Control and Cybernetics

vol. **45** (2016) No. 1

A new concept of an artificial ecosystem algorithm for optimization problems*

by

Dariusz Baczyński

Warsaw University of Technology, Institute of Electrical Power Engineering
Koszykowa 75, 00-662 Warszawa, Poland
dariusz.baczynski@ien.pw.edu.pl

Abstract: This article provides, first, a review of applications of the ecosystem idea in different computational intelligence methods. The article presents the bases of ecosystem operation and a new concept for modelling the phenomena occurring in an ecosystem, with the aim of using these for optimization purposes. The author's original form of the Artificial Ecosystem Algorithm (AEA) and its constituent parts are presented. The construction of the proposed algorithm was dedicated for continuous optimisation. The operation of the Artificial Ecosystem Algorithm is also compared with an Evolutionary Algorithm and PSO for six test functions for various numbers of variables. Conclusions concerning operation, structure and complexity of AEA are provided at the end.

Keywords: artificial ecosystem algorithm, optimization, computational intelligence methods

1. Introduction

During the last half of century, a lot of various computational intelligence methods were developed, originating in a large proportion from some natural phenomena. One can list, among them: Artificial Neural Networks (see Hertz, 1991), Evolutionary Algorithms (Michalewicz, 1992), Fuzzy Systems (see Zadeh, 1965), Simulated Annealing (see Kirkpatrick et al., 1983), Artificial Immune Systems (see de Castro and Timmis, 2003), Particle Swarm Optimization (see Kennedy and Eberhart, 1995), Ant Algorithms (see Dorigo et al., 2006), Bee Algorithms (see Bonabeau et al., 1999), Harmony Search (see Geem et al., 2001), Cultural Algorithms (see Reynolds, 1994), Memetic Algorithm (see Moscato, 1989), Bacterial Algorithms (see Bremermann, 1974), Firefly Algorithm (see Krishnanand and Ghose, 2005), Bat Algorithm (see Yang, 2010a), Tabu Search (see Glover, 1990), Cuckoo Search (see Yang and Deb, 2009), Eagle Strategy (see Yang,

*Submitted: April 2016; Accepted: July 2016.

2010a), Photosynthetic and Enzyme Algorithm (see Yang, 2010a), River Formation Dynamics (see Rabanal et al., 2007), Roach Infestation Optimisation Algorithm (see Havens et al., 2008), Mosquito Host-Seeking (see Feng et al., 2009), Slime Mould Life Cycle (see Monismith and Mayfield, 2008), Intelligent Water Drops (see Shah Hosseini, 2007), Shuffled Frog Leaping Algorithm (see Eusuff and Lansey, 2003), Weed Colonization Optimization (see Mehrabian and Lucas, 2006), Biogeography-Based Optimization (see Simon, 2008), or Symbiotic Organisms Search (see Cheng and Prayogo, 2014).

The Computational Intelligence methods, which are used for optimisation, can be arguably classified as being inspired by single, 'isolated' natural phenomena. This is the case for, among others, algorithms imitating various aspects of living organisms. For instance, there is a group of algorithms imitating the way animals seek their food (e.g., PSO, Ant, or Bee algorithms), another group of algorithms mimics the expansion of plants (e.g., the Invasive Weed Optimization), yet another one is inspired by the phenomena of natural evolution. However, nature has the ability to adapt to different, often very diverse environments, using a synergy of various phenomena. The effectiveness of nature in this process suggests that its emulation should provide an effective optimisation method. In looking for inspirations for a model based on natural phenomena, whose scope of operation is broader than that of the single-process oriented ones, discussed at this point, attention has been paid to the phenomenon of an ecosystem. Ecosystems are still emerging in very different places on Earth. We can see ecosystems ranging from very small to very large. They give the opportunity to live and to evolve for different organisms. Usually, organisms use the ecosystem resources optimally, building together a stable 'construction'. But forming of stable ecosystem may take a lot of time after number of ecosystem collapses.

When we look into an ecosystem, we can see:

- interactions between organisms belonging to the same species (for short periods of time), which can be modelled by, for instance, such paradigms as PSO, Ant Algorithms, Bee Algorithms, Firefly Algorithms, but also a host of other algorithmic structures,

- interactions between organisms belonging to different species (for short periods of time), which can be modelled by, say, PSO, Symbiotic Organisms Search, or Biogeography Based Optimisation,

- interactions between the environment and the organisms of a single specie (for very long periods of time), which can be modelled as evolutionary pressure by, definitely, the Evolutionary Algorithms,

- interactions between organisms belonging to different species (for very long periods of time), which can be modelled as evolutionary pressure, also, certainly, the Evolutionary Algorithms.

Combining all of these interactions in one algorithm should give an opportunity to build a better optimisation method; we must remember, naturally, that this can be a very slow method and a very complicated one (simply because of the characteristics of the source metaphor).

The literature that would discuss the characteristics, and even more so the parameters and the modelling of an ecosystem, is, as of now, scarce. This situation leaves a broad scope for research and, possibly, an effective application of the algorithms developed on this basis in various fields of technology. The work of Binitha and Siva Sathya (2012) presents an overview of three methods classified as those inspired by ecology and ecosystems. One of them is PS^{2}O (see Chen and Yunlong, 2008), being an algorithm emulating the symbiosis between species in an ecosystem. This process is modelled using the parallel PSO algorithms, which exchange information from time to time. The second method is Invasive Weed Optimization (see Mehrabian and Lucas, 2006), which models the spreading of plants over a certain area. The last of the here mentioned method is Biogeography-Based Optimization (see Simon, 2008) which models the spread of species across habitats.

A somewhat different approach to ecosystem phenomena has been adopted by de Boer and Hogeweg (2012), who use an ecosystem model consisting of prey, predators and scavengers (which eat the leftovers of the prey – the final link in the food chain). The purpose of the algorithm, operating in this form is to define the form of the function (in the Lisp language) whose points in space, represented by prey, are known. The task of the co-evolution of the two remaining types of organisms is the possibly best adaptation to the prey – here represented by the potential of eating the prey with no leftovers. The success, whose achievement terminates the running of the respective algorithm, is to generate a predator or a predator-scavenger pair which will eat all prey with no leftovers.

The proposal of Adham and Bentley (2014), who suggested the Artificial Ecosystem Algorithm for the purpose of solving the travelling salesman problem, is reduced to creating a solution from the individuals being its parts (parts of the itinerary sought). These individuals are subject to recombination operations, by which they are able to create a better solution by 'cooperating' with each other.

A solution which employs the relationship between a population of prey and a population of predators is used in Vulli and Agarwal (2008). However, here only the prey is subject to evolution, trying to adopt the best possible 'camouflage' for the environment they live in, which makes them successful in the evolutionary process. An interesting aspect of that paper is also that it discusses the dynamics of particular populations. A model for a simple food chain is used in Hai-Fei and Ding-Wei (2006) for building an optimization algorithm. Flows of energy between different types of artificial organisms (*artorgs*), which depend on their concentration in the solution space, are used in the proposed method to suggest the desirable exploration paths to the individuals.

The idea of an ecosystem is also often used for modelling and developing Digital Business Ecosystems, which, as their authors argue, are much more effective than the service-oriented architectures (see Briscoe et al., 2007, or Briscoe et al., 2011).

Research is also underway concerning the evolution of artificial organisms in artificial ecosystems. Thus, for instance, Lafusa (2007) proposes on this basis the conclusions related to the dynamics of internal components of ecosystems. In Pichler and Canamero (2007), this approach is used for obtaining complex structures of individuals – agents characterized by the desirable behaviours in the respective environment. Likewise, the co-evolution of predators and prey presented in Tzima et al. (2007) produces agents using effective behavioural rules in an artificial environment.

2. The concept of ecosystem algorithm

In creating an Artificial Ecosystem Algorithm (AEA), it would be highly convenient to use an ecosystem model or definition of an ecosystem as such. Each researcher perceives this notion in a specific way, which results in a high variety of papers on ecosystems, as well as in the already mentioned optimization methods.

The following general definition, which was proposed by Weiner, in Weiner (2012), provides the starting point for further analysis: 'A system which carries out production and decomposition processes using energy and supporting the elements circulation cycle, is called the ecosystem'. In addition, as observed by the author quoted, apart from systems which comprise living organisms, no other natural systems are known to have such properties. The above definition is sufficiently general to allow one to consider both the biosphere – on the entire Earth, or a water droplet, to be an ecosystem. For the purpose of this paper, we will interpret an ecosystem as a dynamic system created by the existing and often interdependent, simple or more complex, living organisms, influenced by the inanimate environment.

This meaning is also represented by other definitions of ecosystem (see Weiner, 2012):

- 'an ecosystem is the entirety of organisms inhabiting a certain area, being in mutual relationship to each other, together with their abiotic environment',
- 'an ecosystem denotes an ecological unit which covers all organisms in a given area (i.e., biocoenosis) and interacts with the physical environment in such a way that the flow of energy leads to a distinctly defined food structure, biotic diversity and circulation of matter (i.e., the exchange of elements and compounds) between living and abiotic parts of that unit',
- 'An ecosystem can be defined as an environmental unit composed of different biotic and abiotic components which are interrelated by processes of exchange of chemical compounds and energy. Simply speaking, an ecosystem includes all organisms inhabiting a given environment and abiotic elements of their environment'.

In the research on ecosystems, see Sagoff (2003), there is a distinct division into two principal threads. One of them describes the problems related to the flow of matter and energy inside an ecosystem and between interlinked ecosystems (see Ulanowicz, 2000). This involves research dealing with what is called trophic networks (food webs), which are, in actual practice, reduced to food chains. The second thread focuses on relationships between organisms in terms of natural selection and evolution.

Definitely, these threads should complement each other, because focusing on a single aspect of the interplay between ecosystem components significantly reduces the potential for structuring and explaining a specific behaviour of such ecosystem as a whole as well as of its constituent parts. For instance, food webs cannot explain the co-existence of species which exchange no energy or matter between them.

Now, based on the present author's experience and on previous reviews of different Computational Intelligence methods, together with the assumptions adopted for the ecosystem, an algorithm will be built which is based on relationships between organisms in their environment rather than on an exact representation of a food web, meant as a balance of food flows in the ecosystem as a whole. An assumption was namely made that the relationships between individuals should be modelled both by their mutual relationships within a single group and the relationships between individuals belonging to different groups, co-existing as a part of the ecological system. The relationships existing within a single class of individuals are, for instance: the possibility of crossover between individuals and having offspring, or the capacity for certain collaboration in foraging. One standard example of relationships between different groups of individuals is provided by the evolutionary pressure exerted by some organisms on others.

The following basic assumptions are therefore adopted:

- the abiotic environment of the ecosystem determines the fitness function (being in relation to the objective function of the problem to be optimized),
- three principal types of organisms: plants, herbivores and predators, exist in the ecosystem
- plants are representatives of autotrophs (producers) and process the components existing in the ecosystem,
- plants are eaten by herbivores,
- predators prey on herbivores and eat them,
- all types of organisms reproduce (cross over and mutate) within their own species.

3. The proposed form of the ecosystem algorithm

3.1. Assumptions as to the functioning of the ecosystem algorithm

Based on the concept of the ecosystem and the criteria of its complexity presented in Section 2 above, the following detailed assumptions are proposed for the ecosystem algorithm.

Assumptions on the ecosystem as a whole:

- the ecosystem will consist of three types/groups of interacting organisms: plants, herbivores and predators; this interaction is presented in Fig. 1 in

the form of a trophic network,
- the abiotic environment for the ecosystem determines the fitness function, and the ecosystem algorithm seeks its maximum value; in addition, the fitness function is in relationship to the objective function of the problem to be optimized.

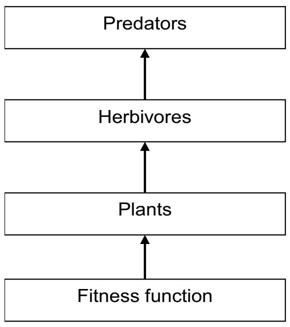


Figure 1. Trophic network of an artificial ecosystem

The assumptions for the particular groups of organisms:
- the fitness function for plants models the content of nutrients in particular points of the ecosystem,
- plants are representatives of autotrophs (producers) and process the components existing in the ecosystem; the higher the content of nutrients in the particular point, the larger the size of the plant growing at that point,
- plants do not change their position,
- the position of the offspring of a pair of plants is determined by the crossover of information from both parents,
- the fitness function for herbivores models a favourable environment – such that provides food (plants) and provides a shelter from predators,
- the offspring of a pair of herbivores receives in its genetic material – inherited from parent individuals – information about how to move and information on their position,
- the fitness function for predators models an environment potentially rich in food (prey in the form of herbivores),
- a predator must eat at least one herbivore during a specific number of iterations, otherwise it dies of hunger,
- the offspring of a pair of predators inherits from the parent individuals, in its genetic material, information about how to move and information on their position,
- each type of organism is able to mutate: in plants this results in the relocation of a plant in the solution space, and in herbivores and predators it results in a change of the movement strategy.

The assumptions on relationships between organisms:

- in each iteration, herbivores eat a specific weight/portion of a selected plant; if plants are smaller than the portion required by the herbivore in each iteration, the herbivore is eating them in such quantity that their combined weights are equal to the required quantity,
- herbivores may transfer plant seeds to their hides,
- herbivores are looking for good locations for their hides using information about the position of the plant they feed on and other information, using a specific movement strategy,
- predators prey on herbivores, and the effect of preying may be twofold: the herbivore may be either scared away and escape in a random direction or it may be eaten by the predator; the effect depends on the relative effectiveness of environment exploration by the herbivore and the predator.

A visual diagram of the structure of individuals of the ecosystem algorithm and mutual relationships is presented in Fig. 2.

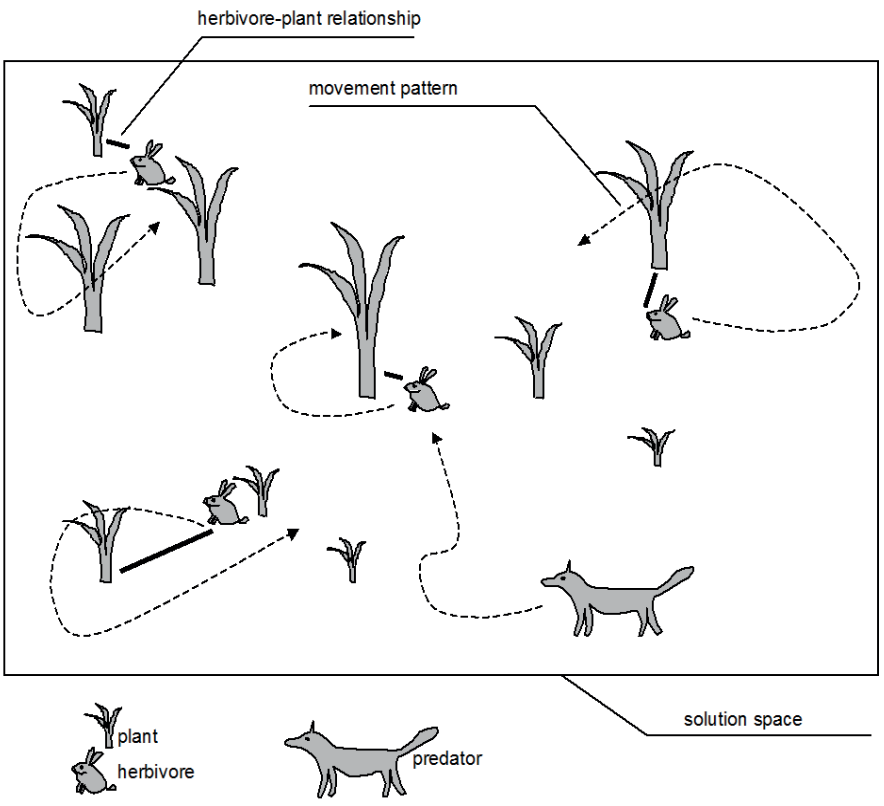


Figure 2. Relationships between individuals (own elaboration)

The assumptions made for the relationships between particular types of organisms and the operation of the algorithm as a whole result in the need for individual organisms to store information about their statuses (Fig. 3). The information can be split into two groups.

One group is the kind of the organism's genotype. For plants, which are stationary, the genotype is defined only by the point in the solution space, represented by the position co-ordinates of the plant. For organisms, which are able to move (herbivores and predators), the genotype includes additional variables, defining how organisms move in the solution space. They define how the organism's movements are affected by:

- its own experience,
- other organisms of the same species, and
- selected organisms from the group of organisms on which it feeds.

Another set of information stored, by the organism, is the supplementary information, defining the quality of the solution represented by a given organism, its experiences so far, and feeding relationships.

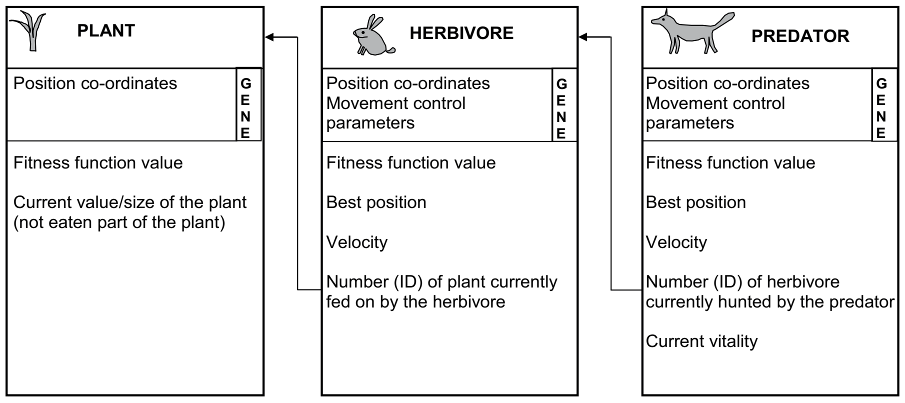

PLANT
Position co-ordinates
GENE
Fitness function value
Current value/size of the plant (not eaten part of the plant)
HERBIVORE
Position co-ordinates
Movement control parameters
GENE
Fitness function value
Best position
Velocity
Number (ID) of plant currently fed on by the herbivore
PREDATOR
Position co-ordinates
Movement control parameters
GENE
Fitness function value
Best position
Velocity
Number (ID) of herbivore currently hunted by the predator
Current vitality

Figure 3. Information stored by particular types of organisms (own elaboration)

3.2. Structure and elements of ecosystem algorithm

A key element of any optimization algorithm is constituted by how it transforms the decision variables, used for describing solutions for a specific kind of problem, into variables on which the algorithm actually operates. In the proposed basic version of the ecosystem algorithm, which operates on real numbers, an encoding in evolutionary algorithms is adopted, called 'natural' encoding. This means that each decision variable of a problem corresponds to a single variable, which represents the position of the given kind of organism in the solution space. This space is the abiotic environment of the ecosystem – a niche, in which the ecosystem as a whole will develop.

From the practical point of view: each organism stores in itself a set of real numbers representing its position. The general structure of the proposed ecosystem algorithm is presented in Fig. 4. The particular blocks of this diagram are

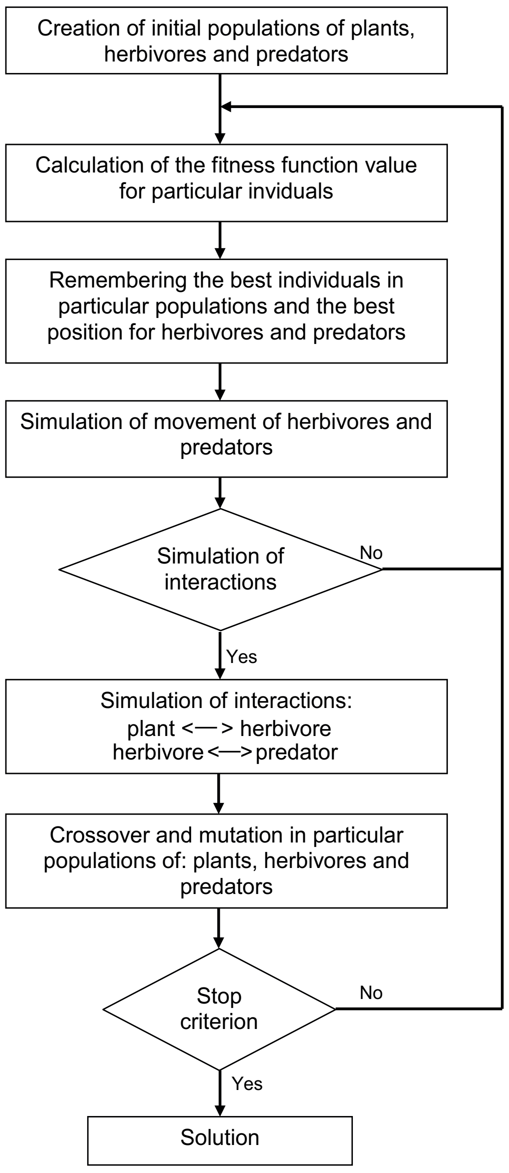
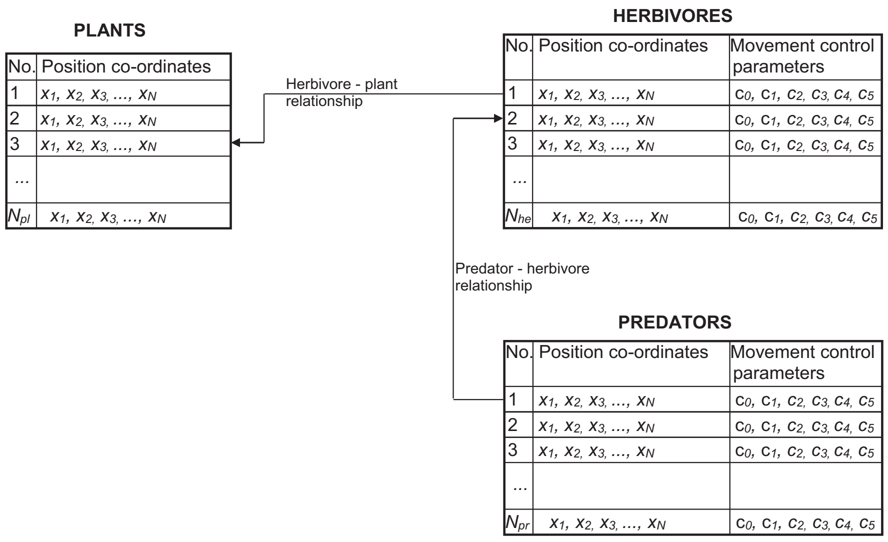

Creation of initial populations of plants, herbivores and predators
Calculation of the fitness function value for particular inviduals
Remembering the best individuals in particular populations and the best position for herbivores and predators
Simulation of movement of herbivores and predators
Simulation of interactions
No
Yes
Simulation of interactions: plant <—> herbivore herbivore<—>predator
Crossover and mutation in particular populations of: plants, herbivores and predators
Stop criterion
No
Yes
Solution

Figure 4. General structure of the proposed ecosystem algorithm

described in the subsequent parts of the present sub-chapter, describing the phases of algorithm operation (in accordance with the respective block designations).

1) Establishing initial populations: The algorithm starts from establishing initial populations of all types of organisms, with the following population sizes: plants – N_{pl}, herbivores – N_{he}, and predators – N_{pr} (Fig. 5). The solution space is assumed to be limited, that is – each of its N dimensions can take values from a specific range only. Each plant will be randomly assigned a point in the solution space to be placed in. For predators and herbivores, in addition to the random assignment of their positions, randomly chosen are, as well, the parameters, which govern the way they move (c_0, c_1, c_2, c_3, c_4, c_5, based on the relationships, described in 4.3 and 4.4) within the ranges ($\min_c$, $\max_c$) for the respective c^{th} parameter.


PLANTS
HERBIVORES
PREDATORS
Herbivore - plant relationship
Predator - herbivore relationship

PLANTS

No.	Position co-ordinates
1	$x_1, x_2, x_3, \ldots, x_N$
2	$x_1, x_2, x_3, \ldots, x_N$
3	$x_1, x_2, x_3, \ldots, x_N$
...	
N_{pl}	$x_1, x_2, x_3, \ldots, x_N$

HERBIVORES

No.	Position co-ordinates	Movement control parameters
1	$x_1, x_2, x_3, \ldots, x_N$	$c_0, c_1, c_2, c_3, c_4, c_5$
2	$x_1, x_2, x_3, \ldots, x_N$	$c_0, c_1, c_2, c_3, c_4, c_5$
3	$x_1, x_2, x_3, \ldots, x_N$	$c_0, c_1, c_2, c_3, c_4, c_5$
...		
N_{he}	$x_1, x_2, x_3, \ldots, x_N$	$c_0, c_1, c_2, c_3, c_4, c_5$

PREDATORS

No.	Position co-ordinates	Movement control parameters
1	$x_1, x_2, x_3, \ldots, x_N$	$c_0, c_1, c_2, c_3, c_4, c_5$
2	$x_1, x_2, x_3, \ldots, x_N$	$c_0, c_1, c_2, c_3, c_4, c_5$
3	$x_1, x_2, x_3, \ldots, x_N$	$c_0, c_1, c_2, c_3, c_4, c_5$
...		
N_{pr}	$x_1, x_2, x_3, \ldots, x_N$	$c_0, c_1, c_2, c_3, c_4, c_5$

Figure 5. Randomly assigned elements in the initial populations (own elaboration)

One plant, meant to 'feed' on, is randomly assigned to each herbivore (a single individual). Similarly, each predator is assigned an initial vitality level Z_{ppr} and one herbivore to prey on.

2) Calculating the value of the fitness function and remembering the best individuals: The next stage of the algorithm is to assign, for each i^{th} individual from each population, the value of the fitness function, $fitness_i$. This function is identical for plants, herbivores and predators – all three groups are solving the same optimization problem. There is no difference in terms of shape, gradients and local optimums of the hypersurface on which the individuals move. This allows for the direct comparisons between the solutions attained by the individuals from different populations. This is of key importance for the selection of best solutions and for the emulation of interactions between herbivores and predators. The fitness function should be linked to the form of the target function of the problem to be solved. Like in the evolutionary algorithms, the fitness function is assumed to be positive for all individuals, whereas the algorithm will seek to maximize this function. Based on the value of the fitness function, the best-performing individuals in particular populations are selected and remembered, and the best solution found so far is remembered. Likewise, each herbivore and predator remembers its best position found so far. For plants, in the first step (and following each 'birth' of a new plant), their size is also determined, and the size depends on the fitness function. The average initial size of plants in a population is assumed to be 1, and the size $W_{ppl,i}$ of the i^{th} plant will be determined by the following equation:

$$W_{ppl,i} = \frac{fitness_i}{\sum\limits_{i=0}^{N_{pl}} fitness_i} * N_{pl} * W_{ppl}. \tag{1}$$

3) Simulation of movements of individuals: The next step of the algorithm is to simulate the movement of herbivores and predators in the solution space. The positions of these organisms are determined like in the PSO algorithm. The position of an organism is determined separately in each dimension and depends on the previous position and on the current velocity of the individual in a given dimension (as expressed in equation (2) below):

$$x_k^{t+1} = x_k^t + v_k^{t+1}. \tag{2}$$

The velocity of organisms, assuming that the duration of a single iteration is 1, defines, in fact, the difference (vector) between the initial and the final position. The velocity of a herbivore in the k^{th} dimension in step t+1 is in accordance with equation (3), given below. Thus, the herbivore's movement incorporates:

- its own experience, being the previous velocity and the best position found so far,

- other organisms of the same species, by incorporating the best position found so far by all organisms and the best position found by a neighbour of the particular organism,

- selected organisms from the group of organisms, on which it feeds, by taking into account the position of the plant, to which it is assigned, and the plant with the best value of the fitness function,

$$\begin{aligned} v_k^{t+1} &= c_0 r_0^t v_k^t + c_1 r_1^t (y_k^t - x_k^t) + c_2 r_2^t (y_k^{p*t} - x_k^t) + c_3 r_3^t (y_k^{n*t} - x_k^t) + \\ &+ c_4 r_4^t (y_k^{plant*t} - x_k^t) + c_5 r_5^t (y_k^{bplant*t} - x_k^t) \end{aligned} \tag{3}$$

where: $r_0^t, r_1^t, r_2^t, r_3^t, r_4^t, r_5^t$ – are random numbers from the range of (0,1), taken with uniform distribution,

v_k^t – herbivore's velocity in the previous step in the k^{th}dimension,

x_k^t – herbivore's co-ordinate in the previous step in the k^{th}dimension,

y_k^t – the co-ordinate of the herbivore, representing its best position found so far,

y_k^{p*t} – the co-ordinate of the best position found so far (of all the herbivores in all iterations),

y_k^{n*t} – the co-ordinate of the best position found in the previous iteration among neighbours of the respective herbivore within the radius of w_n from its position on the list of herbivores,

$y_k^{plant*t}$ – the co-ordinate of the plant the herbivore is assigned to and which is eaten by the herbivore,

$y_k^{bplant*t}$ – the co-ordinate of the plant with the so-far best fitness function among plants,

c_0, c_1, c_2, c_3, c_4, c_5 – weights, determining the contribution of the particular components to the resultant velocity of the herbivore.

The velocity of a predator in the k^{th} dimension in the i^{th} step proceeds in accordance with Equation (4). Thus, the predator's movement incorporates, like for the herbivore, its own experience and that of the other organisms of the same species. In addition, the best positions found so far by all herbivores and by the herbivore preyed on by the particular predator are incorporated.

$$\begin{aligned} v_k^{t+1} &= c_0 r_0^t v_k^t + c_1 r_1^t (y_k^t - x_k^t) + c_2 r_2^t (y_k^{p*t} - x_k^t) + c_3 r_3^t (y_k^{n*t} - x_k^t) + \\ &+ c_4 r_4^t (y_k^{her*t} - x_k^t) + c_5 r_5^t (y_k^{bher*t} - x_k^t) \end{aligned} \tag{4}$$

where:

$r_0^t, r_1^t, r_2^t, r_3^t, r_4^t, r_5^t$ – are random numbers from the range (0,1) with uniform distribution,

v_k^t – predator's velocity in the previous step in the k^{th} dimension,

x_k^t – predator's co-ordinate in the previous step in the k^{th} dimension,

y_k^t – predator's co-ordinate representing its best position found so far,

y_k^{p*t} – the co-ordinate of the best position found so far (of all the predators in all iterations),

y_k^{n*t} – the co-ordinate of the best position found in the previous iteration among neighbours of the respective predator within the radius of w_n from its position on the list of herbivores,

y_k^{her*t} – the co-ordinate of the best position of the herbivore on which the predator preys,

y_k^{bher*t} – the co-ordinate of the best position found so far by herbivores,

c_0, c_1, c_2, c_3, c_4, c_5 – weights, determining the contribution of the particular component to the resultant velocity of the predator.

As previously mentioned, weights $c_0, \cdots, c_5$ will be randomly selected from specific ranges, both for herbivores and predators. In addition, the value of these parameters will change by the operation of the crossover and mutation operators. The adoption of broad variability ranges for these parameters, including negative values, will provide for some optimization of the way the solution space is explored by these organisms. This means that the algorithm will be optimizing the solution to the problem, as well as improving its own operation at the same time – i.e., in some sense, it will be a self-adaptive algorithm.

4) Simulation of interactions between groups of individuals: After performing the simulation of movements of herbivores and predators, a condition is tested as to whether there should be a shift to the simulation of interactions between individuals. In its simplest form: the condition verifies if a specific number of movement simulation iterations have been performed. If this condition is met, the first step shifts to simulating the interactions between individuals of different types.

First, interactions between plants and herbivores are simulated. Each herbivore eats a certain weight of plants H_{pl} (food requirement). First, it feeds on the plant it is assigned to, and reduces its size. If the plant is smaller than the consumer's requirement, the consumer randomly selects another plant and continues feeding. It eats the new plants so long as its requirement is met (see further on for the potential consequences of lack of possibility to fulfill this condition).

Then, preying of each predator on the herbivores assigned to it, is simulated. The outcome of preying depends on the comparison of the best solutions found so far by the predator and its prey. If, during its past lifetime, the predator has found a better solution than the herbivore, the predator eats the herbivore, and its vitality level is increased by one unit. Otherwise, the herbivore runs away in a random direction, and the predator remembers the place of encounter (as its best position), but due to the failure of hunt its vitality level decreases. If the vitality of a given predator decreases below zero, the predator dies. The initial vitality level of each predator is determined at its 'birth' as equal Z_{ppr}.

5) Crossover and mutation: An interplay between groups of organisms leads to the reduction of populations. Therefore, functions are needed to simulate the birth of new individuals in particular populations. In the adopted solution, two organisms of one type cross over, forming a new offspring organism. This solution is different from those adopted in the typical evolutionary algorithms, in which two offspring individuals are usually formed following the crossover. Crossover is a random process, both in terms of the selection of individuals for breeding and in terms of the selection of the parental traits, forming an offspring individual. On the one hand, crossover, by which a single descendant is obtained, allows one to test the combinations of traits of a larger number of individuals.

On the other hand, a crossover, typical to evolutionary algorithms, makes it possible to test in one step two different combinations of parental individuals. In the ecosystem algorithm, crossover with one descendant is applied – due to the basic function of crossover, i.e., to replenish the quantity of particular populations and due to its simple implementation. For plants, crossover affects only the information about the position of parents (Fig. 6).

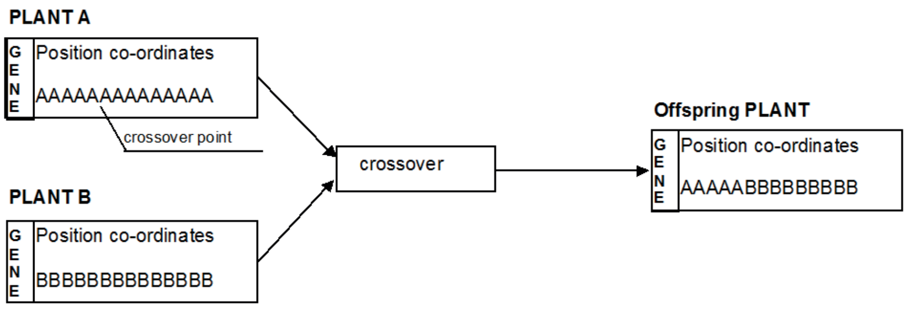


Figure 6. Crossover of plants

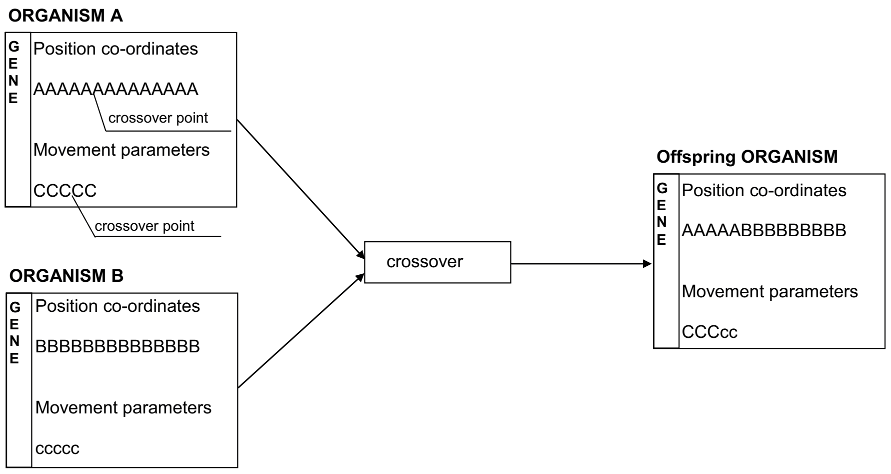


Figure 7. Crossover of predators and herbivores

However, for herbivores and predators, crossover is applied both to the position of the parents and to the parameters describing their movement, that is, the factors $c_0, \ldots, c_5$ from Equations (3) and (4), see Fig. 7. In all cases, traditional one-point crossover is used. The crossover in the proposed algorithm fulfils two functions. One (for all groups of organisms – the crossover of information on the position) is the function that is fulfilled by crossover in a classical evolutionary algorithm. Due to this, the offspring individual can combine information on how to solve the problem held by the parental individuals. The other function of the crossover (for herbivores and predators – the crossover of factors $c_0, \cdots, c_5$) allows one to create offspring individuals, which behave in a way resulting from how the parent organisms behaved. Behaviour is taken to mean a pattern of movement in foraging.

To ensure the stability of both the algorithm and of the evolutionary pressure, the numbers of organisms in all populations are assumed to be stable in all generations. The ratios of the number of individuals in particular groups should be similar to those observed in nature, which means that the number of plants should be much higher (at least by an order of magnitude) than the number of herbivores, and the number of the latter should be much higher than the number of predators. The precondition for successful completion of the algorithm is to preserve the quantity of organisms on each level of the food web. For if the quantitative proportions between particular groups of individuals break down, the ecosystem may fail as well, say, for instance, when all plants get eaten up in several iterations. This, in turn, results in breaking the operation of the ecosystem algorithm, and hence in the failure to achieve the optimum of the target function sought.

The next step of the algorithm is simulation of mutation – identical to the mutation used in evolutionary algorithms. For plants, it applies to the position of the plant (Fig. 8), and for herbivores and predators, subject to mutation are factors $c_0, \ldots, c_5$ (Fig. 9). In both cases, a simple one-point mutation is used. Each group of individuals has its characteristic probability of mutation: plants – p_{mpl}, herbivores – p_{mhe}, predators – p_{mpr}.

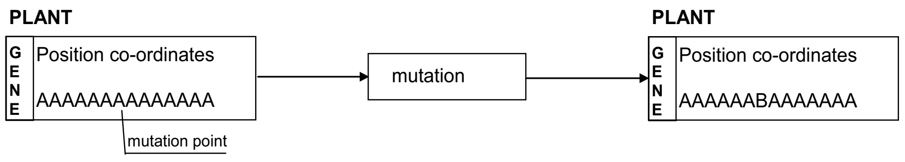


Figure 8. Mutation of plants

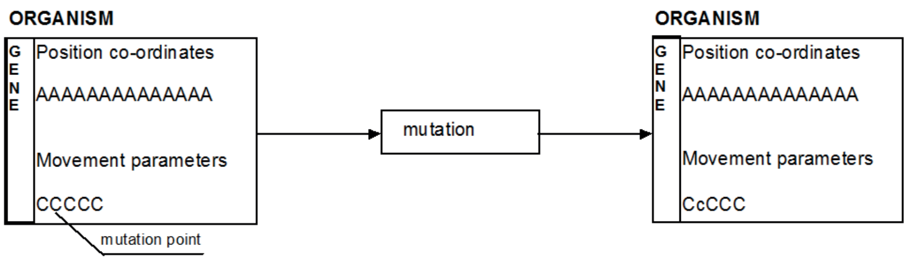


Figure 9. Mutation of predators and herbivores

The mimicking of the transfer of plant seeds by herbivores is a process similar to mutation – they may be brought to, e.g. the herbivore's shelter. Conform to the logic of the assumptions made, it means that the plant is transferred to the best point in the solution space, found so far by the herbivore feeding on a given plant. The number of such movements is represented by the value of the parameter N_{mpl}.

6) Operation ending: The final part of the algorithm is to test the pre-defined condition for successful completion of the operation. This condition can take any form – depending on the particular implementation and the problem to be solved. In the solution modelled here, the stop criterion is a specific number of iterations N_{iter}.

7) Parameters of the ecosystem algorithm:

N_{iter} – number of iterations of the algorithm (the stop criterion),

N_{in} – number of simulation iterations between interactions between individuals,

N_{pl} – size of plant population in the ecosystem,

N_{he} – size of herbivore population in the ecosystem,

N_{pr} – size of predator population in the ecosystem,

p_{mpl} – probability of plant mutation,

p_{mhe} – probability of herbivore mutation,

p_{mpr} – probability of predator mutation,

N_{mpl} – number of transfers of plants by herbivores,

Z_{ppr} – initial vitality level of a predator,

W_{ppl} – coefficient of initial size of a plant,

H_{pl} – herbivore's food requirement,

w_n – extent (radius) of the neighbourhood, in which the best predator or herbivore are sought (Equations (3) and (4)); at this point, the neighbourhood is taken to be the relationship between individuals in terms of mutual position on the list of individuals of a specific type as opposed to the distance in the solution space; such approach is much less computation-intensive than determining the geometric distance between the individuals,

min_c, max_c – lower and upper boundaries of the range, in which values of factors c_0, c_5 from Equations (3) and (4) are randomly selected.

Practical ranges of parameters:

N_{pl}, N_{he}, N_{pr} – ratios of the number of individuals in particular groups should be similar as in nature, which means that the number of plants should be much higher (at least by an order of magnitude) than the number of herbivores, and the number of the latter should be much higher than the number of predators. In order to obtain good results one should start with the number of plants of few hundred up to few thousands. This would entail the appropriate number of herbivores and predators.

p_{mpl}, p_{mhe}, p_{mpr} – mutation probabilities should be (like in other EAs) between 0.01 and 0.3.

N_{mpl} – $(1; 10)$

N_{in} – $(5; 500)$

H_{pl} – $(0.02; 0.5)$

Z_{ppr} – $(50; 200)$
w_n – $(1; 5)$
min_c, max_c – $(-2.0; 2.5)$
N_{iter} – $(1000; 1000000)$
W_{ppl} – $(0.5; 3.0)$ coefficient of initial size.

These values are based on the experience gathered with other algorithms (PSO, EA), but they do also result from the construction of the algorithm (number of organisms in populations), and finally, they result, as well, from the tests being made during the work on this algorithm.

4. The tests

The proposed Artificial Ecosystem Algorithm (AEA) was tested on six commonly used test functions with simple bounds as constraints (see Yang, 2010b, or Bratton and Kennedy, 2007), that is: Ackley's, Rosenbrock's, DeJong's, Griewank's, Rastrigin's and Schwefel's function.

The tests were performed for a large number of algorithm iterations (10^5), meant to find which method can explore the solution space continuously when it has such an opportunity. This is a significant feature of an optimisation algorithm, used in the so called 'off-line' problems, where obtaining a slightly better solution is much more important than obtaining it fast.

The results were compared with the results obtained from the Evolutionary Algorithm (EA) and the Particle Swarm Optimization (PSO). All of these mentioned algorithms were implemented in the C++ language (DEV-C++ IDE environment).*

The first testing function was the Ackley's function (see Yang, 2010b):

$$f(x) = -20 \exp\left[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right] - \exp\left[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right] + 20 + e \tag{5}$$

where: $n = 1, 2, ...$; and $-32.768 \leqslant x_i \leqslant 32.768$ for $i = 1, 2, ..., n$. This function has the global minimum $f_* = 0$ at $x_* = (0,\ 0,\ ...,\ 0)$.

The second was the Rosenbrock's function (see Yang, 2010b, or Bratton and Kennedy, 2007):

$$f(x) = \sum_{i=1}^{n-1}\left[(x_i - 1)^2 + 100\left(x_{i+1} - x_i^2\right)^2\right] \tag{6}$$

where: $n = 1, 2, ...$; and $-30.0 \leqslant x_i \leqslant 30.0$ for $i = 1, 2, ..., n$. This function has the global minimum $f_* = 0$ at $x_* = (1,\ 1,\ ...,\ 1)$.

*The respective project files are available to the interested researchers upon e-mailed request.

The third was De Jong's function (see Yang, 2010b, or Bratton and Kennedy, 2007):

$$f(x) = \sum_{i=1}^{n} x_i^2 \tag{7}$$

where: $n = 1, 2, ...$; and $-100.0 \leqslant x_i \leqslant 100.0$ for $i = 1, 2, ..., n$. This function has the global minimum $f_* = 0$ at $x_* = (0,\ 0,\ ...,\ 0)$.

The fourth was the Griewank's function (see Yang, 2010b, or Bratton and Kennedy, 2007):

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{8}$$

where: $n = 1, 2, ...$; and $-600.0 \leqslant x_i \leqslant 600.0$ for $i = 1, 2, ..., n$. This function has the global minimum $f_* = 0$ at $x_* = (0,\ 0,\ ...,\ 0)$.

The fifth was the Rastrigin's function (see Yang, 2010b, or Bratton and Kennedy, 2007):

$$f(x) = 10n + \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i)\right] \tag{9}$$

where: $n = 1, 2, ...$; and $-5.12 \leqslant x_i \leqslant 5.12$ for $i = 1, 2, ..., n$. This function has the global minimum $f_* = 0$ at $x_* = (0,\ 0,\ ...,\ 0)$.

The sixth was the Schwefel's function (see Yang, 2010b, or Bratton and Kennedy, 2007):

$$f(x) = -\sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right) \tag{10}$$

where: $n = 1, 2, ...$; and $-500 \leqslant x_i \leqslant 500$ for $i = 1, 2, ..., n$. This function has the global minimum $f_* \approx -418.9829{\cdot}n$ at $x_* = (420.9687, 420.9687, ..., 420.9687)$.

To ensure that the evaluation function be positive for all individuals and all the possible numbers of dimensions, the following modification of Schwefel's function was considered:

$$f(x) = -\frac{1}{n}\sum_{i=1}^{n} x_i \sin\left(\sqrt{|x_i|}\right) + 500 \tag{11}$$

This function has the global minimum of $f_* \approx 81.0171$ at the same point as the original function,
$x_* = (420.9687, 420.9687, ..., 420.9687)$.

The optimisation task, for all the here presented test functions, is their minimisation (minimisation of the test - objective function). But all of the compared algorithms were built in such a way that they seek the maximum of the fitness function. Therefore, it was necessary to apply an appropriate transformation of the objective function. All of the compared algorithms had the same fitness function calculated on the basis of the objective function as:

$$F_{fitness} = \frac{1}{F_{obj} + 1}. \tag{12}$$

Of course, such a transformation can make it more difficult to find the optimum for some of the test problems. Yet, if one is not comparing the results to those of other researchers, it does not influence the overall rating of the proposed algorithm.

The here presented AEA algorithm has 15 basic parameters. During simulations, the following values were used (tested):

- N_{pl} – {250; 500; 1000};
- N_{he} – {25; 50; 75; 100; 200};
- N_{pr} – {3; 5; 10};
- p_{mpl}, p_{mhe}, p_{mpr} – {0.07};
- N_{mpl} – {2};
- N_{in} – {10; 100; 200};
- H_{pl} – {0.1};
- Z_{ppr} – {100};
- w_n – {3};
- min_c, max_c – (-0.5; 2.0);
- N_{iter} – {100 000, 1 000 000};
- W_{ppl} – {1} coefficient of initial size.

In the simulations, the following values for EA were used (tested):

- population size {50; 100; 200};
- crossover probability {0.2; 0.5; 0.8};
- mutation probability {0.03; 0.07; 0.15};
- linear scaling of fitness function {0; 1}(off/on);
- elitist selection {0; 1}(off/on);
- one point crossover, arithmetic crossover.

In the respective simulations, the following values for PSO were used (tested):

- coefficient c_0 {0.7; 1.7; 2.0};
- coefficient c_1 {0.1; 0.5};
- coefficient c_2 {0.1; 0.2; 0.5};
- coefficient c_3 {0.1; 0.2; 0.5};
- extent (radius) of the neighbourhood {3; 5; 10};
- number of particles {50; 100};
- number of iterations between subsequent disturbances {50; 160; 500}(disturbance means random relocation of all particles).

The adopted formula for the particle velocity in the PSO was:

$$v_k^{t+1} = c_0 r_0^t v_k^t + c_1 r_1^t (y_k^t - x_k^t) + c_2 r_2^t (y_k^{p*t} - x_k^t) + c_3 r_3^t (y_k^{n*t} - x_k^t)$$

x_k^t – particle position in previous iteration in $k-$th dimension,

y_k^t – the best position of this particle found so far,
y_k^{p*t} – the best position within the swarm found so far,
y_k^{n*t} – the best position within the particle's neighbourhood found so far.

As a part of the algorithm stability testing, the impact of the parameters, which control particular algorithms and the impact of random selection on the results was analysed for both functions with 10 and 100 variables (n =10 and n =100). For each algorithm, combinations of control parameters were selected which, as the experience shows, should lead to obtaining good results. Thirty-five such combinations were selected for the Evolutionary Algorithm, 36 for the PSO, and 33 for the Ecosystem Algorithm. Ten test runs were performed for each combination of parameter settings (with different staring points of the random number generator) in order to test how randomness affects the results. The total number of simulations for (n =10 variables) AEA was 1980 (33 (number of parameter combinations) x 10 (runs for each combination) x 6 (test functions)). Consequently, the total number of simulations for all algorithms and all dimension sets exceeded 12 000.

The results were summarized for two scenarios – for 10 and 100 variables of the target function (Tables 1 - 6) and for 1000 variables (Tables 7 - 12).

The tables represent:

- for the combination of control parameters values, for which the best results were obtained: the minimum value of the criterion function (min_Best), the maximum value (max_Best) and the average value (avg_Best) from 10 test runs; the time to obtain the best result (time_min_Best) and the average time to obtain the results (time_avg_Best) from 10 test runs are provided; the number of fitness function evaluations to obtain the best result (ffe_min_Best) and the average number of fitness function evaluations to obtain results (ffe_avg_Best) from 10 test runs are provided,
- for all test runs: the average value (avg_All) and the maximum value (max_All) of the criterion function.

All algorithms performed 100,000 iterations.

Based on the test results for the target function with 100 variables, the best settings for particular algorithms were selected. For these settings, tests with target function with 1,000 variables were conducted. Like in the previous tests, all algorithms performed 100,000 iterations.

The results presented in Tables 7 through 11 show that all optimization problems with 1,000 variables are complex ones, and that the Ecosystem Algorithm managed not to get stuck in the local optimum and 'attempted' to reach the global optimum in most cases.

To check the behaviour of the algorithms for a large number of iterations, simulations with one million (10^6) iterations were performed (Figs. 10 through 15). Fig. 10 presents how the outputs varied with time for Ackley's function. The different lengths of curves for particular algorithms result from different durations per iteration for particular algorithms. The Ecosystem Algorithm achieves the successive levels of the target function: 10^{-1} after about 2,300 [s], 10^{-2} after about 3,000 [s] and 10^{-3} after about 3,900 [s]. Finally, it arrives at

Table 1. The comparison of factors describing the stability of particular algorithms for the Ackley's function with 10 and 100 variables (Best values are indicated in italics, in this and in the following tables)

Variable number	10			100		
Algorithm	EA	PSO	AEA	EA	PSO	AEA
Min_Best	4.1E-03	*4.0E-15*	*4.0E-15*	4.1E-01	1.1E-04	*1.9E-10*
Max_Best	4.1E-03	*4.0E-15*	*4.0E-15*	7.4E-01	1.9E-04	*3.1E-08*
Avg_Best	4.1E-03	*4.0E-15*	*4.0E-15*	5.4E-01	1.5E-04	*7.2E-09*
Time_min_Best [s]	*1.6*	5.2	5.6	*47.8*	231.1	130.4
Time_avg_Best [s]	*2.8*	8.3	8.7	*49.3*	211.4	125.3
Ffe_min_Best	*599 600*	1 510 900	1 150 300	*4 836 250*	9 969 800	5 292 634
Ffe_avg_Best	*1 038 520*	2 392 910	1 780 045	*4 955 365*	9 102 970	5 278 690
Avg_All	3.3E-01	4.4E-01	*1.1E-03*	8.2E+00	8.5E+00	*9.3E-01*
Max_All	1.2E+01	4.3E+00	*7.2E-02*	2.1E+01	1.9E+01	*8.6E+00*

Table 2. Comparison of factors describing the stability of particular algorithms for the Rosenbrock's function with 10 and 100 variables

Variable number	10			100		
Algorithm	EA	PSO	AEA	EA	PSO	AEA
Min_Best	4.5E-04	1.0E-14	*2.0E-16*	1.3E+02	1.8E-01	*8.7E-05*
Max_Best	1.5E+01	1.5E-14	*1.0E-15*	3.4E+02	*7.0E+01*	1.5E+02
Avg_Best	5.3E+00	1.2E-14	*6.8E-16*	2.8E+02	*8.8E+00*	7.8E+01
Time_min_Best [s]	52.2	34.0	*33.3*	61.0	154.0	113.5
Time_avg_Best [s]	30.9	*35.6*	37.1	*61.9*	160.8	105.7
Ffe_min_Best	18 914 200	9 368 000	*9 145 485*	4 907 350	5 819 200	*3 044 410*
Ffe_avg_Best	11 200 600	*9 596 160*	10 197 909	4 968 765	6 134 140	*2 832 436*
Avg_All	5.9E+00	3.4E+01	*4.3E-03*	1.8E+07	2.6E+07	*1.4E+04*
Max_All	2.2E+01	5.8E+02	*6.7E-01*	4.6E+08	1.4E+08	*4.5E+06*

Table 3. Comparison of factors describing the stability of particular algorithms for De Jong's function with 10 and 100 variables

Variable number	10			100		
Algorithm	EA	PSO	AEA	EA	PSO	AEA
Min_Best	2.3E-08	6.2E-17	*3.6E-17*	2.9E-02	1.8E-15	*5.4E-16*
Max_Best	5.2E-07	*1.1E-16*	1.1E-16	8.4E-02	2.9E-15	*9.8E-16*
Avg_Best	1.8E-07	9.2E-17	*9.1E-17*	5.2E-02	2.2E-15	*7.6E-16*
Time_min_Best [s]	11.2	*4.0*	9.6	*146.5*	213.5	238.6
Time_avg_Best [s]	11.1	*3.7*	14.4	*146.8*	216.3	217.1
Ffe_min_Best	4 829 100	*1 257 400*	2 081 870	19 802 600	*9 775 900*	13 064 440
Ffe_avg_Best	4 713 115	*1 197 600*	3 119 115	19 822 600	*9 902 920*	11 859 492
Avg_All	8.3E-04	1.9E+00	*4.6E-05*	1.5E+04	2.1E+04	*2.3E+01*
Max_All	4.4E-02	3.9E+01	*1.4E-02*	1.7E+05	8.3E+04	*5.9E+03*

Table 4. Comparison of factors describing the stability of particular algorithms for the Griewank's function with 10 and 100 variables

Variable number	10			100		
Algorithm	EA	PSO	AEA	EA	PSO	AEA
Min_Best	4.9E-04	*0.0E+00*	1.1E-16	5.0E-01	3.9E-13	*1.1E-16*
Max_Best	9.6E-02	*3.0E-02*	5.2E-02	9.8E-01	*1.5E-02*	*1.5E-02*
Avg_Best	5.7E-02	*1.3E-02*	1.6E-02	7.0E-01	3.2E-03	*1.5E-03*
Time_min_Best [s]	9.0	*1.4*	20.2	296.0	299.4	*292.8*
Time_avg_Best [s]	*7.9*	15.4	36.5	294.6	290.2	*273.8*
Ffe_min_Best	3 030 700	*350 200*	6 479 280	19 994 200	*10 000 000*	10 826 460
Ffe_avg_Best	*2 623 535*	4 015 390	11 776 482	19 828 700	*9 983 640*	10 253 494
Avg_All	5.7E-02	3.9E-01	*4.1E-02*	1.4E+02	1.9E+02	*2.0E-01*
Max_All	2.1E-01	1.4E+00	*1.6E-01*	1.6E+03	7.5E+02	*1.7E+01*

Table 5. Comparison of factors describing the stability of particular algorithms for the Rastrigin's function with 10 and 100 variables

Variable number	10			100		
Algorithm	EA	PSO	AEA	EA	PSO	AEA
Min_Best	3.8E-05	*0.0E+00*	*0.0E+00*	*2.3E+00*	7.0E+00	6.3E-03
Max_Best	2.2E-04	*0.0E+00*	*0.0E+00*	*5.7E+00*	6.7E+01	1.9E+01
Avg_Best	1.2E-04	*0.0E+00*	*0.0E+00*	*4.0E+00*	3.2E+01	1.1E+01
Time_min_Best [s]	13.9	*0.3*	5.0	*62.4*	262.3	328.5
Time_avg_Best [s]	13.2	*1.0*	6.2	*63.7*	261.4	328.5
Ffe_min_Best	4 990 000	*93 300*	1 795 460	*4 868 750*	9 970 000	12 995 060
Ffe_avg_Best	4 721 490	*271 760*	2 187 460	*4 973 340*	9 959 900	12 964 746
Avg_All	*7.1E-04*	7.5E+00	2.8E-02	1.4E+02	6.1E+02	*8.5E+01*
Max_All	*3.0E-02*	2.3E+01	4.1E+00	1.6E+03	1.2E+03	*5.2E+02*

Table 6. Comparison of factors describing the stability of particular algorithms for the Schwefel's (11) function with 10 and 100 variables

Variable number	10			100		
Algorithm	EA	PSO	AEA	EA	PSO	AEA
Min_Best	*8.1E+01*	*8.1E+01*	*8.1E+01*	8.2E+01	1.0E+02	*8.1E+01*
Max_Best	*8.1E+01*	*8.1E+01*	*8.1E+01*	*8.2E+01*	1.4E+02	9.2E+01
Avg_Best	*8.1E+01*	*8.1E+01*	*8.1E+01*	*8.2E+01*	1.2E+02	8.5E+01
Time_min_Best [s]	6.7	4.0	*1.9*	*51.8*	247.1	262.5
Time_avg_Best [s]	13.1	24.3	*5.8*	*51.5*	249.0	262.2
Ffe_min_Best	2 991 500	1 190 600	*439 150*	*4 990 800*	9 949 700	15 499 285
Ffe_avg_Best	5 879 860	7 149 830	*1 342 830*	*4 976 060*	9 989 820	15 465 357
Avg_All	8.1E+01	8.9E+01	*8.1E+01*	1.5E+02	1.9E+02	*1.4E+02*
Max_All	9.5E+01	1.8E+02	*1.2E+02*	4.0E+02	3.3E+02	*2.9E+02*

Table 7. Comparison of factors of particular algorithms for the Ackley's function with 1,000 variables

Variable number	1000		
Algorithm	EA	PSO	AEA
Min_Best	1.9E+01	1.9E+01	**3.2E+00**
Max_Best	1.9E+01	2.0E+01	**4.9E+00**
Avg_Best	1.9E+01	2.0E+01	**4.2E+00**
Time_min_Best [s]	1463.8	**276.3**	1066.3
Time_avg_Best [s]	1437.2	**101.2**	1062.9
Ffe_min_Best	19 571 200	**1 248 600**	5 298 994
Ffe_avg_Best	19 287 820	**448 490**	5 297 812

Table 8. Comparison of factors of particular algorithms for the Rosenbrock's function with 1,000 variables

Variable number	1000		
Algorithm	EA	PSO	AEA
Min_Best	3.1E+09	8.6E+06	**3.2E+03**
Max_Best	3.5E+09	1.1E+08	**3.9E+03**
Avg_Best	3.4E+09	2.6E+07	**3.6E+03**
Time_min_Best [s]	**541.7**	1116.6	3034.7
Time_avg_Best [s]	**530.1**	1144.1	3045.0
Ffe_min_Best	4 989 850	**4 505 400**	12 999 280
Ffe_avg_Best	4 863 405	**4 609 910**	12 996 668

Table 9. Comparison of factors of particular algorithms for the De Jong's function with 1,000 variables

Variable number	1000		
Algorithm	EA	PSO	AEA
Min_Best	1.1E+06	4.4E+02	**4.9E-01**
Max_Best	1.2E+06	5.4E+02	**2.8E+00**
Avg_Best	1.1E+06	5.0E+02	**1.0E+00**
Time_min_Best [s]	**1197.7**	2088.7	3648.7
Time_avg_Best [s]	**1187.0**	2104.4	3655.5
Ffe_min_Best	19 753 200	**9 987 200**	21 000 340
Ffe_avg_Best	19 511 000	**9 988 020**	20 998 901

Table 10. Comparison of factors of particular algorithms for the Griewank's function with 1,000 variables

Variable number	1000		
Algorithm	EA	PSO	AEA
Min_Best	9.7E+03	4.8E+00	**1.1E-01**
Max_Best	1.0E+04	6.3E+00	**8.0E-01**
Avg_Best	1.0E+04	5.6E+00	**4.9E-01**
Time_min_Best [s]	**2729.7**	2761.4	2787.5
Time_avg_Best [s]	**2684.7**	2794.9	2766.5
Ffe_min_Best	19 463 600	**9 987 100**	10 991 520
Ffe_avg_Best	19 115 940	**9 990 070**	10 906 660

Table 11. Comparison of factors of particular algorithms for the Rastrigin's function with 1,000 variables

Variable number	1000		
Algorithm	EA	PSO	AEA
Min_Best	9.5E+03	4.1E+03	**1.0E+03**
Max_Best	9.9E+03	5.1E+03	**1.6E+03**
Avg_Best	9.7E+03	4.7E+03	**1.2E+03**
Time_min_Best [s]	**539.3**	2089.2	3136.4
Time_avg_Best [s]	**542.9**	2099.3	3171.8
Ffe_min_Best	**4 886 150**	8 262 500	12 962 560
Ffe_avg_Best	**4 911 670**	8 250 460	12 987 958

Table 12. Comparison of factors of particular algorithms for the Schwefel's (11) function with 1,000 variables

Variable number	1000		
Algorithm	EA	PSO	AEA
Min_Best	3.2E+02	**1.7E+02**	2.0E+02
Max_Best	3.3E+02	**2.2E+02**	2.5E+02
Avg_Best	3.2E+02	**1.8E+02**	2.4E+02
Time_min_Best [s]	**468.0**	2409.0	2509.0
Time_avg_Best [s]	**466.9**	2366.5	2484.5
Ffe_min_Best	**4 955 700**	9 933 300	15 486 995
Ffe_avg_Best	**4 950 570**	9 949 680	15 451 750

4.66 x 10^{-8} after 9,577 [s]. Fig. 11 shows how the outputs varied with time for Rosenbrock's function with 10^6 iterations. Unfortunately, the Ecosystem Algorithm did not come as close to the local optimum as it did in the previous test. Yet, it was exploring the solution space until the end of simulation.

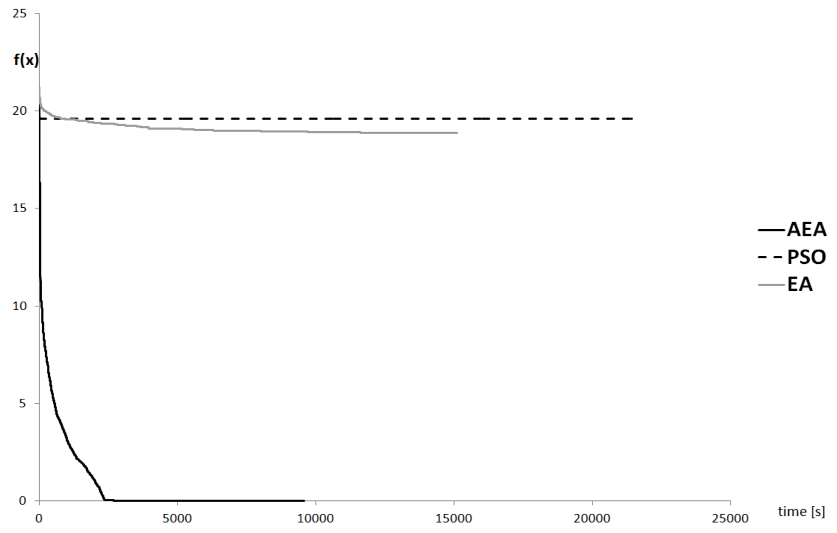


Figure 10. Ackley's function in the simulations with 10^6 iterations

5. Conclusions

The development and functioning of natural ecosystems are by all means fascinating. Virtually any place on the Earth is a sort of ecosystem and is used, in one way or another, by various organisms, with different relationships between them. What is also surprising is the efficiency with which organisms use the ecosystem's resources and a specific self-organisation of the ecosystem, which usually leads to its self stabilisation.

However, ecosystem modelling is a complex problem. The ecosystem model proposed in this paper is – obviously – a simplified one, however, the principal relations between its constitutive parts are maintained. The primary use of this model has been to develop an effective optimization method on its basis. The ecosystem method thus established is one of the most complex Computational Intelligence methods, with a relatively high number of control parameters. The proposed Ecosystem Algorithm is also rather complex in structure, because it is based on three different populations, related to each other through various relationships. Mutual interactions and relationships between populations mimic definite selected processes occurring in a natural ecosystem. Each population is governed by different rules for searching through the fitness function, and is subject to different rules of survival. The tests performed show that the

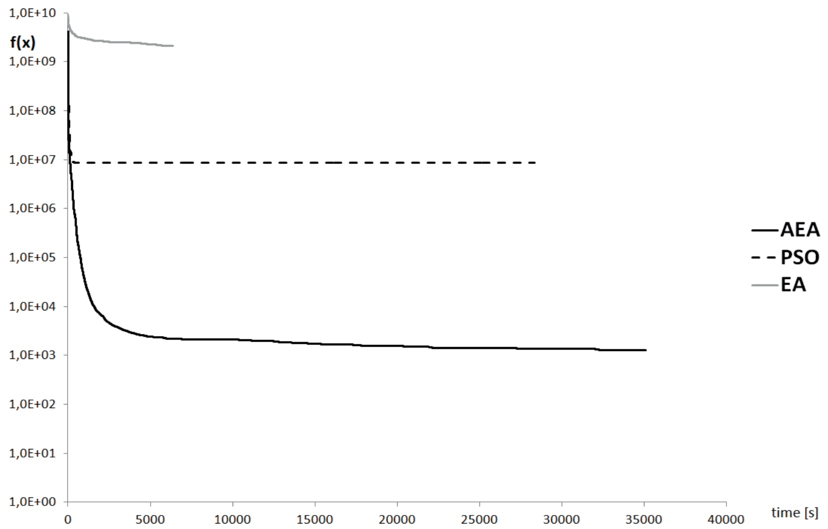


Figure 11. Rosenbrock's function in simulations with 10^6 iterations

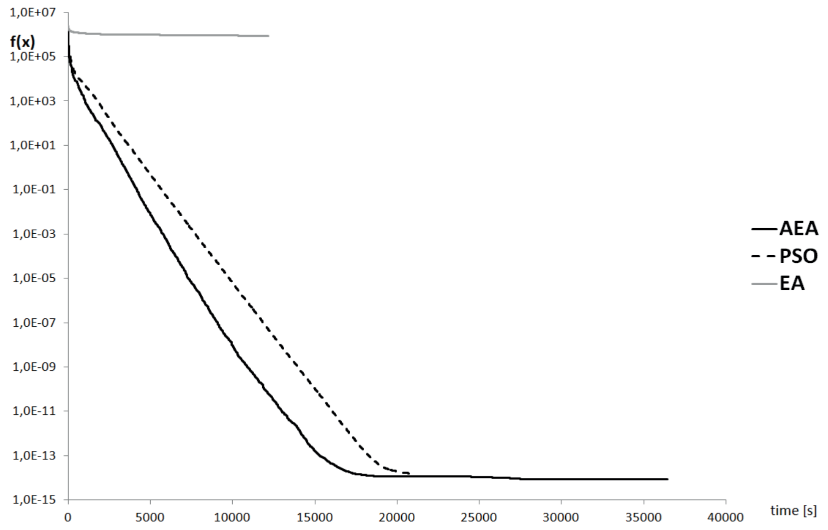


Figure 12. De Jong's function in simulations with 10^6 iterations

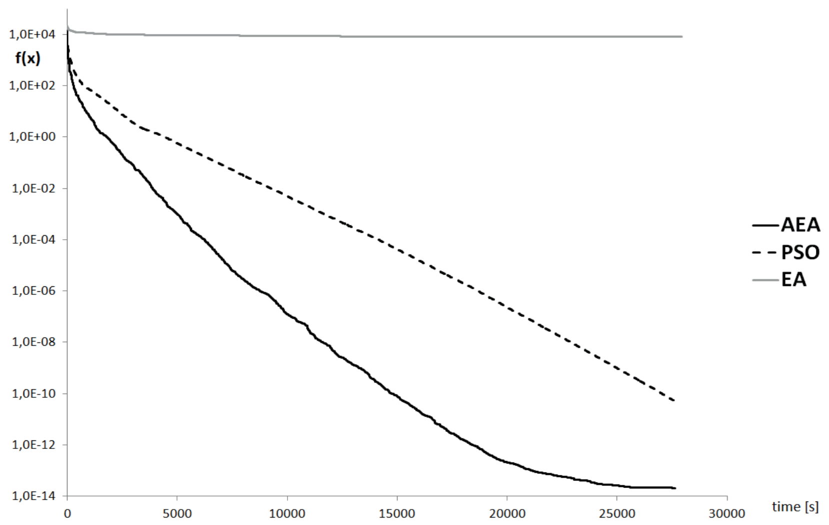


Figure 13. Griewank's function in simulations with 10^6 iterations

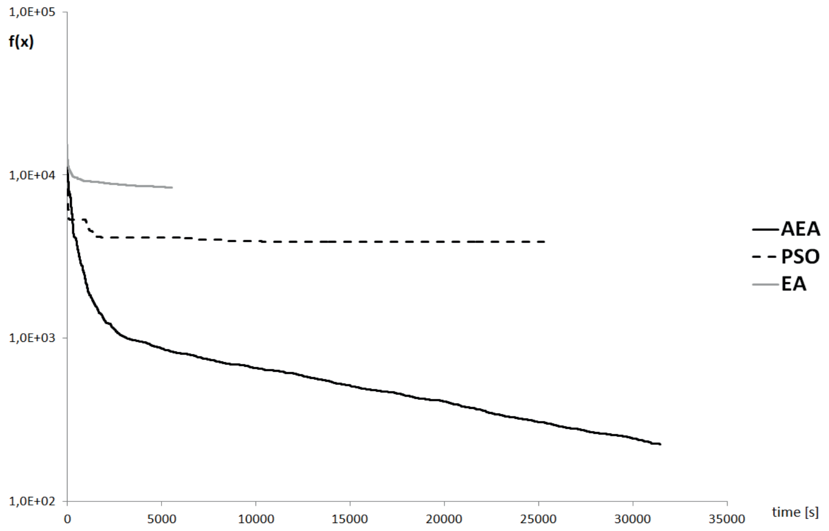


Figure 14. Rastrigin's function in simulations with 10^6 iterations

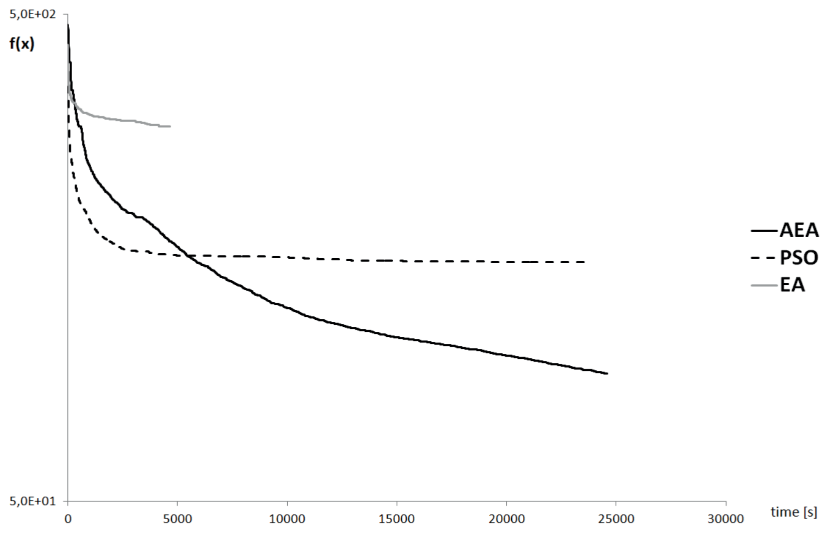


Figure 15. Schwefel's (11) function in simulations with 10^6 iterations

structural complexity provides no advantage to the Ecosystem Algorithm over reference methods for the simplest test problems with 10 variables. However, the advantage is clear for more complex problems with 100 and 1,000 variables. For the problems with 1,000 variables, the proposed algorithm was the only one that came close to the optimum solution in most cases.

The Artificial Ecosystem Algorithm in this form proved also to be useful in solving continuous and combinatorial problems in electrical power engineering (see Baczyński, 2013). However, this form should be treated as an initial proposal for the use of ecosystem relations for optimisation purposes, which shall be developed.

References

ADHAM, M.T. and BENTLEY, P.J. (2014) An Artificial Ecosystem Algorithm applied to static and Dynamic Travelling Salesman Problems. Evolvable Systems (ICES), *2014 IEEE International Conference on*, 149-156, 9-12 Dec. 2014.

BACZYŃSKI D. (2013) Metody inteligencji obliczeniowej w elektroenergetyce (in Polish: *Computational Intelligence Methods in Electrical Power Engineering*), Monograph, Warsaw University of Technology Scientific Works, Elektryka, issue 145, ISSN 0137-2319, WUT Publishing Office, Warsaw.

BINITHA, S. and SIVA SATHYA, S. (2012) A Survey of Bio inspired Optimization Algorithms. *International Journal of Soft Computing and Engineering* (IJSCE) ISSN: 2231-2307, **2**, 2, May 2012.

BONABEAU, E., DORIGO, M., THERAULAZ, G. (1999) *Swarm Intelligence,* *From Natural to Artificial Systems.* Oxford University Press.

BRATTON, D. and KENNEDY, J. (2007) Defining a Standard for Particle Swarm Optimization. *Proceedings of the 2007 IEEE Swarm Intelligence Symposium* (SIS 2007).

BREMERMANN, H. (1974). Chemotaxis and optimization. *Journal of Franklin Institute* 297, 397–404.

BRISCOE, G., SADEDIN, S., DE WILDE, P. (2011) Digital Ecosystems: Ecosystem–Oriented Architectures. *Natural Computing* 10: 1143–1194, Springer.

BRISCOE, G., SADEDIN, S., PAPERIN, G. (2007) Biology of Applied Digital Ecosystems. *Digital EcoSystems and Technologies Conference*, 2007. DEST '07. Inaugural IEEE-IES, 458-463, 21-23 Feb. 2007.

CHEN, H. and YUNLONG, Z. (2008) Optimization Based on Symbiotic Multispecies Coevolution. *Journal on Applied Mathematics and Computation*, 205.

CHENG, M.Y. and PRAYOGO, D. (2014) Symbiotic Organisms Search: a new metaheuristic optimization algorithm. *Comput. Struct.* 139, 98–112.

DE BOER, F. K. and HOGEWEG, P. (2012) Co-evolution and ecosystem based problem solving. *Ecological Informatics* 9, 47–58.

DE CASTRO, L.N. and TIMMIS, J. (2003) Artificial immune systems as a novel soft computing paradigm. *Soft Computing 7,* 526–544, Springer-Verlag

DORIGO, M., BIRATTARI, M. and STUTZLE, T. (2006) Ant colony optimization. *Computational Intelligence Magazine*, IEEE, **1**, 4, 28-39, Nov. 2006

EUSUFF, M.M. and LANSEY, K.E. (2003) Optimization of water distribution network design using the shuffled frog leaping algorithm. *Journal of Water Resources Planning and Management* **129** (3), 210–225.

FENG, X., LAU, F.C.M. and GAO, D. (2009) A new bio-inspired approach to the traveling salesman problem. *Complex Sciences*, *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, February, **5**, 310–1321, Springer, Berlin–Heidelberg.

GEEM, Z.W., KIM, J.-H. and LOGANATHAN, G.V. (2001) A new heuristic optimization algorithm: harmony search. *Simulation* **76**(2), 60–68.

GLOVER, F. (1990) Tabu Search - part II. *ORSA Journal of Computing*, **2**(1).

HAI-FEI, Yu and DING-WEI, Wang (2006) Design and Analysis of Food-Chain Algorithm. *Computational Intelligence and Security, 2006 International Conference on*, **1**, 453-456, Nov. 2006.

HAVENS, T., SPAIN, C., SALMON, N. and KELLER, J. (2008) Roach infestation optimization. *IEEE Swarm Intelligence Symposium*, September, 1–7.

HERTZ, J., KROGH, A. and PALMER, R. (1991) *Introduction to the Theory of Neural Computation.* Addison Wesley, Amsterdam.

KENNEDY, J. and EBERHART, R. (1995) Particle swarm optimization. *Neural Networks*, *Proceedings, IEEE International Conference on*, **4**, 1942-

1948, Nov/Dec 1995.

KIRKPATRICK, S., GELATT, C.D. and VECCHI, M. P. (1983) Optimization by simulated annealing. *Science*, **220** (4598), 671–680.

KRISHNANAND, K.N. and GHOSE, D. (2005) Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. *Swarm Intelligence Symposium, SIS 2005. Proceedings 2005 IEEE*, 84- 91, 8-10 June 2005

LAFUSA, A. (2007) Studying Long-term Evolution with Artificial Life. *Artificial Life, 2007. ALIFE '07. IEEE Symposium on*, 5-22, 1-5 April 2007.

MEHRAB IAN, A.R. and LUCAS, C. (2006) A novel numerical optimization algorithm inspired from weed colonization. *Ecological Informatics* 1, 355–366.

MICHALEWICZ, Z. (1992) *Genetic algorithms + data structures = evolution programs.* Springer-Verlag, Berlin Heidelberg.

MONISMITH, D. and MAYFIELD, B. (2008) Slime mold as a model for numerical optimization. *IEEE Swarm Intelligence Symposium*, 1–8.

MOSCATO, P. (1989) *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms.* Technical report, California Institute of Technology.

PICHLER, P.P. and CANAMERO, L. (2007) An Evolving Ecosystems Approach to Generating Complex Agent Behaviour. *Artificial Life, ALIFE '07. IEEE Symposium on*, 303-310, 1-5 April 2007.

RABANAL, P., RODRÍGUEZ, I. and RUBIO, F. (2007) Using river formation dynamics to design heuristic algorithms. *Unconventional Computation*, **LNCS 4618**, 163–177. Springer.

REYNOLDS, R. G. (1994) An Introduction to Cultural Algorithms. *Proceedings of the Third Annual Conference on Evolutionary Programming*, February 24-26, 1994, San Diego, California, 131-139.

SAGOFF, M. (2003) The plaza and the pendulum: two concepts of ecological science. *Biology and Philosophy* **18**(4), Springer Netherlands.

SHAH HOSSEINI, H. (2007) Problem solving by intelligent water drops. *Evolutionary Computation, CEC 2007. IEEE Congress on*, 3226-3231, 25-28 Sept. 2007.

SIMON, D. (2008) Biogeography-Based Optimization. *Evolutionary Computation, IEEE Transactions on*, **12**, 6, 702-713, Dec. 2008.

TZIMA, F.A., SYMEONIDIS, A.L., MITKAS, P.A. (2007) Symbiosis: Using Predator-Prey Games as a Test Bed for Studying Competitive Coevolution. *Integration of Knowledge Intensive Multi-Agent Systems, KIMAS 2007. International Conference on*, 115-120, April 30 2007-May 3 2007.

ULANOWICZ, R. (2000) *Growth and Development, Ecosystems Phenomenology.* toExcel/iUniverse 2000

VULLI, S.S. and AGARWAL, S. (2008) Individual-Based Artificial Ecosystems for Design and Optimization. *GECCO'08*, July 12–16, 2008, Atlanta, Georgia, USA.

WEINER, J. (2012) *Życie i ewolucja biosfery* (in Polish: *Life and the Evolution of the Biosphere*). Wydawnictwo Naukowe PWN, Warszawa.

YANG, X.S. (2010a) *Nature-Inspired Metaheuristic Algorithms.* Luniver Press.

YANG, X.S. (2010b) Test problems in optimization. *Engineering Optimization: An Introduction with Metaheuristic Applications* (Xin-She Yang, ed.), John Wiley & Sons.

YANG, X.S. and DEB, S. (2009) Cuckoo Search via Lévy flights. *Nature & Biologically Inspired Computing, NaBIC 2009. World Congress on*, 210-214, 9-11 Dec. 2009.

ZADEH, L.A. (1965) Fuzzy sets. *Information and Control*, **8**, 338-353.