The Role And Impact of The Baldwin Effect on Genetic Algorithms

Karolina Kęsik
Institute of Mathematics
Silesian University of Technology
Kaszubska 23, 44-100 Gliwice, Poland
Email: karola.ksk@gmail.com

***Abstract*—The increasing influence of optimization methods on modern technologies means that not only new algorithms are designed but old ones are improved. One of the classic optimization methods is the genetic algorithm inspired by the phenomenon of biological evolution. One of the modifications is Baldwin's approach as genetic assimilation. It is a process that creates permanent structures as an effect on the frequent use of specific organs. This paper presents four different algorithms that have been used as a Baldwin effect in the classical genetic algorithm. All techniques have been tested using 9 test functions and the obtained results are discussed.**

I. Introduction

Describing things and process in nature very often comes to the mathematical description using equation or function. Especially when there are some parameters that make it impossible to determine what proportions should be maintained depending on other external factors. Moreover, even mathematical tools are not able to simply and quickly determine what is the solution for a more complex function or their dependencies.

One of the best existing methods of finding a global extreme for multi-variable functions are algorithms inspired by phenomena occurring in nature or behavior of a specific group of animals. It is visible on the developed research around the world and emerging publications. One of the most dynamically developing branches of optimization are heuristic algorithms, ie those that do not guarantee the correct solution in a finite time. An example of such a technique is the dragonfly algorithm [8] which is inspired by the static and dynamic behavior of dragonfly's swarm. Again in [12], the authors described the mathematical model of behavior of polar bears which are forced to move on ice floe and hunt for seals in arctic conditions.

Not only the algorithms are developed but the applications that have complicated functions. One of such example is analysis of samples taken from a fluorescence microscope. In [19], optimization algorithms were used in the detection of bacterial shapes at various stages of life. Again in [17], this idea was used for charge and discharge electronic vehicles using building energy management system. Another application area is biometric security. In [10], voice samples were interpreted as two-dimensional images where important features to identify the owner of the voice were sought. Other application is chat-bots were heuristic can be used as an engine [15]. For these type of apps, different notation are developed [3]. In [13], heuristic were used to extract brain tumor from MR images. Similarly in the area of smart technology optimization has a significant role. In [2], it was used in wireless sensor network, and in [16], modified technique has found application in optimization of coverage in visible light communication in smart homes systems.

In this paper, the idea of improving genetic algorithm by the introduction of Baldwin effect is presented.

II. Optimization problem formulation

It is quite often that some problems can be described by some function. And then, the smallest or largest value for a given function is wanted. Such a task is called an optimization problem. Let's assume that $\mathbf{x}$ is a point in n–dimension, and function will be described in the following way $f(x) : \mathbb{R}^n \to \mathbb{R}$, then minimization problem can be defined as

$$\begin{aligned} &\text{Minimize} && f(\overline{x}) \\ &\text{subject to} && g(\overline{x}) \geq 0 \\ & && L_i \leq x_i \leq R_i \qquad i = 0, 1, \ldots, n-1, \end{aligned} \tag{1}$$

where $g(\cdot)$ is inequality constraint and each spatial components of point $\mathbf{x}$ is in the range $\langle L_i, R_i \rangle$.

III. Genetic approach to optimization

Through many ages, people observe the changes in natural processes. Then, many times, they tried to use this knowledge in science by creating computer methods based on those notices. In 70s of 20th century, the man called John Holland created the Genetic Algorithm, which was also the first evolutionary algorithm [4]. The inspiration of creating this technique was desire of making computer be able to solve problems likewise it goes in the natural evolution process. The initial idea of Genetic Algorithm was to create a population which contain some individuals and each of them had a unique binary genetic code assigned to it. Set codes correspond to chromosomes exist in organisms and that makes the elements of code equivalent to the genes. Subsequently, after many researches, there were implemented some improvements occurred during the noticed biological process of evolution. The observed operations start from the mechanism, which transforms the binary codes similarly as crossing–over, according to the

Table I: Selected test function for described optimization problem.

Name	Function	Input domain	$\mathbf{x}$	Global minimum
Ackley	$f_1(\mathbf{x}) = -20\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$	$\langle -32.8, 32.8\rangle$	$(0,\ldots,0)$	0
Griewank	$f_2(\mathbf{x}) = \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$\langle -600, 600\rangle$	$(0,\ldots,0)$	0
Rastrigin	$f_3(\mathbf{x}) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	$\langle -5.12, 5.12\rangle$	$(0,\ldots,0)$	0
Schwefel	$f_4(\mathbf{x}) = 418.9829n - \sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	$\langle -500, 500\rangle$	$(0,\ldots,0)$	0
Rotated hyper-ellipsoid	$f_5(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{i} x_j^2$	$\langle -65.5, 65.5\rangle$	$(0,\ldots,0)$	0
Sum squares	$f_6(\mathbf{x}) = \sum_{i=1}^{n} i x_i^2$	$\langle -5.12, 5.12\rangle$	$(0,\ldots,0)$	0
Zakharov	$f_7(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i\right)^4$	$\langle -5, 10\rangle$	$(0,\ldots,0)$	0
Rosenbrock	$f_8(\mathbf{x}) = \sum_{i=1}^{n-1} 100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2$	$\langle -5, 10\rangle$	$(0,\ldots,0)$	0
Styblinski-Tang	$f_9(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i$	$\langle -5, 5\rangle$	$(-2.903534,\ldots,-2.903534)$	$-39.16599n$

process of inheritance. Then they based on mutation and selection. Holland discover that by adding the logic operators and using selection, the average value of population fitness can increase. The most important part of that is using encoding and genetic operators to solve problems. Nowadays, in the age of new technology, it is obvious to use for that the potential of advanced programming with high–performance computers. In this paper, the version of the genetic algorithm operating on numbers in the decimal notation was selected.

1) Genetic Algorithm (GA): The Genetic Algoritms is made of few steps. The first one is initialization in which the initial population is created. That step is about generating the size of population, the encoding of chromosomes and the formula for fitness function $f(\cdot)$ with the appropriate restrictions. The main assumption here is about setting the size of population, because it is the parameter that affects on accuracy of the algorithm. If the population is too small then the algorithm finishes calculation without reaching the correct solution. On the other hand, when the number of individuals is too large, then it can aggravate the computer and make the calculations last very long. Other elements of initial population are set randomly.

Afterwards, there is made a selection of individuals. Some of them are passed to the next stage of calculations. That process is called reproduction. It can be done in many ways. Description one of them requires the use of probability that one individual $\mathbf{x_i}$ is chosen to new generation. The probability of setting this individual in next population depends on the value of fitness function at exact point $f(\mathbf{x_i})$. The higher value of fitness function results in a higher probability at this point. The clue of this step is to create a random value p_r for further description of new population selected from the existing generation P^t. It can be described as

$$\begin{cases} p_r(\mathbf{x_i^t}) = \frac{f(\mathbf{x_i^t})}{\mathbf{x_i^t}} \\ \mathbf{x_i^t} \in P^t \end{cases}. \tag{2}$$

After few sampling of this variable, it is possible to introduce the distribution function of reproduction probability as

$$P_r(\mathbf{x_i^t}) = \sum_{j=1}^{i} p_r(\mathbf{x_i^t}). \tag{3}$$

In the next step there is a selection of individuals – if it is reproduced or not. The factor that determines this choice is a random variable α from $\langle 0, 1\rangle$. The individual $\mathbf{x_i}$ is reproduced as long as the following formula

$$P_r(\mathbf{x_{i-1}^t}) < \alpha \leq P_r(\mathbf{x_i^t}) \tag{4}$$

is fulfilled. Genetic operators, like mutation and crossover, are responsible for creating a diverse population but they also take control of reproducing.

The first genetic operator mentioned above, the mutation, is about replacing the chromosomes. In this process a random variable ξ belongs to $\langle 0, 1\rangle$ is added to the chromosome

$$\mathbf{x_i^{t+1}}_{mutated} = \mathbf{x_i^t} + \xi. \tag{5}$$

Then let the λ be the vector of random values and length equivalent to the number of individuals in generation. If for p_m, which is the probability of mutation, is followed the ineqality

$$\lambda_i < p_m, \tag{6}$$

then the chromosome is mutated.

The second of genetic operators is crossover. This process is about exchanging genetic material. Two chromosomes are

called the parents and that exchange takes place between them. As a result of crossover, two new individuals are generated and attached to the population. Similarly to Eq. (5), taking a random real ξ from $\langle 0, 1\rangle$ allows to define the formula of creating new individual as an average of parental chromosomes $\mathbf{x_i}$ and $\mathbf{x_{i+1}}$

$$\mathbf{x_i}_{des1}^{t+1} = \mathbf{x_i}^t + \xi(\mathbf{x_{i+1}^t} - \mathbf{x_i^t})\mathbf{x_i}_{des2}. \tag{7}$$

Then whole new population has to be evaluated whether they fit to the conditions in environment. This process, called succession, is replacing the old generation with the new one.

Replacing all individuals in population is only one of the ways to carry out succession. The other way is to change just a part of old generation. Here, there becomes a need to choose a method of selection chromosomes. Anyway, a parameter g from $\langle 0, 100\rangle$ can describe an amount of individuals from old generation that remained. Decision, which of individuals should be replaced, can be taken on many different levels:

- the ones that are similar to individuals in next population,
- the ones that are the least adapted,
- the ones that are selected by succession
- the ones that are chosen in random way.

The method which let the best chromosomes survive is succession. In that way, it doesn't matter in which population some individuals appear – if they are ones of the best, then they stay to the end of algorithm. In every iteration individuals are ordered by value of fitness function $f(\mathbf{x_i})$. Then, some of the worst are replaced with new ones. By using some kind of "back-up" generation, the best individuals are stored through all the algorithm. It ensures that better ones will not be replaced by worse ones.

Algorithm 1 Genetic algorithm

```
Start,
Define all parameters - T, f(·), n,
Generate the initial population,
t := 1;
while t ≤ T do
    Select the best individuals for reproduction,
    Apply crossover operation using Eq. (7),
    Apply mutation operation using Eq. (5),
    Replace the worst individuals with new one,
    t++
end while
Return x,
Stop.
```

IV. Baldwin effect

After observing the evolution process, James Mark Baldwin claimed that there exists an ability to acquire new behaviours during all life that depends on environment. Furthermore, the offspring derive those skills from their ancestors [1]. This mechanism is called the Baldwin Effect or also organic selection. The driving force behind this phenomenon is survival instinct. It is very simple to give examples of this observation. If there is a population in the environment, then those individuals, which will learn the most needed peculiar abilities, those will survive. And they will have bigger offspring. What is more, their offspring inherit genes with exceptional properties. After more researches on the Baldwin's theory, it was noticed that not skills but only the ability to acquire some skills is inherited. This can explain the language predispositions in people.

In practice, Baldwin effect is reflected in the value of fitness function because it is the only significant skill to survive. It is visible in additional local search.

A. Gradient descend

Analysis of the direction of gradient decrease is one of the classic optimization methods [14]. Assume that $\mathbf{x} = (x_0, \ldots, x_{n-1})$ is a start point. To find the direction, we need to calculate the negative gradient for each spatial coordinates according to

$$-\nabla f_{x_i} = -\frac{\partial f(x_0, \ldots, x_{n-1})}{\partial x_i}. \tag{8}$$

Using above calculation of $-\nabla$, the new coordinate is calculated as

$$x_i^t = x_i^t + \lambda(-\nabla f_{x_i^t}), \tag{9}$$

where λ is a step and t means current iteration. After finding the new value of point, method compare it with the previous one using fitness function $f(\cdot)$. If new value has better adaptation, it replace the old one.

Algorithm 2 Gradient descent

```
Start,
Define fitness condition f(·), the step λ, t,
Take a staring point x,
t := 1,
while t ≤ T do
    Calculate x' by Eq. (9),
    if f(x') ≤ f(x) then
        x = x',
    end if
    t++,
end while
Return x,
Stop.
```

B. Iterated local search

Iterated local search is one of the classic optimization technique called *hill climbing* [7]. The name is adequate to the operation of the algorithm, which for a given point $\mathbf{x}$ performs perturbation (i.e. random displacement) as

$$\mathbf{x}' = \mathbf{x} - \xi, \tag{10}$$

where $\xi \in \langle 0, 1\rangle$. It allows to escape from the local minimum. Then local search in the neighborhood is done. For this purpose, another method is used. If the obtained point is better in terms of the fitness function, it overwrites the initial one.

Algorithm 3 Iterated local search

Start,
Define $f(\cdot)$, T,
Take a starting point $\mathbf{x}$,
Local search at $\mathbf{x}$,
$t := 1$,
while $t \leq T$ **do**
  Create $\mathbf{x}'$ using Eq. (10),
  Apply Gradient Descent staring in $\mathbf{x}'$ (described in Alg. 2) to find $\mathbf{x}''$,
  if $f(\mathbf{x}'') \leq f(\mathbf{x})$ **then**
    $\mathbf{x} = \mathbf{x}''$,
  end if
  $t++$,
end while
Return $\mathbf{x}$,
Stop.

C. Variable neighborhood search

Variable neighborhood search is another *hill climbing* method [9]. The algorithm assumes the creation of a set, then the use of another local search technique for each representative in this set and comparison with a given point. If the new point proves to be better (using fitness function), the original one is overwritten by the neighbor.

Algorithm 4 Variable neighborhood search

Start,
Define a set N_k, k_{max},
Take a starting point $\mathbf{x}$,
$k := 1$,
$t := 1$,
while $t \leq T$ **do**
  while $k \leq k_{max}$ **do**
    Generate $\mathbf{x}' \in N_k$ in random way,
    Apply Gradient Descent staring in $\mathbf{x}'$ (described in Alg. 2) to find $\mathbf{x}''$,
    if $f(\mathbf{x}'') \leq f(\mathbf{x})$ **then**
      $\mathbf{x} = \mathbf{x}''$,
      $k := 1$,
    else
      $k++$,
    end if
  end while
  $t++$,
end while
Return $\mathbf{x}$,
Stop.

D. Simulated annealing

Simulated annealing is a probabilistic technique for optimization purpose. It was described for the first time in [6] by Kirkpatrick and others. The technique is a model of the annealing process in thermodynamics, and more precisely how metals cool and anneal. It assumes that the temperature is high at beginning of the process what gives high probability of change. As the temperature decreases, the probability of change is smaller. In practice, this means that the probability of adopting a worse solution is higher at high temperature and smaller at lower.

The algorithm uses a modified thermodynamic equation what can be defined as

$$P(E) \approx e^{-\frac{\delta}{T}}, \tag{11}$$

where δ is the difference in the quality of a given point $\mathbf{x}$ and new, random one $\mathbf{x}'$ in relation to the fitness condition

$$\delta = f(\mathbf{x}') - f(\mathbf{x}). \tag{12}$$

A new solution is adopted when the following condition is satisfied

$$\gamma < e^{-\frac{\delta}{T}}, \tag{13}$$

where $\gamma \in \langle 0, 1 \rangle$. Another important aspect is temperature reducing calculated as

$$T_{k+1} = r \cdot T_k, \tag{14}$$

where T means the temperature value and k is the k-th iteration and r is constant value.

Algorithm 5 Simulated annealing

Start,
Define the start temperature T, fitness condition f, number of iteration t and parameter r,
Take a staring point $\mathbf{x}$,
k:=0,
while $k \leq t$ **do**
  $i := 0$,
  for i **to** k **do**
    Create random solution $\mathbf{x}'$,
    Calculate the difference δ according to Eq. (12),
    if $\delta < 0$ **then**
      $\mathbf{x} = \mathbf{x}'$
    else
      Generate γ,
      if equation (13) is satisfied **then**
        $\mathbf{x} = \mathbf{x}'$,
      end if
    end if
  end for
  Reduce the temperature by Eq. (14),
  $k++$,
end while
Return $\mathbf{x}$,
Stop.

Table II: Obtained average results.

	GA	GA with Baldwin effect			
		Gradient descent	Iterated local search	Variable neighborhood search	Simulated annealing
f_1	$2.19932 \cdot 10^{-6}$	$2.13421 \cdot 10^{-6}$	$2.21328 \cdot 10^{-6}$	$2.120981 \cdot 10^{-6}$	$1.912247 \cdot 10^{-6}$
f_2	$1.18919 \cdot 10^{-4}$	$1.23193 \cdot 10^{-4}$	$1.01239 \cdot 10^{-4}$	$1.129815 \cdot 10^{-3}$	$1.01923 \cdot 10^{-4}$
f_3	$-0.28311 \cdot 10^{-3}$	$-0.29172 \cdot 10^{-4}$	$0.28872 \cdot 10^{-4}$	$0.47623cdot10^{-4}$	$-0.18926 \cdot 10^{-4}$
f_4	$0.19172 \cdot 10^{-3}$	$0.22898 \cdot 10^{-3}$	$0.19863 \cdot 10^{-3}$	$0.199859 \cdot 10^{-3}$	$-0.12172 \cdot 10^{-3}$
f_5	$-2.12812 \cdot 10^{-12}$	$1.907319 \cdot 10^{-12}$	$2.98221 \cdot 10^{-11}$	$-2.00193 \cdot 10^{-12}$	$-2.00012 \cdot 10^{-12}$
f_6	$3.95276 \cdot 10^{-14}$	$-3.07329 \cdot 10^{-13}$	$3.55311 \cdot 10^{-14}$	$3.624782 \cdot 10^{-14}$	$-4.21424 \cdot 10^{-15}$
f_7	$2.19272 \cdot 10^{-4}$	$1.78139 \cdot 10^{-5}$	$2.00712 \cdot 10^{-5}$	$1.759836 \cdot 10^{-5}$	$1.37127 \cdot 10^{-5}$
f_8	$-1.78342 \cdot 10^{-3}$	$-1.81293 \cdot 10^{-3}$	$2.19813 \cdot 10^{-3}$	$3.1128 \cdot 10^{-3}$	$1.47516 \cdot 10^{-3}$
f_9	-195.7821	-195.7802	-195.7652	-195.7541	-195.8148

Table III: Obtained average errors.

	GA	GA with Baldwin effect			
		Gradient descent	Iterated local search	Variable neighborhood search	Simulated annealing
f_1	$-2.19932 \cdot 10^{-06}$	$-2.13421 \cdot 10^{-06}$	$-2.21328 \cdot 10^{-06}$	$-2.12098 \cdot 10^{-06}$	$-1.91247 \cdot 10^{-06}$
f_2	-0.000118919	-0.000123193	-0.000101239	-0.001129815	-0.000101923
f_3	0.00028311	0.000029172	-0.000028872	0.000047623	0.000018926
f_4	-0.00019172	-0.00022898	-0.00019863	-0.000199859	0.00012172
f_5	$2.12812 \cdot 10^{-12}$	$-1.90732 \cdot 10^{-12}$	$-2.98221 \cdot 10^{-11}$	$2.00193 \cdot 10^{-12}$	$2.00012 \cdot 10^{-12}$
f_6	$-3.95276 \cdot 10^{-14}$	$3.07329 \cdot 10^{-13}$	$-3.55311 \cdot 10^{-14}$	$-3.62478 \cdot 10^{-14}$	$4.21424 \cdot 10^{-15}$
f_7	-0.000219272	$-1.78139 \cdot 10^{-05}$	$-2.00712 \cdot 10^{-05}$	$-1.75984 \cdot 10^{-05}$	$-1.37127 \cdot 10^{-05}$
f_8	0.00178342	0.00181293	-0.00219813	-0.0031128	-0.00147516
f_9	-0.04785	-0.04975	-0.06475	-0.07585	-0.01515

V. Experiments

All techniques were implemented and tested (Baldwin effect for 20 steps, GA – 10000 iterations and 100 individuals) using all nine test functions. Each algorithm was made 100 times to get 100 solutions. The obtained points were used to calculate the average value of the fitness function as follows

$$\frac{1}{100}\sum_{i=1}^{100} f(\overline{x_i}), \tag{15}$$

and then the error for each function was calculated as

$$\left| f\left(\overline{x}_{ideal}\right) - \frac{1}{100}\sum_{i=1}^{100} f\left(\overline{x}_i\right) \right|. \tag{16}$$

All obtained results are presented in Tab. II and III. In almost every modification interpreted as Baldwin effect, the results were corrected what increased accuracy. Unfortunately, the improvement took place at the expense of increasing the time by adding additional calculations made at the local search. The best solution turned out to be the solution gained from genetic algorithm with simulated annealing in all cases.

Iterated and variable neighborhood returned improved results of the original algorithm, but in comparison to other used methods, they are the least favorable due to the number of calculations. The reason is a structure that uses an additional algorithm which doubles the number of operations.

VI. Conclusions

Baldwin effect can be interpreted as correction for obtained results in each iteration. The obtained results showed the superiority of simulated annealing over other tested methods. In the case of iterated or neighborhood technique, the results obtained were better than the original version, but the execution time has increased quite significantly. It is worth noting that the Baldwin effect was performed only for 20 iteration what is not a large number.

Any modification that aims to increase the precision of the algorithm for the selected point in a given place (only local space-searching was considered in this paper) is worth implementing in the case of creating hybrid methods and when the accuracy is significant.

References

[1] J. M. Baldwin. A new factor in evolution. *The american naturalist*, 30(354):441–451, 1896.

[2] S. Bouarafa, R. Saadane, and M. D. Rahmani. Inspired from ants colony: Smart routing algorithm of wireless sensor network. *Information*, 9(1):23, 2018.

[3] S. Chodarev and J. Porubän. Development of custom notation for XML-based language: A model-driven approach. *Computer Science and Information Systems (ComSIS)*, 14(3), 2017.

[4] J. H. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM Journal on Computing*, 2(2):88–105, 1973.

[5] T. Kapuściński, R. K. Nowicki, and C. Napoli. Comparison of effectiveness of multi-objective genetic algorithms in optimization of invertible s-boxes. In *International Conference on Artificial Intelligence and Soft Computing*, pages 466–476. Springer, 2017.

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

[7] H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search. In *Handbook of metaheuristics*, pages 320–353. Springer, 2003.

[8] S. Mirjalili. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications*, 27(4):1053–1073, 2016.

[9] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

[10] D. Połap. Neuro-heuristic voice recognition. In *Computer Science and Information Systems (FedCSIS), 2016 Federated Conference on*, pages 487–490. IEEE, 2016.

[11] D. Połap and M. Woźniak. Detection of important features from images using heuristic approach. In *International Conference on Information and Software Technologies*, pages 432–441. Springer, 2017.

[12] D. Połap and M. Wozniak. Polar bear optimization algorithm: Meta-heuristic with fast population movement and dynamic birth and death mechanism. *Symmetry*, 9(10):203, 2017.

[13] V. Rajinikanth, S. C. Satapathy, S. L. Fernandes, and S. Nachiappan. Entropy based segmentation of tumor from brain mr images–a study with teaching learning based optimization. *Pattern Recognition Letters*, 94:87–95, 2017.

[14] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[15] A. Shaikh, G. Phalke, P. Patil, S. Bhosale, and J. Raghatwan. A survey on chatbot conversational systems. *International Journal of Engineering Science*, 3117, 2016.

[16] G. Sun, Y. Liu, M. Yang, A. Wang, S. Liang, and Y. Zhang. Coverage optimization of vlc in smart homes based on improved cuckoo search algorithm. *Computer Networks*, 116:63–78, 2017.

[17] S. Umetani, Y. Fukushima, and H. Morita. A linear programming based heuristic algorithm for charge and discharge scheduling of electric vehicles in a building energy management system. *Omega*, 67:115–122, 2017.

[18] A. Venckauskas, A. Karpavicius, R. Damaševičius, R. Marcinkevičius, J. Kapočiūte-Dzikiené, and C. Napoli. Open class authorship attribution of lithuanian internet comments using one-class classifier. In *Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 373–382. IEEE, 2017.

[19] M. Woźniak, D. Połap, L. Kośmider, and T. Cłapa. Automated fluorescence microscopy image analysis of pseudomonas aeruginosa bacteria in alive and dead stadium. *Engineering Applications of Artificial Intelligence*, 67:100–110, 2018.

[20] M. Woźniak, D. Połap, C. Napoli, and E. Tramontana. Application of bio-inspired methods in distributed gaming systems. *Information Technology And Control*, 46(1):150–164.

[21] M. Wózniak, D. Połap, R. K. Nowicki, C. Napoli, G. Pappalardo, and E. Tramontana. Novel approach toward medical signals classifier. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–7. IEEE, 2015.

[22] M. Wróbel, J. T. Starczewski, and C. Napoli. Handwriting recognition with extraction of letter fragments. In *International Conference on Artificial Intelligence and Soft Computing*, pages 183–192. Springer, 2017.