Modified bat algorithm with quaternion representation

Iztok Fister, Janez Brest, Iztok Fister Jr.
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
Email: iztok.fister1@um.si

Xin-She Yang
Middlesex University
School of Science and Technology
London NW4 4BT, United Kingdom

***Abstract*—This paper introduces a modified bat algorithm using quaternion representation of individuals. Quaternions are a number system, which extends complex numbers. They are successfully applied to problems of theoretical physics and to those areas needing fast rotation calculations. We propose the application of quaternions in optimization, more precisely, we have been using quaternions for representation of individuals in bat algorithm belonging to a swarm intelligence family. It is expected that the problems with stagnation in swarm intelligence should be reduced or even eliminated. We believe that this representation could successfully be applied also to the other swarm intelligence and evolutionary algorithms.**

***Keywords*-quaternion; swarm intelligence; evolutionary computation; representation of individuals; bat algorithm**

I. Introduction

The last decades, scientists tend to build an automatic problem solver that would be able to solve different problems in computer science, mathematics, economy and engineering. They often borrow a solutions from nature. For instance, swarm intelligence (SI) is an artificial intelligence (AI) discipline concerned with the design of intelligent multi-agent systems that was inspired by the collective behavior of social insects like ants, termites, bees, and wasps, as well as from other animal societies like flocks of bird or shoals of fish [1]. Algorithms from this field have been applicable primarily for optimization problems and robotics. The more notable swarm intelligence disciplines are as follows: Ant Colony Optimization (ACO) [4], [16], Particle Swarm Optimization (PSO) [14], Artificial Bees Colony optimization (ABC) [13], [6], Firefly Algorithm (FA) [20], Cuckoo Search (CS) [24], Bat Algorithm (BA) [22], etc.

The representation of solution plays an important role in the performance and quality of the swarm intelligence algorithms. Therefore, the tasks of designing such algorithms are to find a proper representation for the problem and to develop appropriate search operators [18]. The proper representation needs to encode all possible solutions of the optimization problem. On the other hand, the appropriate search operator should be applicable to the proper representation. In general, no theoretical methods exist nowadays for describing the effects of representation on the performance. The proper representation of a specific problem mainly depends on the intuition of the algorithm's designer. Therefore, developing a new representation is often a result of the repeated 'trial-and-error' principle.

This paper proposes a new representation for individuals using quaternions that have at first time been used by Fister et al. in [7]. In mathematics, the quaternions extend complex numbers. Quaternion algebra is connected with special features of the geometry of the appropriate Euclidian spaces. The idea of quaternions occurred to William Rowan Hamilton in 1845 [11] whilst he was walking along the Royal Canal to a meeting of the Royal Irish Academy in Dublin.

Quaternions are especially appropriate within those areas where it is necessary to compose rotations with minimal computation, e.g., programming the video games or controllers of spacecraft [2]. For instance, 3-dimensional rotation can be specified by a single quaternion, whilst a pair of quaternions are needed for 4-dimensional rotation. The quaternion calculus are introduced in several physical applications, like: crystallography, the kinematics of rigid body motion, the Thomas precession, the special theory of relativity, and classical electromagnetism [10]. A step forward in the popularization of quaternions was achieved by Joachim Lambek in 1995 [17], who stated that quaternions can provide a shortcut for pure mathematicians who wish to familiarize themselves with certain aspects of theoretical physics.

In this study, the representation of quaternions was tested in a Bat algorithm [22] that obtains good results when optimizing the low-dimensional problems, but become worse when optimizing high-dimensional problems. The experiments showed that the results of the Bat algorithm could be improved when the same algorithm was hybridized using the representation of individuals with quaternions (QBA). Moreover, comparing the results of the QBA algorithm with the results of other algorithms, like PSO, DE, and ABC showed that these are also comparable with the other algorithms in experiments.

The rest of this paper is as follows: In Section 2, quaternion algebra is presented in detail. Section 3, firstly, describes the Bat algorithm and then the modified QBA algo-

978-1-4799-7492-4/15/$31.00

rithm when using representation with quaternions. Section 4 focuses on the experiments and results. This paper concludes by summarizing our preliminary work on representation with quaternions, and the direction for further work is pointed-out.

II. Quaternions

Quaternions are formal expressions $q = x_0 + x_1 i + x_2 j + x_3 k$, where x_0, x_1, x_2, x_3 are real values and they constitute the algebra over the reals generated by basic units i, j, k (also the imaginary part) that satisfy Hamilton's celebrated equations:

$$\begin{array}{lll} ij = k, & jk = i, & ki = j, \\ ji = -k, & kj = -i, & ik = -j, \\ & i^2 = j^2 = k^2 = -1. & \end{array} \quad (1)$$

The quaternions $q \in \mathbf{H}$ describes a 4-dimensional space $\mathbb{R}^4$ over the real numbers. Using this notation, a pair of quaternions is denoted as $q_0 = x_0 + x_1 i + x_2 j + x_3 k$ and $q_1 = y_0 + y_1 i + y_2 j + y_3 k$. The quaternion algebra defines the following operations on quaternions [5]:

- *addition and subtraction*: is defined by

$$\begin{aligned} q_0 \pm q_1 &= (x_0 + x_1 i + x_2 j + x_3 k) \pm (y_0 + y_1 i + y_2 j + y_3 k) \\ &= (x_0 \pm y_0) + (x_1 \pm y_1) i + (x_2 \pm y_2) j + (x_3 \pm y_3) k. \end{aligned} \quad (2)$$

- *scalar multiplication*: is defined over the basic units i, j, k by Eq. (1).
- *multiplication*: of quaternions is defined by

$$\begin{aligned} q_0 q_1 &= (x_0 + x_1 i + x_2 j + x_3 k)(y_0 + y_1 i + y_2 j + y_3 k) \\ &= (x_0 y_0 - x_1 y_1 - x_2 y_2 - x_3 y_3) + \\ &\quad (x_0 y_1 + x_1 y_0 + x_2 y_3 - x_3 y_2) i + \\ &\quad (x_0 y_2 - x_1 y_3 + x_2 y_0 + x_3 y_1) j + \\ &\quad (x_0 y_3 + x_1 y_2 - x_2 y_1 + x_3 y_0) k. \end{aligned} \quad (3)$$

  Multiplication is not commutative because the product $q_0 q_1 \neq q_1 q_0$ in general.
- *conjugate*: is unary arithmetical operation defined by

$$q^* = (x_0 + x_1 i + x_2 j + x_3 k)^* = x_0 - x_1 i - x_2 j - x_3 k. \quad (4)$$

  The conjugate of quaternions satisfies the properties $(q^*)^* = q$ and $(q_0 q_1)^* = q_0^* q_1^*$.
- *norm*: is defined by

$$N(q) = N(x_0 + x_1 i + x_2 j + x_3 k) = \sqrt{x_0^2 + x_1^2 + x_2^2 + x_3^2}. \quad (5)$$

  The norm is a real-valued function that satisfies the properties $N(q^*) = N(q)$ and $N(q_0 q_1) = N(q_0) N(q_1)$. This function is suitable for mapping the elements of vector (individual, solution) from 4-dimensional quaternion in genotype space to 1-dimensional real-valued element in phenotype space.
- *multiplicative inverse*: of quaternion q is denoted as q^{-1} and has the property $q q^{-1} = q^{-1} q = 1$. It is constructed as

$$q^{-1} = q^* / N(q), \quad (6)$$

  where the division of a quaternion by a real-values scalar is division of each component by norm. The inverse operation satisfies the properties $(q^{-1})^{-1} = q$ and $(q_0 q_1)^{-1} = q_0^{-1} q_1^{-1}$.
- *division*: of quaternions q_0 and q_1 is defined by

$$q_0 / q_1 = q_0 q_1^{-1}. \quad (7)$$

In addition to the pure quaternion algebra, two unary functions are added by us as follows:

- *qrand*: is a quaternion defined as

$$qrand() = \{x_i = N(0,1) \mid \text{for } i = 1 \ldots 4\}, \quad (8)$$

  where $N(0,1)$ denotes a random number drawn from a Gaussian distribution with zero mean and standard deviation one. In other words, each component is initialized with the random generated number.
- *qzero*: is a quaternion defined as

$$qzero() = \{x_i = 0 \mid \text{for } i = 1 \ldots 4\}, \quad (9)$$

  where each component of quaternion is initialized with zero.

These operations of quaternion algebra serve as a reach basis for developing the different variation operators used in swarm intelligence and evolutionary algorithms. In the rest of the paper, the usage of these operations by developing the operators within the new Bat algorithm with quaternion representation of individuals, are presented in detail.

III. Bat algorithm

The echolocation behavior of bats inspired Yang [22], [12] by developing the Bat algorithm (BA). That means, bats emit a sound pulse and listen to the echo bouncing back from obstacles whilst flying. Bats are the only mammals with wings. Although some species are not blind, they use echolocation as their primary mechanism for orientation towards a surface. Additionally, this mechanism serves as a tool for finding their prey and discriminating different types of insects.

In echolocation, three parameters are important, i.e., the range of frequencies, the rate of pulse emission, and the loudness. The frequency of pulse emission depends on the prey's size, i.e. the smaller the prey, the higher the frequency. The rate of pulse emission speeds-up when the bats move near their prey when hunting. The loudness is higher when the bats are hunt for prey and lower when they are flying homeward.

A. Original Bat algorithm

The echolocation behavior of bats can be formulated as a new optimization algorithm, where this behavior is captured into an objective function of an optimization problem to be solved. However, the following approximations of bat behavior are idealized during development:

- All bats use echolocation to sense distance to target objects.
- Bats fly randomly with the velocity v_i at position x_i, the frequency $Q_i \in [Q_{min}, Q_{max}]$ (also the wavelength λ_i), the rate of pulse emission $r_i \in [0, 1]$, and the loudness $A_i \in [A_0, A_{min}]$. The frequency (and wavelength) can be adjusted depending on the proximity of their target.
- The loudness varies from a large (positive) A_0 to a minimum constant value A_{min}.

Algorithm 1 Original Bat algorithm

Input: Bat population $\mathbf{x_i} = (x_{i1}, \ldots, x_{iD})^T$, pulse rate $\mathbf{r}$, and loudness $\mathbf{A}$.
Output: The $\mathbf{best}$ solution and fitness value $f_{min} = \min(f(\mathbf{x}))$.

```
1: init_bat(); // initialize the bat population x_i, velocity v_i and frequency Q_i
2: best=best_bat(); // obtain best solution and determine the best fitness f_min
3: while t ≤ MAX_T do
4:   for i = 1 to Np do
5:     Q_i = adjust_freq_bat(); // adjust frequency values
6:     y = move_bat(x_i, v_i); // update velocity and determine a new position
7:     if N(0,1) > r_i then
8:       y = neigh_bat(best); // generate solution in neighborhood of best
9:     end if
10:    f_new = eval_bat(y); // evaluate the quality of solution
11:    if f_new ≤ f_i and N(0,1) < A_i then
12:      x_i = y; f_i = f_new; // save the local best solution
13:    end if
14:    if f_new ≤ f_min then
15:      best = x_i; f_min = f_new; // save the global best solution
16:    end if
17:    adjust_parm_bat(i, r, A); // adjust algorithm parameters
18:  end for
19: end while
```

The pseudo-code of a BA algorithm is illustrated in Algorithm 1. The BA algorithm starts with a population of individuals $\mathbf{x}_i = \{x_{ij}\}$ for $i = 1 \ldots n$ and $j = 1 \ldots D$ with elements x_{ij} representing bat positions in the search space. Hence, parameter *Np* denotes the number of individuals in the population, whilst D is the dimension of the problem. In addition, vectors of the pulse rate $\mathbf{r}$ and the loudness $\mathbf{A}$ both of dimension D, are predefined. The task of this algorithm is to find the optimal (minimal) value of the objective function $f(\mathbf{x})$ and the corresponding **best** solution. The original bat algorithm consists of the following components:

- initialization (lines 1-2): initializing the bat population $\mathbf{x}_i = \{x_{ij}\}$, the velocity $\mathbf{v} = \{v_i\}$, and the frequency $\mathbf{Q} = \{Q_i\}$. Then, the **best** solution is obtained and its fitness f_{min} determined.
- movement of visual bats (lines 5-6): firstly, the frequency value Q_i is adjusted (function 'adjust_freq_bat'), then, velocity $\mathbf{v}_i$ is updated, and finally, new bat $\mathbf{y}$ at i-position is determined (function 'move_bat').
- local search part (lines 7-9): solution $\mathbf{y}$ is generated in the neighborhood of the current $\mathbf{best}$ solution according to the proximity of pulse rate r_i. The function 'neigh_bat', therefore, can be viewed as a kind of local search.
- evaluation of objective function (line 10): the function 'eval_bat' is an implementation of objective function $f(\mathbf{x})$.
- loudness (lines 11-13): the parameter A_i determines a proximity within which the new better solution $\mathbf{y}$ replaces the original less fitter solution $\mathbf{x}_i$. This step is close to simulated annealing logic [15], where the best solution is a new possible less fitter solution with a predefined proximity.
- saving the best solution (line 14-16): the current solution $\mathbf{y}$ with a better fitness value replaces the current best solution $\mathbf{best}$ and its fitness f_{min}.
- adjusting the control parameters (line 17): control parameters, the rate of pulse emission r_i, and the loudness A_i are updated.

Movement of virtual bats obeys the following equations:

$$\begin{aligned} Q_i^{(t)} &= Q_{min} + (Q_{max} - Q_{min})N(0,1), \\ \mathbf{v}_i^{(t+1)} &= \mathbf{v}_i^{(t)} + (\mathbf{x}_i^{(t)} - \mathbf{best})Q_i^{(t)}, \\ \mathbf{x}_i^{(t+1)} &= \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t)}, \end{aligned} \tag{10}$$

where $N(0,1)$ denotes a random number drawn from a Gaussian distribution with zero mean, and standard deviation one.

Note that this movement is close to the definition of movement in particle swarm optimization which, next to the second term in Eq. (10) identifying the distance to the global best solution, also includes an additional term reflecting the distance of the current solution to the local best. As mentioned before, the local search part implements a kind of random walk with direct exploitation, according to equation:

$$\mathbf{y^{(t)}} = \mathbf{best} + \epsilon A_i^{(t)}(2 \times N(0,1) - 1), \tag{11}$$

where $N(0,1)$ denotes a random number drawn from a Gaussian distribution with zero mean and standard deviation one, ϵ is the scaling factor, and $A_i^{(t)}$ the loudness. Obviously, the term inside the parenthesis provides that the random generated number is spanned to interval $[-1, 1]$.

The loudness $A_i^{(t)}$ and the rate of pulse emission $r_i^{(t)}$ can be changed during the search process. Naturally, the loudness decreases and the rate of pulse emission increases when a bat finds a prey. This characteristic can be formulated in a Bat algorithm, using following equations:

$$A_i^{(t+1)} = \alpha A_i^{(t)}, \quad r_i^{(t)} = r_i^{(0)}[1 - exp(-\gamma\epsilon)], \tag{12}$$

where α and γ are constants. Actually, the α parameter plays a similar role as the cooling factor in a simulated annealing algorithm, and controls the convergence rate [15].

B. Bat algorithm with a quaternion representation

The new Quaternion's Bat algorithm (QBA) introduces a novel representation of individuals using quaternions in the original BA algorithm. The motivation behind using quaternions was to avoid stagnation of the original BA algorithm which is often accompanied by it. In line with this, each 1-dimensional element of a real-valued solution is represented by the 4-dimensional quaternion. Indeed, the quaternion search space is explored when discovering the original solution space. Although this search space is much greater than the original solution space, it is expected that the fitness landscape determined using quaternion's norm function is much smoother without many local optima and, therefore, the global optimum could be easier to find.

The new QBA algorithm is similar to the original BA algorithm except that the real-valued elements of vectors $\mathbf{v}_i, \mathbf{x}_i, \mathbf{y}, \mathbf{best} \in \mathbb{R}^D$ now become quaternions $\mathbf{v}_i, \mathbf{q}_i, \mathbf{p}, \mathbf{qbest} \in \mathbf{H}^D$.

Similar to Algorithm 1, the QBat also starts with a population of individuals $\mathbf{q}_i = \{q_{ij}\}$ for $i = 1 \dots Np$ and $j = 1 \dots D$ with quaternions $q_{ij} \in \mathbf{H}$ representing the bat position in the quaternion's search space. Vectors of the pulse rate $\mathbf{r}$ and the loudness $\mathbf{A}$ are also predefined by the user. The task of this algorithm is to find the minimal value of objective function $f^* = \min(f(norm(\mathbf{qbest})))$, and the corresponding best solution $\mathbf{best} = norm(\mathbf{qbest})$. Note that the function $norm(\mathbf{qbest})$ maps the quaternion's best solution **qbest** into real-valued vector **best**.

Other components of the QBat algorithm are implemented as follows. The population of quaternions is initialized randomly as

$$\mathbf{q}_i = \{q_{ij} \mid q_{ij} = qrand();\ \text{for } i = 1 \dots 4\}, \tag{13}$$

whilst the velocity is initialized with zero as

$$\mathbf{v}_i = \{v_{ij} \mid v_{ij} = qzero();\ \text{for } i = 1 \dots 4\}. \tag{14}$$

The movement of virtual bats is expressed in quaternions algebra, as follows

$$\begin{aligned} \mathbf{v}_i^{(t+1)} &= \mathbf{v}_i^{(t)} + (\mathbf{q}_i^{(t)} - \mathbf{best})Q_i^{(t)}, \\ \mathbf{q}_i^{(t+1)} &= \mathbf{q}_i^{(t)} + \mathbf{v}_i^{(t)}. \end{aligned} \tag{15}$$

Note that $Q_i^{(t)}$ in Eq. (15) is calculated similarly to that in Eq. (10). The local search part is now expressed as

$$\mathbf{p}^{(\mathbf{t+1})} = \mathbf{best} + \epsilon A_i^{(t)} N(-1, 1), \tag{16}$$

which is in essence Eq. (11) expressed in the quaternions algebra. Formulation of the objective function depends on the problem to be solved. Additionally, the QBA algorithm uses a function $span$ for mapping the quaternion into 1-dimensional real vector defined as follows:

$$span(q_{ij}) = \frac{Ub_j - Lb_j}{2} \sin(norm(q_{ij})), \quad \text{for } j = 1 \dots D, \tag{17}$$

where Ub_j denotes the upper and Lb_j the lower bounds of the j-th quaternion, $\sin$ is sinus and $norm$ the standard quaternion function. The sinus function normalizes the quaternion $norm$ to the interval in $[-1, 1]$ which must be spanned to the interval $[Lb_j, Ub_j]$ in order to satisfy the constraints $x_{ij} \in [Lb_j, Ub_j]$.

IV. Experiments and results

The goal of our experimental work was to show that quaternions can be appropriate for the representation of individuals in swarm intelligence and evolutionary algorithms. The BA algorithm was used as a test-bed for demonstrating this assumption by solving function optimization problems. In line with our assumptions, it was expected that the new QBA algorithm should improve the results of the original BA algorithm significantly. On the other hand, the QBA algorithm was compared with other swarm intelligence and evolutionary algorithms, like ABC, and DE, in order to see how competitive were the results achieved by using this new representation.

The function optimization problem belongs to a class of continuous optimization problems and is defined as follows. Let us assume, an objective function $f(\mathbf{x})$ is given, where $\mathbf{x} = (x_1, \dots, x_D)$ is a vector of D design variables in a decision space S. The design variables $x_j \in \{Lb_j, Ub_j\}$ are limited by their lower $Lb_j \in \mathbb{R}$ and upper bounds $Ub_j \in \mathbb{R}$. The task of optimization is to find the minimum of the objective function.

In the rest of paper the test suite is described, the experimental setup is defined, the configuration of PC, on which the experiments were executed, is introduced, and the results of experiments are illustrated in detail.

A. Test suite

The test suite consisted of ten functions, which were selected from two references. The primary reference presents Karaboga's paper [13], in which the ABC algorithm was introduced. The secondary reference was the paper of Yang [21] that proposed a set of optimization functions suitable for testing the newly-developed algorithms. Actually, the first five functions were taken from the first reference, whilst the rest from the second reference.

The functions within the test suite can be divided into *unimodal* and *multimodal*. The multimodal functions have two or more local optima. The function is *separable*, when the set of variables can be rewritten as a sum of the function of just one variable. The separable and multimodal functions are more difficult to solve. The more complex functions are those that have an exponential number of local optima

randomly distributed within the search space. The definitions and characteristics of functions constituting the test suite are defined in Tables I-II, respectively.

Each function in the mentioned tables is tagged with its sequence number from f_1 to f_{10}. Typically, the problem becomes heavier to solve when the dimensionality of the benchmark functions is increased. Therefore, benchmark functions of more dimensions needed to be optimized during the experimental work.

Table II
PROPERTIES OF BENCHMARK FUNCTIONS

f	f^*	x^*	Domain
f_1	0	$(0, 0, \ldots, 0)$	$[-600, 600]$
f_2	0	$(0, 0, \ldots, 0)$	$[-15, 15]$
f_3	0	$(1, 1, \ldots, 1)$	$[-15, 15]$
f_4	0	$(0, 0, \ldots, 0)$	$[-32.768, 32.768]$
f_5	0	$(0, 0, \ldots, 0)$	$[-500, 500]$
f_6	0	$(0, 0, \ldots, 0)$	$[-600, 600]$
f_7	-1	$(\pi, \pi, \ldots, \pi)$	$[-2\pi, 2\pi]$
f_8	-1.8013[1]	$(2.20319, 1.57049)$[1]	$[0, \pi]$
f_9	0	$(0, 0, \ldots, 0)$	$[-2\pi, 2\pi]$
f_{10}	0	$(0, 0, \ldots, 0)$	$[-5, 10]$

The lower and upper bounds of the design variables denote domains determining the size of the search space. The wider this domain, the wider the search space. Note that the domains were selected so that the search space was wider than those proposed in the standard literature. Another difficulty was represented by the dimensions of the functions. Typically, the higher the dimensional function, the more difficult to optimize.

B. Experimental setup

In this experimental study, the results of the following algorithms were compared: BA, QBA, DE, and ABC. Actually, DE belongs to the evolutionary algorithms, whilst the other algorithms are members of the swarm intelligence community. In this paper as a termination condition, the number of fitness function evaluations (FEs) was considered although the original algorithms use the number of generations primarily. However, the population size is a crucial parameter for all population-based algorithms that have a great influence on their performance. In line with this, extensive experiments had been run in order to determine the most appropriate setting of this parameter by all algorithms in the test. As a result, the most appropriate setting of this parameter $Np = 100$ was considered during the experiments.

The specific BA parameters were set as follows: the loudness $A_0 = 0.5$, the pulse rate $r_0 = 0.5$, minimum frequency $Q_{max} = 0.0$, and maximum frequency $Q_{max} = 0.1$. The same parameters were also used for the QBA algorithm because both algorithms are the same except in representation of individuals.

[1]Valid for 2-dimensional parameter space.

The DE parameters were configured as follows: the amplification factor of the difference vector $F = 0.5$, and the crossover control parameter $CR = 0.9$. The percentage of onlooker bees for the ABC algorithm was 50% of the colony, the employed bees represented another 50% of the colony, whilst one scout bee was generated in each generation (i.e., $limits = 100$, when the population size is $Np = 100$).

All the algorithms were run 25 times. The results from the algorithms were accompanied according to five standard measures, like: *Best*, *Worst*, *Mean*, *StDev*, and *Median* values.

C. PC configuration

All runs were made on HP Compaq using the following configurations:

1) Processor - Intel Core i7-2600 3.4 (3.8) GHz
2) RAM - 4GB DDR3
3) Operating system - Linux Mint 12

All versions of the tested algorithms were implemented within the Eclipse Indigo CDT framework.

D. Results

Our experimental work was led by the following subjects:

- what impact the dimensionality of problems had on the results of QBA,
- how much the new representation of individuals using quaternions improved the results of the original BA, and
- what the obtained results of QBA implied when comparing them with the results of other optimization algorithms, like DE and ABC.

In the rest of paper these subjects are quantified in detail.

1) Impact on the dimensionality of problems: In order to show how the dimension of the problem affects the performance of algorithms, the dimensions of the functions $D = 10$, $D = 30$, and $D = 50$ were taken into consideration. In line with this, the maximum number of generations was varied according to an expression $MAX\_T = 5000D/Np$ in order to obtain a suitable number of fitness evaluations. In our case, the number of fitness evaluations for $D = 10$ was limited to $50,000$, for $D = 30$ to $150,000$, and for $D = 50$ to $250,000$ fitness evaluations.

The results of QBA algorithm optimizing ten test functions are presented in Table III, from which it can be seen that the QBA algorithm successfully found the global optimum for functions f_1, f_2, f_6, and f_{10} with dimension $D = 10$. The global optimum was not found for functions of higher dimensions ($D = 30$, and $D = 50$). On average, those functions with higher dimension $D = 50$ were more difficult to solve.

Table I
DEFINITIONS OF BENCHMARK FUNCTIONS

f	Function name	Definition
f_1	Griewangk's function	$f(\mathbf{x}) = -\prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + \sum_{i=1}^{n} \frac{x_i^2}{4000} + 1$
f_2	Rastrigin's function	$f(\mathbf{x}) = n * 10 + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$
f_3	Rosenbrock's function	$f(\mathbf{x}) = \sum_{i=1}^{n-1} 100\,(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$
f_4	Ackley's function	$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left(20 + e^{-20} e^{-0.2\sqrt{0.5(x_{i+1}^2 + x_i^2)}} - e^{0.5(\cos(2\pi x_{i+1}) + \cos(2\pi x_i))}\right)$
f_5	Schwefel's function	$f(\mathbf{x}) == 418.9829 * D - \sum_{i=1}^{D} s_i \sin(\sqrt{\lvert s_i \rvert})$
f_6	De Jong's sphere function	$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$
f_7	Easom's function	$f(\mathbf{x}) = -(-1)^D (\prod_{i=1}^{D} \cos^2(x_i)) \exp[-\sum_{i=1}^{D} (x_i - \pi)^2]$
f_8	Michalewicz's function	$f(\mathbf{x}) = -\sum_{i=1}^{D} \sin(x_i)[\sin(\frac{ix_i^2}{\pi})]^{2 \cdot 10}$
f_9	Xin-She Yang's function	$f(\mathbf{x}) = (\sum_{i=1}^{D} \lvert x_i \rvert) \exp[-\sum_{i=1}^{D} \sin(x_i^2)]$
f_{10}	Zakharov's function	$f(\mathbf{x}) = \sum_{i=1}^{D} x_i^2 + (\frac{1}{2}\sum_{i=1}^{D} i x_i)^2 + (\frac{1}{2}\sum_{i=1}^{D} i x_i)^4$

Table III
RESULTS ACCORDING TO DIMENSIONS

D	Meas.	f_1	f_2	f_3	f_4	f_5
10	Best	0.00E+00	0.00E+00	1.13E-01	4.44E-16	7.55E-01
	Worst	0.00E+00	0.00E+00	1.13E-01	4.44E-16	7.55E-01
	Mean	0.00E+00	0.00E+00	1.13E-01	4.44E-16	7.55E-01
	StDev	0.00E+00	0.00E+00	1.42E-17	0.00E+00	0.00E+00
	Median	0.00E+00	0.00E+00	1.13E-01	4.44E-16	7.55E-01
30	Best	7.76E-01	3.65E+01	5.68E+01	6.52E-01	4.11E+02
	Worst	7.76E-01	3.65E+01	5.68E+01	6.52E-01	4.11E+02
	Mean	7.76E-01	3.65E+01	5.68E+01	6.52E-01	4.11E+02
	StDev	3.40E-16	2.18E-14	1.45E-14	3.40E-16	0.00E+00
	Median	7.76E-01	3.65E+01	5.68E+01	6.52E-01	4.11E+02
50	Best	6.94E+00	2.15E+02	7.24E+04	6.09E+00	5.22E+03
	Worst	1.07E+01	3.22E+02	1.43E+05	7.15E+00	5.37E+03
	Mean	8.72E+00	2.67E+02	1.12E+05	6.59E+00	5.26E+03
	StDev	8.75E-01	2.82E+01	1.95E+04	2.65E-01	3.92E+01
	Median	8.65E+00	2.70E+02	1.17E+05	6.53E+00	5.24E+03
D	Meas.	f_6	f_7	f_8	f_9	f_{10}
10	Best	0.00E+00	-9.64E-01	-3.94E+00	0.00E+00	0.00E+00
	Worst	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	Mean	0.00E+00	-9.64E-01	-3.36E+00	0.00E+00	0.00E+00
	StDev	0.00E+00	5.67E-16	2.38E-01	0.00E+00	0.00E+00
	Median	0.00E+00	-9.64E-01	-3.28E+00	0.00E+00	0.00E+00
30	Best	6.92E+01	-3.27E-15	-7.93E+00	5.78E-11	2.28E+00
	Worst	6.92E+01	0.00E+00	0.00E+00	5.78E-11	3.21E+00
	Mean	6.92E+01	-3.27E-15	-7.72E+00	5.78E-11	2.66E+00
	StDev	1.45E-14	1.61E-30	5.11E-02	3.30E-26	2.50E-01
	Median	6.92E+01	-3.27E-15	-7.71E+00	5.78E-11	2.62E+00
50	Best	2.36E+04	-3.77E-53	-1.06E+01	1.76E-17	1.11E+01
	Worst	3.77E+04	0.00E+00	0.00E+00	1.63E-16	3.41E+01
	Mean	3.04E+04	-6.80E-54	-9.53E+00	7.68E-17	1.64E+01
	StDev	3.39E+03	9.99E-54	3.76E-01	3.78E-17	6.48E+00
	Median	3.04E+04	-1.67E-54	-9.33E+00	6.84E-17	1.32E+01

2) Comparative study: An intention of this experiment was twofold. Firstly, to show how the new representation of individuals with quaternions can improve the results of an original BA algorithm [23] by optimizing a suite of ten test functions. Secondly, to show how good the results obtained using the new QBA were when compared with the other well-known algorithms, like ABC [13] and DE [19]. Thereby, one experimental setup was designed to run each algorithm on the function test suite, and then the analysis of the results was made in the senses of both pursued objectives. This analysis was substantiated using the Friedman statistical tests for evaluating the obtained results.

The results of the experiments are illustrated in Table IV, where the mean values and corresponding standard deviations are presented according to ten functions (f_1 to f_{10}) and four algorithms (BA, QBA, DE and ABC). Note that the results of the algorithms are kept into three groups distinguished by the dimensions of the optimized function ($D = \{10, 30, 50\}$). The best results from the algorithms are displayed bold in the table.

Table IV
COMPARING ALGORITHMS

F	D	BA	QBA	DE	ABC
f_1	10	8.3E+00 ± 5.4E+00	**0.0E+00 ± 0.0E+00**	1.5E+00 ± 2.3E-01	1.6E+00 ± 8.4E-01
	30	7.9E+01 ± 2.7E+01	**7.8E-01 ± 3.4E-16**	1.0E+00 ± 2.2E-02	1.1E+00 ± 1.2E-01
	50	1.4E+02 ± 3.2E+01	8.7E+00 ± 8.8E-01	**1.0E+00 ± 5.2E-02**	1.4E+00 ± 5.8E-01
f_2	10	1.5E+02 ± 6.5E+01	**0.0E+00 ± 0.0E+00**	7.0E+01 ± 1.1E+01	4.7E+01 ± 1.5E+01
	30	1.0E+03 ± 3.3E+02	**3.6E+01 ± 2.2E-14**	2.3E+02 ± 1.3E+01	7.3E+01 ± 2.2E+01
	50	1.8E+03 ± 4.5E+02	2.7E+02 ± 2.8E+01	4.2E+02 ± 1.7E+01	**1.5E+02 ± 3.9E+01**
f_3	10	1.6E+05 ± 2.1E+05	**1.1E-01 ± 1.4E-17**	2.4E+03 ± 1.5E+03	1.6E+03 ± 1.5E+03
	30	3.2E+06 ± 1.8E+06	**5.7E+01 ± 1.5E-14**	4.6E+02 ± 2.3E+02	5.2E+02 ± 4.7E+02
	50	4.8E+06 ± 2.3E+06	1.1E+05 ± 1.9E+04	**5.1E+02 ± 2.5E+02**	1.5E+03 ± 1.1E+03
f_4	10	1.1E+01 ± 2.4E+00	**4.4E-16 ± 0.0E+00**	4.7E+00 ± 5.6E-01	9.2E+00 ± 1.6E+00
	30	1.3E+01 ± 9.6E-01	**6.5E-01 ± 3.4E-16**	1.8E+00 ± 3.2E-01	7.2E+00 ± 1.0E+00
	50	1.4E+01 ± 8.4E-01	6.6E+00 ± 2.6E-01	**1.1E+00 ± 2.6E-01**	8.3E+00 ± 9.7E-01
f_5	10	2.2E+03 ± 3.7E+02	**7.5E-01 ± 0.0E+00**	1.8E+03 ± 1.6E+02	1.0E+03 ± 1.6E+02
	30	8.9E+03 ± 4.0E+02	**4.1E+02 ± 0.0E+00**	7.6E+03 ± 4.4E+02	2.6E+03 ± 3.3E+02
	50	1.6E+04 ± 7.8E+02	5.3E+03 ± 3.9E+01	1.4E+04 ± 3.4E+02	**4.8E+03 ± 4.2E+02**
f_6	10	4.0E+04 ± 3.8E+04	**0.0E+00 ± 0.0E+00**	2.0E+03 ± 7.7E+02	2.5E+03 ± 3.0E+03
	30	3.6E+05 ± 9.2E+04	**6.9E+01 ± 1.5E-14**	1.8E+02 ± 7.1E+01	1.6E+02 ± 2.0E+02
	50	6.4E+05 ± 1.7E+05	3.0E+04 ± 3.4E+03	**1.6E+02 ± 5.6E+01**	4.2E+02 ± 5.5E+02
f_7	10	-2.2E-02 ± 1.1E-01	**-9.6E-01 ± 5.7E-16**	-6.7E-01 ± 1.1E-01	-2.2E-03 ± 1.1E-02
	30	-6.7E-18 ± 3.4E-17	**-3.3E-15 ± 1.6E-30**	-2.8E-175 ± 0.0E+00	-1.8E-136 ± 8.8E-136
	50	**-3.6E-27 ± 1.8E-26**	-6.8E-54 ± 1.0E-53	0.0E+00 ± 0.0E+00	-3.2E-261 ± 0.0E+00
f_8	10	-5.9E+00 ± 8.9E-01	-3.4E+00 ± 2.4E-01	-5.2E+00 ± 4.7E-01	**-7.4E+00 ± 3.7E-01**
	30	-1.4E+01 ± 1.7E+00	-7.7E+00 ± 5.1E-02	-1.1E+01 ± 6.7E-01	**-2.3E+01 ± 7.0E-01**
	50	-2.1E+01 ± 2.6E+00	-9.5E+00 ± 3.8E-01	-1.5E+01 ± 6.7E-01	**-3.7E+01 ± 8.6E-01**
f_9	10	1.3E-03 ± 4.6E-04	**0.0E+00 ± 0.0E+00**	2.4E-03 ± 2.2E-04	9.2E-04 ± 1.6E-04
	30	2.5E-11 ± 3.7E-11	5.8E-11 ± 3.3E-26	2.5E-11 ± 1.2E-12	**1.1E-11 ± 1.9E-12**
	50	**7.8E-20 ± 1.3E-19**	7.7E-17 ± 3.8E-17	1.3E-19 ± 7.9E-21	3.4E-17 ± 6.8E-18
f_{10}	10	4.1E+01 ± 2.8E+01	**0.0E+00 ± 0.0E+00**	2.8E+00 ± 9.7E-01	3.5E+01 ± 1.1E+01
	30	2.5E+02 ± 2.0E+02	**2.7E+00 ± 2.5E-01**	3.8E+01 ± 8.7E+00	2.5E+02 ± 3.2E+01
	50	8.8E+02 ± 2.9E+02	**1.6E+01 ± 6.5E+00**	1.8E+02 ± 3.0E+01	5.6E+02 ± 4.5E+01

The Friedman test [8], [9] compares the average ranks of the algorithms. A null-hypothesis states that two algorithms are equivalent and, therefore, their ranks should be equal. If the null-hypothesis is rejected, i.e., the performance of the algorithms is statistically different, the Bonferroni-Dunn test [3] is performed that calculates the critical difference between the average ranks of those two algorithms. When the statistical difference is higher than the critical difference, the algorithms are significantly different. The equation for the calculation of critical difference can be found in [3].

Friedman tests were performed using the significance level 0.05. The results of the Friedman non-parametric test are presented in Fig. 1 being divided into three diagrams that show the ranks and confidence intervals (critical differences) for the algorithms under consideration. The diagrams are organized according to the dimensions of functions. Thus, the better as the algorithm, the nearer to the rank one. Two algorithms are significantly different if their intervals in Fig. 1 do not overlap.

The first diagram in Fig. 1 shows that the QBA algorithm outperforms all the other algorithms in the test. Moreover, the results of QBA are significantly better than the results of all the other algorithms in test. In the second graph, the QBA still improves the results of the other algorithms, but this difference is significant only when comparing with the results of the original BA algorithm. Finally, the third graph shows that the best results are obtained by DE. In summary, QBA outperformed the results of BA significantly regarding all three observed dimensions of the benchmark functions that confirmed our assumption before conducting the experiments.

E. Discussion

In summary, we showed that the new representation of individuals with quaternions improves the behavior of original BA significantly when optimizing the selected test-suite of functions. On the other hand, the results of QBA are comparable with the results of the other well-known swarm intelligence and evolutionary algorithms, like ABC and DE. Anyway, in the future there should more tests be conducted using more complex benchmark function families like CEC or BBOB in order to show the power of quaternions in optimization.

V. CONCLUSION

This paper proposed a modified bat algorithm using quaternion representation of individuals. Quaternions are especially appropriate for programming the video games and controllers of spacecraft, where it is necessary to compute rotations with minimal computation effort. Especially useful, they have been applied to the problems of theoretical physics. Here, the quaternions are used for optimization purposes.

Designing the new representation also demands the defining of the appropriate search operators. There, using the well defined quaternions algebra is inevitable. In order to show, that this algorithm can also be applied to the optimization of higher-dimensional functions, the new representation with quaternions was implemented within the original BA algorithm. The new QBA was significantly improved the results

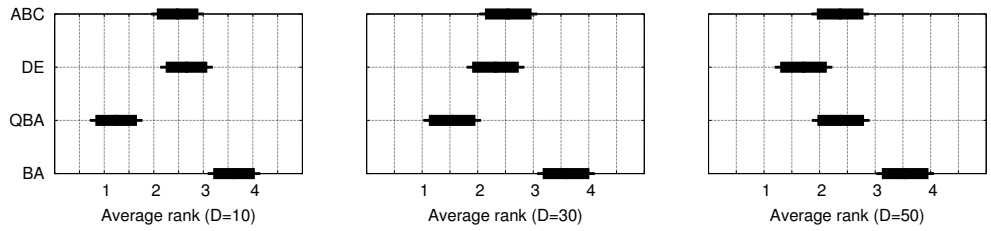

Figure 1. Results of the Friedman non-parametric test

of BA by optimizing the benchmark function suite. Moreover, the obtained results were comparable with the results of other well-known swarm intelligence and evolutionary algorithms, like ABC and DE.

REFERENCES

[1] C. Blum and X. Li. Swarm intelligence in optimization. In C. Blum and D. Merkle, editors, *Swarm Intelligence: Introduction and Applications*, pages 43–86. Springer Verlag, Berlin, 2008.

[2] J.H. Conway and D.A. Smith. *On Quaternions and Octonions: Their Geometry, Arithmetic, and Symmetry*. A K Reters, Wellesley, Massachusetts, 2003.

[3] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[4] M. Dorigo and G. Di Caro. The ant colony optimization metaheuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw Hill, London, UK, 1999.

[5] D. Eberly. Quaternion algebra and calculus. *Magic Software, Inc*, 21, 2002.

[6] Iztok Fister, Iztok Fister Jr., Janez Brest, and Viljem Žumer. Memetic artificial bee colony algorithm for large-scale global optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–8, 2012.

[7] Iztok Fister, Xin-She Yang, Janez Brest, and Iztok Fister Jr. Modified firefly algorithm using quaternion representation. *Expert Systems with Applications*, 40(18):7220 – 7230, 2013.

[8] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32:675–701, December 1937.

[9] M. Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11:86–92, March 1940.

[10] P.R. Girard. The quaternion group and modern physics. *European Journal of Physics*, 5:25–32, 1984.

[11] W.R. Hamilton. *Elements of quaternions*. Longmans, Green and Co., 1899.

[12] Iztok Fister Jr., Simon Fong, Janez Brest, and Iztok Fister. A novel hybrid self-adaptive bat algorithm. *The Scientific World Journal*, 2014:1–12, 2014.

[13] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3):459–471, 2007.

[14] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948. IEEE, 1995.

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[16] Peter Korošec, Jurij Šilc, and Bogdan Filipič. The differential ant-stigmergy algorithm. *Information Sciences*, 192(0):82–97, 2012.

[17] J. Lambek. If Hamilton had prevailed: quaternions in physics. *The Mathematical Intelligencer*, 17:7–15, 1995.

[18] Franz Rothlauf. *Representations for genetic and evolutionary algorithms (2. ed.)*. Springer, 2006.

[19] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[20] X. S. Yang. Firefly algorithm. *Nature-Inspired Metaheuristic Algorithms*, pages 79–90, 2008.

[21] X.-S. Yang. Appendix a: Test problems in optimization. In X.-S. Yang, editor, *Engineering Optimization*, pages 261–266. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2010.

[22] X. S. Yang. A new metaheuristic bat-inspired algorithm. In C. Cruz, J.R. Gonzlez, and G. Terrazas N. Krasnogor, D.A. Pelta, editors, *Nature Inspired Cooperative Strategies for Optimization (NISCO 2010), Studies in Computational Inteligence*, volume 284, pages 65–74. Springer Verlag, Berlin, 2010.

[23] X. S. Yang. A new metaheuristic bat-inspired algorithm. *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, pages 65–74, 2010.

[24] X. S. Yang and S. Deb. Cuckoo search via levy flights. In *World Congress on Nature & Biologically Inspired Computing (NaBIC 2009)*, pages 210–214, IEEE Publications, 2009.