithm to achieve the optimum solution for test function 1. This statement was strengthened by the bar plot pictured in Figure 2 where MABSA was significantly better to achieve the optimum solution as compared to optimum value compiled in *CEC 2006* or other algorithms. Indeed, the *worst* result from the MABSA; –30,700.2654 is still a better result than the optimum value or the *best* result from other established algorithms. However, MABSA is less robust to solve the problem as shown by the higher value of *standard deviation* when compared to other listed algorithms.

4.2 Test function 2

In test function 2, the performance of MABSA was also compared with the 14 established algorithms. The algorithms are CRGA, SAPF, CULDE, SMES, ASCHEA,

PSO-DE, SR, DELC, DEDS, HEA-ACT, ISR, α simplex, ABC2 and MBA. The statistical results obtained by all algorithms including MABSA are shown in Table 3 while the *worst*, *median*, *mean* and *best* results for each considered algorithm is shown in Figure 3.

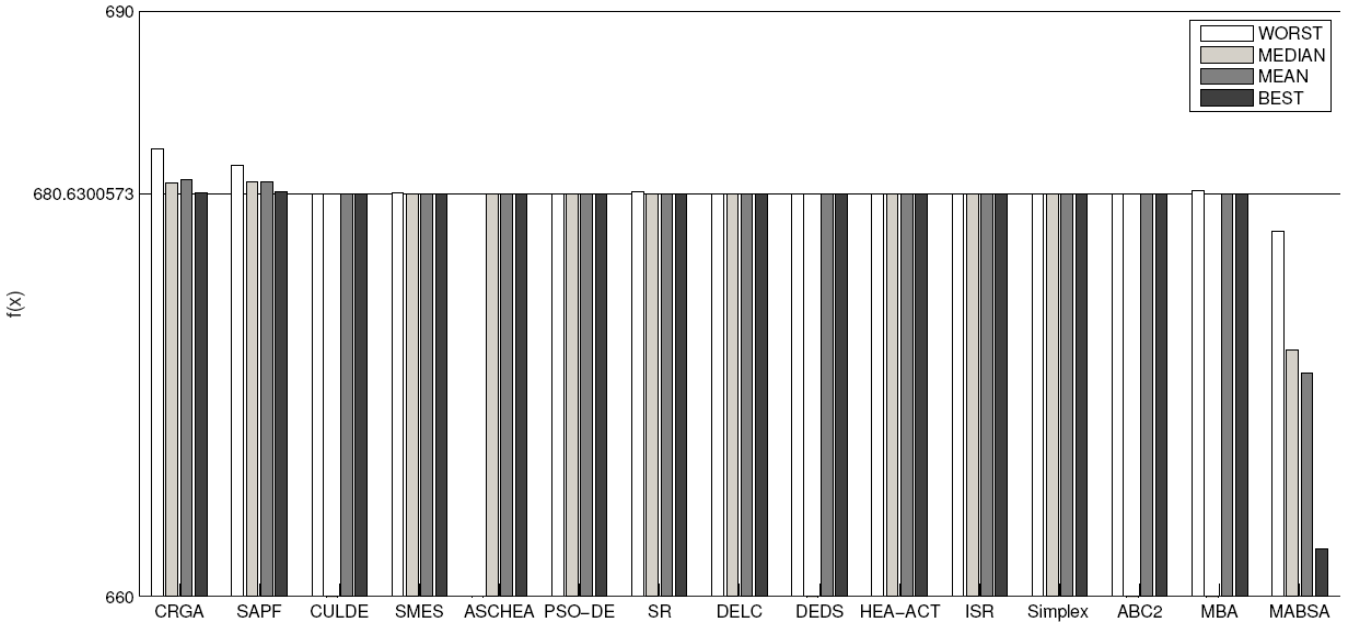
The outstanding performance of MABSA to solve the test function 2 can be seen in both table and bar plot. The fitness function value achieved by MABSA for every statistical criterion was the optimum as compared to other 14 established algorithms as well as the optimum value from *CEC 2006*. In addition to that, the MABSA method was the only algorithm passing the –7,000.0000 value in *median*, *mean* and *best* which was not achievable by other algorithms. Nevertheless, the higher *standard deviation* value achieved by MABSA shows that the algorithm was less robust to solve the test function 2 compared to other algorithms. However, the level of robustness for MABSA to solve this problem was better than the previous problem.

Table 4 Comparison of statistical results obtained using different algorithms for test function 3

Method	*Worst*	*Median*	*Mean*	*Best*	*Std. dev.*	*NFEs*
CRGA	682.9650	681.2040	681.3470	680.7260	$5.7000e^{-01}$	50,000
SAPF	682.0810	681.2350	681.2460	680.7730	$3.2200e^{-01}$	500,000
CULDE	680.6301	n/a	680.6301	680.6301	0.0000	100,100
SMES	680.7190	680.6420	680.6430	680.6320	$1.5500e^{-02}$	240,000
ASCHEA	n/a	680.6350	680.6410	680.6300	n/a	1,500,000
PSO-DE	680.6301	680.6301	680.6301	680.6301	$4.6000e^{-13}$	140,100
SR	680.7630	680.6410	680.6560	680.6300	$3.4000e^{-02}$	350,000
DELC	680.6300	680.6300	680.6300	680.6300	$3.2000e^{-12}$	80,000
DEDS	680.6300	n/a	680.6300	680.6300	$2.5000e^{-13}$	350,000
HEA-ACT	680.6300	680.6300	680.6300	680.6300	$5.8000e^{-13}$	200,000
ISR	680.6300	680.6300	680.6300	680.6300	$3.2000e^{-13}$	271,200
α simplex	680.6301	680.6301	680.6301	680.6301	$2.9000e^{-10}$	323,427
ABC2	680.6530	n/a	680.6400	680.6340	$4.0000e^{-03}$	240,000
MBA	680.7882	n/a	680.6620	680.6322	$3.3000e^{-02}$	71,750
MABSA	678.7398	672.6514	671.4536	662.4557	4.6726	88,303

Note: 'n/a' means not available.

Figure 4 Bar plot of statistical results obtained using different algorithms for test function 3



4.3 Test function 3

In test function 3, the statistical results between MABSA and 14 other algorithms that are taken from the literature are compared. The algorithms are CRGA, SAPF, CULDE, MES, ASCHEA, PSO-DE, SR, DELC, DEDS, HEA-ACT, ISR, α simplex, ABC2 and MBA. A comparison of statistical results obtained by all algorithms is provided in Table 4. Figure 4 shows a bar plot of worst, *median*, *mean* and *best* solution of all algorithms with a benchmark of the optimum value from *CEC 2006*.

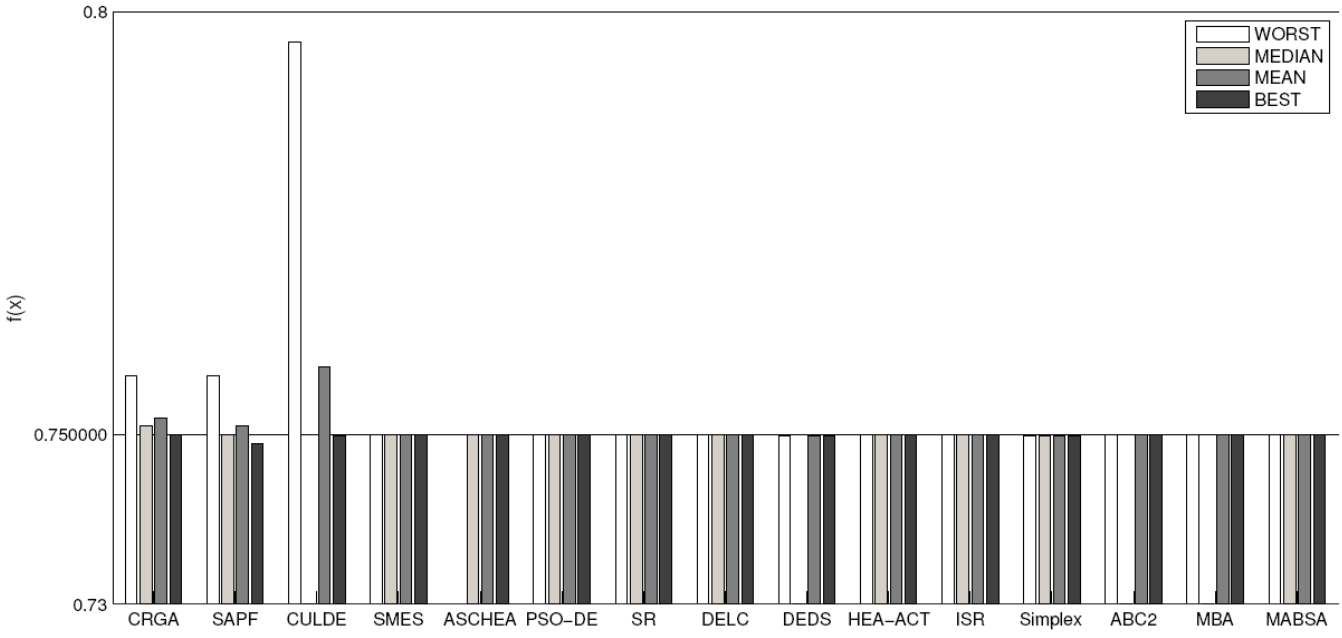
The performance of MABSA was exceptional when compared to other established algorithms to find the optimum fitness function value for test function 3. The MABSA was the sole algorithm that recorded the minimum solution under 680.0000 for all statistical criteria with the *best* solution 662.4557 which was far better than the optimum value from *CEC 2006*. For this test function 3, MABSA was well thought-out to be more robust when compared to the performances in case of test function 1 or test function 2. Despite the fact that the *standard deviation* for MABSA was still larger than 1.0000, the value was acceptable to compromise with the range of *worst*, *median*, *mean* and *best* solution found which was better amongst considered algorithms.

Table 5 Comparison of statistical results obtained using different algorithms for test function 4

Method	*Worst*	*Median*	*Mean*	*Best*	*Std. dev.*	*NFEs*
CRGA	0.7570	0.7510	0.7520	0.7500	$2.5000e^{-03}$	3,000
SAPF	0.7570	0.7500	0.7510	0.7490	$2.0000e^{-03}$	500,000
CULDE	0.7965	n/a	0.7580	0.7499	$1.7138e^{-02}$	100,100
SMES	0.7500	0.7500	0.7500	0.7500	$1.5200e^{-04}$	240,000
ASCHEA	n/a	0.7500	0.7500	0.7500	n/a	1,500,000
PSO-DE	0.7500	0.7499	0.7499	0.7499	$2.5000e^{-07}$	70,100
SR	0.7500	0.7500	0.7500	0.7500	$8.0000e^{-05}$	350,000
DELC	0.7500	0.7500	0.7500	0.7500	0.0000	50,000
DEDS	0.7499	n/a	0.7499	0.7499	0.0000	350,000
HEA-ACT	0.7500	0.7500	0.7500	0.7500	$3.4000e^{-16}$	200,000
ISR	0.7500	0.7500	0.7500	0.7500	$1.1000e^{-16}$	137,200
α simplex	0.7499	0.7499	0.7499	0.7499	$4.9000e^{-16}$	308,125
ABC2	0.7500	n/a	0.7500	0.7500	0.0000	240,000
MBA	0.7500	n/a	0.7500	0.7500	$3.2900e^{-06}$	6,405
MABSA	0.7500	0.7500	0.7500	0.7500	0.0000	89,724

Note: 'n/a' means not available.

Figure 5 Bar plot of statistical results obtained using different algorithms for test function 4



4.4 Test function 4

A set of 14 established algorithms is compared with MABSA in terms of the statistical results obtained for test function 4. These included CRGA, SAPF, CULDE, SMES, ASCHEA, PSO-DE, SR, DELC, DEDS, HEA-ACT, ISR, α simplex, ABC2 and MBA. Table 5 shows the comparison results, while the bar plot of *worst*, *median*, *mean* and *best* solution acquired from all the algorithms with the optimum value from *CEC 2006* is shown in Figure 5.

For test function 4, MABSA successfully achieved results which have the same performance or better than other considered algorithms for all criteria. Indeed, the *median*, *mean* and *best* solution values achieved by MABSA method managed to achieve better than the *CEC 2006* benchmark value; 0.7500. The MABSA recorded 0.7500, 0.7500 and 0.7500 for median, mean and best criteria respectively. According to the results, MABSA is also considered to be more robust to solve the test function 4 as its *standard deviation* value recorded was 0.000000. The robustness ability of MABSA to solve the problem was at par with other considered algorithms and better than CGRA, SAPF, CULDE and SMES.

Table 6 Rank of algorithms for test functions

Algorithm	*MAE*	*Ranking*
MABSA	–66.1095	1
DEDS	$-6.9250e^{-5}$	2
DELC	$-4.4250e^{-5}$	3
HEA-ACT	$-4.4250e^{-5}$	4
ISR	$-4.4250e^{-5}$	4
α simplex	$5.8500e^{-5}$	6
PSO-DE	$7.4750e^{-5}$	7
ABC2	$1.2058e^{-3}$	8
CULDE	$2.0820e^{-3}$	9
MBA	$5.737e^{-3}$	10
ASCHEA	0.0135	11
SMES	0.1357	12
SAPF	3.9220	13
SR	21.4750	14
CRGA	55.8464	15

4.5 Overall comparison of all considered algorithms

The mean absolute errors (*MAEs*) of all algorithms are computed to rank all considered algorithms. *MAE* is a statistical criterion that indicates how far the results are from the actual values as:

$$MAE = \frac{\sum_{z}^{i=1} |m_i - h_i|}{z} \quad (6)$$

where

m_i mean of optimum achieved results

h_i global optimum value

z number of test functions.

All considered algorithms for test functions are ranked in Table 6 based on their corresponding MAE's. The table shows that MABSA is at the highest ranking from 15 considered algorithms.

5 Conclusions

A MABSA has been proposed for solving constrained optimisation problems. The MABSA has been formulated as an improved version of ABSA developed by Yahya et al. (2016). In addition to redefining ABSA parameters, a new strategy, namely the bounce back strategy as a mechanism to control the transmitted beam to fall only within the designated search space, has been incorporated into MABSA.

The MABSA has achieved competitive results on four constrained optimisation benchmark test functions adopted from *CEC 2006* at a relatively better optimum solution value with a low computational cost. From the comparative study, MABSA has shown its ability to handle various constrained optimisation, and its outstanding performance is much better, in terms of statistical metrics, than the established set of algorithms selected from the literature.

The future works will be focus on the application of MABSA to real optimisation problems.

References

Altringham, J.D., Hammond, L. and McOwat, T. (1996) *Bats: Biology and Behaviour*, The Oxford University Press, Oxford, UK.

Amirjanov, A. (2006) 'The development of a changing range genetic algorithm', *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, No. 19, pp.2495–2508.

Becerra, R.L. and Coello, C.A.C. (2006) 'Cultured differential evolution for constrained optimization', *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, No. 33, pp.4303–4322.

Cagnina, L.C., Esquivel, S.C. and Coello, C.A.C. (2008) 'Solving engineering optimization problems with the simple constrained particle swarm optimizer', *Informatica*, Vol. 32, No. 3, pp.319–326.

Garg, H. (2014) 'Solving structural engineering design optimization problems using an artificial bee colony algorithm', *Journal of Industrial and Management Optimization*, Vol. 10, No. 3, pp.777–794.

Hamida, S.B. and Schoenauer, M. (2002) 'ASCHEA: new results using adaptive segregational constraint handling', *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, Honolulu, USA, Vol. 1, pp.884–889.

Hsieh, T-J. (2014) 'A bacterial gene recombination algorithm for solving constrained optimization problems', *Applied Mathematics and Computation*, Vol. 231, No. 1, pp.187–204.

Karaboga, D. and Basturk, B. (2007) 'Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems', *Foundations of Fuzzy Logic and Soft Computing, Lecture Notes in Computer Science*, pp.789–798, Springer.

Liang, J.J., Runarsson, T.P., Mezura-Montes, E., Clerc, M., Suganthan, P.N., Coello, C.A.C. and Deb, K. (2006) *Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization*, Technical Report of School of EEE, Nanyang Technological University, Singapore.

Liu, H., Cai, Z. and Wang, Y. (2010) 'Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization', *Applied Soft Computing*, Vol. 10, No. 2, pp.629–640.

Mezura-Montes, E. and Coello, C.A.C. (2005) 'A simple multimembered evolution strategy to solve constrained optimization problems', *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 1, pp.1–17.

Parsopoulos, K.E. and Vrahatis, M.N. (2005) 'Unified particle swarm optimization for solving constrained engineering optimization problems', *Advances in Natural Computation*, Lecture Notes in Computer Science 3612, pp.582–591, Springer.

Runarsson, T.P. and Yao, X. (2000) 'Stochastic ranking for constrained evolutionary optimization', *IEEE Transactions on Evolutionary Computation*, Vol. 4, No. 3, pp.284–294.

Runarsson, T.P. and Yao, X. (2005) 'Search biases in constrained evolutionary optimization', *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, Vol. 35, No. 2, pp.33–243.

Sadollah, A., Bahreininejad, A., Eskandar, H. and Hamdi, M. (2013) 'Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems', *Applied Soft Computing*, Vol. 13, No. 5, pp.2592–2612.

Takahama, T. and Sakai, S. (2005) 'Constrained optimization by applying the α constrained method to the nonlinear simplex method with mutations', *IEEE Transactions on Evolutionary Computation*, Vol. 9, No. 5, pp.437–451.

Tessema, B. and Yen, G.G. (2006) 'A self adaptive penalty function based algorithm for constrained optimization', *Proceedings of the 2006 IEEE Congress on Evolutionary Computation (CEC'06)*, Vancouver, Canada, pp.246–253.

Wang, L. and Li, L-P. (2010) 'An effective differential evolution with level comparison for constrained engineering design', *Structural and Multidisciplinary Optimization*, Vol. 41, No. 6, pp.947–963.

Wang, Y., Cai, Z., Zhou, Y. and Fan, Z (2009) 'Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique', *Structural and Multidisciplinary Optimization*, Vol. 37, No. 4, pp.395–413.

Yahya, N.M., Tokhi, M.O. and Kasdirin, H.A. (2016) 'A new bats echolocation-based algorithm for single objective optimisation', *Evolutionary Intelligence*, Vol. 9, No. 1, pp.1–20.

Yang, X-S. and Hossein, G.A. (2012) 'Bat algorithm: a novel approach for global engineering optimization', *Engineering Computations*, Vol. 29, No. 5, pp.464–483.

Zahara, E. and Kao, Y-T. (2009) 'Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems', *Expert Systems with Applications*, Vol. 36, No. 2, pp.3880–3886.

Zhang, M., Luo, W. and Wang, X. (2008) 'Differential evolution with dynamic stochastic selection for constrained optimization', *Information Sciences*, Vol. 178, No. 15, pp.3043–3074.

Appendix

Constrained problem 1

Minimise:

$$f(x) = 5.3578547x_3^3 + 0.8356891x_1x_5 + 37.293239x_1 + 40,729.141$$

subject to:

$$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0$$

$$g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 - 0.0022053x_3x_5 \le 0$$

$$g_3(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 100 \le 0$$

$$g_4(x) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^3 + 90 \le 0$$

$$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0$$

$$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0$$

where

$$78.0 \le x_1 \le 102.0$$

$$33.0 \le x_2 \le 45.0$$

$$27.0 \le x_i \le 45.0, \quad i = 3, 4, 5$$

Constrained problem 2

Minimise:

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

subject to:

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$$

$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.85 \le 0$$

where

$$13.0 \le x_1 \le 100.00$$

$$0.0 \le x_2 \le 100.0$$

Constrained problem 3

Minimise:

$$f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

subject to:

$$g_1(x) = 127 - 2x_1^2 - 3x_2^4 - x_3 - 4x_4^2 - 5x_5 \geq 0$$

$$g_2(x) = 282 - 7x_1 - 3x_2 - 10x_3^2 - x_4 - x_5 \geq 0$$

$$g_3(x) = 196 - 23x_1 - 2x_2^2 - 6x_6^2 + 8x_7 \geq 0$$

$$g_4(x) = -4x_1^2 - x_2^2 + 3x_1x_2 - 2x_3^2 - 5x_6 + 11x_7 \geq 0$$

where

$$-10.0 \leq x_i \leq 10.0, \quad i = 1, 2, 3, 4, 5, 6, 7$$

Constrained problem 4

Minimise:

$$f(x) = x_1^2 + (x_2 - 1)^2$$

subject to:

$$h(x) = x_2 - x_1^2 = 0$$

where

$$-1.0 \leq x_i \leq 1.0, \quad i = 1, 2$$