Is Swarm Intelligence Able to Create Mazes?

Dawid Połap, Marcin Woźniak, Christian Napoli and Emiliano Tramontana

Abstract—**In this paper, the idea of applying Computational Intelligence in the process of creation board games, in particular mazes, is presented. For two different algorithms the proposed idea has been examined. The results of the experiments are shown and discussed to present advantages and disadvantages.**

Keywords—**computational intelligence, heuristic algorithm**

I. Introduction

THE first evolutionary algorithms have been shown in the 70s [1], [2]. Where in [1] J. Holland has shown that the nature is giving best optimization techniques, which are implementable in various optimization problems. Further with research on possible applications computer scientists have developed several techniques that map behavior of various animals into computer algorithms. In [3] cuckoos breeding habits were implemented to search for optimum solutions, [4] and [5] present nature based algorithms applied to solve metal solidification modeling, [6] shows how to modify harmony search method to more efficiently optimize differential models of solidification models, [7], [8]; [9], [10] and [11] present evolutionary approach to model and optimize cloud based system for efficient user verification and network traffic positioning. In [12] text data clustering was optimized by application of ant colony, in [13] significant operating points were solved, while in [14] presents dedicated particle swarm modeling for dynamic routing problems. Computational intelligence based on swarm algorithms is also efficient in various image processing problems, like key-point search [15], [16], [17]. Other important application of swarm intelligence leads to implementations with neural networks or other intelligent systems [18], [19], [20], [21] and [22]. Until this day these algorithms found numerous applications as an alternative to existing solutions in almost every field of science.

One of them, where a swarm intelligence has been used, are various games. In [23] evolutionary approach was implemented to automatically solve playing strategies, while in [24] similar approach is presented to efficiently help in optimization of evaluation function for various games. Regardless of the type of game, the environment must be strongly varied in order to enhance playability. Second important feature is proper security that enables players to develop gaming playability [25]. The greater playability, the quality of the game is bigger.

D. Połap and M. Woźniak are with Institute of Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice, Poland, (e-mail: Dawid.Polap@gmail.com, Marcin.Wozniak@polsl.pl)

C. Napoli and E. Tramontana are with Department of Mathematics and Informatics, University of Catania, Viale A. Doria 6, 95125 Catania, Italy, (e-mail: napoli@dmi.unict.it, tramontana@dmi.unict.it)

In this paper we would like present a novel approach to board games automatic development based on dedicated swarm intelligence implementation. In the case of 2D games, board is generated depending to the level of complexity. The higher the level, the board should be more difficult to pass. In most cases, each board for 2D game may be presented as a maze which is a system of many roads, where the majority does not lead anywhere. Mazes are designed in accordance with certain principles which are specified for each game separately. In general, a maze has one entrance and one exit and the player's task is to find the road leading through a maze without crossing the walls. A similar case is for 3D games where the walls have been replaced by models of nature or some dedicated architecture models. The main problem of creating this type of board games that are focused on the playability is to reduce complexity to create an shape which may be an appropriate maze. Many maze generators are based on the graph theory which mainly uses tree search algorithms such as Prim's or Kruskal's algorithms, which are efficient for energy saving routing design [26] and gaming systems [27]. In this paper, we would like to propose an alternative methods to create mazes. We base our approach on Swarm Intelligence (SI) with dedicated strategies for boards construction.

II. Swarm Intelligence Algorithms with Developed Procedures to Compose Mazes

Swarm Intelligence is an algorithmic description of the coordinated moves that all swarm particles do together. This movement is based on communication between them, when information about surrounding conditions is passed to optimize strategy of movement. Mathematical model of this type of optimization is developed on observation of animals. Various species perform several optimization strategies to breed, feed, spread and escape. In this paper we want to concentrate on two algorithms simulating colony of ants and bees.

In nature ants search for sources of food. Information about their locations is left for other ants in special traces of pheromones. Model of this behavior is very efficient in many optimization problems, like heat transfer modeling [28]. Similarly, it is possible to model a colony of bees. Among bees information is passed in a kind of dance that bees performs in the hive if a source of nectar was found. This algorithm can be applied i.e. for key point search in 2D pictures [15]. These two algorithms can be implemented to create mazes. In the following sections we present dedicated versions developed for the purpose of maze construction.

A. Artificial Ant Colony Algorithm

Artificial Ant Colony Algorithm (AACA) was inspired by the life of ants and more specifically the search for food. Ants

are able to find a way back to the nest, mark it for other ants and also carry several times more weight of food than theirs to home. This particular modeling of nature inspired optimization AACA has been applied in various tasks, like solidification modeling [5], image processing [29] and data clustering [12]. In AACA, the movement of ants is done in random directions leaving behind a trail of pheromone which allows them to return to the nest or reconstruct a path to the food. In each case going in search of food and after finding a source of food at the way to home ant is able to move both directions because of the left pheromone track. This pheromone marking process is performed by all ants in the home. Every ant that leaves the home can follow tracks of the others. In the situation, when the number of paths is more than one, ants will choose the road in which trail pheromones is the strongest after a temporary evaporation. This guarantees that before many other ants were traveling this path, so at the and of it a large source of food can be found. Model of this marking is performed in the following iterations in the algorithm. Each iteration means that all the ants left home in the search of food and moved over the space leaving pheromone trials. In the next iteration these trials are evaluated by others and if leading to better source of food improved with new portion of pheromones or if leading to weak source abandoned.

During the first iteration, the pheromone value is everywhere the same. In subsequent iterations, the value is updated by

$$f^{t+1}(\mathbf{x_i}, \mathbf{x_j}) = (1 - \rho)f^t(\mathbf{x_i}, \mathbf{x_j}) + \Gamma_i^t, \qquad (1)$$

where ρ is evaporation rate, t is the current iteration and Γ is th distance between the $\mathbf{x_i}$ and all individuals n in the population. The distance is calculated by

$$\Gamma_i^t = \sum_{i=1}^{n} \frac{1}{L_{ij}^t}, \qquad (2)$$

where L_{ij}^t is the length of the path between ants i and j, which is defined as Cartesian metric

$$L_{ij} = \|\mathbf{x_i} - \mathbf{x_j}\| = \sqrt{\sum_{k=1}^{2}(x_{i,k} - x_{j,k})^2}, \qquad (3)$$

where $\mathbf{x_i}$ and $\mathbf{x_j}$ are points in $R \times R$ space, $x_{i,k}, x_{k,j}$-k-th components of the spatial coordinates $\mathbf{x_i}$ and $\mathbf{x_j}$ representing insect.

The probability of choosing the road to the ant $\mathbf{x_j}$ by $\mathbf{x_i}$ is calculated by

$$p^t(\mathbf{x_i}, \mathbf{x_j}) = \frac{[f^t(\mathbf{x_i}, \mathbf{x_j})]^\alpha \left[\frac{1}{L_{ij}^t}\right]^\beta}{\sum_{\alpha \in N_i^k}\left([f^t(\mathbf{x_i}, \mathbf{x_\alpha})]^\alpha \left[\frac{1}{L_{i\alpha}^t}\right]\right)}, \qquad (4)$$

where N_i^k is a set of unknown roads for k ant which lead to the i, α means the impact of left pheromones.

The movement of ants is based on the highest probability p^t and it is defined as

$$\mathbf{x_i^{t+1}} = \mathbf{x_i^t} + \text{sign}(\mathbf{x_i^t}(\text{ind}(t)) - \mathbf{x_i^t}), \qquad (5)$$

where ind(t) means a set of neighbor indicts after sort. Implementation of the algorithm is shown in Algorithm 1.

Algorithm 1 AACA to Create Mazes

1: Start,
2: Define all coefficients: n size of workers population, α impact of left pheromones, ρ evaporation rate, ζ the minimum value of the pheromone, r number of random alleys,
3: Create array with α values and two different exits,
4: **while** the queen does not pass the entire maze **do**
5:      Update pheromone values using (1),
6:      Calculate distances between worker ants (3),
7:      Calculate possible path to follow by worker i to location j $p^t(\mathbf{x_i}, \mathbf{x_j})$ using (4),
8:      Determine the best position to follow,
9:      Move population of workers using (5),
10: **end while**
11: Recalculate the value of pheromone according to (11),
12: Return array of recalculated values of pheromone.
13: Stop.

B. Artificial Bee Colony Algorithm

Artificial Bee Colony Algorithm (ABCA) reflects the behavior of honey bees during the search for food. When bees are looking for nectar sources, they can communicate with each other. One of the most common communication techniques is the waggle dance, during which bees inform each other about the quality of the found source, distance from the hive and direction. Waggle dance allows to find the location where the best nectar is to be found. In the algorithm, there are different types of bees because of their role in the hive. The first group are scouts who look for food at random way and communicate with others by waggle dance in the hive. The second group are an onlookers who choose the best areas for exploration after the watch of the dance. The last are employed bees who are looking for the best quality nectar in the designated areas. All of them communicate to pass the information. Model of this process is performed in the following iterations in the algorithm. Each iteration means that all the bees left home in the search of food and came back pass the information to the others. In the next iteration other bees fly to these locations to search for food. If a better source of food is found this is information is passed at the end of iteration.

In the first stage of the algorithm, we create an array that reflects the map of meadow where bees are moving in search of the best flowers. At the beginning, all fields (except entrance and exit) has the same value of ζ. The exits are marked threefold value of ζ. Depending on the quality of the place, each bee may modify this value by the following equation

$$\Theta(\zeta) = \begin{cases} \zeta + 0,1 & \text{if} & \Gamma(\mathbf{x_i}, \zeta) < 0,5 \\ \zeta - 0,05 & \text{if} & \Gamma(\mathbf{x_i}, \zeta) > 0,5 \end{cases}, \qquad (6)$$

where $f(\mathbf{x_i}, \zeta)$ is called fitness function and it is calculated by

$$\Gamma(\mathbf{x_i}, \zeta) = \zeta\sqrt{(x_{r,0} - x_{i,0})^2 + (x_{r,1} - x_{i,1})^2}, \qquad (7)$$

where r is a index of exit to which the distance is the smallest from current position.

In the implementation of the algorithm, we assume that each bee is a point in space. When the new sources are found, the bees can leave the current position towards better. Bees are looking for a new source when they receive information from the hive after watching the waggle dance. At the end of each iteration, all the bees are compared in order to obtain information about the location of the best source. It is interpreted as the waggle dance. An onlooker bee choose the best area through by

$$p(\mathbf{x_i}) = \frac{\Gamma(\mathbf{x_i}, 1)}{\sum_{i=1}^{i=n} \Gamma(\mathbf{x_i}, 1)}. \qquad (8)$$

After gaining information about the fitness of bees, they are sorted to find the best positions. After that, onlookers can move in this direction. The movement of bees is made by

$$\mathbf{x_i}^{t+1} = \mathbf{x_i}^{t} + \alpha_k \cdot \Delta\mathbf{x_{ik}}, \qquad (9)$$

where k is a random index from the set of the best sources, α_k is a random number from $\langle -1, 1 \rangle$ and $\Delta\mathbf{x_{ik}}$ is obtained by

$$\Delta\mathbf{x_{ik}} = (x_{ij} - x_{kj}), \qquad (10)$$

where j is randomly chosen spatial coordinate of the chosen bee. The implemented algorithm is illustrated in Algorithm 2. The algorithm returns an array that is a map on which the bees moved.

Algorithm 2 ABCA to Create Mazes

1: Start,
2: Define all coefficients: n size of population, m - number of chosen best bees, ζ the minimum value to create a wall, r number of random alleys,
3: Create position array with value ζ and two different exits,
4: Create population,
5: **while** the queen does not pass the entire maze **do**
6:   Evaluate the values in array according to (6)
7:   Evaluate population using (8)
8:   Sort *bees* according to the value of location,
9:   Select m best locations among all *bees*,
10:   Other *bees* replace with randomly selected bees
11:   using (9) move the *bees* toward nectar source (exits) defined in (10),
12: **end while**
13: Recalculate the value from position array according to (11),
14: Return array,
15: Stop.

C. An Adaptation of AACA and ABCA to Create Mazes

An adaptation of presented Swarm Intelligence algorithms to create mazes requires additional operations to obtain picture of the maze. Each of the presented algorithms returns an array of values (ants an array of pheromones, bees the map of

meadow) which represent the maze. Each of these values is calculated by the following function

$$\Lambda(y) = \begin{cases} y \in \langle 0, \kappa \rangle & \text{empty space} \\ y \in (\kappa, 1 \rangle & \text{walls} \end{cases}, \qquad (11)$$

where κ is the limit value for which it will create a wall of the maze. Presented AACA and ABCA algorithms are used

---

Algorithm 3 The Algorithm for Maze Design Using Swarm Intelligence

1: Start,
2: Use Algorithm 1 or Algorithm 2 to create an array representing a maze,
3: Create a bitmap I,
4: **for all** value v in array **do**
5:   **if** v is 1 **then**
6:     **if** $r > 0$ **then**
7:       **if** random value is less than 0.5 **then**
8:         Create a wall.
9:       **end if**
10:     **else**
11:       Create a wall.
12:     **end if**
13:   **else**
14:     **for all** neighbor of v **do**
15:       **if** neighbor is 1 **then**
16:         Create a wall.
17:       **end if**
18:     **end for**
19:   **end if**
20: **end for**
21: Stop.

---

in Algorithm 3 to create various mazes. However to improve the efficiency of board games creation we have implemented a dedicated stop construction process. This is based on stop condition that is used to verify if it is still necessary to continue to compose of the paths in the maze.

D. Stop Condition

To adapt the AACA and ABCA algorithms described in subsection II-A and II-B, a dedicated stop condition must be adjusted to create a road from entrance to exit of the maze. For this purpose, the queen (a moving supervisor) is added to each of the algorithms. After each iteration, the queen is called to check whether exists a passage through constructed maze. If the passage exits, the algorithm is complete and the maze is ready. Otherwise, the next iteration begins since the queen is not satisfied with the work of their subordinates.

Seeking for an exit, the queen moves according to the Cartesian metric defined in (3) where we assume that

$$L_{ij} = 1. \qquad (12)$$

The possibility of passing diagonally across the maze is locked in this way. That helps to ensure that the queen moves only in horizontal or vertical line. The whole stop condition algorithm

---
Algorithm 4 The imperial march of the Queen

1: Start,
2: Create an array in accordance with (11),
3: Find all the entrances to the maze,
4: **for all** entry to the maze **do**
5:     **while** there is no other movement **do**
6:         Find neighboring fields,
7:         Remove fields in a row, in which the Queen was in the previous step,
8:         **for all** neighboring fields **do**
9:             **if** equation (12) is not true or the field is a wall **then**
10:                 Delete field,
11:             **end if**
12:         **end for**
13:         Select at random one of the existing movements,
14:     **end while**
15:     **if** the last field is one of the entrances **then**
16:         End of the algorithm - there is a way out of the maze,
17:     **end if**
18: **end for**
19: End of the algorithm - there is no way out of the maze,
20: Stop.
---

is shown in Algorithm 4. For even more complicated constructions of mazes, the r parameter can be added. Parameter r will represent the number of random alleys which is the number of walls to be removed in random way. In this case, when an initial board is created we can improve efficiency of the swarm to create the passage.

III. EXPERIMENTS

Presented approach was implemented to create various mazes in different resolutions and combinations. We have performed benchmark tests on various dimensions of boards. Let us present sample results for:

- square mazes (Fig. 1 and Fig. 2) with parameters values $n = 30$, $r = 20$, $\zeta =$, ABCA ($m = 10\%n$), AACA ($\alpha = 0, 4$, $\rho = 0, 3$),
- rectangle mazes (Fig. 3 and Fig. 4) with parameters values $n = 200$, $r = 120$, $\zeta =$, ABCA ($m = 10\%n$), AACA ($\alpha = 0, 4$, $\rho = 0, 3$).

Fig. 5 presents a chart of time comparison for creation of mazes by both AACA (blue line) and ABCA (orange line).
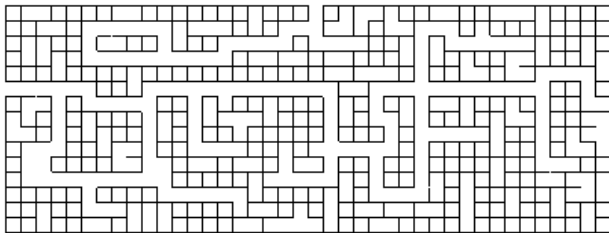

Fig. 1. An example of 40×15 maze generated by the AACA
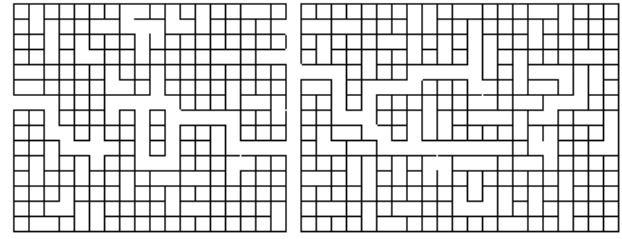

Fig. 2. An example of 40×15 maze generated by the ABCA

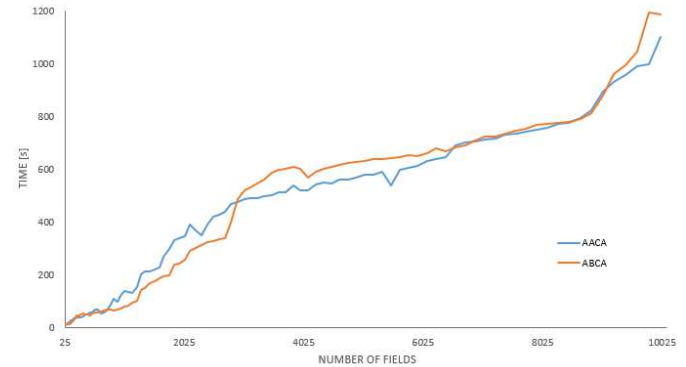Up to 3000 fields in the maze AACA is less efficient in time

Fig. 5. Time to generate the maze for applied algorithms

comparison to ABCA. If the maze we create contains between 3000 and 6000 fields AACA take advantage. From 6000 to 9000 fields both algorithms present similar time efficiency. Above 10000 fields AACA is much faster.

IV. FINAL REMARKS

In the research, many experiments have been realized for different dimensions of mazes. The results allow to conclude that Swarm Intelligence is capable to create mazes. Fig 5 shows that the maze of complex up to 3000 fields is best to use ABCA presented in section II-B, but for larger mazes AACA presented in section II-A seems to be a better choice.

Generating maze is quite a complex process but the results are very satisfied mazes are created randomly which gives a feature of uniqueness. As a result, the algorithms presented in this paper can be used to create not only mazes but environments for 2D or even 3D games and also applicable to smart cities models for prediction and evaluation purposes.

REFERENCES

[1] J. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105, 1973.
[2] J. Baldwin, "A new factor in evolution," *The American Naturalist*, vol. 30, no. 354, pp. 441–451, 1986.
[3] X. Yang and S. Deb, "Cuckoo search via lévy flights," in *NaBIC'2009 Proceedings*, 2009, pp. 210–214.
[4] E. Hetmaniok, D. Słota, and A. Zielonka, "Experimental verification of immune recruitment mechanism and clonal selection algorithm applied for solving the inverse problems of pure metal solidification," *Int. Comm. Heat & Mass Transf.*, vol. 47, pp. 7–14, 2013.
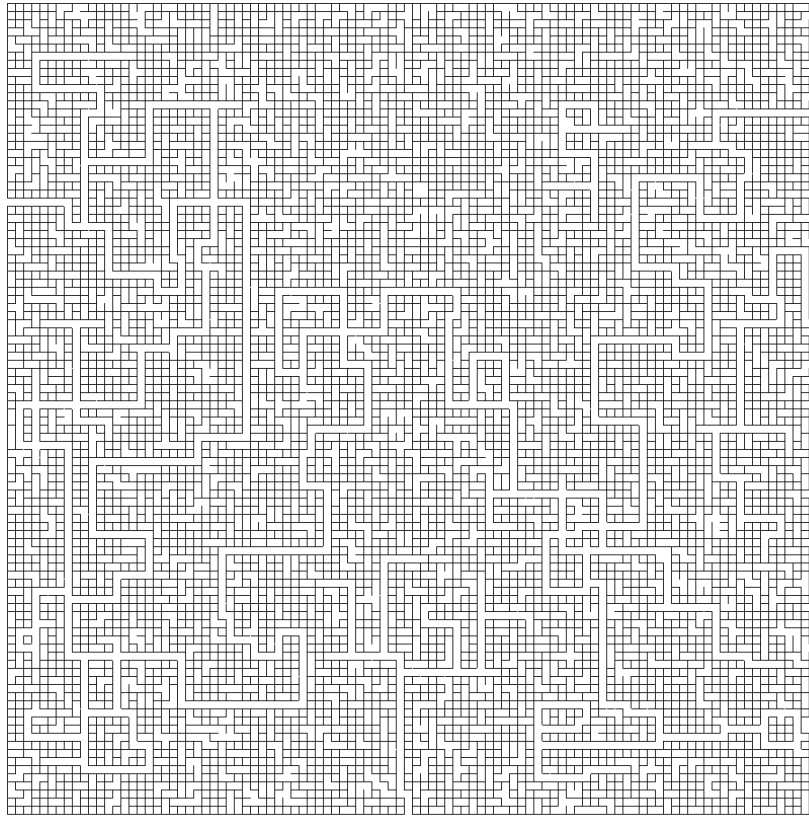
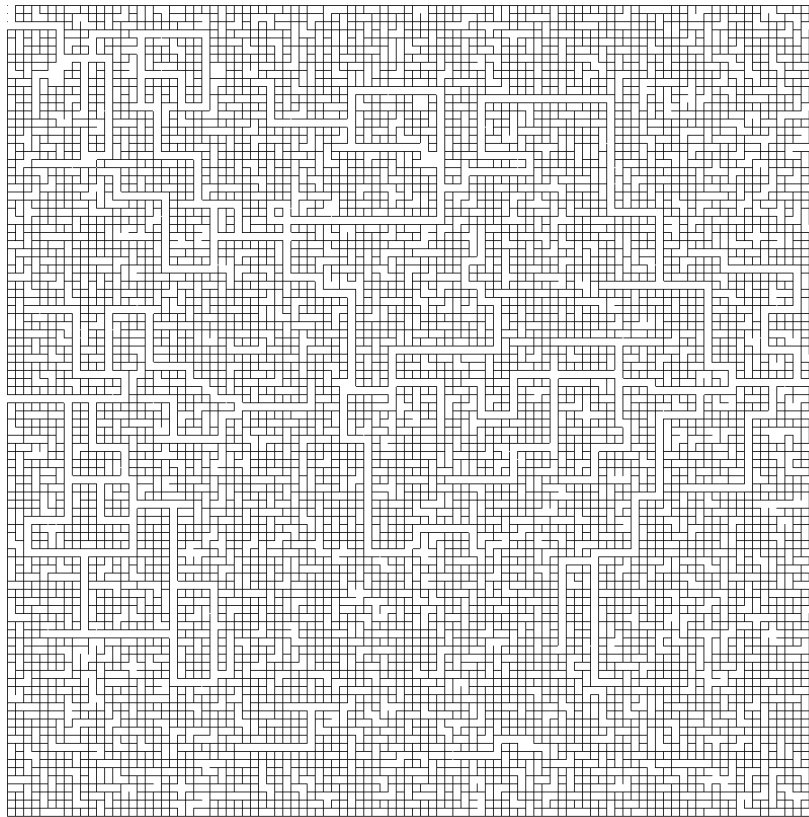Fig. 3. An example of 100×100 maze generated by the AACA

Fig. 4. An example of 100×100 maze generated by the ABCA

[5] R. Brociek and D. Słota, "Application of intelligent algorithm to solve the fractional heat conduction inverse problem," *Communications in Computer and Information Science - ICIST'2015*, vol. 538, pp. 356–365, 2015, DOI: 10.1007/978-3-319-24770-0_31.

[6] E. Hetmaniok, D. Słota, and A. Zielonka, "Solution of the inverse continuous casting problem with the aid of modified harmony search algorithm," in *Parallel Processing and Applied Mathematics, Part I*, ser. LNCS, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski, Eds., vol. 8384. Springer-Verlag, 2014, pp. 402–411.

[7] M. Woźniak, D. Połap, G. Borowik, and C. Napoli, "A first attempt to cloud-based user verification in distributed system," in *Asia-Pacific Conference on Computer Aided System Engineering APCASE'2015*. 14-16 July, Quito, Ecuador: IEEE, 2015, pp. 226–231, DOI: 10.1109/AP-CASE.2015.47.

[8] M. Woźniak, W. M. Kempa, M. Gabryel, R. K. Nowicki, and Z. Shao, "On applying evolutionary computation methods to optimization of vacation cycle costs in finite-buffer queue," *Lecture Notes in Artificial Intelligence - ICAISC'2014*, vol. 8467, pp. 480–491, 2014, DOI: 10.1007/978-3-319-07173-2_41.

[9] C. Napoli, G. Pappalardo, E. Tramontana, and Zappalà, "A cloud-distributed gpu architecture for pattern identification in segmented detectors big-data surveys," *The Compter Journal*, p. bxu147, 2014, DOI: 10.1093/comjnl/bxu147.

[10] C. Napoli, G. Pappalardo, and E. Tramontana, "An agent-driven semantical identifier using radial basis neural networks and reinforcement learning, *Proceedings of XV Workshop From Objects to Agents (WOA)*, vol. 1260, CEUR-WS, September, 2014.

[11] M. Gabryel, M. Woźniak, and R. Damaševičius, "An application of differential evolution to positioning queueing systems," *Lecture Notes in Artificial Intelligence - ICAISC'2015*, vol. 9120, pp. 379–390, 2015, DOI: 10.1007/978-3-319-19369-4_34.

[12] P. Dziwiñski, L. Bartczuk, and J. T. Starczewski, "Fully controllable ant colony system for text data clustering," *Lecture Notes in Computer Science - ICAISC'2012*, vol. 7269, pp. 199–205, 2012, DOI: 10.1007/978-3-642-29353-5.

[13] P. Dziwiñski, L. Bartczuk, A. Przybyl, and E. Avedyan, "A new algorithm for identification of significant operating points using swarm intelligence," *Lecture Notes in Artificial Intelligence - ICAISC'2014*, vol. 8468, pp. 349–362, 2014, DOI: 10.1007/978-3-319-07176-3_31.

[14] M. Okulewicz and J. Mandziuk, "Two-phase multi-swarm PSO and the dynamic vehicle routing problem," in *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIHLI 2014: 2014 IEEE Symposium on Computational Intelligence for Human-Like Intelligence, Proceedings*. 9-12 December, Orlando, Florida, USA: IEEE, 2014, pp. 86–93, DOI: 10.1109/CIHLI.2014.7013391.

[15] M. Woźniak, D. Połap, M. Gabryel, R. K. Nowicki, C. Napoli, and E. Tramontana, "Can we preprocess 2d images using artificial bee colony?" *Lecture Notes in Artificial Intelligence - ICAISC'2015*, vol. 9119, pp. 660–671, 2015, DOI: 10.1007/978-3-319-19324-3_59.

[16] C. Napoli, G. Pappalardo, E. Tramontana, Z. Marszałek, D. Połap, and M. Woźniak, "Simplified firefly algorithm for 2D image key-points search," in *IEEE SSCI 2014 - 2014 IEEE Symposium Series on Computational Intelligence - CIHLI 2014: 2014 IEEE Symposium on Computational Intelligence for Human-Like Intelligence, Proceedings*. 9-12 December, Orlando, Florida, USA: IEEE, 2014, pp. 118–125, DOI: 10.1109/CIHLI.2014.7013395.

[17] M. Woźniak and D. Połap, "Basic concept of cuckoo search algorithm for 2D images processing with some research results : An idea to apply cuckoo search algorithm in 2d images key-points search," in *SIGMAP 2014 - Proceedings of the 11th International Conference on Signal Processing and Multimedia Applications, Part of ICETE 2014 - 11th International Joint Conference on e-Business and Telecommunications*. 28-30 August, Vienna, Austria: SciTePress, 2014, pp. 157–164, DOI: 10.5220/0005015801570164.

[18] C. Napoli, G. Pappalardo, E. Tramontana, R. K. Nowicki, J. T. Starczewski, and M. Woźniak, "Toward automatic work groups classification based on probabilistic neural network approach," *Lecture Notes in Artificial Intelligence - ICAISC'2015*, vol. 9119, pp. 79–89, 2015, DOI: 10.1007/978-3-319-19324-3_8.

[19] M. Woźniak, C. Napoli, E. Tramontana, G. Capizzi, G. Lo Sciuto, R. K. Nowicki, and J. T. Starczewski, "A multiscale image compressor with rbfnn and discrete wavelet decomposition," in *IEEE IJCNN 2015 - 2015 IEEE International Joint Conference on Neural Networks, Proceedings*. 12-17 July, Killarney, Ireland: IEEE, 2015, pp. 1219–1225, DOI: 10.1109/IJCNN.2015.7280461.

[20] M. Woźniak, D. Połap, R. K. Nowicki, C. Napoli, G. Pappalardo, and E. Tramontana, "Novel approach toward medical signals classifier," in *IEEE IJCNN 2015 - 2015 IEEE International Joint Conference on Neural Networks, Proceedings*. 12-17 July, Killarney, Ireland: IEEE, 2015, pp. 1924–1930, DOI: 10.1109/IJCNN.2015.7280556.

[21] I. Martisius and R. Damasevicius, "Class-adaptive denoising for EEG data classification," *Lecture Notes in Artificial Intelligence - ICAISC'2012*, pp. 302–309, 2012.

[22] C. Napoli and E. Tramontana, "An object-oriented neural network toolboox based on design patterns," *Communications in Computer and Information Science - ICIST'2015*, vol. 538, pp. 388–399, 2015, DOI: 10.1007/978-3-319-24770-0_34.

[23] M. Swiechowski and J. Mandziuk, "Self-adaptation of playing strategies in general game playing," *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 6, no. 4, pp. 367–381, 2014, DOI: 10.1109/TCI-AIG.2013.2275163.

[24] K. Waledzik and J. Mandziuk, "An automatically generated evaluation function in general game playing," *IEEE Trans. Comput. Intellig. and AI in Games*, vol. 6, no. 3, pp. 258–270, 2014, DOI: 10.1109/TCI-AIG.2013.2286825.

[25] J. Karkowski and J. Mandziuk, "A new approach to security games," *Lecture Notes in Artificial Intelligence - ICAISC'2015*, vol. 9120, pp. 402–411, 2014, DOI: 10.1007/978-3-319-19369-4_36.

[26] A. Hirao, Y. Nomura, H. Yonezu, and H. Takeshita, "Prim's algorithm based p2mp energy-saving routing design for midori," in *IEEE COIN 2012 - IEEE International Conference on Optical Internet, Proceedings*. 29-31, May, Yokohama, Kanagawa: IEEE, 2012, pp. 86–93.

[27] L. Najman, J. Cousty, and B. Perret, "Playing with kruskal: Algorithms for morphological trees in edge-weighted graphs," *Lecture Notes in Computer Science - MMASIP'2013*, vol. 7883, pp. 135–146, 2013.

[28] E. Hetmaniok, D. Słota, and A. Zielonka, "Determination of the heat transfer coefficient by using the ant colony optimization algorithm," in *Parallel Processing and Applied Mathematics, Part I*, ser. LNCS, R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wa/sniewski, Eds., vol. 7203. Springer, 2012, pp. 470–479.

[29] D. Połap, M. Woźniak, C. Napoli, E. Tramontana, and R. Damaševičius, "Is the colony of ants able to recognize graphic objects?" *Communications in Computer and Information Science - ICIST'2015*, vol. 538, pp. 376–387, 2015, DOI: 10.1007/978-3-319-24770-0_33.