Evaluating the Merits of Big-Bang Big-Crunch Optimization Through a Comparative Analysis Between Nature Inspired Meta-heuristics

*An empirical comparison between global-best particle swarm optimization, differential evolution, and big-bang big-crunch optimization.

JC De Witt
Stellenbosch University
(Department of Computer Science)
Stellenbosch, South Africa
21663904@sun.ac.za

***Abstract*—Nature is a source of inspiration for meta-heuristic algorithms like particle swarm optimization (PSO), and differential evolution (DE). Big bang- big crunch (BB-BC) is an optimization technique derived from the theorized life cycle of the universe. BB-BC optimization is a meta-heuristic algorithm associated with fast convergence. The main contributions of this study are the proposal and evaluation of a BB-BC variant, called modified big bang-big crunch (MBB-BC). MBB-BC is designed to offer superior exploration-exploitation trade-off. To determine whether MBB-BC is an effective optimizer, a comparative analysis is conducted between MBB-BC, BB-BC, PSO, and DE. It is shown that MBB-BC performs better than PSO and traditional BB-BC optimization, across a host of popular benchmark functions.**

***Keywords*—Particle swarm optimization, differential evolution, big bang-big crunch optimization, meta-heuristic algorithm.**

I. Introduction

The term *meta-heuristic*, coined by Fred Glover in 1986 [1], refers to a stochastic optimization procedure predicated upon nature inspired behaviour. All meta-heuristic algorithms propose a trade-off between randomization, and local search [2]. Meta-heuristic based optimization is an iterative process involving a population of entities which improve in *quality* over time. In this context, quality is assessed by means of an objective function evaluation.

Big bang-big crunch optimization is a meta-heuristic that models the life cycle of the universe. As the name suggests, the algorithm has two distinct phases: the big-bang phase, and the big-crunch phase [3]. The big-bang involves stochastic entity dispensation, analogous to the prevailing cosmological model explaining the birth of the universe. Thereafter begins the big-crunch phase. The big-crunch is a convergence operator in which entities coalesce to a singularity, a hypothetical scenario describing the ultimate fate of the universe.

This paper provides a comparison between four meta-heuristics: global best particle swarm optimization (PSO), differential evolution (DE), big bang-big crunch optimization (BB-BC), and a novel adaptation of big bang- big crunch (MBB-BC) that favours exploration. The intended outcome of this analysis is to determine, on average, if big-bang big-crunch outperforms classic meta-heuristic algorithms. The performance of each algorithm is evaluated across a series of benchmark functions. It was determined that big bang-big crunch optimization was competitive, but not superior to differential evolution.

The remainder of this paper is organized as follows: Section II supplies an overview of big-bang big-crunch optimization. Section III outlines the comparative algorithms used in the analysis. Section IV presents the comparison methodology used to conduct the analysis. Section V discloses the benchmark functions and experimental procedure followed to obtain results. Subsequently, the results obtained are presented in Section VI. Lastly, section VII concludes the paper.

II. Big bang-big crunch optimization

Big bang-big crunch optimization (BB-BC) is a meta-heuristic algorithm inspired by the life cycle of the universe. The BB-BC algorithm was developed by Erol and Eksin [3] in 2006.

A. Inspiration

From a cosmological perspective, the big bang marks the birth of the universe. The big bang involves a rapid expansion of matter emanating from a single point. The nature of the expansion is inherently disorderly and random. It is theorized that eventually the density of matter throughout the universe will become sufficiently high that the gravitational attraction between particles will overcome the force of expansion. Entities dispersed throughout the universe mutually attract one another, until the universe violently collapses into a singularity. The relative position of the singularity depends on the distribution of mass at the start of the big crunch phase, as mass is directly related to gravitational attraction.

B. Algorithmic phases

There are two primary stages of the big bang-big crunch optimization algorithm; the big bang phase, and the big crunch phase. The big bang phase of the algorithm is responsible for introducing diversity into the optimization procedure, facilitating adequate *exploration*. Exploration refers to the ability of the model to scout the search space and locate a fruitful region in which an optimum might be found [4]. After the big bang phase of the algorithm, a big crunch phase is initiated to ensure solution *exploitation*. Exploitation is the ability of the algorithm to focus on a small region of the search space to refine a promising solution. The phases of the big bang-big crunch optimization procedure are chronologically disclosed in the proceeding subsections.

1) Initializing the model: To initialize the model, entities are dispersed uniformly throughout the search domain [3]. Although it has been shown that uniformly initializing meta-heuristic populations does not always produce optimal performance [5], it is suggested by the proposing literature that uniform distribution along the search domain introduces acceptable initial diversity. The position of an entity in the search domain represents a candidate solution. The quality of candidate solutions are iteratively improved as the primary phases of the algorithm are executed.

2) The big crunch phase: The big crunch phase is a convergence operator responsible for facilitating improvement in the quality of candidate solutions. A mass is associated with each entity in the population. In the context of objective function minimization, the magnitude of mass is inversely related to objective function evaluation. An inverse relationship between objective function evaluation and mass implies that entities located at near-optimal points in the search space possess significant mass. In the big crunch phase, entities unite at the position corresponding to the collective center of mass. The center of mass equation is given by:

$$\mathbf{x}_c = \frac{\sum_{i=1}^{n} \frac{1}{f_i}\mathbf{x}_i}{\sum_{i=1}^{n} \frac{1}{f_i}} \tag{1}$$

The above notation is defined as follows:

- $\mathbf{x}_c$ denotes the position vector associated with the center of mass of all entities in the search space.
- $\mathbf{x}_i$ is the position of entity i in the search space.
- f_i is the objective function evaluation of entity x_i, the inverse of mass.
- n is the number of entities encompassing the population.

The solution represented by the center of mass centroid $\mathbf{x}_c$ at the end of the optimization procedure represents the optimum identified by the algorithm.

3) The big bang phase: The big bang phase is characterized by the expansion of entities away from the representative centroid. Expansion is a requisite component of entity exploration. The explosive force driving the expansion is inversely related to the optimization iteration. For early iterations, entities are thrust far from the computed center of mass. For later iterations, entities only explore locally. The dependency between distribution force and iterational progression is exhibited in the equation:

$$\mathbf{x}_{new} = \mathbf{x}_c + \frac{l}{k} r \tag{2}$$

The variables in the above equation are defined as:

- $\mathbf{x}_{new}$ is the updated position of an entity, displaced from the centroid.
- l is the upper-bound of the objective function domain. An entity should not be displaced beyond the boundary of the search domain. The displacement function is therefore a function of this boundary.
- r represents a random number sampled from the standard normal distribution, a Gaussian distribution with a mean of zero, and a standard deviation of one. The literature suggests that Gaussian sampling yields favorable performance [3].
- k corresponds to the active iteration of the optimization procedure. It is important to note that the ratio, $\frac{l}{k}$, facilitates quick entity convergence as large values of k scale this term close to zero. If $\frac{l}{k}$ is near zero, entities can no longer be expelled from the centroid, and the optimization process is concluded.

The BB-BC optimization algorithm as proposed by Erol and Eksin has two unique characteristics that make it an attractive algorithm for many applications. There are no control parameters that influence the performance of the BB-BC algorithm. As a result, the algorithm does not need to undergo the computationally expensive task of control parameter tuning. The BB-BC algorithm also converges very quickly. The explosive force factor $\frac{l}{k}$ from equation (2) approaches zero very rapidly in many optimization applications, especially those with a small problem domain. The rapid convergence is not always inherently desirable. Olorunda and Engelbrecht [6] have illustrated the adverse effect of premature convergence in meta-heuristic optimization. The Modified big bang-big crunch optimization technique proposed in this paper is designed to prolong the explorational tendency of BB-BC optimization via an adjustment to equation (2).

C. The big bang-big crunch algorithm

Pseudocode for BB-BC optimization algorithm proposed by Erol and Eksin [3] is provided in Algorithm 1:

D. Modified big bang-big crunch optimization

Standard big bang-big crunch optimization converges to a solution very quickly. In some optimization scenarios, premature convergence leads to the exploitation of low quality optima [7]. To slow the convergence rate of the algorithm, the position calculation equation used during the big bang phase is updated as follows:

$$\mathbf{x}_{new} = \mathbf{x}_c + rF_kD \quad , \qquad \text{where} \quad F_k = \frac{k^x - k}{1-k} \tag{3}$$

The notation of this equation is summarized as:

- $\mathbf{x}_{new}$ is the updated position of an entity, displaced from the centroid.
- r represents a random number sampled from the standard normal distribution.

Algorithm 1 BB-BC Algorithm

```
$Initialize$
$r$ : Normal number,
$N$ : Population size,
$UB$ : Upper bound,
$Itermax$ : Maximum number of iterations
Generate population $\mathbf{X}_i$ of size $n$ with respect to the defined limits
while $Iteration \leq Itermax$ do
    for candidate $\mathbf{X}_i$ in population $P$ do
        $f_i \leftarrow$ Fitness of $\mathbf{X}_i$
    end for
    Calculate the center of mass, $\mathbf{x}_c = \frac{\sum_{i=1}^{n} \frac{1}{f_i}\mathbf{X}_i}{\sum_{1}^{n} \frac{i=1}{f_i}}$
    for Candidate $\mathbf{x}_i$ in population $P$ do
        Determine $\mathbf{x}_i^{new}$ as a function of $\mathbf{x}_i$ using (2) or (3)
    end for
    $iteration \leftarrow iteration + 1$
end while
```

- F_k is a scaling proportion, used to govern the convergence speed of the optimization algorithm. F_k is a monotone decreasing function on the interval [0,1] continuous on its domain, for all choices $k \in (0, 1)$.
- k is a rate parameter used to control the negative slope of the convergence curve F_k. It was determined empirically that $k = 0.001$ offers a fair trade-off between exploration and exploitation. A simulation procedure of independent trials was used to determine a suitable value for k.
- D corresponds to the domain width of the fitness function. It is the absolute difference between the maximum and minimum value that a decision variable can take on.
- x represents the iteration proportion. For early iterations, x is associated with a value close to one. When the current iteration approaches the stopping condition, x is associated with a value near one.

It is desirable that meta-heuristic algorithms provide a varying substitution rate between exploration and exploitation during the optimization procedure [2]. In early phases of the optimization process, it is desirable that entities roam the search domain, only optimizing promising solutions towards the end of the simulation. In the context of BB-BC, it is intended that the explosive magnitude of the big bang should be large during early iterations to guarantee that entities are thrust throughout the search domain. During later stages of optimization, the magnitude should decrease. The curve F_k governs the rate at which exploration is substituted for exploitation. A family of convergence curves is shown in figure 1. The family of curves demonstrate the inverse relationship between iteration proportion, and exploitation.

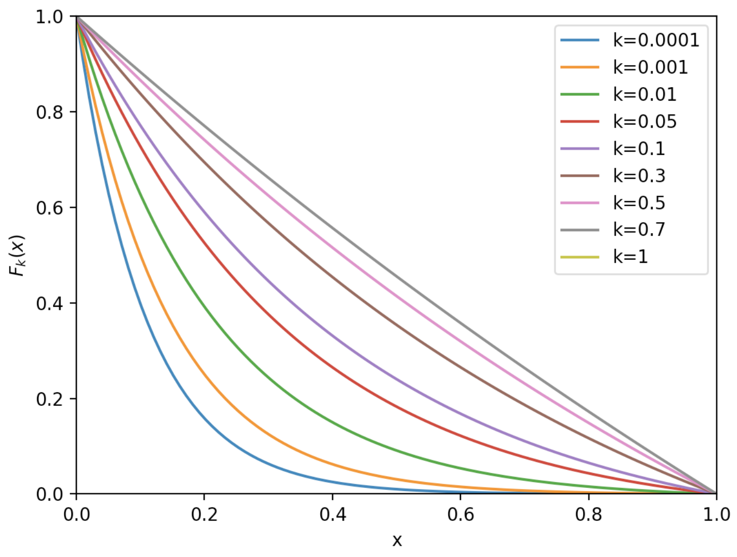

Fig. 1: A family of convergence curves F_k plotted for various values k. Smaller values k correspond to higher convexity in the curve, associated with rapid convergence.

An important distinction between modified BB-BC and the initial algorithm presented by Erol and Eskin is the introduction of a control parameter, k. As presented, the choice for k influences the performance of the algorithm, and should be tuned accordingly to perform optimally. It was however determined that in many optimization scenarios, $k = 0.001$ performs favorably.

III. Comparative algorithms

To evaluate the merit of big bang-big crunch optimization, a comparison is made between BB-BC and other nature inspired meta-heuristic algorithms. The algorithms participating in the comparison are differential evolution, and inertia-weighted particle swarm optimization.

A. Differential evolution

Differential evolution (DE) is a stochastic, population based search algorithm developed by Storn and Price in 1995 [8]. Differential evolution is a hill-climb search algorithm that performs a *mutation* and *crossover* operation on selected population members to generate a trial vector that is subsequently compared to the chosen population member. If the fitness of the trial vector exceeds the fitness of the population member, the population member is replaced.

1) Mutation: The DE mutation operation produces a trial vector for each individual of the population by mutating a selected target vector with a weighted difference vector. For each vector $\mathbf{x}_i(t)$, generate a trial vector $\mathbf{u}_i(t)$ using [8]:

$$\mathbf{u}_i(t+1) = \mathbf{x}_{i_1}(t) + F(\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t)) \quad (4)$$

The notation of (4) is clarified as:

- $\mathbf{x}_{i_1}(t)$, $\mathbf{x}_{i_2}(t)$ and $\mathbf{x}_{i_3}(t)$ are three selected population members sampled such that $i \neq i_1 \neq i_2 \neq i_3$
- F is a real constant factor $\in [0,2]$ which controls the amplification of the differential variation term $(\mathbf{x}_{i_2}(t) - \mathbf{x}_{i_3}(t))$.
- t is a generation coefficient corresponding to the active iteration number.

The approached used in this paper to perform DE mutation follows the DE/Rand/1 scheme proposed by Storn and Price [8].

2) Crossover: Crossover constructs a trial vector by combining components of a selected vector and the mutant vector obtained from (4) [9]. The crossover operation requires a control parameter $Cr \in [0,1]$, which governs the intensity of solution variation between successive generations. Crossover involves transfiguring values in the selected vector in attempt to improve the evaluated fitness of the selected vector. A random integer $n \in [0,D]$ is selected. The crossover procedure begins by transforming the selected vector at index n, iteratively mutating each succeeding index. If the trial vector is successfully transfigured, it replaces the selected solution in the population, otherwise the trial vector is abandoned.

3) The DE algorithm: The differential evolution algorithm is provided in Algorithm 2.

Algorithm 2 DE Algorithm

```
1: $F$ and $Cr$                ▷ These are tuned control parameters
2: $t \leftarrow 0$, set the generation number to zero
3: $D \leftarrow$ the dimension of the problem space
4: $P(0) \leftarrow$ initialized population of $n_s$ individuals
5: while $t < maxGenerations$ do
6:     for each individual, $\mathbf{x}_i(t) \in P(t)$ do
7:         Select $r_1, r_2, r_3, \in [0,D]$ such that $r_1 \neq r_2 \neq r_3$
8:         $\mathbf{x}_i^{temp}(t) \leftarrow \mathbf{x}_i(t)$, save population member
9:         $n \leftarrow$ random integer $\in [0,D]$ for dimension D
10:        $L \leftarrow 0$
11:        repeat
12:            $\mathbf{x}_{i,n}^{temp}(t) \leftarrow \mathbf{x}_{r_1,n}(t) + F[\mathbf{x}_{r_2,n}(t) - \mathbf{x}_{r_3,n}(t)]$
13:            $n \leftarrow (n+1)\ mod\ D$
14:            $L \leftarrow L + 1$
15:            $r \leftarrow$ random number $\in [0,1]$
16:        until $r < Cr$ and $L \leq D$
17:        if $f(\mathbf{x}_i^{temp}) \leq f(\mathbf{x}_i)$ then
18:            $\mathbf{x}_i(t) \leftarrow \mathbf{x}_i^{temp}(t)$
19:        else
20:            $\mathbf{x}_i(t)$ remains unchanged, abandon $\mathbf{x}_i^{temp}(t)$
21:        end if
22:    end for
23: end while
```

B. Particle swarm optimization

Particle swarm optimization (PSO), as established by Kennedy and Eberhart in 1995 [10], is a nature-inspired stochastic optimization algorithm. In PSO, a population of particles traverse a search space guided by both global and local topology. The position of a particle during this procedure represents a candidate solution to a predetermined optimization problem. The quality of a particular solution is assessed by means of an objective function evaluation. The movement characteristics of particles greatly depend on the values of key model parameters.

1) PSO velocity and position update: Various PSO techniques have been developed since the initial contributions of Kennedy and Eberhart. This paper uses the inertia weight model of PSO, developed by Shi and Eberhart [11]. Intertia weight PSO establishes a proportional relationship between a particle's prior velocity and its successive velocity. According to this model, the equations to update the velocity and position of a particle are given by equations (1) and (2):

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

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + c_1\mathbf{r}_1(\mathbf{y}_i^t - \mathbf{x}_i^t) + c_2\mathbf{r}_2(\hat{\mathbf{y}}_i^t - \mathbf{x}_i^t) \tag{6}$$

Bold symbols in the above equations represent n dimensional vectors, where n corresponds to the dimensionality of the search domain. The syllabary of equations (1) and (2) are clarified below:

- $\mathbf{x}_i^t$ and $\mathbf{x}_i^{t+1}$ specify the position of the i^{th} particle during iteration t and $t+1$ respectively. The position of a particle represents a potential solution to the optimization problem.
- $\mathbf{v}_i^t$ and $\mathbf{v}_i^{t+1}$ denote the velocity of the i^{th} particle during iteration t and $t+1$ respectively. It is important to note that the velocity vector of a particle specifies both the direction this particle is moving, as well as the magnitude of the movement. The velocity of all particles are initialized to the zero vector at iteration one.
- $\mathbf{y}_i^t$ represents the position which yields the best objective function evaluation, that has been visited by particle i. This point is commonly referred to as a particle's personal best vector.
- $\hat{\mathbf{y}}_i^t$ denotes the position which yields the best objective function evaluation that any particle in the neighborhood has visited. In this paper the neighborhood corresponds to all particles within the swarm.
- $\mathbf{r}_1$ and $\mathbf{r}_2$ are n dimensional random vectors, where each element in the vector is chosen from a *uniform(0,1)* distribution.
- c_1 is a control coefficient for the term $\mathbf{r}_1(\mathbf{y}_i^t - \mathbf{x}_i^t)$. Collectively this term is referred to as the "cognitive component", which quantifies performance of the particles current position relative to that of the particles personal best. This interaction has aptly been coined "nostalgia" by Kennedy and Eberhart [10].
- c_2 is the control coefficient for the term $\mathbf{r}_2(\hat{\mathbf{y}}_i^t - \mathbf{x}_i^t)$. Collectively, this term is referred to as the "social component", which quantifies performance of the particle's current position relative to that of the particle's neighborhood best. This interaction has similarly been referred to as "envy" by Kennedy and Eberhart [10].

- w regulates the affect that the previous velocity has on the calculation of the velocity in the subsequent iteration. The velocity of the previous iteration is often regarded as the "inertia" or "momentum" of that particle, hence the name for this PSO model - Interia Weight PSO.

C. Particle convergence condition

If the control parameters w, c_1, c_2 are chosen to satisfy:

$$c_1 + c_2 < \frac{24(1-w^2)}{7-5w} \quad \text{for } \; w \in [-1, 1] \tag{7}$$

it has been both theoretically and empirically demonstrated that particles will eventually reach equilibrium [12] [13] [14] [15].

1) The PSO algorithm: The Particle swarm optimization algorithm is provided in Algorithm 3.

Algorithm 3 Standard PSO Algorithm

```
1: Create and initialize an $n_x$-dimensional swarm, S;
2: repeat
3:     for each particle i=1,...,$S.n_s$ do
4:         if $f(S.x_i) < f(S.y_i)$ then
5:             $S.y_i = S.x_i$;
6:         end if
7:         if $f(S.y_i) < f(S.\hat{y_i})$ then
8:             $S.\hat{y_i} = S.y_i$;
9:         end if
10:    end for
11:    for each particle i=1,...,$S.n_s$ do
12:        update the velocity;
13:        update the position;
14:    end for
15: until stopping condition is true
```

IV. Comparison methodology

To investigate the performance quality of PSO, DE, BB-BC, and MBB-BC, multiple independent simulations are conducted under unique optimization contexts. The performance of each meta-heuristic is evaluated using multiple benchmark functions of differing dimensionality, across multiple independent trials. Malan and Engelbrecht [16] demonstrated that consistent population size is prerequisite to a tenable comparative analysis between algorithms. In light of this finding, the performance of each algorithm is not compared to another unless the population size is constant between algorithms. The performance of each meta-heuristic algorithm is evaluated using the following metrics:

- The *quality* of the obtained solution. Solution quality is directly proportional to proximity from the global optimum.
- The variation in performance between independent simulations. It is desirable that high-quality points in the solution space be frequently exploited.
- The exploration, exploitation trade-off. As Blum and Roli [17] demonstrated, meta-heuristics that display initial exploration, moving towards exploitation is a feature of successful optimization.

V. Empirical Procedure

A set of nine well-known benchmark functions [18] are used to evaluate the performance of the meta-heuristic algorithms. The selected benchmark problems are both uni-modal and multi-modal. Scalable functions are tested using both 10 dimensions, and 30 dimensions. All the benchmark functions were shifted such that the global optimum evaluation is zero. In the following functions, D denotes the dimension of the search space.

A. Benchmark functions

1) Elliptic Function:

$$f(\mathbf{x}) = \sum_{n=1}^{D} (10^6)^{\frac{i-1}{d-1}} (\mathbf{x_i}^2) \quad where \; -10 \le x_i \le 10$$

2) Step Function:

$$f(\mathbf{x}) = \sum_{n=1}^{D} (\lfloor x_i^2 \rfloor) \quad where \; -100 \le x_i \le 100$$

3) Cosine-Mixture Function:

$$f(\mathbf{x}) = 0.1 \sum_{n=1}^{D} \cos(5\pi x_i) - \sum_{n=1}^{D} x_i^2 \quad where \; -1 \le x_i \le 1$$

4) Ackley Function:

$$f(\mathbf{x}) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{d} x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) + 20 + e, \quad where \; -32.768 \le x_i \le 32.768$$

5) Rastrigin Function:

$$f(\mathbf{x}) = 10D + \sum_{i=1}^{D} \left(x_i^2 - 10cos(2\pi x_i)\right) \quad where \; -5.12 \le x_i \le 5.12$$

6) Bent-Cigar Function:

$$f(\mathbf{x}) = x_1^2 + 10^6 \sum_{i=2}^{D} x_i^2 \quad where \; -100 \le x_i \le 100$$

7) Salomon Function:

$$f(\mathbf{x}) = 1 - cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2} \quad where \; -100 \le x_i \le 100$$

8) Quartic Function:

$$f(\mathbf{x}) = \sum_{i=1}^{D} i x_i^4 \quad where \; -1.28 \le x_i \le 1.28$$

9) Mishra$_1$ Function:

$$f(\mathbf{x}) = (1+g_n)^{g_n} - 2,\ for$$
$$g_n = D - \sum_{i=1}^{D-1} x_i \quad where\ \ 0 \leq x_i \leq 1$$

B. Simulation parameters

It was empirically determined through a tuning procedure that the following parameter configurations are near-optimal for PSO and DE. Note, BB-BC does not contain control parameters that require tuning.

TABLE I: Tuned control parameter configurations

Function	**PSO** w, c_1, c_2	**DE** F, Cr
Elliptic	$0.53,\ 2.0\ 1.87$	$0.63, 0.73$
Step	$0.66,\ 2.0\ 1.47$	$0.37, 0.60$
Cosine-Mixture	$0.53,\ 2.0\ 1.87$	$0.27, 0.33$
Ackley	$0.73,\ 1.73\ 1.33$	$0.30, 0.47$
Rastrigin	$0.53,\ 2.0\ 1.87$	$0.33, 0.60$
Bent-Cigar	$0.53,\ 2.0\ 1.87$	$0.30, 0.60$
Salomon	$0.53,\ 2.0\ 1.87$	$0.20, 0.40$
Quartic	$0.66,\ 1.87\ 1.47$	$0.27, 0.47$
Mishra	$0.60,\ 1.87\ 1.73$	$0.47, 0.87$

C. Independent variables

The performance of PSO, DE, BB-BC, and MBB-BC are evaluated under the following optimization configurations:

TABLE II: Optimization configurations

Dimension	Population size
10	30
30	30

For every configuration specified in the table II, a series of optimization simulations is performed using PSO, DE, BB-BC, and MBB-BC. Each meta-heuristic optimization algorithm is tested over 20 independent trials of 5,000 iterations each. The performance across all 20 trials is averaged, to reduce stochastic noise. The average variance between trials is also recorded. The results presented and discussed in the subsequent section.

D. Simulation tournament

A tournament is conducted for the configurations specified in the table II. A ranking is assigned to each meta-heuristic algorithm based on the optimization quality. A ranking of 1 is considered best, whilst 4 is considered the worst. In the event of a tie, equal rankings are given to all algorithms participating in the tie. The best algorithm is the one with the lowest cumulative ranking.

VI. Simulation Results

This section presents empirical results and observations regarding the comparison between the meta-heuristic algorithms.

A. 10-dimensional optimization using a population size of 30

1) Tournament results: The performance of each algorithm is tabulated as follows:

TABLE III: Meta-heuristics performance for $D = 10,\ N = 30$

Heuristic	**Function**	**Avg. Minimum**	**rank**	**Avg. Variance**	**rank**
DE	*Ellip.*	$4.19e^{-85}$	1	$2.15e^{-168}$	1
	Step.	$0.00e^{+00}$	1	$0.00e^{+00}$	1
	Mix.	$0.00e^{+00}$	1	$0.00e^{+00}$	1
	Ack.	$3.82e^{-15}$	1	$6.31e^{-31}$	1
	Rast.	$4.97e^{-02}$	1	$4.95e^{-02}$	1
	Bent.	$1.18e^{-126}$	1	$5.39e^{-252}$	1
	Sal	$1.15e^{-01}$	3	$1.34e^{-03}$	2
	Quar.	$1.25e^{-251}$	1	$0.00e^{+00}$	1
	Mish	$4.44e^{-15}$	2	$0.00e^{+00}$	1
PSO	*Ellip.*	$4.00e^{-27}$	2	$3.19e^{-52}$	2
	Step.	$0.00e^{+00}$	1	$0.00e^{+00}$	1
	Mix.	$1.48e^{-02}$	4	$2.07e^{-03}$	3
	Ack.	$2.77e^{-01}$	3	$3.44e^{-01}$	4
	Rast.	$7.22e^{+00}$	4	$2.48e^{+01}$	3
	Bent.	$2.54e^{-47}$	2	$6.32e^{-93}$	2
	Sal	$4.65e^{-01}$	1	$9.08e^{-02}$	3
	Quar.	$2.27e^{-09}$	4	$1.03e^{-16}$	4
	Mish	$6.60e^{-06}$	3	$6.25e^{-10}$	3
BB-BC	*Ellip.*	$1.22e^{-06}$	4	$1.29e^{-13}$	4
	Step.	$0.00e^{+00}$	1	$0.00e^{+00}$	1
	Mix.	$9.61e^{-02}$	2	$2.13e^{-02}$	4
	Ack.	$1.58e^{+00}$	4	$7.71e^{-01}$	3
	Rast.	$6.67e^{+00}$	2	$2.54e^{+01}$	4
	Bent.	$1.76e^{+02}$	3	$1.05e^{+04}$	3
	Sal	$2.63e^{+00}$	4	$9.51e^{-01}$	4
	Quar.	$5.18e^{-16}$	3	$1.03e^{-31}$	3
	Mish	$4.59e^{-01}$	4	$6.42e^{-01}$	4
MBB-BC	*Ellip.*	$7.58e^{-08}$	3	$1.88e^{-15}$	3
	Step.	$0.00e^{+00}$	1	$0.00e^{+00}$	1
	Mix.	$3.69e^{-02}$	3	$6.61e^{-03}$	2
	Ack.	$2.58e^{-04}$	2	$6.35e^{-09}$	2
	Rast.	$6.96e^{+00}$	3	$1.56e^{+01}$	2
	Bent.	$1.53e^{+03}$	4	$2.43e^{+06}$	4
	Sal	$2.05e^{-01}$	2	$2.61e^{-03}$	1
	Quar.	$3.76e^{-19}$	2	$2.12e^{-37}$	2
	Mish	$0.00e^{+00}$	1	$0.00e^{+00}$	1

In the simulation involving a population size of 30 individuals and a 10-dimensional search space , the following totals were obtained:

1) Differential evolution achieved a cumulative ranking of 22.
2) Modified big bang-big crunch achieved a cumulative ranking of 39.
3) Particle swarm optimization achieved a cumulative ranking of 49.
4) Big bang-big crunch achieved a cumulative ranking of 57.

These results suggest that differential evolution is the most successful meta-heuristic algorithm under these conditions.

2) Exploration vs. Exploitation balance: It is important that meta-heuristic algorithms explore the search domain in the early stages of the optimization procedure, only exploiting identified minima during later stages of the simulation. See figure 2.

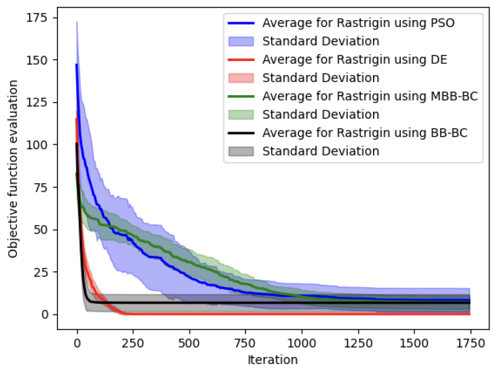

Fig. 2: This figure illustrates the explorative tendency of the PSO and MBB-BC meta-heuristics. This pattern was evident in the following benchmark functions: Rastrigin, Ackley, Salomon, and Cosine-Mixture.

In some cases BB-BC yielded significantly worse solutions than the other algorithms, likely due to this algorithm's rapid convergence. The obtained solution in such a scenario is highly dependent on the position to which the population members were initialized. The explosive magnitude responsible for facilitating exploration in BB-BC approaches zero too quickly, leading to high variation in average solution quality. The BB-BC is the only algorithm that displayed this behaviour. This premature convergence is extremely evident when plotted on the same graph as MBB-BC, an algorithm specifically designed to avoid this behaviour, as evident by figure 3.

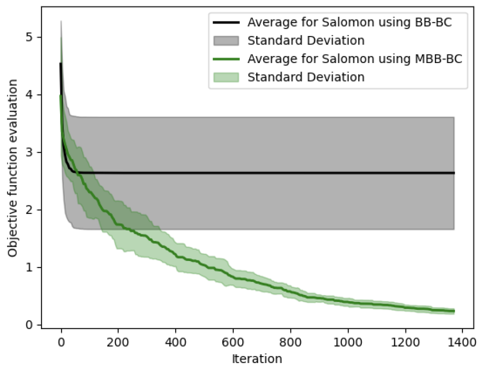

Fig. 3: This figure illustrates the premature exploitation causing BB-BC to obtain low quality solutions, with extremely high variation between trials. This behaviour is also exhibited for the Ackley and Cosine-Mixture functions.

Early convergence did not always lead to poor solution exploitation. In the Step, Elliptic, Quartic, and Bent-Cigar benchmark functions, BB-BC located near-optimal solutions much faster than MBB-BC. See figure 4 and 5.

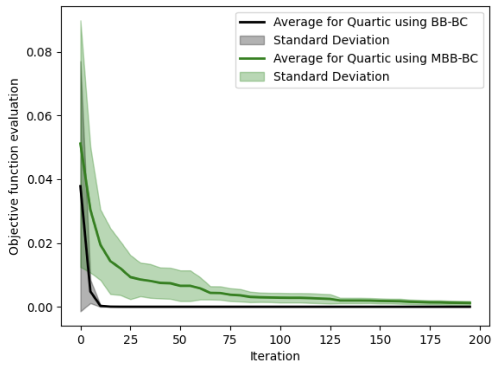

Fig. 4: BB-BC successfully exploiting near-optimal solutions faster than MBB-BC. This behaviour was displayed in the Step, Elliptic, Quartic, and Bent-Cigar benchmark functions.

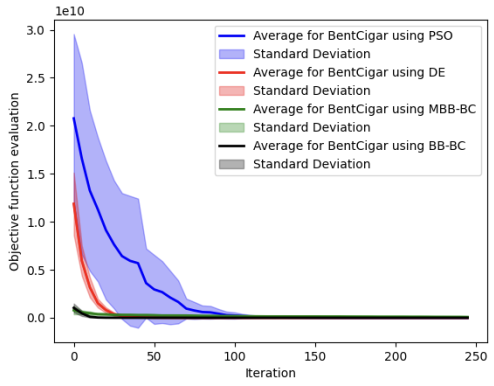

Fig. 5: BB-BC successfully exploiting near-optimal solutions faster than DE, PSO, and MBB-BC, this behaviour was displayed in the Step, Elliptic, Quartic, and Bent-Cigar benchmark functions.

In all optimization scenarios explored in this experiment, PSO demonstrated the strongest explorative tendency. The average variation in obtained solution between independent trials of PSO is very high during the early stages, but approaches zero during later iterations. In figure 6, all 20 independent trials of PSO is plotted on a shared axis. The variance between trials in early iterations, compared to late iterations, is notable.

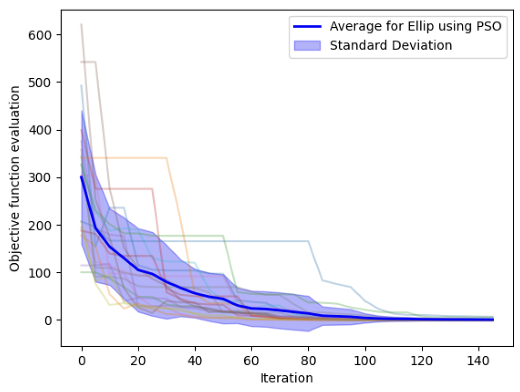

Fig. 6: All independent trials of PSO on the elliptic benchmark function. Each independent trial is plotted on a semi-translucent curve, the aggregation of all trials is plotted in blue. This transition from exploration to exploitation is demonstrated by PSO in every benchmark function.

Overall, for 10-dimensional optimization with a population size of 30, it was determined that differential evolution is the best meta-heuristic optimization algorithm. On average, differential evolution exploited the highest quality solutions, most frequently. In some cases BB-BC exploited solutions of equal quality in fewer iterations, but the extreme variation of BB-BC between benchmark functions indicate that this algorithm cannot easily be trusted to produce desirable solutions across many optimization problems. PSO and MBB-BC produced solutions of similar quality (MBB-BC was slightly better), though PSO demonstrated the best exploration-exploitation trade-off.

B. *30-dimensional optimization using a population size of 30*

1) Tournament results: The performance of each algorithm is tabulated as follows:

TABLE IV: Meta-heuristics performance for $D = 30,\ N = 30$

Heuristic	Function	Avg. Minimum	rank	Avg. Variance	rank
DE	$Ellip.$	$9.11e^{-41}$	1	$6.97e^{-81}$	1
	$Step.$	$0.00e^{+00}$	1	$0.00e^{+00}$	1
	$Mix.$	$0.00e^{+00}$	1	$0.00e^{+00}$	1
	$Ack.$	$7.37e^{-15}$	1	$6.31e^{-31}$	1
	$Rast.$	$1.17e^{-03}$	1	$2.75e^{-05}$	1
	$Bent.$	$1.34e^{-26}$	1	$3.59e^{-51}$	1
	Sal	$2.90e^{-01}$	2	$7.26e^{-03}$	2
	$Quar.$	$1.34e^{-124}$	1	$2.85e^{-247}$	1
	$Mish$	$8.86e^{-15}$	1	$3.56e^{-30}$	1
PSO	$Ellip.$	$1.17e^{-06}$	2	$1.46e^{-11}$	3
	$Step.$	$1.50e^{-01}$	4	$4.50e^{-01}$	4
	$Mix.$	$2.88e^{-01}$	2	$4.02e^{-02}$	4
	$Ack.$	$1.18e^{+00}$	3	$6.78e^{-01}$	2
	$Rast.$	$3.34e^{+01}$	4	$5.34e^{+02}$	4
	$Bent.$	$2.33e^{+04}$	4	$1.02e^{+10}$	4
	Sal	$1.79e^{+00}$	3	$1.55e^{+00}$	4
	$Quar.$	$8.30e^{-05}$	4	$6.11e^{-08}$	4
	$Mish$	$1.80e^{-01}$	3	$1.09e^{-01}$	3
BB-BC	$Ellip.$	$5.63e^{-03}$	4	$6.30e^{-04}$	4
	$Step.$	$0.00e^{+00}$	1	$0.00e^{+00}$	1
	$Mix.$	$1.73e^{-01}$	3	$1.93e^{-02}$	2
	$Ack.$	$1.65e^{+00}$	4	$6.37e^{-01}$	3
	$Rast.$	$1.42e^{+01}$	2	$6.36e^{+01}$	3
	$Bent.$	$7.11e^{+02}$	2	$1.78e^{+05}$	2
	Sal	$4.32e^{+00}$	4	$5.29e^{-01}$	3
	$Quar.$	$1.24e^{-14}$	3	$2.28e^{-29}$	3
	$Mish$	$4.98e^{+00}$	4	$2.41e^{+01}$	4
MBB-BC	$Ellip.$	$2.03e^{-05}$	3	$5.00e^{-10}$	2
	$Step.$	$0.00e^{+00}$	1	$0.00e^{+00}$	1
	$Mix.$	$1.48e^{-01}$	4	$1.61e^{-02}$	3
	$Ack.$	$2.79e^{-01}$	2	$3.43e^{-01}$	4
	$Rast.$	$2.11e^{+01}$	3	$3.59e^{+01}$	2
	$Bent.$	$1.20e^{+03}$	3	$2.41e^{+06}$	3
	Sal	$5.30e^{-01}$	1	$9.58e^{-03}$	1
	$Quar.$	$1.02e^{-16}$	2	$1.98e^{-32}$	2
	$Mish$	$1.03e^{-03}$	2	$7.64e^{-07}$	2

In the simulation involving a population size of 30 individuals and a 30-dimensional search space , the following totals were obtained:

1) Differential evolution achieved a cumulative ranking of 20.
2) Modified big bang-big crunch achieved a cumulative ranking of 63.
3) Particle swarm optimization achieved a cumulative ranking of 52.
4) Big bang-big crunch achieved a cumulative ranking of 44.

These results suggest that differential evolution is the most successful meta-heuristic algorithm under these conditions.

C. Comparison between 30-dimensional performance, and 10-dimensional performance

Many of the performance characteristics exhibited when optimizing 10-dimensional benchmark problems were once again encountered during 30-dimensional optimization. BB-BC still suffered from premature convergence leading to the exploitation of sub-par solutions. See figure 7.

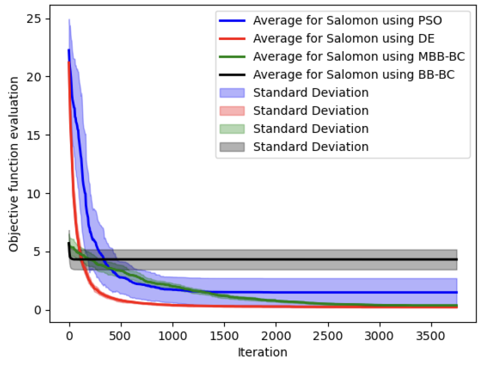

Fig. 7: BB-BC displaying premature convergence, causing the exploitation of low-quality candidate solutions, when compared to other algorithms. BB-BC exhibited poor performance on the Elliptic, Ackley, Salomon, and Mishra$_1$ benchmark functions.

Similar optimization ability between PSO and MBB-BC was once again exhibited. Figure 8 suggests that in many situations PSO and MBB-BC eventually exploit solutions of similar quality, though PSO does so with larger variance in early iterations, on average.

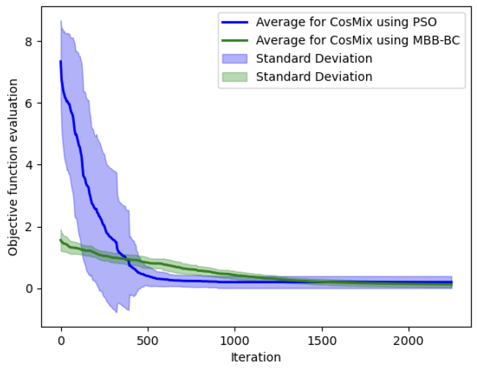

Fig. 8: Similar performance between PSO and MBB-BC was exhibited in the Elliptic, Cosine-Mixture, and Rastrigin benchmark functions.

Interestingly however, PSO failed to identify a suitable solution for the 30-dimensional Ackley benchmark problem. Differential evolution, and MBB-BC did not struggle to identify near-optimal candidate solutions for the Ackley benchmark function. See figure 9.

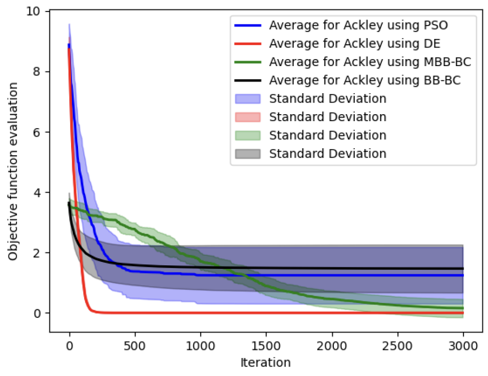

Fig. 9: PSO fails to exploit a near-optimal candidate solution for the 30-dimension Ackley benchmark function.

Overall, differential evolution demonstrated superior optimization ability across the benchmark functions used in this experiment, obtaining a final ranking of 42. Modified big bang-big crunch performed adequately, demonstrating decent exploitative tendencies, whilst still converging to favorable minima. MBB-BC obtained a final ranking of 83. Particle swarm optimization and big bang-big crunch obtained very similar final rankings of 112 and 109 respectively, though the exhibited optimization characteristics were very different. BB-BC often exploited prematurely, whilst PSO did not transfer to exploitation quickly enough.

VII. Conclusion

The intention of this paper was to investigate whether the big bang-big crunch, or the modified big bang-big crunch algorithms offer optimization abilities superior to particle swarm optimization and differential evolution. Evaluation metrics such as solution quality and inter-trial variance were established to asses the success of each meta-heuristic algorithm. The experiment conducted in this paper concluded that modified big bang-big crunch performed better than traditional big bang-big crunch and particle swarm optimization, but worse than differential evolution. Differential evolution exploited near-optimal candidate solutions the most frequently, with the lowest inter-trial variance across a host of diverse benchmark functions and search space dimensionality.

References

[1] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & operations research*, vol. 13, no. 5, pp. 533–549, 1986.

[2] X. Yang, "Metaheuristic Optimization," *Scholarpedia*, vol. 6, no. 8, p. 11472, 2011, revision #91488.

[3] O. K. Erol and I. Eksin, "A new optimization method: big bang–big crunch," *Advances in Engineering Software*, vol. 37, no. 2, pp. 106–111, 2006.

[4] E. T. Oldewage, A. P. Engelbrecht, and C. W. Cleghorn, "The merits of velocity clamping particle swarm optimisation in high dimensional spaces," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, pp. 1–8.

[5] Q. Li, S.-Y. Liu, and X.-S. Yang, "Influence of initialization on the performance of metaheuristic optimizers," *Applied Soft Computing*, vol. 91, p. 106193, 2020.

[6] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 2008, pp. 1128–1134.

[7] H. M. Pandey, A. Chaudhary, and D. Mehrotra, "A comparative review of approaches to prevent premature convergence in ga," *Applied Soft Computing*, vol. 24, pp. 1047–1077, 2014.

[8] R. Storn and K. Price, "Minimizing the real functions of the icec'96 contest by differential evolution," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996, pp. 842–844.

[9] D. Tayal, C. Gupta, and A. Jain, "A new crossover operator in differential evolution for numerical optimization," 10 2012.

[10] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[11] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*. IEEE, 1998, pp. 69–73.

[12] C. W. Cleghorn and A. P. Engelbrecht, "Particle swarm convergence: An empirical investigation," in *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014, pp. 2524–2530.

[13] R. Poli, "Mean and variance of the sampling distribution of particle swarm optimizers during stagnation," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 712–721, 2009.

[14] M. Xu and J. Gu, "Parameter selection for particle swarm optimization based on stochastic multi-objective optimization," in *2015 Chinese automation congress (CAC)*. IEEE, 2015, pp. 2074–2079.

[15] Q. Liu, "Order-2 stability analysis of particle swarm optimization," *Evolutionary computation*, vol. 23, no. 2, pp. 187–216, 2015.

[16] K. M. Malan and A. P. Engelbrecht, "Algorithm comparisons and the significance of population size," in *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 2008, pp. 914–920.

[17] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, p. 268–308, sep 2003. [Online]. Available: https://doi.org/10.1145/937503.937505

[18] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.