MATHEMATICAL MODELING OF ECOLOGICAL SYSTEMS ALGORITHM

Abdel-Razzak Merheb, Hassan Noura, Francois Bateman

Mathematical Modeling of Ecological Systems Algorithm

Article · June 2021

DOI: 10.22453/LSJ-022.2.209-231

3 authors:

Abdel-Razzak Merheb
Lebanese International University
34 PUBLICATIONS **326** CITATIONS
SEE PROFILE

Hassan Noura
Aix-Marseille Université
176 PUBLICATIONS **2,805** CITATIONS
SEE PROFILE

François Bateman
École de l'Air
27 PUBLICATIONS **442** CITATIONS
SEE PROFILE

MATHEMATICAL MODELING OF ECOLOGICAL SYSTEMS ALGORITHM

Abdel-Razzak Merheb[1], Hassan Noura[2], and François Bateman[3]

[1]Lebanese International University, Lebanon
abdelrazzak.merheb@liu.edu.lb
[2]Laboratoire d'Informatique et Système, Aix Marseille University, France
hassan.noura@univ-amu.fr
[3]Centre de Recherche de l'École de l'Air, France
francois.bateman@ecole-air.fr

(Received March 2020 – Accepted June 2021)

ABSTRACT

Merheb, A. R., Noura, H., & Bateman, F. (2021). Mathematical Modeling of Ecological Systems Algorithm. *Lebanese Science Journal,* 22(2), 209 - 231.

In this paper, the mathematical modeling of a new bio-inspired evolutionary search algorithm called Ecological Systems Algorithm (ESA) is presented. ESA imitates ecological rules to find iteratively the optimum of a given function through interaction between predator and prey search species. ESA is then compared to the well-known Genetic Algorithm which is a powerful bio-inspired stochastic search/optimization algorithm used for decades. Simulation results of the two algorithms optimizing ten different benchmark functions are used to investigate and compare both algorithms based on their speed, performance, reliability, and efficiency.

Keywords: Ecological Systems Algorithm, Genetic Algorithms, Benchmark functions, Swarm Intelligence, Bio-Inspired Algorithms.

INTRODUCTION

Swarm intelligence is an evolutionary computation scientific discipline inspired by insect colonies where a single individual has simple and naive behavior that takes trivial actions based on the processing of many sensor inputs. However, the interaction between the individuals produces an intelligent collective behavior that allows insect societies to find optimal solutions for their daily life problems. Scientists imitate social behavior of insects, and form decentralized systems of many autonomous individuals with naive actions distributed in the environment. The result was intelligent systems that are able to solve

http://dx.doi.org/10.22453/LSJ-022.2.209-231

different problems with remarkable flexibility, efficiency, and robustness (Garnier et al., 2007).

In (Zuo et al., 2017), the fruit Fly Optimization Algorithm (FOA), which is a global optimization algorithm that mimics the foraging behavior of a fruit fly swarm, is improved for multiobjective optimization problems. The new algorithm, called Stochastic Fractal based Multiobjective fruit Fly Optimization Algorithm (SFMOFOA), improves the convergence performance of classic FOA algorithm by the introduction of a food source generating method using a stochastic fractal with an adaptive parameter updating strategy. The selection process of FOA is updated with the Pareto domination concept in order to deal with multiobjective optimization problems. The performance of the new algorithm is tested through its application to eighteen different benchmark functions, and results are used to compare the new algorithm with four state of the art search methods. Numerical results show that the new algorithm has competitive performance, and that it is able to converge, with good distributions, to the Pareto fronts of the benchmark functions. In (Lamy, 2018), authors propose a new biologically inspired metaheuristic called Artificial Feeding Birds (AFB). The new algorithm is inspired by the simple behavior of birds searching for food. The new algorithm is compared with four well established biologically inspired search algorithms based on their results in optimizing several benchmark functions and the training of artificial neural networks. AFB is shown to give equivalent or better results produced with Artificial Bee Colony (ABC), Firefly Algorithm (FA), Genetic Algorithm (GA) and Ant Colony Optimization (ACO). The relationship between predator and prey individuals is used to update the well-known Particle Swarm Optimization (PSO) algorithm in (Zhang et al., 2018). Two search groups are generated in the new algorithm where three strategies are added to the properties of search particles: catch, escape and breeding. Moreover, a proportional-integral (PI) control is used to control the number of population so it fluctuates but stays within a relative stability, enhancing population diversity. Testing the new algorithm on 40 benchmark functions with different dimensions shows the superior performance of the new algorithm in comparison with ten other search algorithms.

The interaction between two predator-prey or more species in an ecosystem is described by the Lotka-Volterra equations, also known as the predator-prey equations. Lotka-Volterra equations are a set of two nonlinear, first order differential equations that explain the population change or the dynamics of biological systems. These equations are used not only in biological, ecological and environmental literature, but also in other fields such as atmospheric chemistry, urban growth studies, tourist industry, and economics. In economics, prey-predator equations based economic models are used to study the complex relations between economy, population, labor and capital (Puliafito et al., 2008).

In this paper, the Ecological Systems Algorithm (ESA) is modeled mathematically. Equations needed to update the positions and the health of search individuals is presented in detail. Moreover, a primitive comparison between ESA and the Genetic Algorithm using ten different benchmark functions is also presented. ESA was first developed in 2012 as an intelligent tool to tune the fault tolerant controllers of quadrotor Unmanned Aerial Vehicles (UAVs). Several papers have been published using ESA as tuning tool for different Active

and Passive Fault tolerant Controllers for quadrotor and octorotor UAVs. However, this is the first paper that models ESA mathematically and explains it in detail. The algorithm modeled here differs from the work presented in (Zhang et al., 2018) in many aspects. First, ESA is an independent algorithm that has its own strategies and steps and is not an improvement of an existing algorithm. ESA presents an algorithm that resembles to interacting animals in an environment, where each search individual moves in naive steps but the interaction among different individuals and between the individuals and their environment produces the intelligent behavior of the algorithm. Moreover, ESA has a simple position update technique that does not depend on the velocity vector as in PSO.

The rest of the paper is organized as follows: Section 2 explains the Ecological Systems Algorithm and models it mathematically. In Section 3, ten benchmark functions used in global optimization problems are defined and used to evaluate ESA and compare it with the Genetic Algorithms (GA). The two algorithms are compared regarding speed, reliability, and efficiency. Finally, a summary of the work with some future ideas is stated in the conclusion.

MATERIAL AND METHODS

In this section, the Ecological Systems Algorithm is explained and modeled mathematically. Next, ten benchmark functions used in global optimization problems are defined and used to evaluate ESA and compare it with the Genetic Algorithms (GA).

Ecological systems algorithm

Ecological Systems Algorithm is a biologically inspired algorithm that uses natural selection applied on larger scale. Search individuals of ESA are neither genes like in Genetic Algorithms nor bacteria like in Bacterial Foraging Algorithm (Passino and Liu, 2002), but larger creatures like mammals and birds. Moreover, the search here is realized using two search individual species, predators (predator in Latin) and preys (Litatio in Latin) (Beauchamp et al., 2007). The fitness function is optimized by the interaction between these two species.

Equation (1) defines the search species in ESA which are predator and prey (or litatio) species.

$$\begin{matrix} \boldsymbol{P_j^i}\left(\boldsymbol{x_j^i}, \boldsymbol{y_j^i}, \boldsymbol{H_j^i}, \boldsymbol{A_j^i}\right) \\ \boldsymbol{L_j^i}\left(\boldsymbol{x_j^i}, \boldsymbol{y_j^i}, \boldsymbol{H_j^i}, \boldsymbol{A_j^i}\right) \end{matrix} \qquad (1)$$

Where P_j^i denotes the predator j at iteration i, and L_j^i denotes the prey j at iteration i. Each individual has its own variables that define its location, health, and age situation. x_j^i, y_j^i are the coordinates of individual j in iteration i, while H_j^i and A_j^i are respectively the health and age of individual j in iteration i.

ESA Initialization

The algorithm starts by the generation of the two search species with random positions and health values and an age value equal to zero as presented is equation (2).

$$\begin{matrix} \boldsymbol{P_j^0\left(x_j^0, y_j^0, H_j^0, A_j^0 = 0\right)} \\ \boldsymbol{L_j^0\left(x_j^0, y_j^0, H_j^0, A_j^0 = 0\right)} \end{matrix} \quad (2)$$

Here, $j = 1, 2, \ldots, S_p$ for the predator species, and $j = 1, 2, \ldots, S_l$ for the prey species. S_p and S_l are respectively the number of predators and preys in their species. Equation (3) shows the random generation of the initial coordinates of each individual along with its initial health. The initial coordinates of each individual as well as its initial health value are chosen randomly but within the search boundaries and the maximum allowed health value.

$$\begin{matrix} \boldsymbol{x_j^0 = min\,(rand, max_X)} \\ \boldsymbol{y_j^0 = min\,(rand, max_Y)} \\ \boldsymbol{H_j^0 = min\,(rand, max_H)} \end{matrix} \quad (3)$$

Where max_X, max_Y, and max_H are respectively the maximum allowed x and y coordinate values, and maximum allowed health value. After the generation of the initial search species, ESA starts its iterations to search for an optimal solution to the problem. Each iteration in ESA applies five different steps -which denote the five ecological rules- in its struggling to solve the optimization problem. Each iteration starts with the age update step, continues with the position update step, the health update step, and the reproduction step, and ends with the demise step. Several search variables are initialized before ESA ecological steps are repeated iteratively. The maximum individual age, the individual step sizes, the rates at which the health of an individual increases or decreases, the minimum health value that allows an individual to breed, the minimum health for an individual to survive, and the effective nutrient density thresholds are all initialized before the search starts. These variables will be defined in some details in the ESA ecological steps.

ESA ecological steps

The nutrient density at the prey location is directly related to the function under test. If ESA is used to optimize a two-dimensional function $f(x, y)$ for example, the nutrient density that a prey individual $L_j^i\left(x_j^i, y_j^i, H_j^i, A_j^i\right)$ examines is the value of the optimized function at its location or $f(x_j^i, y_j^i)$. To weight an iteration, the nutrient density values of all existing prey individuals are used to evaluate a given fitness value. A suitable fitness function is the average nutrient density of all existing prey individuals. The fitness value of an iteration can be used to stop the search. If the fitness value of a given iteration is high, it means that most

of the prey search individuals are located in a high nutrient value location and thus optimal solution is found.

Age update step

At each iteration, the age of each individual in both search species is increased by one. Equation (4) depicts the increase in age for individual j at iteration $i + 1$.

$$\boldsymbol{A_j^{i+1} = A_j^i + 1} \qquad (4)$$

The age update step of individual j continues until $A_j^i = N$ where N is a variable chosen at the beginning of the search and defines the maximum age allowed for an individual. When the age of an individual reaches N, the individual expires after the application of the demise step.

Position update step

At each iteration, individuals in the two search species change their positions in a random manner (Beauchamp et al., 2007). Any search individual j updates its coordinates based on its previous position and a random number related to the maximum allowed step size in x and y directions as shown in equation (5).

$$\begin{aligned} \boldsymbol{x_j^{i+1} = x_j^i + Rand_{xj} \times Step_x} \\ \boldsymbol{y_j^{i+1} = y_j^i + Rand_{yj} \times Step_y} \end{aligned} \qquad (5)$$

Where $Rand_{xj}$ and $Rand_{yj}$ are random numbers between 0 and 1 from a standard uniform distribution function generated for individual j before applying the position update step, and $Step_x$ and $Step_y$ are the step size of the individuals. Note that the step sizes of the predator and prey species could be chosen equal or different. Thus, it is more appropriate to choose $Step_{xp}$, $Step_{yp}$ as step size of the predator individuals, and $Step_{xl}$, $Step_{yl}$ as the step size of the prey individuals.

Health update step

The health variable of each individual is updated based on the interaction between the individual with its environment. The first effective agent on the health of a prey is its position. If the new position of a prey has higher nutrient density value compared with its previous position, the health of the individual increases with a predefined scale. On the other hand, if the new position has worse nutrient density value the prey health decreases. The health update step for a prey individual is presented in equation (6).

$$\begin{aligned} \boldsymbol{H_j^{i+1} = C_j^{i+1} \times H_j^i} \\ \boldsymbol{with} \begin{cases} \boldsymbol{C_j^{i+1} > 1, if\ Nutrient^{i+1} > Nutrient^i} \\ \boldsymbol{C_j^{i+1} < 1, if\ Nutrient^{i+1} < Nutrient^i} \end{cases} \end{aligned} \qquad (6)$$

Where $Nutrient^i$ is the nutrient density at the prey location in the previous iteration. The value of C_j^{i+1} should be chosen before the start of the search, and it indicates the rate at which the health of a prey individual increases or decreases related to its new position. In order to add the swarm effect to the health update step for a prey individual, the variables Eps_n and G_n are added. If the nutrient density in the prey location is less than the threshold Eps_n (nutrient is scarce), the prey health decreases even if it comes from a location with less nutrient density. This situation indicates that the nutrient is not enough for the whole swarm, which results in famine. On the other hand, if the nutrient density in the prey location is more than G_n (nutrient is plenty), the prey health increases even if it comes from a better location. This situation indicates that the nutrient is plenty for the whole swarm, and all the preys forage and increase their health. The prey health update in (6) is applied when the individual does not exist in the vicinity of a predator. If a predator is very close to the prey individual, the predator will attack the prey which will affect the health of the prey individual. If the health of the prey is less than the health of the predator, the latter one manages to eat the prey. This means that the health of the prey individual is decreased to zero which results in the prey death in the demise step. On the other hand, if the prey has greater health value, it manages to escape but with some injuries that decrease its health. Note that in case more than one predator exist at the vicinity of a prey, the closest predator is the one eligible for prey attack. Equation (7) shows the health update step for a prey individual in the vicinity of a predator individual.

$$\boldsymbol{H_j^{i+1} = C_j^{i+1} \times K_j^{i+1} \times H_j^i}$$
$$\boldsymbol{with} \begin{cases} \boldsymbol{K_j^{i+1} = 0, if H_{closest\_pred}^i > H_j^i} \\ \boldsymbol{K_j^{i+1} = 1, if\ no\ close\ predator\ exists} \\ \boldsymbol{K_j^{i+1} < 1, if H_{closest\_pred}^i < H_j^i} \end{cases} \quad (7)$$

Where C_j^{i+1} is chosen according to the criteria in the previous set of equations, and $K_j^{i+1} \neq 0$ indicates the rate at which the health of the prey decreases when it manages to escape from the predator attack. Needless to state that K_j^{i+1} should be chosen at the initialization stage.

For the predator individuals, the health update step is simple and straight forward as shown in equation (8).

$$\boldsymbol{H_j^{i+1} = M_j^{i+1} \times H_j^i}$$
$$\boldsymbol{with} \begin{cases} \boldsymbol{M_j^{i+1} < 1, if\ H_j^i < H_{closest\_prey}^i\ or\ no\ close\ prey\ exists} \\ \boldsymbol{M_j^{i+1} > 1, if\ H_j^i > H_{closest\_prey}^i} \end{cases} \quad (8)$$

The equation above means that at a given iteration, if no prey is close to a predator individual the health of the predator decreases severely as a result of famine. In addition, if

there is a very healthy prey close to the predator individual, the prey is able to escape and the predator does not forage resulting in a decrease in its health. On the other hand, if there is a weak prey close to the predator, the predator manages to eat the prey and increases its health by a factor of $M_j^{i+1} > 1$.

Reproduction step

The reproduction step is responsible for increasing the number of individuals in a search species. A prey individual breeds in a new iteration if its health is higher than a given threshold, and if it moves to a better location with higher nutrient density. On the other hand, a predator individual breeds in an iteration if it manages to eat a prey and if its health is higher than a given threshold. For a prey individual, the reproduction step is as follows

Algorithm 1 Prey **Reproduction**

1. **while** not all prey individuals are visited do
2. for individual *j*, **if** $C_j^i > 1$ and $H_j^i > Breed_{Hl}$
3. Breed a prey baby $x_{baby\ j}^i = x_j^i, y_{baby\ j}^i = y_j^i, H_{baby\ j}^0 = rand$
4. $L_{baby\ j}^i\left(x_{baby\ j}^i, y_{baby\ j}^i, H_{baby\ j}^0, 0\right)$
5. **Return** $L_{baby\ j}^i$
6. **end while**

Where $C_j^i > 1$ indicates that the prey individual j moves to a new nutrient rich location in step *i*, and $Breed_{Hl}$ is the health value that allows a prey to breed and is chosen at the beginning of the search. The algorithms in this paper are inspired by the work done by Parpinelli (Parpinelli and Lopes, 2011) and Neumann (Neumann and Witt, 2010). Similarly, the reproduction step of a predator individual is summarized as follows (Zeigler et al., 1978)

Algorithm 2 Predator **Reproduction**

1. **while** not all predator individuals are visited do
2. for individual *j*, **if** $M_j^i > 1$ and $H_j^i > Breed_{Hp}$
3. Breed a predator baby $x_{baby\ j}^i = x_j^i, y_{baby\ j}^i = y_j^i, H_{baby\ j}^0 = rand$
4. $L_{baby\ j}^i\left(x_{baby\ j}^i, y_{baby\ j}^i, H_{baby\ j}^0, 0\right)$
5. **Return** $L_{baby\ j}^i$
6. **end while**

Here $M_j^i > 1$ means that the predator *j* has foraged in iteration *i*, and $Breed_{Hp}$ is the minimum health value that allows the predator to breed. Needless to say that if a species reaches its maximum allowed number of individuals, the reproduction step is not applied. This can be seen in the conditions $\text{Predator Individual number} < S_p$ and $\text{Prey Individual number} < S_l$ in Algorithms 1 and 2.

Demise step

The demise step is very important in the Ecological Systems Algorithm since it ensures the elimination of weak individuals in both search species. After several iterations,

the prey individuals that have moved from bad to better locations (locations with higher nutrient density) get their health values increased. On the other hand, prey individuals that have moved to worse locations experience a decrease in their health values. In the demise step, individuals with health values less than a given threshold or that have reached their maximum iteration age vanish. This ensures that only individuals that have found optimal locations survive for the next set of iterations. After enough number of iterations, all prey individuals will be gathered in the optimal location since individuals in bad locations are not allowed to breed and vanish with iterations. The demise step for prey individuals can be summarized as in Algorithm 3.

Algorithm 3 Demise **Step**

1. **while** not all prey individuals are visited do
2. for individual j, **if** $A_j^i \geq N$ or $H_j^i < Demise_{Hl}$
3. Prey j is eliminated $L_j^{i+1}(0,0,0,0)$
4. **Return** L_j^{i+1}
5. **end while**

Where A_j^i is the age of individual j at iteration i, H_j^i is its health, and $Demise_{Hl}$ is the minimum health for prey individuals to be alive. For predator species, Algorithm 3 is also applied but with $Demise_{Hl}$ is replaced by $Demise_{Hp}$. $Demise_{Hl}$ and $Demise_{Hp}$ are design values chosen at the beginning of the optimization process.

Note that ESA for two-dimensional problem search is explained here since it is more conventional and resembles to the real search of animal species in their environments. For ESA searching a space of dimension D, the initial search species are

$$\begin{matrix} \boldsymbol{P_j^i}\left(\boldsymbol{x_{1j}^i}, \boldsymbol{x_{2j}^i}, \boldsymbol{x_{3j}^i}, \ldots\ldots, \boldsymbol{x_{Dj}^i}, \boldsymbol{H_j^i}, \boldsymbol{A_j^i}\right) \\ \boldsymbol{L_j^i}\left(\boldsymbol{x_{1j}^i}, \boldsymbol{x_{2j}^i}, \boldsymbol{x_{3j}^i}, \ldots\ldots, \boldsymbol{x_{Dj}^i}, \boldsymbol{H_j^i}, \boldsymbol{A_j^i}\right) \end{matrix} \quad (9)$$

Where x_1, x_2,..., x_D are the coordinates of the predator and prey search individuals in D dimensions.

In addition to the regular ecological steps, dispersal property could be added to the algorithm by changing the health update step for prey individuals. The dispersal property gives the algorithm more chance to discover new regions in the search space that were initially far from being tested. Moreover, the dispersal property helps avoiding the local minima problem in ESA with prey individuals, located in local optimal locations, escape their locations and start exploring new scopes. The dispersal property can be summarized as follows: whenever a prey individual is attacked and survived the attack, it is made to move with greater speed and in a random position. The health update step with dispersal property is summarized by Algorithm 4.

Algorithm 4 Updated **Health Step**

1. **while** not all prey individuals are visited do
2. **if** no close predator exists
3. **if** $Nutrient_j^{i+1} > Nutrient_j^i$
4. $C_j^{i+1} > 1$
5. **else** $C_j^{i+1} < 1$
6. $H_j^{i+1} = C_j^{i+1} \times H_j^i$
7. **Return** H_j^{i+1}
8. **else** a close predator exists
9. **if** $H_{pred}^i \leq H_j^i$
10. $H_j^{i+1} = C_j^{i+1} \times K_j^{i+1} \times H_j^i$
11. $x_j^{i+1} = x_j^i + Rand_{xj} \times 2 \times Step_x$
12. $y_j^{i+1} = y_j^i + Rand_{yj} \times 2 \times Step_y$
13. **Return** $L_j^{i+1}(x_j^{i+1}, y_j^{i+1}, H_j^{i+1}, A_j^{i+1})$
14. **else** prey is eliminated $L_j^{i+1}(0,0,0,0)$
15. **Return** L_j^{i+1}
16. **end while**

ESA resembles to real animals in nature where each individual performs primitive actions with no intelligent moves or calculations. The actions taken by different individuals are naive and made without intelligence, but the interaction between different individuals and between the individuals and their environment produces an intelligent behavior for the algorithm. This makes ESA a Swarm Intelligence algorithm. It is important to emphasize that if extinction of the whole predator species occurs, ESA will still be able to search the space and find the optimal solutions. However, the search becomes slower and its ability to explore new regions is diminished. On the other hand, it is impossible for the algorithm to complete a search with only a predator species. Figure 1 shows the flowchart of Ecological Systems Algorithm.

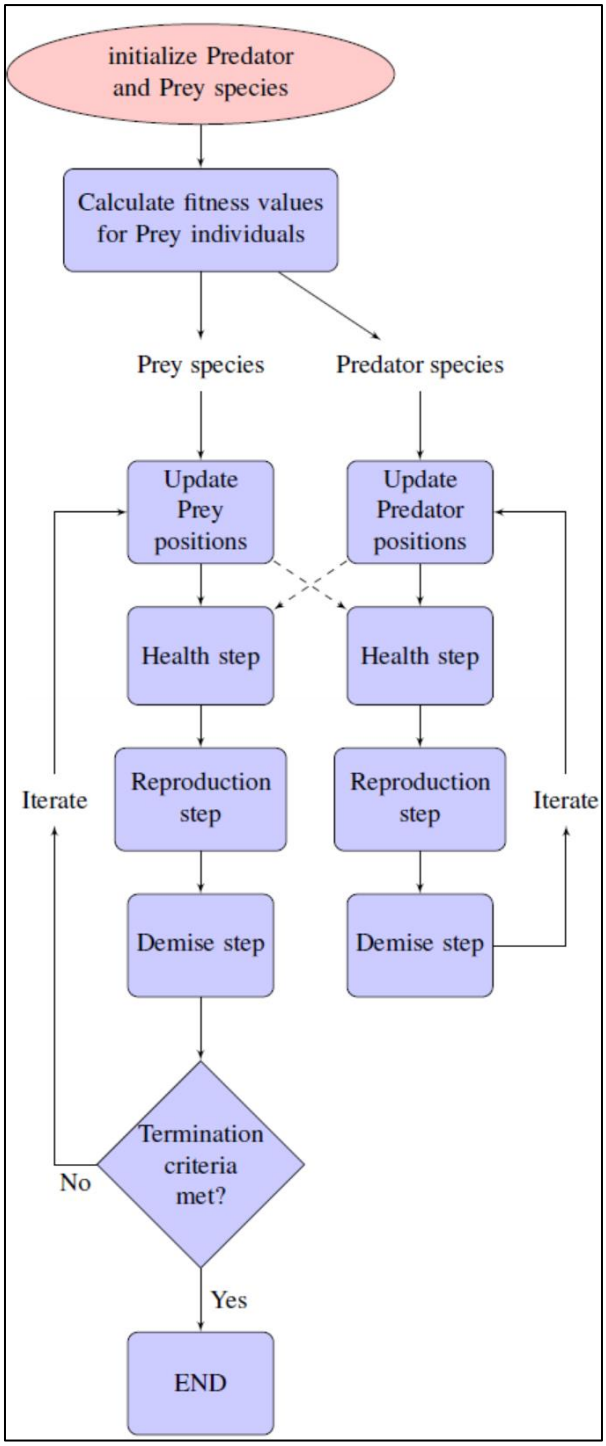

Figure 1: Flowchart of Ecological Systems Algorithm.

Tuning ESA parameters

Tuning the search algorithm for its parameters is a serious problem: the performance of the algorithm is influenced by these parameters as well as by the initial solutions suggested randomly at the beginning of the search. This means that the search algorithm requires another search algorithm to tune its parameters and find the best values to start the search!

The new search algorithm has also some parameters that need to be chosen carefully, and the tuning of its parameters will be a new problem. In the following, a guideline on the selection of the parameter set for ESA is provided. First, one has to choose roughly the dimensions of the search where the optimum is expected to be found. The dimension can be extended if the optimum detected by the algorithm is found to be insufficient. Eps_n and G_n values define the quality of the optimum point that the algorithm is trying to find. By choosing these values carefully, one can set the accepted range of the optimum which is greater than Eps_n and less than G_n values. The factors by which the health of predators and preys increase or decrease $(C_j^{i+1}, K_j^{i+1}, M_j^{i+1})$ affect the search speed and can be chosen by trial and error. The step size a search individual makes in the environment should be chosen influenced by the shape of the environment itself. If the environment has multiple optima with narrow minima and maxima for example, short step size is chosen to ensure that narrow locations are tested with multiple search individuals.

ESA belongs to metaheuristic algorithm family which has no guaranteed global convergence (Yang, 2014). It is thus essential to emphasize the importance of a detailed theoretical study of the convergence of ESA using probabilistic analysis (Liu et al., 2009; Liu and Liu, 2013) or Markov chain theory analysis (Song et al., 2009; He et al., 2018). Such studies are beyond the scope of this introductory paper.

Comparison of ESA and GA

In this section, ESA and GA algorithms will be compared based on several benchmark functions optimization results. GA is one of the first evolutionary search algorithms. It is a well-established, fast, powerful, and efficient algorithm (Kar, 2016). Authors believe that comparing a new biologically inspired algorithm to one of the pioneer bio-inspired algorithms that has been under development and in use in several scopes for decades will reveal the importance and effectiveness of the newly developed algorithm (Karaboga and Basturk, 2007; Lamy, 2018). In this paper, the traditional base-10 single point crossover GA algorithm presented by Kevin Passino in (Passino, 2005) is used. To be able to compare different search algorithms in a benchmark test, it is essential to perform multiple runs of each algorithm on each benchmark function. This is because the search path taken by each algorithm depends on the starting conditions and will be different at each run. Moreover, the effectiveness and performance of the algorithms is expected to be different from a function to another. A robust algorithm is an algorithm that has similar performance regardless of the starting conditions and different benchmark functions. In this section, ESA and GA are used to optimize ten different benchmark functions (Wu et al., 2016). The performance and effectiveness of the two algorithms are compared by performing multiple runs for each algorithm with each benchmark function. The change in the best, worst, average, and mean fitness values, as well as the standard deviation of the fitness values through 100 iterations (Shareef et al., 2015; Karaboga and Basturk, 2007; Margaritis and Digalakis, 2001) are used to compare the two algorithms. The variation in the global optimization results found by both algorithms in 20 different runs of 100 iterations is also used to evaluate the two algorithms.

This number of runs is seen to be sufficient to obtain a general comparative picture of the two algorithms (Dieterich and Hartke, 2012).

Benchmark functions

The benchmark functions used in this paper are non-linear, continuous/discontinuous, unimodal/multimodal, convex/non-convex, and separable/non-separable. The modality of a function is the number of vague peaks in the surface of that function; and a function is multimodal if it has more than one vague peak. This means that a multimodal function has multiple local optima that might disguise the search algorithm. The separability of a function affects the difficulty level of this function. In separable functions, each variable is independent of the other variables which make such functions easier to solve compared to non-separable functions (Shareef et al., 2015). The dimension of benchmark functions affects the optimization difficulty level, and is taken to be two for all functions.

The Ackley function is continuous, differentiable, non-separable, scalable, and multimodal that is widely used to test optimization algorithms. This function has moderate complexity with many local optima spread in its surface. An optimization algorithm searching Ackley function risks of being trapped in one of these optima. The Cross In Tray function is continuous, non-differentiable, and multimodal with multiple global optima. Griewank function is another continuous, differentiable, non-separable, scalable, and strongly multimodal function. The variables in this function are interdependent, and local optima are widespread but distributed regularly. Similar to the previous functions, the Holder Table function is continuous, differentiable, separable, non-scalable, and multimodal. This function has many local optima, but only four global optima. Levi function is continuous and differentiable, and has many local optima. Like the other functions, Matyas function is continuous, differentiable, non-separable, but non-scalable and unimodal. This function has only one global optimum with no local optima that hardens the search. The Perm function is continuous and differentiable function with four global optima. Rastrigin function is highly multimodal and has many local optima that are regularly distributed in the environment. A search algorithm optimizing this function can be trapped in its local optima and fail to find the global optima. Schaffer function is continuous, differentiable, non-separable, scalable, and highly multimodal function. Schwefel function has many peaks and valleys. It is a complex, continuous, differentiable, non-separable, scalable, but unimodal function. This function has a second best optima which drives search algorithms to converge in the wrong direction. The reader is directed to (Jamil and Yang, 2013; Suganthan et al., 2005), and (Finck et al., 2010) for more information about benchmark functions used in this paper. The benchmark functions used to evaluate and compare ESA and GA algorithms are shown in Table 1. Note that these functions are used in two dimensional searches within the search space {(-10, -10), (10,10)} except the Rastrigin and the Shcwefel functions with respectively {(-20, -20), (20,20)} and {(-50, -50), (0,0)} as search spaces.

Table 1: Benchmark functions

Function	General expression	Global optima
Ackley	$f(\vec{x}) = 12.64 + 20 \cdot (e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} + + \ e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} - 20 - e)$	12.64 at [0, 0]
Cross In Tray	$f(x,y) = 0.0001 \cdot (\lvert \sin(x)\sin(y) \cdot \cdot exp(\lvert 100 - \frac{\sqrt{x^2+y^2}}{\pi}\rvert) + 1\rvert)^{0.1}$	2.058, at [1.5, 1.5], [1.5, -1.5], [-1.5, 1.5], [-1.5, -1.5]
Griewank	$f(\vec{x}) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}})$	2.004, 18 global optima
Holder Table	$f(x,y) = \lvert sin(x)cos(y)exp(\lvert 1 - \frac{\sqrt{x^2+y^2}}{\pi}\rvert)\rvert$	19.18, at [8.1, 9.7], [-8.1, 9.7], [8.1, -9.7], [-8.1, -9.7]
Levi	$f(x,y) = 0.01 \cdot (200 - [\sin^2(3\pi x) + ((x-1)^2) \cdot \cdot (1 + \sin^2(3\pi y)) + ((y-1)^2)(1 + \sin^2(2\pi y))])$	2, at [1, 1]
Matyas	$f(x,y) = 100 - \left(0.26 \cdot (x^2 + y^2) - 0.48 \cdot x \cdot y\right)$	100, at [0, 0]
Perm	$f(\vec{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{n} (j^i + \beta \left((\frac{x_j}{j}) - 1 \right)) \right)^2$	6.634e[4], at [10, 10], [-10, 10], [10, -10], [-10, -10]
Rastrigin	$f(\vec{x}) = 10 \cdot n + \sum_{i=1}^{n} x_i^2 - 10 \cdot \cos(2 \cdot \pi \cdot x_i)$	800, at [20, 20], [-20, 20], [20, -20], [-20, -20]
Schaffer	$f(x,y) = 0.5 + \frac{(cos(sin(\sqrt{x^2-y^2})))^2 - 0.5}{[1+0.001 \cdot (x^2+y^2)]^2}$	1, at [0, 0]
Schwefel	$f(\vec{x}) = 418.9829 * n + \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	908. 9, at [-50, -50]

Genetic Algorithm applied to benchmark functions

Genetic Algorithm is a biologically inspired stochastic search algorithm based on natural selection used to solve complicated optimization problems. GA algorithm is a very practical tool where no reformulation of the problem or special mathematical treatment is needed. Moreover, this algorithm which is not subjected to local optimization problem common in conventional search algorithms, is easy to apply to nonlinear search spaces, and can be used in discontinuous and noisy search problems. GA starts by assigning randomly a group of solutions and testing their fitness. The best solution values are then used to generate new solutions using three random methods. Each search individual in the GA algorithm used to optimize the benchmark functions has two traits. This allows GA to search two dimensional environments. Each trait has six genes that are digitized, so that search individuals are changed by toggling digital bits. The probability of mutation process is chosen to be 0.05, while the crossover probability is chosen as 0.8. A total number of 100 individuals in the population is chosen to perform the search. These values were inspired by the extensive study made by Digalakis and Margaritis in (Margaritis and Digalakis, 2001) and (Margaritis and Digalakis, 2002). From the various population size, mutation probability, and crossover

probability values tested in these works, suitable values giving acceptable results for most of the benchmark functions were chosen.

ESA applied to benchmark functions

To search the two-dimensional benchmark functions using ESA, one prey and one predator species are assigned randomly in the environment. The search species are created in the plane, each individual with an age of zero and with random x and y coordinates and health value. By changing their coordinates randomly, search individuals move in the environment to explore new locations. A prey checks the nutrient density in its location by feeding its coordinates to the benchmark function being optimized. High fitness value in a location indicates high nutrient density at that location. After the creation of initial population, the nutrient density at the location of each prey individual is calculated. Prey individuals then move randomly in the environment, and nutrient densities are calculated in their new locations. The prey that had moved from worse to better location increases its health, and a new search individual is created at the same location. Prey individual that had moved to a worse location decreases its health as result of famine. Note that the random steps taken by prey and predator individuals are insured to be within the search boundaries.

For predator individuals, they detect prey individuals found in close locations and attack them. If the health of the prey individual is more than that of the predator individual, the prey manages to escape but with some injuries that decrease its health. On the other hand, if the predator is healthier, it eats the prey, increases its own health, and breed a new baby predator at its location. The predator individual then moves to a new location randomly looking for a new prey. Table 2 and Table 3 show ESA parameters used to tune the benchmark functions. Values in Table 2 are chosen by trial and error, but with careful analysis of the outputs of several algorithm runs. It is found that if the prey health change rate C_j^{i+1} is chosen less than the suggested value, the prey species perishes quickly before the algorithm converges to an optimum. On the other hand, if C_j^{i+1} is chosen more than the suggested value, prey individuals found in local optima survive multiple iterations, the search slows down, and the predator species is rendered ineffective. This will result in the inability of the algorithm to discover new locations. Similarly, the predator health change rate was set following the same observations. It is important to emphasize that ideally, different health change rates would be chosen for each environment. However, the values given in Table 2 give acceptable results for all the benchmark functions used in this paper. The individual step size, the prey detection distance, along with the fitness boundaries Eps_n and G_n in Table 3 are chosen by trial and error but regarding the search environment. The step size of an individual is chosen small to ensure precise search of environments with several local minima such as Rastrigin and Griewank functions. This results in much slower but more effective search. For more homogeneous environments such as Perm and Matyas functions, the individual step size is chosen bigger to ensure fast search. Concerning Eps_n and G_n, a previous knowledge of the expected optimum helps in setting their values. These values allow the algorithm to avoid local minima by ignoring locations with fitness values far less than the optimum (set by Eps_n), and focusing the search around

locations with fitness values close to the optimum (set by G_n). The values of Eps_n and G_n are chosen based on the global optima values of the functions in Table 1. For the Rastrigin function for example its global optima is 200, so locations with fitness values less than 120 (Eps_n) are made less appealing for search, and locations with fitness values equal and more than 160 (G_n) are made popular. More babies will be born here, and individuals searching these locations will have longer life span.

Table 2: Parameters of ESA

Prey individuals		**Predator individuals**	
C_j^{i+1}	1.8, if $Nutrient_j^{i+1} > Nutrient_j^i$	M_j^{i+1}	1.1, if predator forages
$C_j^{i+1} \times K_j^{i+1}$	0.8, if prey is attacked	M_j^{i+1}	0.9, if no close prey exists
C_j^{i+1}	1.3, if $Nutrient_j^{i+1} < Nutrient_j^i$	Predator step size	0.8
S_l	100	S_p	10

Table 3: Parameters of ESA searching the test functions

Function	**Individual step size**	**Prey detection distance**	$\boldsymbol{Demise_{Hl} = Demise_{Hp}}$	$\boldsymbol{Eps_n}$	$\boldsymbol{G_n}$
Ackley	0.5	0.3	0.8	6	10
Cross In Tray	1	0.8	0.8	0.5	1.7
Griewank	0.5	0.8	0.8	0.5	0.9
Holder Table	0.5	0.3	0.8	10	18
Levi	0.5	0.5	0.8	0.2	1.5
Matyas	1	1	0.8	20	95
Perm	1	0.8	0.8	20000	65000
Rastrigin	0.5	0.5	0.8	120	160
Schaffer	1	0.3	0.8	0.7	0.8
Schwefel	5	5	0.8	700	850

RESULTS AND DISCUSSION

Results show that GA possesses superior convergence characteristics compared with ESA, and reaches the global optima in shorter time. However, the comparative results shown in Figure 2 to Figure 4 show that ESA has satisfactory performance, and its results are comparable to those of GA. ESA performed better than GA when optimizing Griewank, Holder table, and Rastrigin functions, which shows the ability of ESA to escape from local optima.

Figure 2 to Figure 4 show the convergence characteristic curves of GA and ESA algorithms obtained while optimizing the benchmark functions. ESA finds the best global optimum or near global optimum in less than 20 iterations on average for all functions except Rastrigin and Schwefel functions as shown in Figure 2. ESA needs some 60 additional iterations to optimize the Rastrigin function. For the Schwefel function, ESA can find the best near-optimum solution in 40 iterations. GA outperformed ESA in finding the best global optimum for Ackley, Rastrigin, and Schwefel functions. On the other hand, ESA outperformed GA in finding the global optimum for Griewank, Holder table, Perm, and Rastrigin functions. The performance of the two algorithms is similar for the remaining functions.

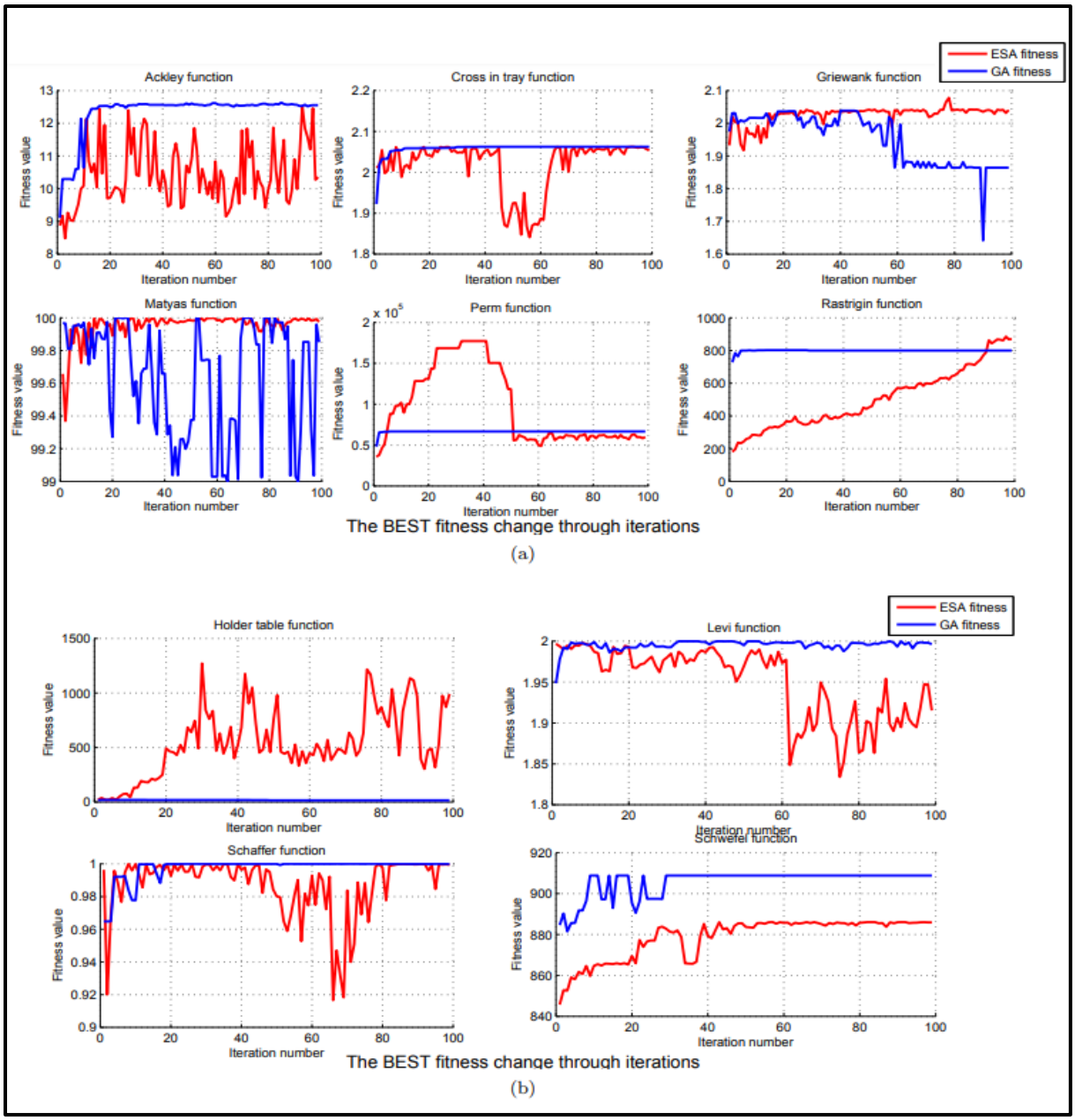

Figure 2: Best fitness values for ESA and GA in optimizing benchmark functions at Table 1 through 100 iterations, Ackley, Cross In Tray, Griewank, Matyas, Perm, and Rastrigin functions in (a), and Holder Table, Levi, Schaffer, and Schwefel in (b).

Figure 3 shows the mean fitness value curves found while optimizing the benchmark functions using ESA and GA through 100 iterations. This figure shows that the performance of GA and ESA is acceptable because their means are close to the optimal solutions for most of the benchmark functions.

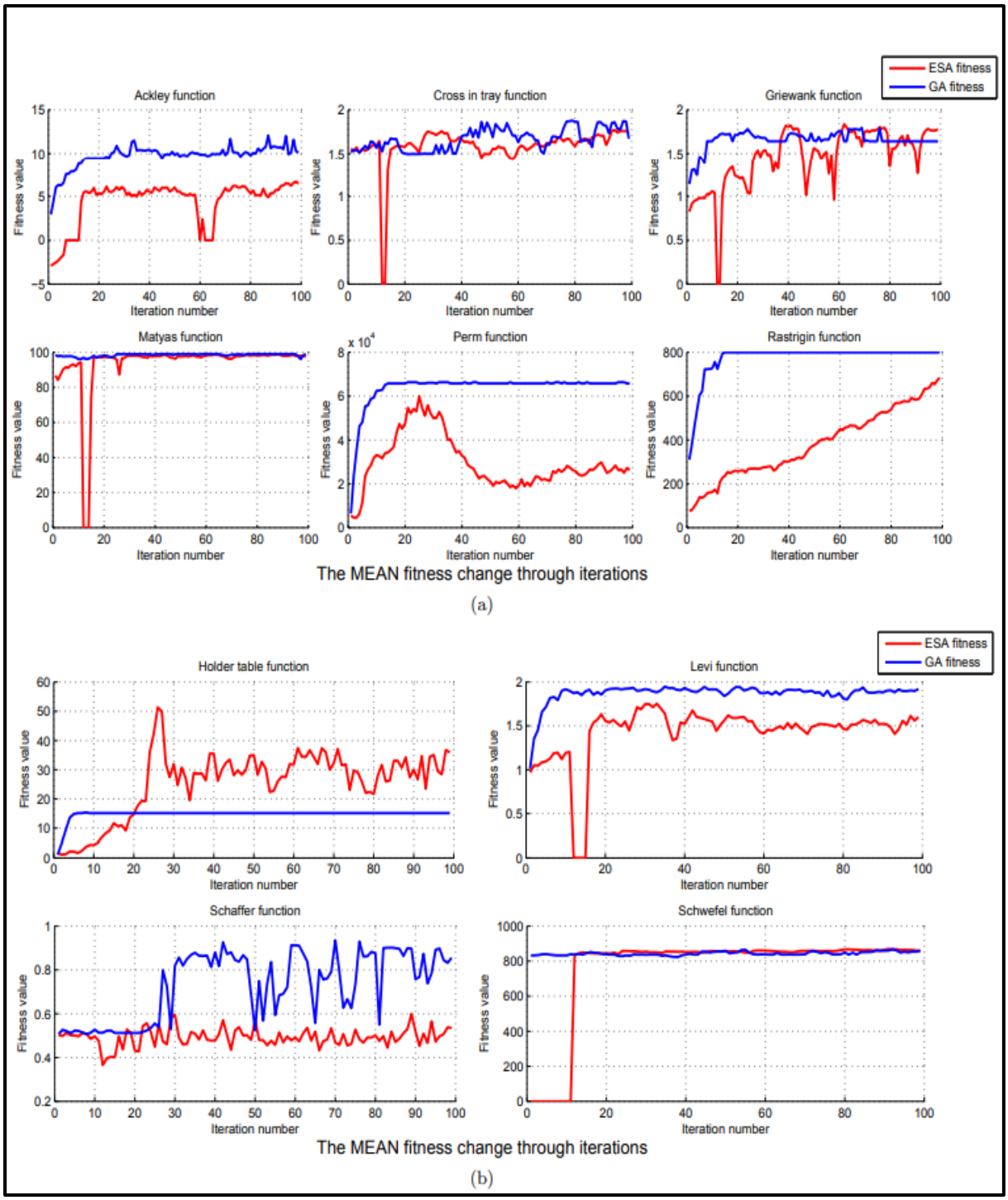

Figure 3: Mean fitness for ESA and GA in optimizing benchmark functions at Table 1 through 100 iterations, Ackley, Cross In Tray, Griewank, Matyas, Perm, and Rastrigin functions in (a), and Holder Table, Levi, Schaffer, and Schwefel in (b).

Figure 4 shows the standard deviation of the search performed by ESA and GA in 100 iterations. It is clear that the standard deviation of ESA exceeds that of GA in most of the functions, and oscillates around a constant value. This is related to the nature of the algorithms. In GA, best genes survive and perform the search resulting in a small standard deviation. In ESA, search individuals are spread in the environment searching for high nutrient density, and their number is defined by Lotka-Volterra equations. The number of predators and preys oscillates according to these equations, which explains the oscillation of the standard deviation. However, ESA performance is still acceptable because it has an overall low standard deviation.

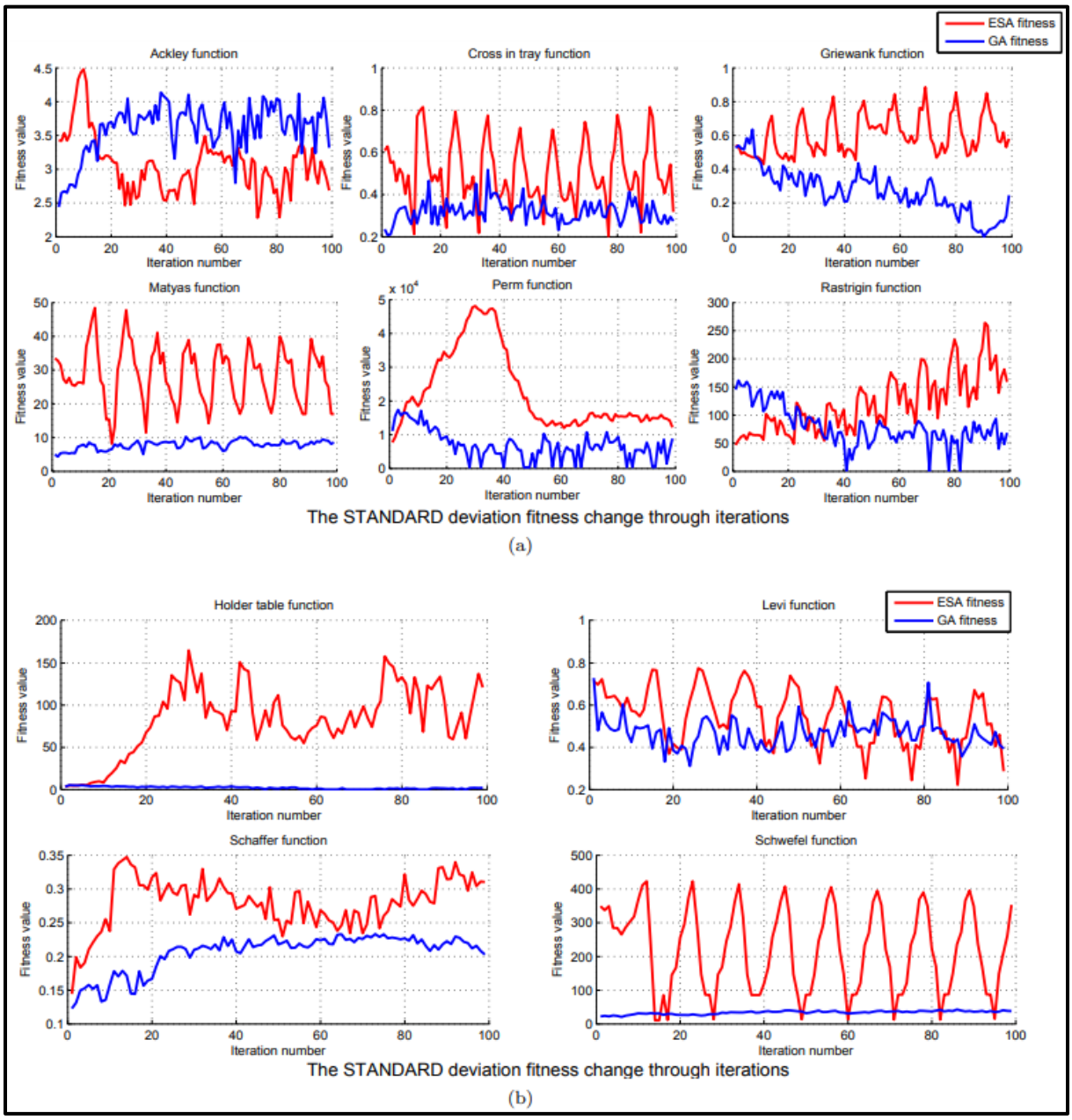

Figure 4. Standard deviation of the fitness function for ESA and GA in optimizing benchmark functions at Table 1 through 100 iterations, Ackley, Cross In Tray, Griewank, Matyas, Perm, and Rastrigin functions in (a), and Holder Table, Levi, Schaffer, and Schwefel in (b).

Experiments have shown that GA have found better global optima for Ackley, Cross In tray, and Levi functions, while ESA shows better performance with higher global optimum. This is related to the manner GA perform the search. In GA, digitized chromosomes perform the search by toggling some bits. This gives GA the opportunity to reach different and very far locations in two consecutive steps (by changing the third bit of 1001101 (77), the number becomes 109). Very large steps could be done between two iterations rather than being bounded with constant step sizes like in ESA. However, chromosomes close to optima may not be able to perform small changes and reach closer values to the optima. On the other hand, ESA search individuals have small step sizes that allow them to approach gradually to the optima. In other words, the fast convergence of GA is done with the sacrifice of more accurate solutions.

Table 4 shows the global optimization results of GA and ESA for the ten different benchmark functions. In many cases, ESA found better solution than GA but takes more time to converge for all benchmark functions except for the Holder Table function. It is important to emphasize that -like all stochastic search algorithms- the results of GA and ESA are tightly related to the initial population created at the beginning of the search as well as to the stop criteria responsible for ending the search. For example, optimizing the Griewank function using ESA may be achieved in 6 iterations or in 60 iterations (Table 5). Moreover, the speed of the search is also related to the situation of the computer processor. In two consecutive search trials for the same function, the computer takes 49 iterations in 7.44 seconds, and 51 iterations in 3.146 seconds.

Table 4: Global optimization results

Function		Optimum	Average	Average conversion time	Average iteration-generation
Ackley	ESA	12.3738	6.9564	0.81075 seconds	13.6
	GA	12.6014	9.5292	0.521 seconds	135
Cross In Tray	ESA	2.0594	1.7312	0.6275 seconds	10.3
	GA	2.0626	1.6210	0.4150 seconds	106.5
Griewank	ESA	2.0405	1.5350	3.364 seconds	47.8
	GA	1.8641	1.6363	0.0934 seconds	20.8
Holder Table	ESA	18.8895	9.4581	0.50466 seconds	6.6
	GA	20.1402	19.9338	0.927 seconds	241.4
Levi	ESA	1.9324	1.5354	0.941 seconds	18.2
	GA	1.9946	1.6	0.5682 seconds	166.4
Matyas	ESA	99.9748	96.3292	1.328 seconds	26
	GA	99.9534	95.3332	0.0934 second	21.4
Perm	ESA	$5.9214e^4$	$2.8104e^4$	0.692 seconds	11.4
	GA	$6.6785e^4$	$6.5889e^4$	0.0616 seconds	6.2
Rastrigin	ESA	870.3864	694.3641	2.049 seconds	40
	GA	800	768.9209	0.0891 seconds	18.4
Schaffer	ESA	0.9999	0.6325	0.638 seconds	11.2
	GA	1	0.7338	0.4366 seconds	129
Schwefel	ESA	884.8125	862.4745	1.754 seconds	34.2
	GA	908.8519	861.0567	0.1308 seconds	31.4

Table 5: Results of speed tests of ESA optimizing Griewank function

	Age = 5 iterations	**Age = 10 iterations**	**Age = 15 iterations**
Average iterations	24.2	36.8	143.3
Average time	1.33 seconds	2.01 seconds	8.2 seconds
Minimum iterations	6	3	45
Minimum time	0.409719 seconds	0.897217 seconds	6.916340 seconds
Maximum iterations	60	103	279
Maximum time	2.942206 seconds	5.053009 seconds	15.469319 seconds

Experiments have shown that the maximum age of individuals affects directly ESA conversion speed. Results of 10 speed tests conducted on Griewank function with three different ages are shown in Table 5. When the maximum age is 5 iterations, all the initial search individuals created at the beginning of the search die in 5 iterations leaving the environment for fresh individuals created as a result of a successful step. This allows the algorithm to find the optimum in average 25 iterations and 1.33 seconds. When the maximum age is increased to 10 iterations, the number of iterations needed for the algorithm to find the optimum is increased to 37. Increasing the maximum age to 15 iterations results in an exponential increase in iteration number and in the time needed for the algorithm to converge.

CONCLUSION

In this paper a novel bio-inspired evolutionary search algorithm based on ecological rules is proposed. The new algorithm -called Ecological Systems Algorithm (ESA)- uses a population of two search individual species to perform the search. Each individual in the predator and prey species has very naive decision making, but the interaction of these individuals with their environment as well as with each other produces an intelligent behavior that is able to solve optimization problems. Seeking comparison, Ecological Systems and Genetic algorithms are used to optimize ten different benchmark functions. MATLAB/SIMULINK results show that ESA was able to compete with GA and find more accurate optima for many benchmark functions. However, the convergence speed of ESA is shown to be lower compared with GA convergence speed. Future work should focus on the improvement of ESA algorithm. The introduction of inter-individual forces that ensure flocking of close individuals of the same species, the introduction of elitism among individuals, and the use of average fitness dependent step size are possible ideas that are expected to fasten and refine the ESA search. Perspectives of this work include the use of ESA to different optimization problems like clustering, and the optimization of multi-objective problems and travelling salesman problem, and its use in real engineering applications such as adaptive controller tuning, path planning for mobile robots, medical image analysis, network routing, and smart energy management.

REFERENCES

Beauchamp, D. and Johnson, B. and Wahl, D. 2007. Predator-Prey Interactions. In C. G. Brown (Ed.), *Analysis and interpretation of inland fisheries data* (1 ed.). American Fisheries Society.

Dieterich, J. and Hartke, B. 2012. Empirical Review of Standard Benchmark Functions Using Evolutionary Global Optimization. *Applied Mathematics, 3*(10), 1552-1564.

Finck, S. and Hansen, N. and Ros, R. and Auger, A. 2010. *Real-Parameter Black-Box Optimization Benchmarking: Presentation of the Noiseless Functions.* Institut National de Recherche en Informatique et en Automatique (INRIA).

Garnier, S. and Gautrais, J. and Theraulaz, G. 2007. The biological principles of swarm intelligence. *Swarm Intelligence*(1), 3-31.

He, X. S. and Wang, F. and Wang, Y. and Yang, X. S. and Yang, X. S. 2018. Global Convergence Analysis of Cuckoo Search Using Markov Theory. In X.-S. Yang (Ed.), *Nature-Inspired Algorithms and Applied Optimization.* Springer, Cham.

Jamil, M. and Yang, X. S. 2013. A literature survey of benchmark functions for global optimization problems. *International Journal of Mathematical Modelling and Numerical Optimisation, 4*(2), 150 - 194.

Kar, A. K. 2016. Bio inspired computing – A review of algorithms and scope of algorithms and scope of applications. *Expert Systems With Applications, 59*(1), 20-32.

Karaboga, D. and Basturk, D. 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization, 39*(1), 459–471.

Lamy, J. B. 2018. Artificial Feeding Birds (AFB): a new metaheuristic inspired by the behavior. In S. S. Shandilya S. (Ed.), *Advances in Nature-Inspired Computing and Applications* (pp. 43-60). Springer.

Liu, H. and Abraham, A. and Snášel, V. 2009. Convergence Analysis of Swarm Algorithm. *World Congress on Nature & Biologically Inspired Computing* (pp. 1714-1719). Coimbatore, India: NaBIC.

Liu, S. and Liu, H. 2013. Particle Swarm Algorithm: Convergence and Applications. In Z. C. Xin-She Yang (Ed.), *Swarm Intelligence and Bio-inspried Computation: Theory and Applications* (pp. 137-168). Elsevier.

Margaritis, K. and Digalakis, J. G. 2002. An Experimental Study of Benchmarking functions for Genetic Algorithms. *International Journal of Computer Mathematics, 79*(4), 403–416.

Margaritis, K. G. and J. G. Digalakis 2001, January. On Benchmarking Functions for Genetic Algorithms. *International Journal of Computer Mathematics*, 1 - 27.

Merheb, A. and Noura, H. and Bateman, F. 2014. Active fault tolerant control of quadrotor uav using sliding mode control. *2014 International Conference on Unmanned Aircraft Systems (ICUAS14)*, (pp. 156 – 166). Orlando, FL, USA.

Merheb, A. and Noura, H. and Bateman, F. 2015. Active fault tolerant control of an octorotor uav. In F. Miranda (Ed.), *Control Theory: Perspectives, Applications and Developments.* NY, USA: Nova science publishers.

Merheb, A. and Noura, H. and Bateman, F. 2015. Design of passive fault tolerant controller of a quadrotor based on sliding mode theory. *International Journal of Applied Mathematics and Computer Science, 25*(3), 561–576.

Neumann, F. and Witt, C. 2010. *Bioinspired Computation in Combinatorial Optimization, Algorithms and Their Computational Complexity* (1 ed., Vol. Natural Computing Series). Springer-Verlag Berlin Heidelberg.

Parpinelli, R. and Lopes, H. 2011. An eco-inspired evolutionary algorithm applied to numerical optimization. *Third World Congress on Nature and Biologically Inspired Computing.* Salamanca, Spain.

Passino, K. M. 2002. Biomimicry of social foraging bacteria for distributed optimization: Models, principles, and emergent behaviors. *Journal Of Optimization Theory And Applications, 115*(3), 603–628.

Passino, K.M. and Y. Liu 2005. *Biomimicry for Optimization, Control, and Automation.* London: Springer.

Puliafito, S. and Puliafito, J. and Grand, M. 2008. Modeling population dynamics and economic growth as competing species: An application to co2 global emissions. *Ecological Economics, 65*(3), 602 – 615.

Shareef, H. and Ibrahim, A. and Mutlag, A. 2015. Lightning search algorithm. *Applied Soft Computing, 36*(1), 315–333.

Song, X. and Sun, L. and Chang, C. 2009. A Hybrid Particle Swarm Algorithm for Job Shop Scheduling Problems and its Convergence Analysis. *2009 International Conference on Artificial Intelligence and Computational Intelligence*, (pp. 99-103). Shanghai, China.

Suganthan, P. and Hansen, N. and Liang, J. and Deb, K. and Chen, Y. and Auger, A. and Tiwari, S. 2005. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. *Natural Computing, 1*(1), 341-357.

Wu, T. Q. and Yao, M. and Yang, J. H. 2016. Dolphin swarm algorithm. *Frontiers of Information Technology & Electronic Engineering, 17*(8), 717-729.

Yang, X. S. 2014. Swarm intelligence based algorithms: a critical analysis. *Evolutionary Intelligence, 7*, 17-28.

Zeigler, M. V. and Bernard, P. 1978. Bacterial Predator-Prey Interaction at Low Prey Density. *Applied and Environmental Microbiology, 36*(1), 11–17.

Zhang, H. and Yuan, M. and Liang, Y. and Liao, Q. 2018. A novel particle swam optimization based on prey-predator relationship. *Applied Soft Computing, 68*(1), 202–218.

Zuo, C. and Wu, L. and Zeng, Z. and Wei, H. 2017. Stochastic fractal based multiobjective fruit fly optimization. *International Journal of Applied Mathematics and Computer Science, 27*(2), 417–433.