Weight Based Algorithms to Increase the Playability in 2D Games

Alicja Winnicka, Karolina Kęsik,
Kalina Serwata and Kamil Książek
Institute of Mathematics
Silesian University of Technology
Kaszubska 23, 44-100 Gliwice, Poland
Email: winnicka.alicja@10g.pl, karola.ksk@gmail.com
kalarcika@gmail.com, kamilksiazek95@gmail.com

***Abstract*—Increasing the level of playability in games depends on the engine's operation. The more accurate and predictable the game is, the greater the difficulty level will be. However, the level of playability may decrease due to the player's interest. To prevent this, we propose a combination of different techniques for universal operation on various two-dimensional games. The used methods have been modified in such a way to show that hybrid forms can be much more advantageous.**

The proposition has been tested on selected games, and the obtained results were analyzed and discussed depending on the introduced modifications. The aim of the discussion was to indicate the various advantages and disadvantages of these techniques to increase the playability.

I. Introduction

The game development is dependent on graphics card producers and player requirements. The more efficient the equipment will be available on the market, the more real and complicated games should be. Particularly the second aspect is important, because even the simplest game can attract human attention. The problem is the lack of holding this attention. To remedy this, games use more and more new algorithms that allow a computer counterattack during the game. However, the perfect operation of the algorithm can cause the player will not have the slightest chance of winning, which will cause the game to quickly be forgotten. This lack of interest should be prevented by the addition of a certain, preferably controlled randomness in his movements.

Particularly the development of artificial intelligence techniques finds its applications in the various aspects of our lives, not only in entertainment, which games are an example. Decision support systems are the most known application of these techniques. One such example is medicine, where computer can confirm or even detect some diseases based on a given samples of X-RAY or CT images. Selected tools for that problems can be heuristic algorithms, what is presented in [15]. Another example are energy networks [1], [2], [10]. Along with the rapid development of artificial intelligence, other branches also sharply advance. This is especially visible in databases and warehousing because more and more information needs to be stored, where specific queries or sorts are performed [8]. Subsequently, authorization of these data is important, as can be seen from the number of publications and scientific papers in the field of biometrics [5], [11].

Entertainment forces a rapid pace of development, which can be seen after indicating new trends such as extended, virtual or mixed realities. In [9], [13], the current open challenge in the field of games are described. In [4], monte carlo tree search method with learning mechanism for video game is presented. Authors of [14] discuss about the idea of reinforcement learning method for the use in multiplayer nonzerosum games. Again in [7], some smart technique for agents in game is described. Important part of artificial intelligence are neural networks, which also have found their place in games [3].

In this paper, we propose a hybrid solution connected with the classic techniques like q-learning, a* and tabu search. Our technique is based on the fitness function which depends on the selection of a specific metric.

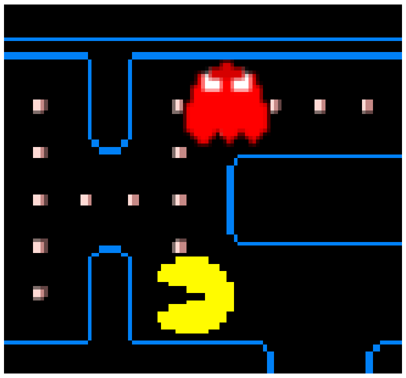
II. Simplified A* with tabu table

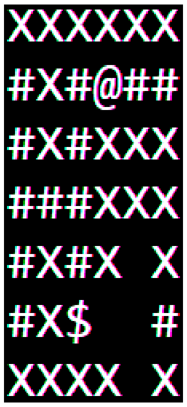
Proposed algorithm is based on path search algorithms. However it does not check whole path to the target, but only the value of points, which are the neighbours of chasing player (or enemy, depending on the point of view). Firstly, it needs to find all possible fields on the board that can be visited. After this action, it is necessary to compute the value of each field $\Theta(\cdot)$. The evaluation is made according to the following formula

$$\Theta(\mathbf{A}, \mathbf{B}) = d(\mathbf{A}, \mathbf{B}) + w_{ij}, \tag{1}$$

where **A** and **B** are the points in two-dimensional space understood as (x_A, y_A) and (x_B, y_B). First point is the possible next field of chasing player and the second one is the localization of target. Function $\Theta(\cdot)$ is the sum of two components, the value of the field calculated as a specific metric and the weight w_{ij}, where i and j are given positions on the board. The weight is selected in random way in the range $[-1, 1]$. Let us assume that d is such a function that

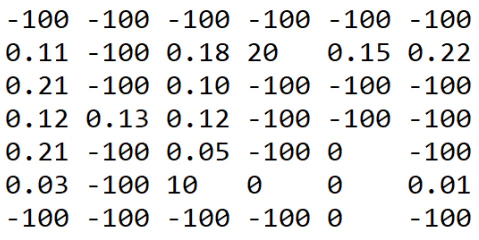
$$d : X \times X \to [0, \infty], \tag{2}$$


(a) The pacman game



```
XXXXXX
#X#@##
#X#XXX
###XXX
#X#X X
#X$  #
XXXX X
```

(b) A console game



-100	-100	-100	-100	-100	-100
0.11	-100	0.18	20	0.15	0.22
0.21	-100	0.10	-100	-100	-100
0.12	0.13	0.12	-100	-100	-100
0.21	-100	0.05	-100	0	-100
0.03	-100	10	0	0	0.01
-100	-100	-100	-100	0	-100

(c) A table which was used to create two simple games.

Figure 1: Visualisation of the game areas.

where X is non-empty set. d is called a metric. For any two points $\mathbf{A}, \mathbf{B} \in X$, the function must satisfy the following properties:

1) distance between two points is equal to 0 if and only if points have the same coordinates

$$d(\mathbf{A}, \mathbf{B}) = 0 \iff \mathbf{A} = \mathbf{B}, \qquad (3)$$

2) symmetry rule – a distance between **A** and **B** is the same like a distance between **B** and **A**

$$d(\mathbf{A}, \mathbf{B}) = d(\mathbf{B}, \mathbf{A}), \qquad (4)$$

3) triangle inequality – a distance between **A** and **B** is less or equal the sum of distances between **A** and **C** (**C** is an intermediate point) and between **C** and **B**

$$d(\mathbf{A}, \mathbf{B}) \leq d(\mathbf{A}, \mathbf{C}) + d(\mathbf{C}, \mathbf{B}). \qquad (5)$$

The most known metric is the Euclidean one defined as:

$$d_E(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \qquad (6)$$

where n is the number of coordinates of points. Another example is the taxicab metric (called also the Manhattan metric), defined as follows:

$$d_M(\mathbf{A}, \mathbf{B}) = \max_{i=1,\ldots,n} |x_i - y_i|. \qquad (7)$$

The last presented case is the jungle river metric understood as

$$d_R(\mathbf{A}, \mathbf{B}) = d_E(\mathbf{A}, \mathbf{C}_1) + d_E(\mathbf{C}_1, \mathbf{C}_2) + d_E(\mathbf{C}_2, \mathbf{B}), \qquad (8)$$

where $\mathbf{C}_1$ and $\mathbf{C}_2$ are orthogonal projections of **A** and **B**, respectively, on the line r (r is called a river).

Each game takes place in a certain area or board. Let us assume that the board size is $w \times h$. In the case when all fields are empty, and the computer players are approaching to the user, the game would be too simple. It is necessary to put some obstacles on the board, what will be marked as a large number, for example 100. A random value w_{ij} will be assigned to each empty field at the position (i, j). Additionally, player will be marked as 10 and the enemies (moved by the computer) as 20. Visualization of such a board is shown on Fig. 1.

The player moves towards the user by selecting the field, in which a value of the fitness function Θ described by Eq. (1) is the smallest. However, it is possible a situation where the algorithm gets stuck. An example of such a setting is a corner, where a character on three sides has a wall (further walk is prevented). Starting from this position, there is a very high probability of returning to the same field. In order to avoid described situation, we introduce a tabu table, where movements that were made and did not bring any benefits (i.e. preventing further walk in the direction of a user) are saved. The proposed algorithm is presented in Alg. 1.

III. Experiments

The proposed solution has been implemented in C# and tested in terms of the number of won games, the number of necessary moves that the algorithm needs to caught the player, and various metrics which have been described earlier. In addition, all tests were carried out depending on the number of fields on the board, the amount of obstacles or even the points which must be collected by the player. As points, we mean in the case of the console the symbols '#', and in the case of pacman – dots.

In Fig. 2 is shown how the average number of moves performed by the algorithm increases depending on the size of the board. The initial size of board was 400 and during subsequent steps 20 fields were added (up to 760). In each case the growth of moves was linear (what is shown thanks to linear regression). However, the slope of the line is relatively small, which indicates the advantage of the proposed solution. Even in case of a large board, presented method was effective. In addition, almost perfect linear growth in the average number of movements was obtained for the jungle river metric. A bit worse results were achieved for the Manhattan metric and the worst metric was the Euclidean one. This is caused by randomly located obstacles at the game board. Again in

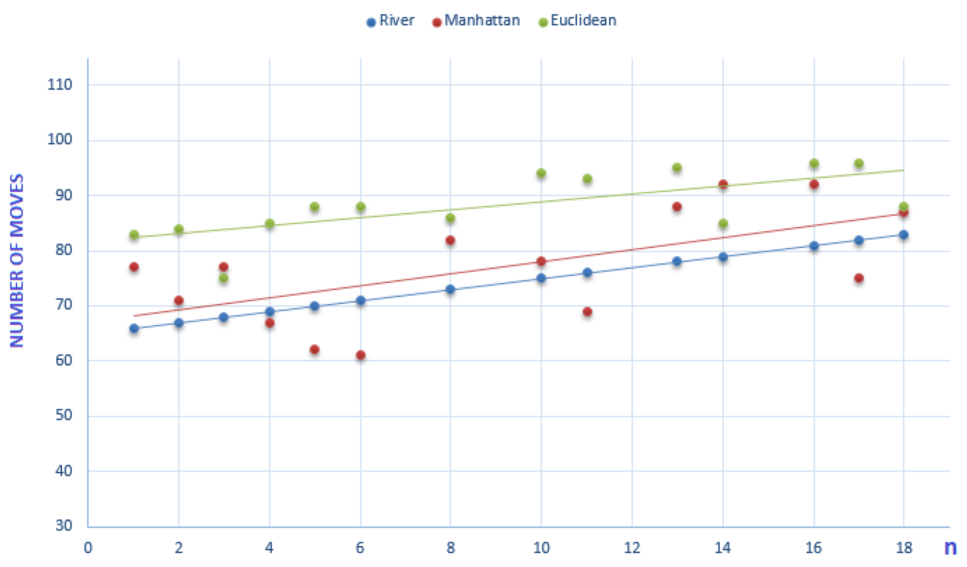


Figure 2: Graph of the average number of moves needed by the algorithm to catch a player on a board of $400 + 20 \cdot n$ fields (where n is the value on the X-axis).

Algorithm 1 Weight-based algorithm for choosing the next movement.

```
1:  Start.
2:  Define all the possible movements in dependence of player's position.
3:  Select randomly one direction.
4:  Calculate a value of the fitness function for selected direction by using Eq. (1) and mark this value as $\alpha$.
5:  Create empty tabu table.
6:  **for each** possible movements **do**
7:      Calculate a value of the fitness function $f$ according to Eq. (1).
8:      **if** $f < \alpha$ **then**
9:          **if** move is not in tabu table **then**
10:             Change the direction.
11:             $\alpha = f$.
12:         **end if**
13:         $k = 0$.
14:         **for each** neighbour **do**
15:             **if** a movement in the direction of this neighbour is forbidden **then**
16:                 $k++$.
17:             **end if**
18:         **end for**
19:         **if** $k = 3$ **then**
20:             Add this movement to tabu table.
21:         **end if**
22:     **end if**
23: **end for**
24: Return the best movement in the indicated direction.
25: Stop.
```

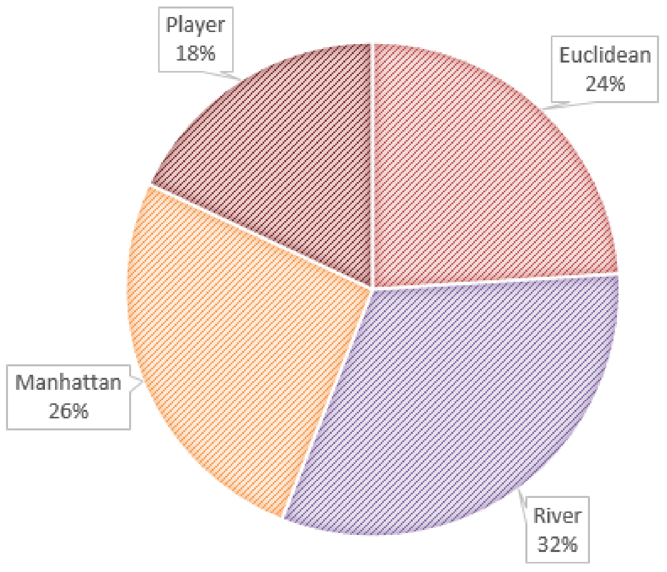


Figure 3: Percentage variety of wins in 100 games played on the board with 400 fields and 50 randomly located points to gain.

Fig. 3, the average number of wins during 100 experiments is presented. The most effective was the proposed algorithm with the jungle river metric. This combination was the most succesful during 32% out of the total number of games. The Manhattan metric was slightly less effective (the winner of 26% games). What is interesting, the weakest results out of presented metrics were achieved by the Euclidean one (24 %). The smallest number of victories had a player – only 18%. It can be concluded that beating the algorithm was not easy but still possible.

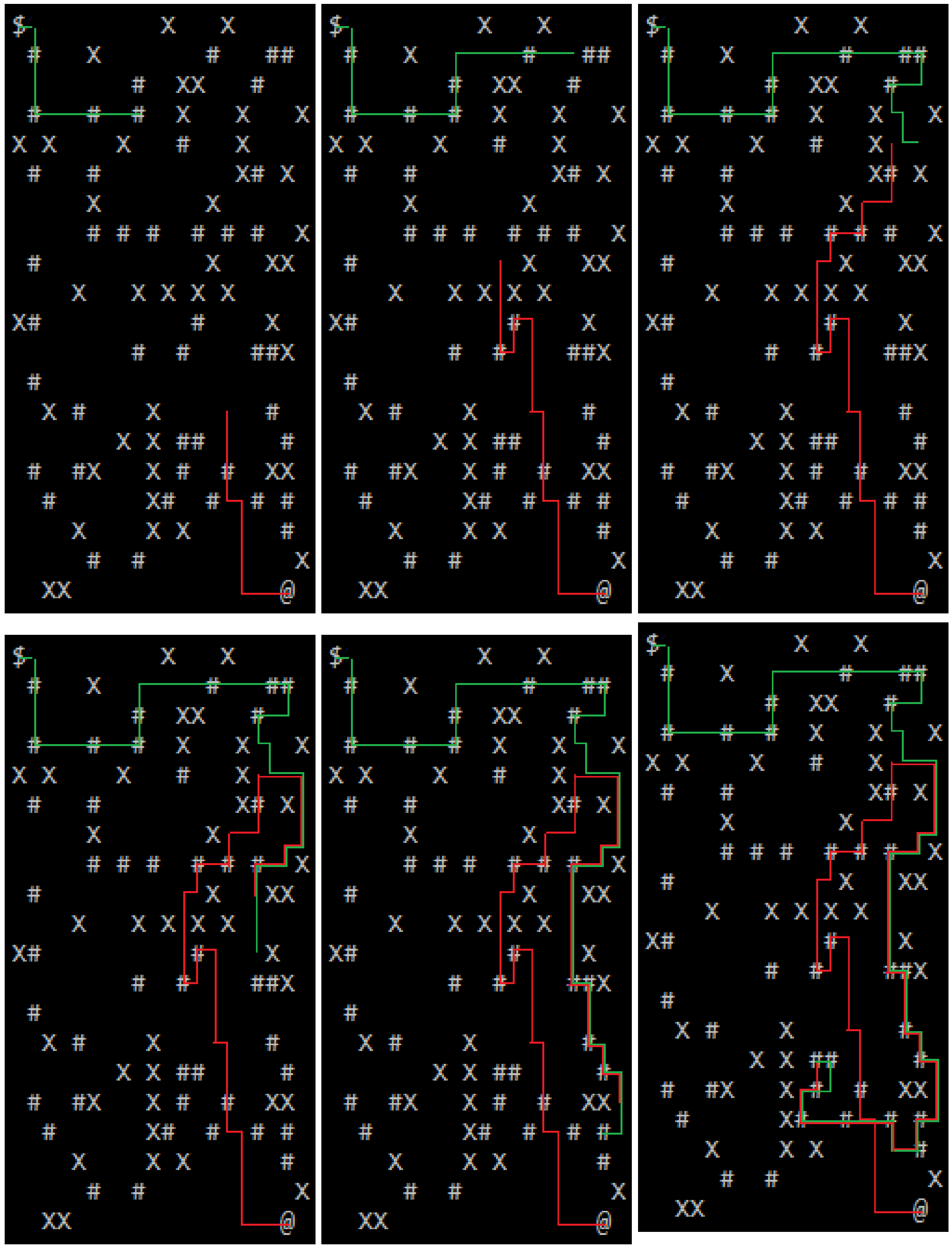


Figure 4: Example of player movements – the green line is a trace of movements made by the user, and the red one – by the proposed algorithm by using the Euclidean metric.

The number of calculations is small in comparison to other graphic algorithms which have to search the entire board. Only the position of the player is the necessary knowledge for applying of this technique. The path to the given object is determined by a dedicated function. In addition, various metrics have different effectiveness, so it allows us to propose a game with varied difficulty level by application of a specific metric. This type of distinction not only diversifies the game, but also does not allow the player to adapt to one level of computer intelligence.

IV. Conclusions

A hybrid solution based on known algorithms has been shown in this paper. Three different metrics were selected: the jungle river, the Manhattan and the Euclidean, which were used interchangeably in the assessment of possible movements to be performed by a computer-controlled player. The obtained results showed that this type of solution has the right to be used as a computational intelligence. Especially, if the difficulty level of the game would be understood as applying a different metric. Measurements for each of presented metrics showed that the method's effectiveness depends on the right choice of them. In addition, they are stable in terms of growth of the calculation, which increases slightly when the size of the board grows.

During further research it is possible to apply other, less known metrics or use described model in other, a bit more complicated games and check efficiency of the method. Measurements prove that such approach can be successfully implemented in similar types of games.

The algorithm can be used in different games when player need to catch moving element in the game, which doesn't need to be player. This model of playing can be applied in most arcade games and many another ones. Presented ideas has a wide range of applications in this brand of science.

References

[1] G. Capizzi, G. L. Sciuto, C. Napoli, and E. Tramontana. Advanced and adaptive dispatch for smart grids by means of predictive models. *IEEE Transactions on Smart Grid*, 2017.

[2] G. Capizzi, G. L. Sciuto, C. Napoli, and E. Tramontana. An advanced neural network based solution to enforce dispatch continuity in smart grids. *Applied Soft Computing*, 62:768–775, 2018.

[3] A. Dobrovsky, C. W. Wilczak, P. Hahn, M. Hofmann, and U. M. Borghoff. Deep reinforcement learning in serious games: Analysis and design of deep neural network architectures. In *International Conference on Computer Aided Systems Theory*, pages 314–321. Springer, 2017.

[4] E. Ilhan and A. Ş. Etaner-Uyar. Monte carlo tree search with temporal-difference learning for general video game playing. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*, pages 317–324. IEEE, 2017.

[5] J. Kapočiūtė-Dzikienė, A. Venčkauskas, and R. Damaševičius. A comparison of authorship attribution approaches applied on the lithuanian language. In *Computer Science and Information Systems (FedCSIS), 2017 Federated Conference on*, pages 347–351. IEEE, 2017.

[6] 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.

[7] A. Khalifa, M. Preuss, and J. Togelius. Multi-objective adaptation of a parameterized gvgai agent towards several games. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 359–374. Springer, 2017.

[8] Z. Marszałek. Performance tests on merge sort and recursive merge sort for big data processing. *Technical Sciences*, 21(1):19–35, 2018.

[9] F. Milani, A. C. B. De Marchi, and R. Rieder. Usability guidelines to develop gesture-based serious games for health: A systematic review. In *Virtual and Augmented Reality (SVR), 2017 19th Symposium on*, pages 188–194. IEEE, 2017.

[10] E. Okewu, S. Misra, R. Maskeliūnas, R. Damaševičius, and L. Fernandez-Sanz. Optimizing green computing awareness for environmental sustainability and economic security as a stochastic optimization problem. *Sustainability*, 9(10):1857, 2017.

[11] D. Połap. Extraction of specific data from a sound sample by removing additional distortion. In *Computer Science and Information Systems (FedCSIS), 2017 Federated Conference on*, pages 353–356. IEEE, 2017.

[12] D. Połap, M. Woźniak, C. Napoli, and E. Tramontana. Real-time cloud-based game management system via cuckoo search algorithm. *International Journal of Electronics and Telecommunications*, 61(4):333–338, 2015.

[13] S. Risi and J. Togelius. Neuroevolution in games: State of the art and open challenges. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(1):25–41, 2017.

[14] R. Song, F. L. Lewis, and Q. Wei. Off-policy integral reinforcement learning method to solve nonlinear continuous-time multiplayer nonzero-sum games. *IEEE transactions on neural networks and learning systems*, 28(3):704–713, 2017.

[15] M. Woźniak and D. Połap. Bio-inspired methods modeled for respiratory disease detection from medical images. *Swarm and Evolutionary Computation*, 2018.

[16] M. Woźniak, D. Połap, M. Gabryel, R. K. Nowicki, C. Napoli, and E. Tramontana. Can we process 2d images using artificial bee colony? In *International Conference on Artificial Intelligence and Soft Computing*, pages 660–671. Springer, 2015.

[17] 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.

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