Improving the convergence velocity of the balancing composite motion optimisation algorithm

Nang Duc Bui[1], Thang Quan Nguyen[1], Hieu Chi Phan[2], Dung-Nhan Nguyen Bui[3*]

[1]Le Quy Don Technical University, 236 Hoang Quoc Viet Street, Nghia Do Ward, Cau Giay District, Hanoi, Vietnam
[2]Memorial University of 6 Newfoundland, St. John's, Canada
[3]University of Transport Technology, 54 Trieu Khuc Street, Thanh Xuan Nam Ward, Thanh Xuan District, Hanoi, Vietnam

Received 2 April 2024; revised 11 June 2024; accepted 10 September 2024

Abstract:

The balancing composite motion optimisation (BCMO), is a metaheuristic algorithm. However, the strengths of BCMO lie in its ability to extract information from the current generation and balance local and global searches. As a result, the convergence velocity of BCMO is superior to other optimisation methods, with a parameter-free and simplified computational procedure. This study proposes an enhanced algorithm, BCMO_A, designed to further increase the convergence velocity of the original BCMO. The proposed enhancement involves introducing an individual selection operator for the new generation. Three candidate options for the selection process were explored, showing minor differences. The best individuals from the two available populations were selected before interaction. The study also emphasises the importance of the terminal condition, which is based on the best value and the mean of objective functions in the current population. Numerical experiments were conducted on 23 classical benchmark functions, and the results were analysed using official statistical tests. The findings demonstrate that the original BCMO has been significantly improved, achieving better convergence speed, with optimised values reached earlier.

Keywords: **algorithms, balancing composite motion optimisation, convergence velocity, metaheuristic, optimisation.**

Classification numbers: **1.3, 2.3**

1. Introduction

Recently, numerous metaheuristic optimisation algorithms have been proposed, yielding promising results across various fields. Natural evolution serves as the inspiration and foundation for several algorithms, such as the Genetic Algorithm (GA) [1], and later, Differential Evolution (DE) [2]. Other optimisation algorithms mimic natural phenomena, including Particle Swarm Optimisation (PSO) [3-5], Ant Colony Optimisation (ACO) [6, 7], Elephant Herding Optimisation [8], among others. Unlike conventional approaches, all metaheuristic algorithms employ an exchange mechanism between local search and global exploration (i.e., exploration and exploitation), with the solution evolution process commonly governed by random motion.

In such intelligent algorithms, local search focuses on a limited domain to identify the best solution within that domain, while global discovery explores a broader domain in search of an improved solution. These search mechanisms work in tandem to achieve the best solution, and the balance between them significantly impacts the quality of optimisation algorithms. The defining characteristic of each metaheuristic optimisation algorithm is the method used to achieve this balance. Another critical attribute of metaheuristic algorithms is the convergence velocity of the analysis process, particularly for large-scale problems.

Introduced by T.L. Duc, et al. (2020) [9] with some preliminary applications [10], the BCMO is a novel optimisation algorithm that focuses on balancing local and global motion. The key feature of the algorithm is the determination of relative local and global motions, based on information extracted from the current population and the temporary global optimisation point (serving as a surrogate for the true solution of the problem). The BCMO algorithm has been tested and demonstrated superior performance compared to other algorithms. These features, observed in the original study, will be further elaborated upon in this study.

Early application of such algorithm has been applied for the practical optimizing problem in [10]. H.T. Duong, et al. (2020) [10] applied the algorithm to the axial load capacity of a concrete-filled steel tube column. A secondary

*Corresponding author: Email: nhanbnd@utt.edu.vn

deep learning model was successfully developed, based on the dataset generated from a large-scale implementation of BCMO (i.e., 1000 samples), providing a rough estimation of the optimisation design results.

Clearly, the convergence velocity and computational expense of the algorithm are crucial factors contributing to the success of such complex models, where a large number of optimised variables are required. Metaheuristic algorithms are well-known for their adaptability, allowing for further improvements. It has been observed in the literature that original algorithms can be significantly enhanced with either minor or major adjustments to improve their performance [11-13]. For instance, an adjustment was made by incorporating a Greedy Adaptive control parameter into DE [11]. In contrast to DE, BCMO has the advantage of being parameter-free, requiring only the number of individuals and generations to be preset. The Elephant Herding Optimization (EHO) [8] was modified using feasible rules to solve an optimisation problem with constraints [12]. Similarly, G. Liu, et al. (2016) [13] proposed an inertia weight update strategy to enhance search space coverage and speed for the real-time multiple object tracking problem. Numerous other examples of successful algorithm enhancements can be found in [14-20].

During the algorithm implementation process, the authors observed that BCMO could be adjusted and refined to achieve faster convergence and reduce computational expense. The adjustment involves supplementing the individual selection operator within the procedure and emphasising the importance of the terminal condition, which was mentioned in the original paper [9] but not included in the provided code. This paper is organised as follows: Section 2 summarises the fundamentals of the BCMO algorithm; Section 3 proposes approaches for improving the convergence velocity and speed of the original algorithm. Section 4 presents an extensive numerical comparison between the original and adjusted algorithms (i.e., BCMO versus BCMO_A, respectively).

2. The original balancing composite motion optimisation

The initial population of the BCMO is generated in a manner analogous to other metaheuristic algorithms, by randomly selecting design variables with uniform distribution within their corresponding bounds, as shown in Eq. 1:

$$x_i = x_i^L + rand(1,d) \times (x_i^U - x_i^L) \quad (1)$$

where x_i^U; x_i^L are the upper and lower bounds of the i^{th} design variables; rand(1,d) is a uniform random vector of size d[0,1].

The new position of the i^{th} (i>0) individual of a given generation in the next generation can be searched by Eqs. 2 and 3:

$$x_i^{t+1} = x_i^t + v_i \quad (2)$$

$$v_i = v_{i/j} + v_j \quad (3)$$

Thus, the new position of the i^{th} individual in Eqs. 2 and 3 is similar to that in PSO, ACO, FA, and DE. The distinction of BCMO lies in the definition of the motion vector v_i, where v_i and v_j represent the motion of i and j relative to the global solution O; $v_{i/j}$ is the relative motion of i^{th} individual to the j^{th} individual, and the j^{th} individual performs better than the i^{th}.

Clearly, v_i depends on two points, one of which is the global solution, or the actual solution to the problem, which has not yet been obtained. Therefore, an alternative for O is proposed by the so-called "instant global optimisation point" O_{in}.

If individuals in the population are ranked in descending order based on the objective function (for minimisation), j is smaller than i, or i=j=1. In the simplest case, O_{in} can be taken as the top-ranked individual of the population. However, this may lead to premature convergence due to the consistency of over several generations. To avoid premature termination, O_{in} must be updated by Eq. 4:

$$x_{O_{in}}^t = \begin{cases} u_1^t & if\ f(u_1^t) < f(x_1^t) \\ x_1^{t-1} & otherwise \end{cases} \quad (4)$$

where u_1^t is the trial vector, obtained by extracting information of the current population, as shown in Eq. 5:

$$u_1^t = u_{c+} v_{k1/k2}^t + x_{k2/1}^t \quad (5)$$

where u_c is the centre point of the design space [LB, UB]:

$$u_c = \frac{LB+UB}{2} \quad (6)$$

where $v_{k1/k2}^t$, $x_{k2/1}^t$ are the pseudo relative movements of the k_1^{th} individual to the k_2^{th} and k_2^{th} to the first individual, respectively. k_2^{th} is randomly chosen between [2, NP], NP is the size of the population. The values of are obtained in the same way as $v_{i/j}$ and v_i.

With the new location of the individual calculated as in Eqs. 2 and 3, BCMO incorporates the local and global

searches, represented by v_j and $v_{i/j}$, respectively. The global search is defined by Eq. 7:

$$v_j = \propto_j (x_{Oin} - x_j) \quad (7)$$

where α_j is the first-order derivative of the movement distance $x_{Oin} - x_j$ and found by Eq. 8:

$$\propto_j = L_{GS} * dv_j \quad (8)$$

where L_{GS} is the length of the global step size of the j^{th} individual, dv_j is the direction vector with a sign of 0.5:0.5 for positive:negative. L_{GS} and dv_j are found by Eq. 9, which depends on the random number TV_j uniformly distributed within [0, 1]:

$$L_{GS} = \begin{cases} e^{\frac{1}{d}\frac{j}{NP}r_j^2} & if\ TV_j > 0.5 \\ e^{\frac{-1}{d}(1-\frac{j}{NP})r_j^2} & otherwise \end{cases} \quad (9)$$

$$dv_j = \begin{cases} rand(1,d) & if\ TV_j > 0.5 \\ -rand(1,d) & otherwise \end{cases} \quad (10)$$

where d is the number of dimensions or direction of S space, NP is the size of the population, rand (1,d) is a random number within [0, 1], and r_j is the distance from the j^{th} individual to O_{in} as in Eq. 11:

$$r_j = \|x_j - x_{Oin}\| \quad (11)$$

Analogous to v_j, $v_{i/j}$ can be found as in Eqs. 12 and 13:

$$v_{i/j} = \propto_{ij} (x_j - x_i) \quad (12)$$

$$\propto_{ij} = L_{LS} \times dv_{ij} \quad (13)$$

where L_{LS} is fixed at 1 to balance the exploration and exploitation of the i^{th} individual; dv_{ij} is the direction vector of i^{th} and j^{th} individual as in Eq. 10.

The i^{th} individual may move towards or away from the O_{in} and j^{th} individual with better performance. If $v_{i/j}$ is positive, the i^{th} individual moves into the local exploration domain of the j^{th} individual, and vice versa. Similarly v_j can move away or towards the instant global optimisation point O_{in} after each generation. To balance the exploration and exploitation of each individual in the solution domain, the proportion of positive and negative values for $v_{i/j}$ and v_j should be equal.

Figure 1 provides the procedure of the original BCMO proposed in [9], along with numerical experiments conducted on 23 classical benchmark functions for optimisation, the CEC 2014 benchmark functions, and three real-life engineering design problems, all of which yielded promising results. The key advantages of BCMO can be summarised as follows:

(I) The motion of the individual to the new location depends on:

(Ia) information of the best individual at the current generation $[X]^t_{best}$ and

(Ib) the individual considered as the pseudo global point $[X]^t_{Oin}$ and

(Ic) the reversal of the motion directions, with 50% balancing the local and global searches.

(II) Unlike other algorithms, BCMO is a parameter-free optimisation algorithm, reducing the need for fine-tuning parameters to obtain optimal solutions.

(III) The convergence velocity of BCMO is notably fast, especially for problems with a large number of design variables. This is crucial for the potential application of BCMO in developing deep learning models, where optimising a high-dimensional set of weights is required, and where multiple computationally expensive algorithms are involved.

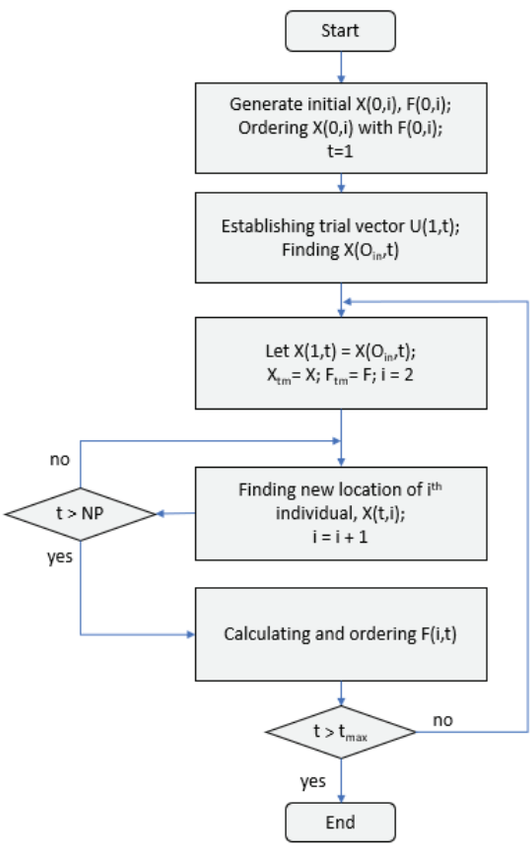

Fig. 1. Flow chart of the original balancing composite motion optimisation.

3. The adjusted algorithm balancing composite motion optimisation_A

3.1. Clarifying the terminal conditions

As with other metaheuristic algorithms, the original BCMO applied 2 terminal conditions simultaneously: a maximum number of interactions and an error value, with the algorithm terminating upon meeting either condition first. However, the code provided with the paper [9] did not include the error value condition. Without this condition, the calculation process may unnecessarily repeat. This can be observed in Eqs. 2 and 3, where the number of vectors v_j and $v_{i/j}$ depends on the relative locations of i^{th} and j^{th} and O_{in} as in Eqs. 7 and 8. When the algorithm nears convergence at a minimum point (whether local or global), the individuals move closer to one another, and v_i, $v_{i/j}$ will approach zero. In such cases, no motion occurs, yet the number of interactions increases.

To address this drawback, the terminal condition from [2] is proposed. This condition is based on the best and mean objective function values of individuals in the current population, as shown in Eq. 14:

$$\left|F(x)_{Min} - \frac{\sum_{i=1}^{Np} F(x)_i}{Np}\right| \leq \varepsilon \quad (14)$$

3.2. Implementing the selecting operator

Metaheuristic algorithms commonly employ a population selection operator before proceeding to the next iteration. The original BCMO did not discuss this operator. This study proposes incorporating a selection operation during the analysis process to improve the algorithm's convergence speed.

Due to the random nature of the local and global searches, some individuals may have poorer adaptability, affecting the overall population's performance. At the end of an iteration, two populations exist simultaneously (i.e., X_{t-1} and X_t). Using only one population, X_t, for the next iteration results in the loss of valuable information. This study proposes extracting that information by selecting the best NP individuals from the two populations (i.e., selecting from 2NP individuals).

In summary, the flow chart of the adjusted BCMO, termed BCMO_A (presented in Fig. 2), will include the terminal condition outlined in Eq. 14 and the selection operation. The following sections will present the numerical experiments.

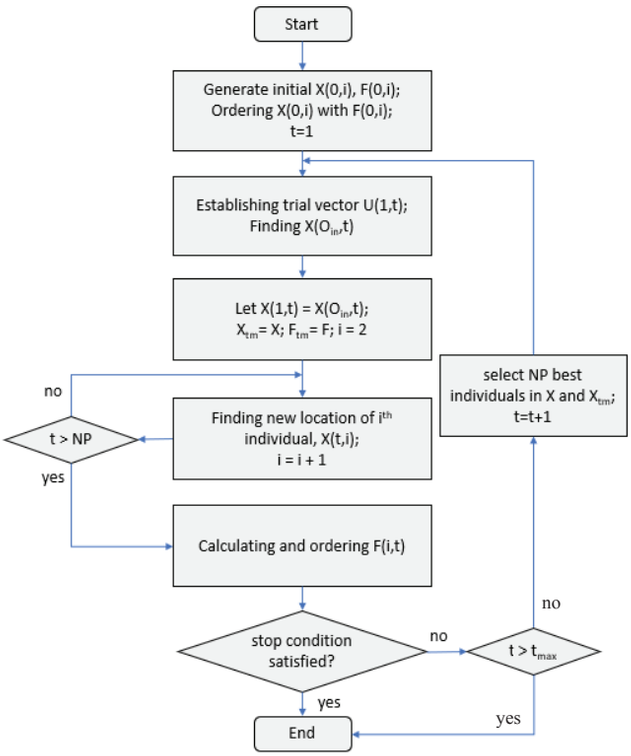

Fig. 2. Flowchart of the adjusted algorithm, balancing composite motion optimisation_A.

4. Numerical experiments

4.1. The validated optimisation problems

To validate the quality of the optimisation algorithm, benchmark functions as described in [2, 21-23] are commonly used. In this study, 23 conventional benchmark functions were employed, as listed in Table 1. The first group consists of single minimum functions, numbered from F01 to F07. Functions F08 to F13 have multiple minima, and functions F14 to F23 have fixed design variables.

4.2. Establishing code for balancing composite motion optimisation and balancing composite motion optimisation_A

Based on T.L. Duc, et al. (2020) [9] and the associated code provided, the program allows for the use of the flowchart in either Fig. 1 or 2. The results generated by the present code for the original BCMO are designated as BCMO* to differentiate them from the results in [9]. Table 2 compares the results of 23 selected functions between BCMO* and BCMO, using the same flowchart as in Fig. 1. Each implementation was conducted with NP=100 and N_{thmax}=1000 as the terminal condition. Such optimisation analyses were implemented on a computer with a configuration of CPU Intel Core i3, 2.53 GHz, 4 GB RAM.

Table 1. Benchmark functions.

ID	I. Functions with single minimum/maximum	nd	Boundaries	Theoretical solution
F01	**Sphere model** $f_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100, 100]	F_{min}=f (0, 0,...,0)=0
F02	**Schewefel's problem 2.22** $f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10, 10]	F_{min}=f (0, 0,...,0)=0
F03	**Schewefel's problem 1.2** $f_2(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j^2 \right)$	30	[-100, 100]	F_{min}=f (0, 0,...,0)=0
F04	**Schewefel's problem 2.21** $f_4(x) = max_i[\lvert x_i \rvert, 1] \leq i \leq n$	30	[-100, 100]	F_{min}=f (0, 0,...,0)=0
F05	**Generalised rosenbrock's function** $f_1(x) = \sum_{i=1}^{n-1} \left[1000(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$	30	[-30, 30]	F_{min}=f (1, 1,...,1)=0
F06	**Step function** $f_6(x)=(\lvert x_i+0.5 \rvert)^2$ *[f(x)=floor (xi +0.5)²]]*	30	[-100, 100]	F_{min}=f (0, 0,...,0)=0
F07	**Quartic function with nose** $f_7(x) = \sum_{i=1}^{n} i\, x_i^4 + rand$	30	[-1.28,1.28]	F_{min}=f (0, 0,...,0)=0
	II. Functions with multiple minima/maxima			
F08	**Generalised schwefel's problem 2.26** $f_8(x) = \sum_{i=1}^{n} -x_i sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	[-500, 500]	F_{min} = f(421,...,421)=- 12569.5
F09	**Generalised rastrigin function** $f_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10cos(2\pi x_i) + 10 \right]$	30	[-5.12, 5.12]	F_{min}=f (0, 0,...,0)=0
F10	**Ackley's function** $f_{10}(x) = -20exp\left(-0.2 \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} \right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)\right) + 20 + e$	30	[-32, 32]	F_{min}=f (0, 0,...,0)=0
F11	**Generalised griewank function** $f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600, 600]	F_{min}=f (0, 0,...,0)=0
F12	**Generalised penalised function** $f_{12(x)} = \frac{\pi}{30}[10sin^2(\pi y_1) + \sum_{1}^{n-1} (y_i - 1)^2[1 + 10sin^2(\pi y_{i+1}) + (y_n - 1)^2]] + \sum_{i=1}^{30} u(x_i, 10,100,4)$ $y_i = 1 + \frac{x_i - 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & if\ x_i > a \\ 0 & if - a < x_i < a \\ k(-x_i - a)^m \end{cases}$	30	[-50, 50]	F_{min}=f(1, 1,...,1)=0
F13	$f_{13}(x) = 0.1[sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[sin^2(2\pi x_n)]] + \sum_{i=1}^{n} u(x_i, 5,100,4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & if\ x_i > a \\ 0 & if - a < x_i < a \\ k(-x_i - a)^m \end{cases}$	30	[-50, 50]	F_{min}=f(1, 1,...,1)=0
	III. Functions with multiple minima/maxima and fixed design variables			
F14	**Shekel's foxholes function** $f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \right]^{-1}$ $a_{ij} = \begin{bmatrix} -32, -16, 0, 16, 32, -32 \dots 0, 16, 32 \\ -32, -32, -32, -32, -32, -16 \dots 32, 32, 32 \end{bmatrix}$	2	[-65, 65]	F_{min}=f(-32, -32)= 0,998004

ID	I. Functions with single minimum/maximum	nd	Boundaries	Theoretical solution

F15	**Kowalik's function**
	$f_{15}(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$
	a=[0.1957 0.1947 0.1735 0.1600 0.0844 0.0627 0.0456 0.0342 0.0323 0.0235 0.0246];
	b=[4 2 1 1/2 1/4 1/6 1/8 1/10 1/12 1/14 1/16];
		4	[-5, 5]	F_{min}=f(0.1928, 0.1908, 0.1231, 0.1358)= 0.0003075

F16	**Six-hump camel-back function** $f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[-5, 5]	F_{min}=f(0.08983, -0.7126)=f(-0.08983, 0.7126)=-1.0316285
F17	**Branin function** $f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosx_1 + 10$	2	[-5, 5]	F_{min}=f(-3.142, 12.275);(3.142, 2.275);(9.425,2.275)=0.397887
F18	**Goldstein-price function** $f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30(2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[-2,2]	F_{min}=f (0, -1)=3
F19	**Hartman's family** $f_{19}(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\right]$ $a = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$; p=$\begin{bmatrix} 0.3689 & 0.1170 & 0.2673 \\ 0.4699 & 0.4387 & 0.7470 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$; c=$[1;\ 1.2;\ 3;\ 3.2]^T$	3	[0.1]	F_{min}=f(0:114; 0:556; 0:852)=-3.86
F20	$f_{20}(x) = -\sum_{i=1}^{4} c_i exp\left[-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\right]$ $a = \begin{bmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$; ; c=$[1;\ 1.2;\ 3;\ 3.2]^T$ p= $\begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$	6	[0, 1]	F_{min}=f(0.201; 0.150; 0.477; 0.275; 0.311; 0.657)=-3.32
21	**Shekel's family** $f_{(21)}(x) = -\sum_{i=1}^{5}[(x_i - a_i)(x_i - a_i)^T + c_i]^{-1}$ $a = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix}$; $c = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.5 \end{bmatrix}$	4	[0, 10]	$F_{min,21}$=f(4.00004, 4.00013, 4.00004, 4.00013)= -10.1532;
22	**Shekel's family** $f_{(22)}(x) = -\sum_{i=1}^{7}[(x_i - a_i)(x_i - a_i)^T + c_i]^{-1}$ $a = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix}$; $c = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.5 \end{bmatrix}$	4	[0, 10]	$F_{min,22}$=f(4.00057,4.00069, 3.99949, 3.99961)=-10.403;
23	**Shekel's family** $f_{(23)}(x) = -\sum_{i=1}^{10}[(x_i - a_i)(x_i - a_i)^T + c_i]^{-1}$ $a = \begin{bmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{bmatrix}$; $c = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.5 \end{bmatrix}$	4	[0, 10]	$F_{min,23}$=f(4.00075, 4.00059, 3.99949, 3.99961)=-10.5364

Table 2. Validating results of the present code.

No. F	BCMO[1]	BCMO*	No. F	BCMO	BCMO*
F01	2.180E-42	1.240E-39	F13	5.100E-03	5.126E-03
F02	3.172E-24	3.977E-25	F14	9.980E-01	9.980E-01
F03	2.937E-30	9.636E-73	F15	3.991E-04	4.601E-04
F04	1.686E-15	3.035E-123	F16	-1.032E+00	-1.032E+00
F05	2.749E+01	2.748E+01	F17	3.979E-01	3.979E-01
F06	2.284E-06	0.000E+00	F18	3.000E+00	3.000E+00
F07	1.960E-02	2.235E-04	F19	-3.863E+00	-3.863E+00
F08	-9.256E+03	-9.242E+03	F20	-3.274E+00	-3.259E+00
F09	0.000E+00	0.000E+00	F21	-7.740E+00	-7.475E+00
F10	8.882E-16	4.441E-16	F22	-9.474E+00	-7.455E+00
F11	0.000E+00	0.000E+00	F23	-9.684E+00	-8.653E+00
F12	1.040E-02	1.037E-02			

[1]: results from [9]. BCMO: balancing composite motion optimisations. BCMO*: the results generated by the present code for the original BCMO to differentiate them from the results.

For statistically testing the difference between results from T.L. Duc, et al. (2020) [9] and those of our present code, the Wilcoxon test was conducted, as shown in Table 3. With a p-value of 0.792, the null hypothesis "there is no difference between the results of the two coding programs" - cannot be rejected.

Table 3. Wilcoxon test on the original and adjusted balancing composite motion optimisations.

Methods	Sign test			Wilcoxon test				
Parameter	R^-	R^+	R^0	R^-	R^-avg	R^+	R^+avg	p_value
BCMO* - BCMO	9	8	6	60	6.67	93	11.63	0.792

BCMO*: are the results generated by the present code for the original BCMO to differentiate them from the results.

4.3. Options of the selection operation

As discussed earlier, at the end of each interaction, two populations, $[X]_{t-1}$ and $[X]_t$, are generated and there are three potential options that can be used as the selection operator to obtain $[X]_{t+1}$. They are:

- Option 1: From the 2×NP individuals in $[X]_t$ and $[X]_{t-1}$ at the beginning of an iteration, the NP best individuals are selected for the next iteration $[X]_{t+1}$;

- Option 2: Pairs of NP 2 individuals, each from $[X]_t$ and $[X]_{t-1}$, are compared, and the individual with better performance is selected for $[X]_{t+1}$;

- Option 3: From the 2×NP individuals, the best NP/2 individuals from each population are selected for the next iteration, $[X]_{t+1}$.

The chosen option is determined based on statistical analyses for the 23 benchmark functions (Table 4). Three metrics are used to assess each option in determining the final choice:

Table 4. Statistical features of options for selection operator.

F	F_{opt}*	Option 1			Option 2			Option 3		
		F_{opt_mean}	N_{opt}	Time	F_{opt_mean}	N_{opt}	Time	F_{opt_mean}	N_{opt}	Time
F1	0.000E+00	4.430E-10	13	0.294	5.398E-14	0	240.0	0.000E+00	30	0.357
F2	0.000E+00	2.313E-09	16	0.380	2.304E-11	0	292.0	0.000E+00	30	0.536
F3	0.000E+00	3.193E-11	7	0.337	8.230E-19	0	203.0	0.000E+00	30	0.432
F4	0.000E+00	2.737E-11	7	0.096	6.976E-19	0	106.0	0.000E+00	30	0.312
F5	0.000E+00	2.674E+01	0	3.633	2.736E+01	0	1000.0	2.898E+01	0	0.610
F6	0.000E+00	0.000E+00	30	0.967	0.000E+00	30	438.0	0.000E+00	30	0.210
F7	0.000E+00	2.673E-04	0	5.785	3.939E-03	0	5.97	5.943E-04	0	6.549
F8	-1.043E+04	-6.474E+03	0	3.275	-8.272E+03	0	1000.0	-3.277E+03	0	0.793
F9	0.000E+00	4.184E-10	12	0.284	8.911E-14	16	212.0	0.000E+00	30	0.359
F10	0.000E+00	2.710E-09	0	0.514	5.375E-11	0	318.0	4.441E-16	0	0.639
F11	0.000E+00	6.621E-10	10	0.379	3.962E-14	6	212.0	0.000E+00	30	0.527
F12	0.000E+00	1.381E-05	0	3.680	1.037E-02	0	984.0	1.621E+00	0	0.813
F13	0.000E+00	4.324E-01	0	4.494	1.080E-01	0	1000.0	5.399E+00	0	0.812
F14	9.980E-01	3.016E+00	10	0.265	1.197E+00	24	168.0	8.105E+00	0	0.335
F15	3.075E-04	1.457E-03	0	2.645	6.247E-04	2	912.0	2.410E-02	0	0.279
F16	-1.032E+00	-1.032E+00	30	0.093	-1.032E+00	30	100.0	-9.192E-01	0	0.283
F17	3.979E-01	3.979E-01	29	0.097	3.979E-01	30	102.0	5.498E-01	0	0.264
F18	3.000E+00	3.000E+00	30	0.088	3.000E+00	30	106.0	7.773E+00	0	0.276
F19	-3.863E+00	-3.863E+00	30	0.118	-3.863E+00	30	105.0	-3.793E+00	0	0.301
F20	-3.322E+00	-3.233E+00	11	0.259	-3.251E+00	12	190.0	-2.601E+00	0	0.334
F21	-1.015E+01	-7.984E+00	20	0.199	-8.560E+00	22	176.0	-1.581E+00	0	0.381
F22	-1.040E+01	-7.826E+00	19	0.191	-9.207E+00	25	163.0	-1.622E+00	0	0.337
F23	-1.054E+01	-9.245E+00	25	0.025	-9.353E+00	25	190.0	-1.846E+00	0	0.374

F_{opt}*: represents the theoretical results for the corresponding benchmark function.

1. The mean of optimised results after 30 analyses on the same benchmark function, F_{opt_mean}.

2. The number of times the adjusted BCMO algorithm reaches the theoretical results, N_{opt}.

3. The mean computational time. F_{opt}* represents the theoretical results for the corresponding benchmark function. Statistical results of the three options, after running each benchmark function 30 times, are presented in Table 4, with a maximum number of generations set to 1000 and an error value ε is 1.e-8. Paired statistical tests with options 1-2 and 1-3 are given in Table 5. The adjusted BCMO with chosen options is designated as BCMO_A.

Table 5. Statistical tests of the difference between Option 1 to Options 2 and 3.

Method	Parametric	Option 1 - Option 2			Option 1 - Option 3		
		F_{mean}	N_{opt}	Time	F_{mean}	N_{opt}	Time
Sign test	R^-	4	6	22	15	6	17
	R^+	14	6	1	7	10	6
	R^0	5	11	0	1	7	0
Wilcoxon test	R-	42	34	275	225	46	154
	Mean(R-)	10.5	5.67	12.5	15	7.67	9.06
	R+	129	44	1	28	90	122
	Mean(R-)	9.21	7.33	1.0	4	9	20.33
	P_Value	0.606	0.693	0.004	0.974	0.105	0.317

It can be seen in Table 5 that F_{mean} of Option 1 versus Option 2 and Option 1 versus Option 3 are statistically similar, with the corresponding p-values consistently greater than the critical value of 0.05 (i.e., 0.606 and 0.974, respectively). Likewise, for N_{opt}, no significant differences are seen, although the p-value of Option 1 versus Option 3 is somewhat lower at 0.105 but still exceeds the critical p-value threshold. The only noticeable difference arises in the computation time for Option 1 compared to Option 2, with a p-value of 0.004. In the case of Option 1 versus Option 3, no statistically significant difference in time is observed. Therefore, Option 1 was selected for the selection operation in this study.

4.4. Comparing original and adjusted balancing composite motion optimisation

The results comparing the original and adjusted BCMO algorithms are presented in Table 6. For each benchmark function, both the original algorithm (BCMO) and the adjusted algorithm (BCMO_A) were run 30 times. The recorded data include: the mean value of optimised objective functions (i.e., F_{mean} and F_{mean_A} for BCMO and BCMO_A respectively), the best of optimised objective functions (i.e., F_{best} and F_{best_A} for BCMO and BCMO_A respectively), the number of times the algorithm reach the theoretical optimised value (i.e., N_{opt} and N_{opt_A} for BCMO and BCMO_A, respectively, maximum=30), and the computational time required for the algorithm on the same computer configuration as mention in Section 4.2 (i.e., T and T_A for BCMO and BCMO_A, respectively). The T_A/T ratio is also provided to offer a quick overview of the improvement in computation time due to the selection operator.

It can be seen that in most cases (except for F7), the additional operations successfully reduced computation time, from 0.8739 (F15) to 0.0266 (F23).

For functions with a single minimum or maximum (i.e., F1 to F7), F_{mean} values of BCMO are superior to those of BCMO_A. However, the probability of converging to the theoretical optimised value is higher for BCMO_A than for BCMO. For functions with multiple minima or maxima, the selection operation does not significantly affect the capacity of the algorithm to find the global minimum or maximum. The most significant observation in Table 6 is the difference in computation time between the two algorithms. The adjusted BCMO_A, with the added selection operation, converges faster than the original algorithm. The only exception in computation time occurs with F7, where both BCMO and BCMO_A stop due to reaching the maximum number of generations. Formal statistical tests on the differences between the two algorithms are presented in Table 7. The results indicate that: (1) there is no statistical difference in the solution-searching capacity or the ability to reach the theoretical solutions between BCMO and BCMO_A, and (2) BCMO_A significantly reduces computation time compared to the original BCMO. Thus, while preserving the quality of solution searching, BCMO_A demonstrates improved convergence speed.

Table 6. Comparison of balancing composite motion optimisation and balancing composite motion optimisation_A (NP=100; N_{thmax}=1000; ε=1.0E-8).

Function	F_{opt}*	BCMO_A				BCMO				T_A/T
		F_{mean_A}	F_{best_A}	N_{opt}	T_A (s)	F_{mean}	F_{best}	N_{opt}	T (s)	
F1	0.000E+00	4.430E-10	0.000E+00	13	0.294	6.641E-18	4.419E-28	0	1.737	0.1693
F2	0.000E+00	2.313E-09	0.000E+00	16	0.380	7.306E-13	3.208E-17	0	2.142	0.1774
F3	0.000E+00	4.966E-11	0.000E+00	5	0.337	6.334E-21	5.835E-55	0	0.600	0.5617
F4	0.000E+00	4.806E-11	0.000E+00	5	0.096	2.141E-20	8.155E-31	0	0.632	0.1519
F5	0.000E+00	2.674E+01	2.625E+01	0	3.633	2.745E+01	2.732E+01	0	4.227	0.8595
F6	0.000E+00	0.000E+00	0.000E+00	30	0.967	0.000E+00	0.000E+00	30	2.580	0.3748
F7	0.000E+00	2.673E-04	6.980E-07	0	5.785	1.244E-02	1.560E-03	0	5.623	1.0288
F8	-1.043E+04	-6.474E+03	-9.012E+03	0	3.275	-9.353E+03	-1.067E+04	0	5.778	0.5668
F9	0.000E+00	4.184E-10	0.000E+00	12	0.284	0.000E+00	0.000E+00	30	1.890	0.1503
F10	4.441E-16	2.710E-09	4.441E-16	0	0.514	1.244E-12	4.441E-16	0	3.060	0.1680
F11	0.000E+00	6.621E-10	0.000E+00	10	0.379	6.291E-17	0.000E+00	29	2.664	0.1423
F12	0.000E+00	1.982E-06	2.894E-08	0	3.680	6.911E-03	3.313E-03	0	7.483	0.4918
F13	0.000E+00	4.324E-01	2.412E-08	0	4.494	5.502E-03	3.495E-07	0	7.726	0.5817
F14	9.980E-01	3.016E+00	9.980E-01	10	0.265	9.980E-01	9.980E-01	30	0.900	0.2944
F15	3.075E-04	5.658E-04	3.075E-04	4	3.090	5.211E-04	3.075E-04	19	3.536	0.8739
F16	-1.032E+00	-1.032E+00	-1.032E+00	30	0.093	6.282E-13	-1.032E+00	0	2.150	0.0433
F17	3.979E-01	3.979E-01	3.979E-01	29	0.097	3.979E-01	3.979E-01	30	0.485	0.2000
F18	3.000E+00	3.000E+00	3.000E+00	30	0.088	3.000E+00	3.000E+00	30	0.516	0.1705
F19	-3.863E+00	-3.863E+00	-3.863E+00	30	0.118	-3.863E+00	-3.863E+00	30	0.601	0.1963
F20	-3.322E+00	-3.233E+00	-3.322E+00	11	0.259	-3.255E+00	-3.322E+00	13	1.061	0.2441
F21	-1.015E+01	-7.984E+00	-1.015E+01	20	0.199	-7.483E+00	-1.015E+01	18	0.85	0.2341
F22	-1.040E+01	-7.184E+00	-1.040E+01	16	0.191	-8.330E+00	-1.040E+01	21	0.88	0.2170
F23	-1.054E+01	-9.245E+00	-1.054E+01	25	0.025	-8.391E+00	-1.054E+01	21	0.94	0.0266

BCMO: balancing composite motion optimisations. F_{opt}*: represents the theoretical results for the corresponding benchmark function.

Table 7. Statistical tests on the performance of balancing composite motion optimisation versus balancing composite motion optimisation_A.

Test method / Metric	Sign test			Wilcoxon test				
	R^-	R^+	R^0	R^-	R^-_{avg}	R^+	R^+_{avg}	*p-value*
F_{mean_A} / F_{mean}	6	13	4	77	12.83	113	12.83	0.606
F_{best_A} / F_{best}	10	1	12	55	5.5	11	11	0.659
T_A / T	21	2	0	273	13	3	1.5	0.0017

5. Conclusions

The original BCMO, which balances global and local searches, is a promising optimisation algorithm, particularly for optimising functions with large design variables. Its major advantage is its parameter-free nature, which makes it both practical and convenient for users. This study proposed a modification to the original BCMO by adding a selection operation to improve the algorithm. The effectiveness of three candidate selection methods for choosing the new

population from the 2×NP individuals was investigated. Numerical experiments comparing the performance of BCMO and BCMO_A show that while maintaining the original algorithm's quality, the proposed adjustment significantly improves convergence speed.

CRediT author statement

Nang Duc Bui: Conceptualisation, Supervision; Thang Quan Nguyen: Conceptualisation, Methodology, Writing - Original draft preparation; Hieu Chi Phan: Conceptualisation, Validation, Writing - Original draft preparation; Dung-Nhan Nguyen Bui: Conceptualisation, Writing - Reviewing and Editing.

COMPETING INTERESTS

The authors declare that there is no conflict of interest regarding the publication of this article.

REFERENCES

[1] Z. Michalewicz (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, DOI: 10.1007/978-3-662-03315-9.

[2] K.V. Price, R.M. Storn, J.A. Lampinen (2006), *Differential Evolution: A Practical Approach to Global Optimization*, Springer, DOI: 10.1007/3-540-31306-0.

[3] Q. Bai (2010), "Analysis of particle swarm optimization algorithm", *Computer and Information Science*, **3(1)**, pp.180-184, DOI: 10.5539/cis.v3n1p180.

[4] R.C. Eberhart, Y. Shi (2001), "Particle swarm optimization: Developments, applications and resources", *Proceedings of The 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, **1**, pp.81-86, DOI: 10.1109/CEC.2001.934374.

[5] J. Kennedy, R.C. Eberhart, Y. Shi (2001), *Swarm Intelligence*, Elsevier, DOI: 10.1016/B978-1-55860-595-4.X5000-1.

[6] M. Dorigo, M. Birattari, T. Stutzle (2006), "Ant colony optimization", *IEEE Computational Intelligence Magazine*, **1(4)**, pp.28-39, DOI: 10.1109/MCI.2006.329691.

[7] M. Dorigo, G.D. Caro (1999), "Ant colony optimization: A new meta-heuristic", *Proceedings of The 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, DOI: 10.1109/CEC.1999.782657.

[8] G.G. Wang, S. Deb, X.Z. Gao, et al. (2016), "A new metaheuristic optimisation algorithm motivated by elephant herding behaviour", *International Journal of Bio-Inspired Computation*, **8(6)**, pp.394-409, DOI: 10.1504/IJBIC.2016.081335.

[9] T.L. Duc, Q.H. Nguyen, H.N. Xuan (2020), "Balancing composite motion optimization", *Information Sciences*, **520**, pp.250-270, DOI: 10.1016/j.ins.2020.02.013.

[10] H.T. Duong, H.C. Phan, T.T. Le, et al. (2020), "Optimization design of rectangular concrete-filled steel tube short columns with balancing composite motion optimization and data-driven model", *Structures*, **28**, pp.757-765, DOI: 10.1016/j.istruc.2020.09.013.

[11] M. Leon, N. Xiong (2016), "Adapting differential evolution algorithms for continuous optimization via greedy adjustment of control parameters", *Journal of Artificial Intelligence and Soft Computing Research*, **6(2)**, pp.103-118, DOI: 10.1515/jaiscr-2016-0009.

[12] A. Alihodzic, E. Tuba, R.C. Hrosik, et al. (2017), "Unmanned aerial vehicle path planning problem by adjusted elephant herding optimization", *2017 25th Telecommunication Forum (Telfor)*, pp.1-4, DOI: 10.1109/TELFOR.2017.8249468.

[13] G. Liu, Z. Yeung, H.W.F. Chung, et al. (2016), "A new weight adjusted particle swarm optimization for real-time multiple object tracking", *International Conference on Neural Information Processing*, Springer, pp.643-651, DOI: 10.1007/978-3-319-46672-9_72.

[14] E. Tuba, M. Tuba, E. Dolicanin (2017), "Adjusted fireworks algorithm applied to retinal image registration", *Studies in Informatics and Control*, **26(1)**, pp.33-42, DOI: 10.24846/v26i1y201704.

[15] B.S. Yıldız, S. Kumar, N. Pholdee, et al. (2022a), "A new chaotic Lévy flight distribution optimization algorithm for solving constrained engineering problems", *Expert Systems*, **39(8)**, DOI: 10.1111/exsy.12992.

[16] S. Kumar, B.S. Yildiz, P. Mehta, et al. (2023), "Chaotic marine predators algorithm for global optimization of real-world engineering problems", *Knowledge-Based Systems*, **261**, DOI: 10.1016/j.knosys.2022.110192.

[17] B.S. Yıldız, S. Kumar, N. Panagant, et al. (2023), "A novel hybrid arithmetic optimization algorithm for solving constrained optimization problems", *Knowledge-Based Systems*, **271**, DOI: 10.1016/j.knosys.2023.110554.

[18] B.S. Yıldız, P. Mehta, N. Panagant, et al. (2022b), "A novel chaotic Runge Kutta optimization algorithm for solving constrained engineering problems", *Journal of Computational Design and Engineering*, **9(6)**, pp.2452-2465, DOI: 10.1093/jcde/qwac113.

[19] P. Mehta, B.S. Yıldız, S.M. Sait, et al. (2023), "A novel hybrid Fick's law algorithm-quasi oppositional-based learning algorithm for solving constrained mechanical design problems", *Materials Testing*, **65(12)**, pp.1817-1825, DOI: 10.1515/mt-2023-0235.

[20] D. Gürses, P. Mehta, S.M. Sait, et al. (2023), "A multi-strategy boosted prairie dog optimization algorithm for global optimization of heat exchangers", *Materials Testing*, **65(9)**, pp.1396-1404, DOI: 10.1515/mt-2023-0082.

[21] E.P. Adorio, U. Diliman (2005), *MVF-multivariate Test Functions Library in C for Unconstrained Global Optimization*, http://www.geocities.ws/eadorio/mvf.pdf, accessed 21 January 2024.

[22] K. Hussain, M.N.M. Salleh, S. Cheng, et al. (2017), "Common benchmark functions for metaheuristic evaluation: A review", *JOIV: International Journal on Informatics Visualization*, **1(4-2)**, pp.218-223, DOI: 10.30630/joiv.1.4-2.65.

[23] M. Hellwig, H.G. Beyer (2019), "Benchmarking evolutionary algorithms for single objective real-valued constrained optimization - A critical review", *Swarm and Evolutionary Computation*, **44**, pp.927-944, DOI: 10.1016/j.swevo.2018.10.002.