Expert Systems with Applications 40 (2013) 7220–7230

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Modified firefly algorithm using quaternion representation

Iztok Fister[a,*], Xin-She Yang[a,b], Janez Brest[a], Iztok Fister Jr.[a]

[a] University of Maribor, Faculty of Electrical Engineering and Computer Science, Smetanova 17, 2000 Maribor, Slovenia
[b] School of Science and Technology, Middlesex University, United Kingdom

ARTICLE INFO

Keywords:
Quaternions
Representation of individuals
Firefly algorithm
Optimization

ABSTRACT

Quaternions are a number system, which extends complex numbers. They are especially useful in areas where fast rotation calculations are needed, e.g., programming video games or controllers of spacecraft. This paper proposes to use quaternion for the representation of individuals in firefly algorithm so as to enhance the performance of the firefly algorithm and to avoid any stagnation. The preliminary results of our experiments after optimizing a test-suite consisting of ten standard functions, showed that the proposed firefly algorithms using quaternion's representation improved the results of the original firefly algorithm.

1. Introduction

Optimization is a process of finding one or more solutions to a problem for satisfying certain limitations (also constraints), and minimizes or maximizes one or more objectives (Deb, 2001). Those problems that arise daily in practice are typically complex according to the time and space needed for solving them on digital computers. Therefore, exact solving of problems ordinarily fail, where all possible solutions need to be checked. As a result, a lot of algorithms have now arisen that solve the problems heuristically. Usually, a search process involved within such heuristic algorithms when seeking good solutions is influenced by both randomization global and deterministic local searches. These kinds of heuristics are therefore also referred as meta-heuristics. Typically, these algorithms are inspired by Nature. Essentially, there are three sources that influence the development of algorithms for problem-solving on digital computers:

- the human brain (Russel & Norvik, 2010),
- the Darwinian evolution (Darwin, 1859), and
- the social behavior of insects and other animals (Kennedy & Eberhart, 2001).

The first source has been led to emergence of *artificial intelligence* (AI), the second to *evolutionary computation* (EC), and the third to *swarm intelligence* (SI). This paper focuses on swarm intelligence, more precisely on the *firefly algorithm* (FA) as being one of the more famous representatives of this class of algorithm.

Fireflies are insects, the main characteristic of which is their flashing lights that can be admired in the summer sky at night. These lights have two fundamental functions, i.e., to attract mating partners and to warn off potential predators. The flashing lights' intensity I decreases as the distance r increases according to the term $I \propto I/r^2$. This phenomenon inspired Yang (2008) to formulate the firefly algorithm.

Results from experiments running FA have shown that, on the one hand, the FA is appropriate for solving multi modal problems while, on the other hand, it can either be mostly stuck into the local minimum (premature convergence) or the results do not improved furthermore (stagnation). It seems that both phenomenons could be connected with the exploration and exploitation components of a process space. Črepinšek, Mernik, and Liu (2011) asserted that too much exploitation induces premature convergence, while too much exploration slows down the convergence. Several approaches could be used in order to avoid premature convergence or stagnation. For instance, Fister, Yang, Brest, and Fister (2013) proposed explicit balancing the process of exploration and exploitation regarding the diversity metric.

This paper focuses on the same problem, i.e., how to avoid premature convergence and/or stagnation within the FA algorithm by using the quaternion's representation of individuals. In mathematics, quaternions extend complex numbers. Quaternion algebra is connected with special features of geometry of the appropriate Euclidian spaces. Quaternions are especially appropriate within those areas where it is necessary to compose rotations with minimal computation, e.g., programming video games or controllers of spacecraft (Conway & Smith, 2003). For instance, 3-dimensional rotation can be specified by a single quaternion, while a pair of quaternions are needed for 4-dimensional rotation. Quaternion calculus is introduced within several physical applications, like crystallography, the kinematics of rigid body motion, the Thomas

* Corresponding author. Tel.: +386 31419573.
E-mail addresses: iztok.fister@uni-mb.si (I. Fister), x.yang@mdx.ac.uk (X.-S. Yang), janez.brest@uni-mb.si (J. Brest), iztok.fister2@uni-mb.si (I. Fister Jr.).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.06.070

precession, the special theory of relativity, and classical electromagnetism (Girard, 1984). A step forward in the popularization of quaternions was achieved by Lambek (1995), who stated that quaternions could provide a shortcut for those pure mathematicians who wished to familiarize themselves with certain aspects of theoretical physics.

The aim of selecting the proper problem representation is twofold. First, to encode all possible solutions and second, to apply operators that fit well to the properties of the representation (Rothlauf, 2006). Each optimization problem can be represented in many ways. Moreover, some representations are problem-specific. Representation affects the performances of SI algorithms essentially, although yet no theoretical evidence exists for this behavior. Therefore, designers are obliged to trust their intuition in selecting the appropriate representation.

When using a quaternions' representation, it is expected that a balance between the exploration and exploitation can be performed. A quaternion's search space is explored in place of exploring the original search space. That is, each 1-dimensional real-values element of the solution vector is mapped into a 4-dimensional quaternion. Although the quaternion search space is huge when compared to the original search space, exploring this space is easier because the fitness landscape, as determined by fitness function is less complex within this space. As a result, the search process is implicitly directed towards the more promising areas of the search space.

Our experimental work started with the original FA algorithm by optimizing a suite of well-known functions taken from literature (Yang, 2010). The purpose of the experiments was to show what impact the quaternion's representation of individuals into FA (also QFA) had on the results of optimization. The experiments showed that the results of the FA algorithm could be improved when the same algorithm was hybridized using the representation of individuals with quaternions (QFA). Moreover, when comparing the results of the QFA algorithm with the results of other algorithms like PSO, DE, and ABC it showed that these were also comparable with the other algorithms in the experiments.

1.1. Related work

The swarm intelligence principle is based on several unsophisticated entities that cooperate in order to exhibit the desired behavior, in place of a sophisticated controller that governs the global behavior of the system (Blum & Li, 2008). For instance, an ant as an individual does not imply much, but an ant colony is capable of performing significant tasks when cooperating (as building the anthills). However, no global master-plan exists for an individual. Moreover, ants do not even use direct communication between each other. In place of direct communication, they apply indirect communication via chemical pheromone trails that direct the individual ants to easily finding the shortest path. This behavior inspired Dorigo and Di Caro (1999) to develop ant colony optimization (ACO).

On the other hand, the social behavior of animals and insects, e.g., bird flocking, fish schooling, and animal herding, inspired Kennedy and Eberhart (1999) to develop the particle swarm optimization algorithm (PSO). Similarly to natural swarm of individuals (like birds) that move thorough the landscape seeking the regions rich with food, artificial particles representing a solution to the problem also moves through a search-space looking for optimal solutions.

Honey bees' behavior demonstrates all the characteristics needed for efficient meta-heuristic, i.e., moving bees in groups, exploration and exploitation, attraction of more promising regions of the landscape. Bees move in groups in order to transport nectar from a food source to a hive. This movement is self-organized and based on the interactions between bees, rather than from a hierarchical center (Beekman, Sword, & Simpson, 2008). Exploitation means the usage of existing information. That is, bees exploit areas rich with nectar. In contrast, exploration means the collection of new information. This information is realized by bees through scout bees that seek for new promising areas of the landscape. The mobilization of onlooker bees waiting for decision-making in order to exploit more attractive areas of a landscape is made through the so-named "waggle dance", where the scout informs the onlooker bees where a new food source rich with nectar can be found. The bees' behavior inspired Karaboga and Basturk (2007) to develop an artificial bee colony algorithm (ABC).

The bat algorithm (BA) that exploits the so called echolocation of bats was developed by Yang (2010). Bats use echo-locating as a sonar in order to detect and avoid obstacles. Bats have additional night vision because rays can be transmitted and received also in complete dark. It is generally known that sound pulses are transformed to frequency reflected from obstacles. Bats measure the time from emission to reflection and use it for their navigation. They emit very loud, ultrasonic sound pulses. The pulse rate is usually defined as 10 to 20 times per second. After hitting and reflecting, bats transform their own pulses to be informed about how far away the prey is.

Some cuckoo species behave as obligate brood parasites by laying their eggs in the nests of other host birds. This means that the host birds then nurture foreign offspring. This behavior inspired Yang and Deb (2009) to formulate a cuckoo search algorithm.

Swarm intelligence is concerned mainly with the design of intelligent multi-agent systems (Wooldridge, 2009). Algorithms from this field have been applicable primarily for optimization problems, and the control of robots. The more notable works relating to this paper with regard to swarm intelligence algorithms are as follows: ant colony optimization (Dorigo & Di Caro, 1999; Korošec, Šilc, & Filipič, 2012), particle swarm optimization (Kennedy & Eberhart, 1999), artificial bees colony optimization (Fister, Fister, Brest, & Žumer, 2012; Karaboga & Basturk, 2007), firefly algorithm (Fister, Yang, Fister, & Brest, 2012), cuckoo search (Yang & Deb, 2009), bat algorithm (Fister, Fister, & Yang, 2013), etc.

1.2. Structure of this paper

The structure of the reminder of this paper is as follows. Section 2 deals with optimization problems and their complexities. Quaternions' algebra is represented in Section 3. In Section 4, a detailed description follows of the FA using quaternion's representation of individuals. The experiments and result are discussed in Section 5. The paper concludes with a summary of our work and an outline of the directions of further work.

2. Complexity of optimization problems

An optimization problem can be formally defined as a quadrille $\langle I, S, f_o, g\rangle$, where

- I: is a set of instances $x \in I$,
- S: is the function that to each instance $x \in I$ assigns a set of feasible solutions $S(x)$,
- f_o: is the objective function that to each feasible solution $s \in S(x)$ of instance $x \in I$ assigns the value $f_0(s) \in \mathbb{R}$,
- g: is the goal that determines whether the minimum ($g = \min$) or maximum ($g = \max$) value of the objective function is searched for.

Usually, the fitness function f is used in place of the objective function f_o. If we suppose that $\max(f_o(s)) = \min(-f_o(s))$, then the goal g = max can always be transformed into g = min. In this manner, we always search for the minimum value of the transformed objective function. As a result, a definition of the optimization problem can be reduced to a triple $\langle I, S, f\rangle$. Note that the optimization problem can appear in three different forms, as follows:

- Construction form: The optimal solution s^* and to it the belonging value of the fitness function $f(s^*)$ needs to be calculated for the instance $x \in S$.
- Non-construction form: The optimal value of the fitness function $f(s^*)$ is needed for the instance $x \in S$.
- Decision form: For the particular instance $x \in S$, it should be discovered whether the optimal value of fitness function $f(s^*)$ is better than a certain prescribed constant K, more formally if $f(s^* \leqslant \text{K})$.

Further, optimization problems can be further divided into continuous, discrete, and mixed. Discrete optimization problems are also referred to as combinatorial. The decision variables for a continuous optimization problem can occupy values within the domain of real values whilst the decision variables for a combinatorial problem have discrete values. Mixed type has some variables taking continuous, whilst other variables take discrete values.

According to the number of objectives involved in the optimization problem, this can be divided into single-objective and multi-objective (also multi-criteria) (Zhou et al., 2011). The task of a single-objective optimization is to find the optimal solution according to only one objective function. When the optimization problem involves more than one objective function, the task is to find one or more optimal solutions regarding each objective (Deb, 2001). Here, a solution that is good with respect to one objective can be worse for another, and vice versa. Therefore, the goal of multi-objective optimization is to find a set of solutions that are optimal with respect to all other objectives, and such a set of solutions forms a so-called Pareto front. Interestingly, most real-world problems are multi-objective.

In some optimization problems, not all the possible combinations of decision variables represent feasible solutions. This class of problems is also known as constraint problems. Constraint problems can be divided into two different types: constraint satisfaction problems (CSP) and constraint optimization problems (COP). In contrast, if the problem is unconstrained, it is referred to as a free optimization problem (FOP) (Eiben & Smith, 2003). CSP is defined as a pair $\langle S, \phi\rangle$, where S denotes a search space and ϕ is a Boolean function on S that represents a feasibility condition. In fact, this function divides the search space S into feasible and infeasible regions. A solution of the CSP problem is each $s \in S$ with $\phi(s) = true$. On the other hand, COP is defined as a triple $\langle S, f, \phi\rangle$ where S denotes a search space, f is a real valued fitness function, and ϕ is a Boolean function on S. A solution of this problem is the $s \in S(x)$ with $\phi(s) = true$ and $S(x) = S(x^*)$.

Optimization problems for dynamic (also non-stationary) environments change over time. These changes can be implicitly defined in a dynamic environment or there exists uncertainty about the nature of the expected change, the magnitude of the changes, and the duration of any static period between changes (Morrison, 2004). Algorithms for solving these problems should be able to respond to changes in the environment. On the other hand, noise concerns a fitness function. For instance, the fitness function is noisy as a result of simulation/measurement errors or approximation errors (in the case where surrogates are used in place of the computationally expensive high fidelity fitness function).

Engineering applies scientific principles to design or develop structures, machines, processes, devices, and materials. That is, engineering applications use an optimization theory and apply it to problems arising in engineering. Most of these applications implement those numerical calculations involved in optimization algorithms that are intended for running on computer systems. In this sense, the modern swarm intelligence methods and applications are indispensable.

The time complexity of an algorithm determines the way in which the increase in the instance size influences the time complexity (Garey & Johnson, 1979). This relationship can be expressed by the so-called asymptotic time complexity function $O(f(n))$ that determines the upper-bound of time complexity for a given optimization problem. The algorithmic theory divides problems into two classes with regard to the asymptotic time complexity function: P-hard and NP-hard. Those problems that have polynomial time complexity $O(n^k)$ and are treated as easy belong to the first class. In contrast, problems of class NP-hard demonstrate the exponential time complexity $O(2^n)$ and are, therefore, treated as hard. That is, the exponential time complexity may cause that some increase in the input data can increase the solution time of the problem exponentially. In the worst case, we could be waiting for the solution over an infinite period of time.

3. Quaternions' algebra

Quaternions Hamilton (1899) 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 real numbers generated by basic units i, j, k (also the imaginary part) that satisfy Hamilton's equations:

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

The quaternions $q \in \mathbb{H}$ 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 (Eberly, 2002):

- *addition and subtraction*: is defined by

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

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

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

Multiplication is not commutative because the product $q_0 q_1 \neq q_1 q_0$ in general.

- *conjugate*: is a unary arithmetical operation defined by

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

The conjugation of quaternions satisfies the properties $(q^*)^* = q$ and $(q_0 q_1)^* = q_0^* q_1^*$.

- *norm*: is defined by

$$\|q\| = \|x_0 + x_1 i + x_2 j + x_3 k\| = \sqrt{x_0^2 + x_1^2 + x_2^2 + x_3^2}. \tag{5}$$

The norm is a real-valued function that satisfies the properties $\|q^*\| = \|q\|$ and $\|q_0 q_1\| = \|q_0\|\|q_1\|$. This function is suitable for mapping the elements of vector (individual, solution) from a 4-dimensional quaternion to a 1-dimensional real-valued element in phenotype space.

- *multiplicative inverse*: of quaternion q is denoted as q^{-1} and has the property $qq^{-1} = q^{-1}q = 1$. It is constructed as

$$q^{-1} = q^*/\|q\|, \tag{6}$$

where the division of a quaternion by a real-value scalar is the division of each component by the 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

$$q_0/q_1 = q_0 q_1^{-1}. \tag{7}$$

- *distance*: of quaternions q_0 and q_1 is defined by

$$dist(q_0, q_1) = \sqrt{(x_0 - y_0)^2 + (x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}. \tag{8}$$

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

- *qrand*: is a quaternion defined as

$$qrand() = \{x_i = N(0,1) \mid \text{for } i = 0 \ldots 3\}, \tag{9}$$

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

$$qzero() = \{x_i = 0 \mid \text{for } i = 0 \ldots 3\}, \tag{10}$$

where each component of quaternion is initialized with zero.

These operations of the quaternion algebra serve as a reach basis for developing the different variation operators used in swarm intelligence. The rest of the paper presents the usage of these operations by developing the operators within the new FA algorithm with the quaternion's representation of individuals, in detail.

4. Firefly algorithm using a quaternion's representation of individuals

Fireflies (Coleoptera: Lampyridae) are well known for bioluminescent signaling, which is used for species recognition and mate choice (Long et al., 2012). Bioluminiscence that comprises a complicated set of chemical reactions is not always a sexual signal only but also warns off potential predators. In reminder of the paper, the original FA algorithm is described that captures the bioluminiscent behavior of fireflies within fitness function. Further, this algorithm is then modified using the quaternion's representation of individuals (QFA).

4.1. Original firefly algorithm

The light-intensity I of flashing firefly decreases as the distance from source r decreases in terms of $I \propto 1/r^2$. Additionally, the air absorption causes the light to become weaker and weaker as the distance from the source increases. This flashing light represented the inspiration for developing the FA algorithm by Yang (2008). Here, the light-intensity is proportional to the objective function of the problem being optimized (i.e., $I(\mathbf{s}) \propto f(\mathbf{s})$, where $\mathbf{s} = S(\mathbf{x})$ represent a candidate solution) (Fister, Fister, Yang, & Brest, 2013).

In order to formulate the FA, some flashing characteristics of fireflies were idealized, as follows:

- All fireflies are unisex.
- Their attractiveness is proportional to their light intensity.
- The light intensity of a firefly is affected or determined by the landscape of the objective function.

Note that light-intensity I and attractiveness are in some way synonymous. While the intensity I is referred to as an absolute measurement of emitted light by firefly, the attractiveness is relative measurement the light that should be seen in the eyes of beholders and judged by the other fireflies (Yang, 2008). The light intensity I varied with distance r is expressed by the following equation

$$I(r) = I_0 e^{-\gamma r^2}, \tag{11}$$

where I_0 denotes the intensity light at the source, and γ is a fixed light absorption coefficient. Similarly, the attractiveness β that also depends on the distance r is calculated according to the following generalized equation

$$\beta(r) = \beta_0 e^{-\gamma r^2}, \quad \text{for } k \geqslant 1. \tag{12}$$

The distance between two fireflies i and j is represented as the Euclidian distance

$$r_{ij} = \|\mathbf{s}_i - \mathbf{s}_j\| = \sqrt{\sum_{k=1}^{D} s_{ik} - s_{jk}}, \tag{13}$$

where s_{ik} is the k-th element of the i-th firefly position within the search-space, and D denotes the dimensionality of a problem. Each firefly i moves to another more attractive firefly j, as follows

$$\mathbf{s}_i = \mathbf{s}_i + \beta_0 e^{-\gamma r_{ij}^2}(\mathbf{s}_j - \mathbf{s}_i) + \alpha \cdot N_i(0,1). \tag{14}$$

Eq. (14) consists of three terms. The first term determines the position of the i-th firefly. The second term refers to the attractiveness, while the third term is connected with the randomized move of the i-th firefly within the search-space. This term consists of the randomization parameter α, and the random numbers $N_i(0,1)$ drawn from a Gaussian distribution. The scheme of FA is sketched in Algorithm 1.

Algorithm 1. Original FA algorithm

```
1: $t = 0$; $\mathbf{s}^* = \emptyset$; $\gamma = 1.0$; // initialize: gen.counter, best
   solution, attractiveness
2: $P^{(t)} = \text{InitFA}()$;   // initialize the firefly population
   $\mathbf{s_i^{(0)}} \in \mathbf{P}^{(0)}$
3: while $t \leqslant$ MAX_GEN
4:    $\alpha^{(t)} = \text{AlphaNew}()$;   // determine a new value of $\alpha$
5:    EvaluateFA($\mathbf{P}^{(t)}$,$f(\mathbf{s})$);   // evaluate $\mathbf{s}_i^{(t)}$ according to $f(\mathbf{s}_i)$
6:    OrderFA($\mathbf{P}^{(t)}$,$f(\mathbf{s})$);   // sort $\mathbf{P}_i^{(t)}$ according to $f(\mathbf{s}_i)$
7:    $s^*$ = FindTheBestFA($\mathbf{P}^{(t)}$,$f(\mathbf{s})$);   // determine the best
   solution $\mathbf{s}^*$
8:    $\mathbf{P}^{(t+1)}$ = MoveFA($\mathbf{P}^{(t)}$);   // vary attractiveness according
   Eq. (14)
9:    $t = t + 1$;
10: endwhile
11: return $\mathbf{s}^*, f(\mathbf{s})$;   // post process
```

The FA algorithm (Algorithm 1) runs on the population of fireflies $P^{(t)}$ that are represented as real-valued vectors $\mathbf{s}_i^{(t)} = s_{i0}^{(t)}, \ldots s_{in}^{(t)}$, where $i = 1 \ldots NP$ and NP denotes the number of fireflies in population $P^{(t)}$ at generation t. Note that each firefly $\mathbf{s}_i^{(t)}$ is of dimension D. The population of fireflies is initialized randomly (function InitFFA) according to equation

$$\mathbf{s}_{ij}^{(0)} = (ub_i - lb_i) \cdot \mathrm{rand}(0,1) + lb_i, \tag{15}$$

where ub_i and lb_i denote the upper and lower bounds, respectively. The main loop of the firefly search process that is controlled by the maximum number of generations *MAX_GEN* consists of the following functions:

- AlphaNew: calculating new values for the randomization parameter α. This parameter is modified according to the following equation

$$\begin{aligned} \Delta &= 1 - 10^{-4}/0.9^{1/MAX\_GEN}, \\ \alpha^{(t+1)} &= 1 - \Delta \cdot \alpha^{(t)}, \end{aligned} \tag{16}$$

where Δ determines the step size of changing the parameter $\alpha^{(t+1)}$. Note that this parameter monotony descends with the increasing of generation counter t.

- EvaluateFA: evaluating the new solution $\mathbf{s}_i^{(t)}$ according to a fitness function $f(\mathbf{s}^{(t)})$, where $\mathbf{s}_i^{(t)} = S(\mathbf{x}_i^{(t)})$.
- OrderFA: ordering solutions $\mathbf{s}_i^{(t)}$ for $i = 1 \ldots NP$ with respect to the fitness function $f(\mathbf{s}_i^{(t)})$ ascending, where $\mathbf{s}_i^{(t)} = S(\mathbf{x}_i^{(t)})$.
- FindTheBestFA: determining the best solution in the population $P^{(t)}$. Normally, the best solution becomes $\mathbf{s}^* = \mathbf{s}_0^{(t)}$.
- MoveFA: moving the fireflies towards the search space according to the attractiveness of their neighbors' solution (Eq. (14)).

In the reminder of this paper, the modification using quaternions is discussed in more detail.

4.2. Modified firefly algorithm

The modified FA is based on the original FA, where the representation of virtual fireflies is moved from an Euclidian space to a quaternion space. In the Euclidian space, each virtual firefly is represented as D-dimensional real-values vector $\mathbf{s}_i = \{s_{i0}, \ldots, s_{in}\}$, where $s_{ij} \in \mathbb{R}^n$, while in quaternion space as a D-dimensional vector of quaternions $\mathbf{q}_i = \{q_{i0}, \ldots, q_{in}\}$, where $q_{ij} \in \mathbb{H}^n \wedge \mathbb{H} \in \mathbb{R}^4$. In line with this, it is expected that although the quaternion space is more complex, on the one hand, it is smoother for exploration, on the other hand. Therefore, the search-process could be directed towards the more promising areas of the search-space. The modified algorithm can be seen in Algorithm 2.

Algorithm 2. Modified QFA algorithm

```
1: $t = 0$; $\mathbf{q}^* = Qzero()$; $\gamma = 1.0$;   // initialize: gen.counter,
   best solution, attractiveness
2: $Q^{(t)}$ = InitQFA();   // initialize the firefly population
   $\mathbf{x}_i^{(0)} \in \mathbf{P}^{(0)}$
3: while $t \leqslant$ MAX_GEN do
4:    $\alpha^{(t)}$ = AlphaNew();   // determine a new value of $\alpha$
5:    EvaluateQFA($\mathbf{Q}^{(t)}$,$f(\|\mathbf{q}\|)$);   // evaluate $\mathbf{q}_i^{(t)}$ according to
   $\|\mathbf{q}_i\|$
6:    OrderQFA($\mathbf{Q}^{(t)}$, $\|\mathbf{q}\|$);   // sort $\mathbf{Q}^{(t)}$ according to $\|\mathbf{q}_i\|$
7:    $\mathbf{q}^*$ = FindTheBestQFA($\mathbf{Q}^{(t)}$, $\|\mathbf{q}\|$);   // determine the best
   solution $\mathbf{q}^*$
8:    $\mathbf{Q}^{(t+1)}$ = MoveQFA($\mathbf{Q}^{(t)}$);   // vary attractiveness
   according Eq. (14)
9:    $t = t + 1$;
10: endwhile
11: return $\mathbf{q}^*$, $\|\mathbf{q}\|$;   // post process
```

The modified FA differs from the original FA by using the quaternion's representation of individuals. In place of operations in Euclidian space, quaternion algebra is employed in order to move virtual fireflies towards the more promising areas of search space. In fact, each 1-dimensional element of fireflies is represented by the 4-dimensional quaternion's representation. On this representation, however, the quaternion algebra is applied.

The QFA algorithm acts as follows. The population of quaternions is initialized in *InitQFA*() using the *qrand*() function. The solution $\mathbf{s} = (s_0, \ldots, s_D)$ in the Euclidian space is obtained from i-th quaternions' vector $\mathbf{q}_i$ using the norm function as follows:

$$s_j = \|q_{ij}\|, \quad \text{for } j = 1 \ldots D, \tag{17}$$

and evaluated by function *EvaluateQFA*() the same as in the original FA. The ordering of the virtual fireflies in the population stays unchanged in the *OrderQFA*() function. The same is also true when finding the best solution in the *FindTheBestQFA*() function. The focus of the modified algorithm represents the *MoveQFA*() function that moves a population of virtual fireflies through the quaternion's search-space. Calculating the distance between the fireflies in the search-space is expressed in the modified algorithm as follows:

$$r_{ij} = dist(\mathbf{q}_i, \mathbf{q}_j), \tag{18}$$

where $\mathbf{q}_i$ is the i-th virtual firefly position, and $\mathbf{q}_j$ is the j-th virtual firefly position in the search-space. Moving the firefly i to another more attractive firefly j is expressed as follows:

$$\mathbf{q}_i = \mathbf{q}_i + \beta_0 e^{-\gamma r_{ij}^2}(\mathbf{q}_j - \mathbf{q}_i) + \alpha \cdot \varepsilon \cdot Qrand(), \tag{19}$$

where r_{ij} represents the distance between the i-th and j-th fireflies in the quaternion's space, α is the randomization parameter, ε the scale, and the *Qrand*() is a random generated quaternion vector. After moving the virtual fireflies, a verification function is launched. It ensures that the new firefly position is under the prescribed limitations, i.e., $lb_i \leqslant \|q_i\| \leqslant ub_i$.

5. Experiments and results

The goal of our experimental work was to show that quaternions can be appropriate for the representation of individuals in QFA., Essentially, this paper focuses on the problem of whether this representation of individuals can reduce the stagnation that has often arisen in the original FA algorithm. In line with this, the developed QFA was applied to the function optimization problems as proposed by Yang (2010). The aim of the experimental work was twofold: on the one hand to show that QFA improves the results of the original FA algorithm, and on the other hand that the obtained results were comparable with the results of the other EA algorithms like DE, and the other SI algorithms like BA and ABC.

As mention before, this study focuses on solving the function optimization problem. The function optimization belongs to a class of continuous optimization problems and is defined as follows. Let us assume, an objective function $f(\mathbf{s})$ is given, where $\mathbf{s} = (s_1, \ldots, s_D)$ is a vector of D design variables in a decision space S. The design variables $s_j \in \{lb_j, ub_j\}$ 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 reminder of this paper, the test suite is described, the experimental setup is presented, the configuration of the PC is illustrated on which these experiments were run, and finally the obtained results are presented. These results are then analyzed in the sense of the appointed experimental goals.

5.1. Test suite

The test suite consisted of ten functions which were selected from two references. The first five functions were taken from Karaboga's paper (Karaboga & Basturk, 2007), in which the ABC algorithm was introduced, while the last five were from the paper of Yang (2010) that proposed a set of optimization functions suitable for testing the newly-developed algorithms.

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, can be summarized as follows:

- Griewangk's function:

$$f_1(\mathbf{s}) = -\prod_{i=1}^{D}\cos\left(\frac{s_i}{\sqrt{i}}\right) + \sum_{i=1}^{D}\frac{s_i^2}{4000} + 1, \tag{20}$$

where $s_i \in [-600, 600]$. The function has the global minimum $f^* = 0$ at $\mathbf{x}^* = (0, 0, \ldots, 0)$. It is highly multimodal, when the number of variables is higher than 30.

- Rastrigin's function:

$$f_2(\mathbf{s}) = D * 10 + \sum_{i=1}^{D}(s_i^2 - 10\cos(2\pi s_i)), \tag{21}$$

where $s_i \in [-15, 15]$. The function has the global minimum $f^* = 0$ at $\mathbf{x}^* = (0, 0, \ldots, 0)$ and is also highly multimodal.

- Rosenbrock's function:

$$f_3(\mathbf{s}) = \sum_{i=1}^{D-1}100(s_{i+1} - s_i^2)^2 + (s_i - 1)^2, \tag{22}$$

where $s_i \in [-15, 15]$ and whose global minimum $f^* = 0$ is at $\mathbf{s}^* = (1, 1, \ldots, 1)$. This function, also known as the 'banana function' has several local optima. Gradient-based algorithms are especially difficult to converge to the global optima by optimizing this function.

- Ackley's function:

$$f_4(\mathbf{s}) = \sum_{i=1}^{D-1}(20 + e - 20e^{-0.2\sqrt{0.5(s_{i+1}^2+s_i^2)}} - e^{0.5(\cos(2\pi s_{i+1})+\cos(2\pi s_i))}), \tag{23}$$

where $s_i \in [-32.768, 32.768]$. The function has the global minimum $f^* = 0$ at $\mathbf{s}^* = (0, 0, \ldots, 0)$ and it is highly multimodal.

- Schwefel's function:

$$f_5(\mathbf{s}) = 418.9829 * D - \sum_{i=1}^{D}s_i\sin\left(\sqrt{|s_i|}\right), \tag{24}$$

where $s_i \in [-500, 500]$. The Schwefel's function has the global minimum $f^* = 0$ at $\mathbf{s}^* = (1, 1, \ldots, 1)$ and is highly multimodal.

- De Jong's sphere function:

$$f_6(\mathbf{s}) = \sum_{i=1}^{D}s_i^2, \tag{25}$$

where $s_i \in [-600, 600]$ and whose global minimum $f^* = 0$ is at $\mathbf{s}^* = (0, 0, \ldots, 0)$. The function is unimodal and convex.

- Easom's function:

$$f_7(\mathbf{s}) = -(-1)^D\left(\prod_{i=1}^{D}\cos^2(s_i)\right)\exp\left[-\sum_{i=1}^{D}(s_i - \pi)^2\right], \tag{26}$$

where $s_i = [-2\pi, 2\pi]$. The function has several local minimum and the global minimum $f^* = -1$ at $\mathbf{s}^* = (\pi, \pi, \ldots, \pi)$.

- Michalewicz's function:

$$f_8(\mathbf{s}) = -\sum_{i=1}^{D}\sin(s_i)\left[\sin\left(\frac{is_i^2}{\pi}\right)\right]^{2\cdot 10}, \tag{27}$$

where $s_i = [0, \pi]$. The function has the global minimum $f^* = -1.8013$ at $\mathbf{s}^* = (2.20319, 1.57049)$ in two-dimensional parameter space. In general, it has several local optima.

- Xin-She Yang's function:

$$f_9(\mathbf{s}) = \left(\sum_{i=1}^{D}|s_i|\right)\exp\left[-\sum_{i=1}^{D}\sin(s_i^2)\right], \tag{28}$$

where $s_i = [-2\pi, 2\pi]$. The function is not smooth because it has several local optima and the global minimum $f^* = 0$ at $\mathbf{s}^* = (0, 0, \ldots, 0)$.

- Zakharov's function:

$$f_{10}(\mathbf{s}) = \sum_{i=1}^{D}s_i^2 + \left(\frac{1}{2}\sum_{i=1}^{D}is_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{D}is_i\right)^4, \tag{29}$$

where $s_i = [-5, 10]$. The function has the global minimum $f^* = 0$ at $\mathbf{s}^* = (0, 0, \ldots, 0)$ with no local optima.

The lower and upper bounds of the design variables determine intervals that limit the size of the search space. The wider this interval, the wider the search space. Note that the intervals 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.

5.2. Experimental setup

In this experimental study, the results of the following algorithms were compared: FA, DE, BA, ABC, and QFA. In fact, DE belongs to the evolutionary algorithms, while the other algorithms are members of the swarm intelligence community. As a termination condition, the number of fitness function evaluations (*FEs*) was considered, in this paper. This value depends on the dimension of the problem regarding the expression $FEs = 5000 \cdot D$. The maximal number of generations (MAX_T) is expressed as $MAX\_T = FEs/NP$, where *NP* represents the population size. 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 FA parameters were set as follows: $\alpha = 0.1$, $\beta = 0.2$, and $\gamma = 0.9$. The same parameters were also used for the QFA algorithm because both algorithms are the same except in representation of individuals.

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 DE parameters were configured as follows: the amplification factor of the difference vector $F = 0.9$, and the crossover control parameter $CR = 0.5$. 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, as follows: the *Best*, the *Worst*, the *Mean*, the *Stdev*, and the *Median* values.

5.3. 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 – 4 GB DDR3
3. Operating system – Linux Mint 12

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

5.4. Results

An aim of our experimental work was to find answers to the following questions:

- what influence the number of evaluations had on the results during the QFA run,
- what influence the dimensionality of problems had on the results of QFA,
- how good are the results of QFA when comparing with the results of other optimization algorithms, like FA, DE, BA, and ABC,
- how the quaternion's representation of individuals can reduce the stagnation in the original FA algorithm.

In the rest of paper, answers to these questions are quantified in detail.

5.4.1. The influence of the number of evaluations

In this experiment, the best fitness values were recorded over three different phases of the QFA's search process, that is:

- at the beginning (i.e., $\frac{1}{5}$ of the maximum number of fitness evaluations),
- in the middle (i.e., $\frac{1}{2}$ of the maximum number of fitness evaluations),
- at the end of the run.

Indeed, the behavior of QFA was observed. The results of this experiment by optimizing the functions of dimension $D = 30$ are illustrated in Table 1. In this table, the first measuring was taken after 30,000 evaluations of fitness function, the second after 75,000, and third after 150,000 evaluations of fitness function.

As can be seen from Table 1, the stagnation problem cannot be detected during the QFA search process.

That is, the *Mean* values by optimizing the functions from the test suite constantly decreased, when the number of evaluations increased. The same behavior was also observed for other measured values, like the *Worst*, the *Best*, the *Stdev*, and the *Median*.

Moreover, function f_7 converged into the value zero by more than $\frac{1}{5}$ of the maximum number of fitness evaluations *MAX_FE*.

5.4.2. The influence of 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 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 QFA algorithm optimizing ten test functions are presented in Table 2, from which it can be seen that the QFA algorithm achieved the better results when optimizing the functions with lower dimensions. Therefore, the best results were obtained by optimizing the functions with dimension $D = 10$.

Table 1
Detailed results of QFA ($D = 30$).

Evaluations	Measures	f_1	f_2	f_3	f_4	f_5
3.00E+004	Best	6.27E+00	3.32E+02	3.12E+04	6.82E+00	3.68E+03
	Worst	1.18E+02	4.17E+02	1.71E+05	2.03E+01	6.84E+03
	Mean	3.20E+01	3.67E+02	1.03E+05	8.46E+00	5.10E+03
	Stdev	1.99E+01	2.20E+01	3.73E+04	2.51E+00	7.28E+02
	Median	2.98E+01	3.70E+02	1.04E+05	8.13E+00	5.09E+03
7.50E+004	Best	6.27E+00	2.60E+02	6.35E+03	5.40E+00	3.08E+03
	Worst	1.91E+01	3.27E+02	6.84E+04	2.03E+01	6.08E+03
	Mean	1.49E+01	2.90E+02	1.95E+04	6.44E+00	4.41E+03
	Stdev	2.79E+00	1.81E+01	1.19E+04	2.89E+00	6.67E+02
	Median	1.51E+01	2.93E+02	1.63E+04	5.93E+00	4.44E+03
1.50E+005	Best	5.02E+00	5.42E+01	4.48E+01	7.17E−01	2.62E+03
	Worst	7.35E+00	1.92E+02	9.92E+02	2.00E+01	5.52E+03
	Mean	6.07E+00	1.20E+02	2.94E+02	2.59E+00	3.93E+03
	Stdev	4.56E−01	3.03E+01	2.99E+02	3.66E+00	6.71E+02
	Median	6.09E+00	1.15E+02	1.71E+02	1.93E+00	3.94E+03
		f_6	f_7	f_8	f_9	f_{10}
3.00E+004	Best	2.11E+004	0.00E+000	−1.43E+001	1.11E−009	1.90E+001
	Worst	3.77E+005	0.00E+000	−9.04E+000	3.02E−007	4.63E+001
	Mean	1.17E+005	0.00E+000	−1.12E+001	7.61E−008	2.98E+001
	Stdev	5.84E+004	0.00E+000	1.13E+000	8.61E−008	7.75E+000
	Median	1.10E+005	0.00E+000	−1.10E+001	5.66E−008	2.88E+001
7.50E+004	Best	2.11E+004	0.00E+000	−1.51E+001	1.02E−010	7.62E+000
	Worst	3.77E+005	0.00E+000	−1.07E+001	9.69E−008	1.60E+001
	Mean	6.95E+004	0.00E+000	−1.29E+001	8.81E−009	1.19E+001
	Stdev	6.48E+004	0.00E+000	1.10E+000	2.05E−008	2.24E+000
	Median	5.69E+004	0.00E+000	−1.27E+001	2.62E−009	1.16E+001
1.50E+005	Best	1.61E+004	0.00E+000	−2.29E+001	1.23E−011	6.56E−002
	Worst	3.77E+005	0.00E+000	−1.68E+001	1.73E−011	2.90E−001
	Mean	3.37E+004	0.00E+000	−1.96E+001	1.49E−011	1.57E−001
	Stdev	7.14E+004	0.00E+000	1.39E+000	1.29E−012	5.51E−002
	Median	1.97E+004	0.00E+000	−1.95E+001	1.49E−011	1.47E−001

Table 2
Results of QFA according to dimensions.

Dimension	Measure	f_1	f_2	f_3	f_4	f_5
10	Best	2.08E+00	1.09E+01	7.41E+00	1.24E−01	4.15E+02
	Worst	2.75E+00	4.79E+01	3.67E+02	2.00E+01	1.62E+03
	Mean	2.52E+00	2.25E+01	6.23E+01	2.90E+00	1.10E+03
	Stdev	1.77E−01	9.82E+00	9.27E+01	6.44E+00	2.86E+02
	Median	2.53E+00	2.10E+01	1.08E+01	4.18E−01	1.11E+03
30	Best	5.02E+00	5.42E+01	4.48E+01	7.17E−01	2.62E+03
	Worst	7.35E+00	1.92E+02	9.92E+02	2.00E+01	5.52E+03
	Mean	6.07E+00	1.20E+02	2.94E+02	2.59E+00	3.93E+03
	Stdev	4.56E−01	3.03E+01	2.99E+02	3.66E+00	6.71E+02
	Median	6.09E+00	1.15E+02	1.71E+02	1.93E+00	3.94E+03
50	Best	7.51E+00	1.55E+02	7.70E+01	1.55E+00	5.14E+03
	Worst	9.81E+00	2.96E+02	3.44E+03	2.02E+01	8.53E+03
	Mean	9.02E+00	2.21E+02	5.91E+02	8.57E+00	6.81E+03
	Stdev	5.84E−01	3.98E+01	9.95E+02	8.82E+00	9.12E+02
	Median	9.20E+00	2.16E+02	2.30E+02	2.29E+00	6.77E+03
		f_6	f_7	f_8	f_9	f_{10}
10	Best	5.76E+003	0.00E+000	−8.35E+000	2.35E−014	1.77E−003
	Worst	7.03E+003	0.00E+000	−4.81E+000	2.14E−003	2.37E−002
	Mean	6.80E+003	0.00E+000	−7.06E+000	1.64E−004	6.93E−003
	Stdev	3.62E+002	0.00E+000	7.84E−001	5.68E−004	5.65E−003
	Median	7.03E+003	0.00E+000	−7.04E+000	3.16E−014	4.75E−003
30	Best	1.61E+004	0.00E+000	−2.29E+001	1.23E−011	6.56E−002
	Worst	3.77E+005	0.00E+000	−1.68E+001	1.73E−011	2.90E−001
	Mean	3.37E+004	0.00E+000	−1.96E+001	1.49E−011	1.57E−001
	Stdev	7.14E+004	0.00E+000	1.39E+000	1.49E−011	5.51E−002
	Median	1.97E+004	0.00E+000	−1.95E+001	1.29E−012	1.47E−001
50	Best	2.88E+004	0.00E+000	−3.40E+001	5.16E−020	3.75E−001
	Worst	3.77E+004	0.00E+000	−2.32E+001	1.33E−019	1.39E+000
	Mean	3.22E+004	0.00E+000	−2.85E+001	6.59E−020	8.63E−001
	Stdev	2.15E+003	0.00E+000	−2.84E+001	6.08E−020	2.69E−001
	Median	3.19E+004	0.00E+000	2.58E+000	1.97E−020	8.43E−001

Stagnation during the QFA search process was also unobserved for any of the dimensions taken into account, i.e., $D = 10$, $D = 30$, and $D = 50$. On average, optimizing the functions with higher dimensions was more difficult than for those with lower dimensions.

5.4.3. Comparative study

The intention of this subsection was to show how the new representation of individuals in QFA can improve the results of an original FA and how good these results are when compared with the other well known algorithms, like DE, BA, and ABC. In line with this, all algorithms optimized the same suite of ten functions under the same conditions. Finally, the results of this comparison were substantiated using the Friedman statistical tests in order to evaluate the obtained results.

The results of this experimental study are illustrated in Table 3, where the mean values and corresponding standard deviations are presented according to ten functions (f_1 to f_{10}) and five algorithms (FA, DE, BA, ABC, and QFA). Although experiments in all dimensions were performed, only the results for dimension $D = 50$ are presented. In summary, each function was optimized $5 \times 25 \times 3 = 375$ times (the number of algorithms $\times$ the number of runs $\times$ the number of dimensions).

As can be seen from Table 3, the ABC algorithm reached the best results optimizing almost the all functions except the function f_{10}, where the QFA algorithm was more successful, and f_7, where all algorithms achieved the same results, i.e., zero. Three Friedman non-parametric tests were conducted according five characteristic measures that captured the results with regard to different dimensions.

In order to estimate the quality of results, the Friedman test was conducted. The Friedman test (Friedman, 1937, 1940) 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 (Demšar, 2006) 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 Demšar (2006).

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. Two algorithms are significantly different if their intervals in Fig. 1 do not overlap.

The first diagram in Fig. 1 shows that the ABC algorithm significantly outperforms the results of the other four algorithms, i.e., QFA, BA, FA, and DE, according to dimension $D = 10$. Amongst the other four algorithms, the QFA was substantially better than the other algorithms in test. The situation remained the same even when the results were compared regarding the dimensions $D = 30$ and $D = 50$. That is, the ABC significantly improved the results of the other four algorithms, whilst the QFA was substantially better than the other algorithms in the test.

In summary, these results showed that the quaternion's representation of individuals substantially improved the original FA algorithm (Hypothesis I) and that the obtained results were comparable with other algorithms in the test except for the ABC (Hypothesis II). In order to improve the results of ABC, however, other mechanisms like self-adaptation, etc., should be applied to QFA. Thus, both two placed hypothesis were approved.

Table 3
Detailed results of algorithms ($D = 50$).

Function	Measure	FA	DE	BA	ABC	QFA
f_1	Mean	9.56E−001	1.39E+003	1.57E+002	1.05E−001	9.02E+000
	Stdev	8.53E−002	1.08E+002	3.81E+001	5.45E−002	5.84E−001
f_2	Mean	5.19E+002	1.55E+004	1.80E+003	2.29E+001	2.21E+002
	Stdev	3.52E+001	1.55E+003	4.10E+002	7.14E+000	3.98E+001
f_3	Mean	1.54E+004	1.59E+009	1.14E+006	1.72E+002	5.91E+002
	Stdev	1.36E+004	3.12E+008	7.38E+005	1.30E+002	9.95E+002
f_4	Mean	2.12E+001	2.09E+001	1.42E+001	2.05E+000	8.57E+000
	Stdev	3.32E−002	3.32E−002	8.88E−001	3.57E−001	8.82E+000
f_5	Mean	1.65E+004	2.83E+003	1.57E+004	2.49E+003	6.81E+003
	Stdev	5.09E+002	5.12E+002	7.40E+002	2.36E+002	9.12E+002
f_6	Mean	1.43E+001	5.60E+006	6.05E+005	3.79E−002	3.22E+004
	Stdev	2.25E+000	5.64E+005	1.30E+005	5.35E−002	2.15E+003
f_7	Mean	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
	Stdev	0.00E+000	0.00E+000	0.00E+000	0.00E+000	0.00E+000
f_8	Mean	−3.60E+000	−5.86E+000	−7.30E+000	−9.63E+000	−7.06E+000
	Stdev	6.72E−001	1.05E+000	4.15E−001	2.42E−002	7.84E−001
f_9	Mean	2.85E−002	1.32E−003	3.86E−003	4.54E−004	1.64E−004
	Stdev	2.17E−002	6.33E−004	7.59E−004	1.82E−009	5.68E−004
f_{10}	Mean	3.27E+005	5.13E+003	5.20E+002	5.34E+002	8.63E−001
	Stdev	2.33E+005	2.44E+003	1.52E+002	3.96E+001	2.69E−001

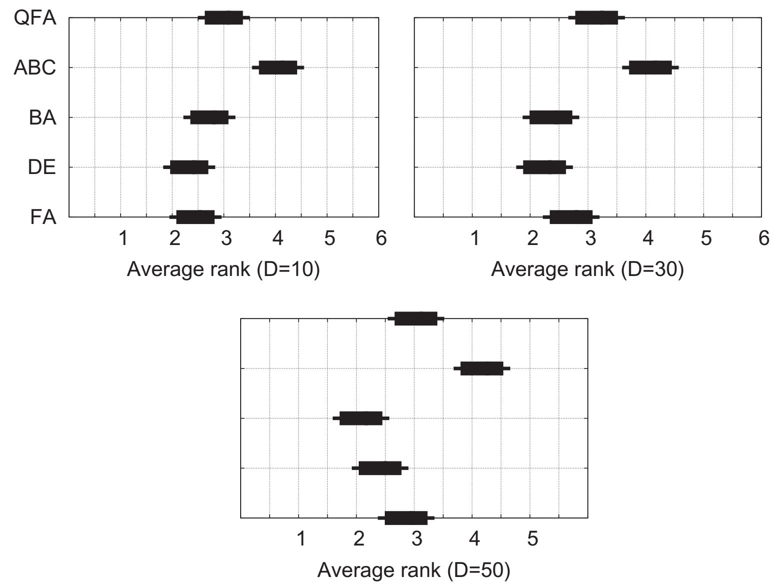

Fig. 1. Results of the Friedman non-parametric test.

5.4.4. Convergence plots

Finally, the question was posted as to how quaternion's representation of individuals reduces the stagnation within the QFA search process. In line with this, those convergence graphs were analyzed that were obtained by acting the search processes within the QFA and FA algorithms by optimizing the functions f_2, f_3, and f_{10} for $D = 10$. Two convergence diagrams were plotted for each observed function. The former represents the behavior of the best value modified through the generations, whilst the latter the average values modified throughout the generations. Note that the corresponding maximum number of fitness evaluations was fixed at 50,000 by $D = 10$.

All three best and average convergence plots in Figs. 2–4 illustrate the similar behavior of algorithms QFA and FA when optimizing the functions f_2, f_3, and f_{10}. That is, the best value persistently decreased when the number of generations was increased by the QFA. In contrast, the best value of the FA decreased to the some extent and then did not improve anymore (stagnation). The average value by the QFA algorithm describes a ridged curve, in contrast to FA, where the curve is smoother.

On the other hand, the curve decreased persistently by the QFA, whilst it became stable after some generations by the FA. As a result, the QFA search process retained a higher population diversity that seemed to avoid the algorithm to getting stuck or falling into stagnation.

Summary, the quaternion's representation of individuals within the QFA algorithm caused that the search process not to fall into stagnation. In contrast, the convergence time increased slightly, but it seems that the solution will be found when the maximum number of fitness evaluations is increased.

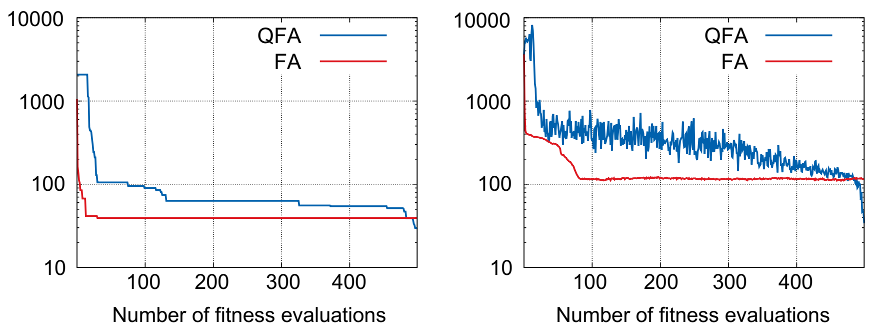

Fig. 2. Best and average convergence plots for f_2 $(D = 10)$.

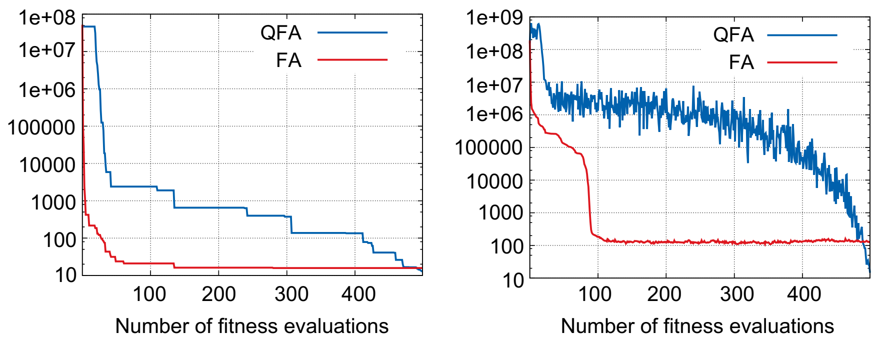

Fig. 3. Best and average convergence plots for f_3 $(D = 10)$.

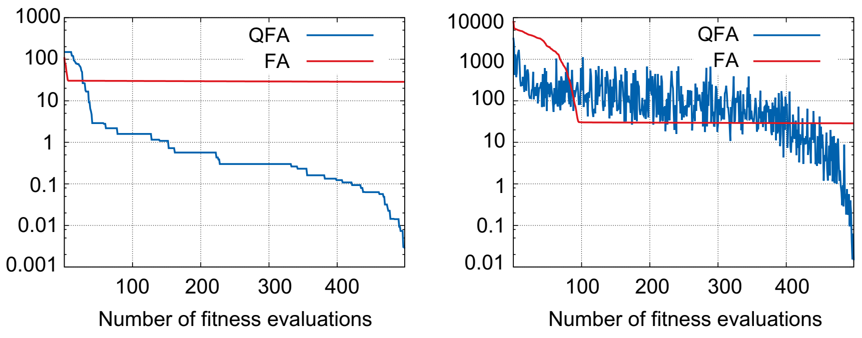

Fig. 4. Best and average convergence plots for f_{10} $(D = 10)$.

6. Conclusion

This paper proposed a novel representation of individuals with quaternions in the QFA algorithm. Quaternions are especially appropriate for programming the video games and controllers of spacecraft, where it is necessary to compute rotations with minimal computation. On the other hand, they have been successfully applied to the problems of theoretical physics. Quaternions are rarely applied to solving the optimization problems.

Representation of individuals in population-based algorithms (like EA and SI) has a crucial effects on the performance of such algorithms. Additionally, the appropriate representation demands also powerful operators that are able to take care on the exploration of new solutions. These operators should not be discriminatory to some areas of the search space. That is, each solution in definite search space must be achieved with the some probability using these parameters.

This paper proposes the quaternion's representation of individuals in the QFA population-based algorithm that maps each 1-dimensional real-valued element to 4-dimensional quaternion. This representation seems to increase the search space enormously, but it turns out that exploring this quaternion's search space is more effectively. The primary intention of this study was to show that the stagnation problem often arisen in the SI algorithms could be mitigated or even avoided using the quaternion's representation of individuals. Beside the original FA algorithm, the second speculation should also be confirmed on the other SI algorithms, in general.

However, this study has brought a new way in solving the hardest optimization problems. As this paper is the preliminary study only, many alternative ways in using the quaternion's representation of individuals remains still open for discovering in the future. At first, what effect has the span function on the results of optimization.

Acknowledgement

This work was supported in part by the Slovenian Research Agency under program P2-0041 "Computer Systems, Methodologies, and Intelligent Services". Authors are very thankful to professor Peter Cafuta, Institute for robotics and automatics for his discussions on the quaternions.

References

Beekman, M., Sword, G. A., & Simpson, S. J. (2008). Biological foundations of swarm intelligence. In C. Blum & D. Merkle (Eds.), *Swarm intelligence: introduction and applications* (pp. 3–41). Berlin: Springer-Verlag.

Blum, C., & Li, X. (2008). Swarm intelligence in optimization. In C. Blum & D. Merkle (Eds.), *Swarm intelligence: introduction and applications* (pp. 43–86). Heidelberg: Springer-Verlag.

Conway, J. H., & Smith, D. A. (2003). *On quaternions and octonions: their geometry, arithmetic, and symmetry*. Wellesley, MA: A K Peters.

Črepinšek, M., Mernik, M., & Liu, S. H. (2011). Analysis of exploration and exploitation in evolutionary algorithms by ancestry trees. *International Journal of Innovative Computing and Applications, 3*, 11–19.

Darwin, C. (1859). *The origin of species*. New York: P.F. Collier.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: John Wiley & Sons.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research, 7*, 1–30.

Dorigo, M., & Di Caro, G. (1999). The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization* (pp. 11–32). London, UK: McGraw Hill.

Eberly, D. (2002). *Quaternion algebra and calculus*. Magic Software, Inc..

Eiben, A. E., & Smith, J. E. (2003). *Introduction to evolutionary computing*. Berlin: Springer-Verlag.

Fister, I., Fister, I., Jr., Brest, J., & Žumer, V. (2012). Memetic artificial bee colony algorithm for large-scale global optimization. In *IEEE congress on evolutionary computation, Brisbane, Australia* (pp. 3038–3045). IEEE Publications.

Fister, I., Jr., Fister, D., & Yang, X.-S. (2013). A hybrid bat algorithm. *Electrotechnical Review, 80*, 1–7.

Fister, Iztok, Fister, Iztok, Jr., Yang, Xin-She, & Brest, Janez (2013). A comprehensive review of firefly algorithms. *Swarm and Evolutionary Computation*. accepted for publication.

Fister, I., Yang, X.-S., Brest, J., & Fister, I. Jr., (2013). Memetic self-adaptive firefly algorithm. In X.-S. Yang, R. Z. C. Xiao, A. H. Gandomi, & M. Karamanoglu (Eds.), *Swarm intelligence and bio-inspired computation: theory and applications* (pp. 73–102). Amsterdam: Elsevier.

Fister, I., Jr., Yang, X.-S., Fister, I., & Brest, J. (2012). Memetic firefly algorithm for combinatorial optimization. In B. Filipič & J. Šilc (Eds.), *Bioinspired optimization methods and their applications: proceedings of the fifth international conference on bioinspired optimization methods and their applications – BIOMA 2012* (pp. 75–86). Jožef Stefan Institute.

Friedman, M. (1937). 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.

Friedman, M. (1940). A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics, 11*, 86–92.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. New York, USA: W.H. Freeman & Co..

Girard, P. R. (1984). The quaternion group and modern physics. *European Journal of Physics, 5*, 25–32.

Hamilton, W. R. (1899). *Elements of quaternions*. Longmans Green and Co..

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

Kennedy, J., & Eberhart, R. C. (1999). The particle swarm optimization: social adaptation in information processing. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization* (pp. 379–387). London, UK: McGraw Hill.

Kennedy, J. F., & Eberhart, R. C. (2001). *Swarm intelligence*. Burlington, MA: Morgan Kaufman.

Korošec, P., Šilc, J., & Filipič, B. (2012). The differential ant-stigmergy algorithm. *Information Sciences, 192*, 82–97.

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

Long, S. M., Lewis, S., Jean-Louis, L., Ramos, G., Richmond, J., & Jakob, E. M. (2012). Firefly flashing and jumping spider predation. *Animal Behaviour, 83*, 81–86.

Morrison, R. W. (2004). *Designing evolutionary algorithms for dynamic environments*. Berlin: Springer Verlag.

Rothlauf, F. (2006). *Representations for genetic and evolutionary algorithms*. Berlin: Springer-Verlag.

Russel, S., & Norvik, P. (2010). *Artificial intelligence: a modern approach*. Upper Saddle River: Prentice Hall.

Wooldridge, M. (2009). *An introduction to multiagent systems*. Chichester, UK: John Wiley & Sons.

Yang, X.-S. (2008). Firefly algorithm. In X.-S. Yang (Ed.), *Nature-inspired metaheuristic algorithms* (pp. 79–90). Wiley Online Library.

Yang, X.-S. (2010). Appendix A: test problems in optimization. In X.-S. Yang (Ed.), *Engineering optimization* (pp. 261–266). John Wiley & Sons, Inc..

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm. In C. Cruz, J. R. Gonzlez, N. Krasnogor, D. A. Pelta, & G. Terrazas (Eds.). *Nature inspired cooperative strategies for optimization (NISCO 2010)* (Vol. 284, pp. 65–74). Berlin: Springer-Verlag.

Yang, X.-S., & Deb, S. (2009). Cuckoo search via Levy flights. In *World congress on nature & biologically inspired computing (NaBIC 2009)* (pp. 210–214). IEEE Publications.

Zhou, A., Qu, B.-Y., Li, H., Zhaom, S.-Z., Suganthan, P. N., & Zhang, Q. (2011). Multiobjective evolutionary algorithms: a survey of the state of the art. *Swarm and Evolutionary Computation, 1*, 32–49.