The Opponent's Movement Mechanism in Simple Games Using Heuristic Method

Alicja Winnicka
Institute of Mathematics
Silesian University of Technology
Kaszubska 23, 44-100 Gliwice, Poland
Email: winnicka.alicja@10g.pl

***Abstract*—All kinds of games force continuous development of not only hardware technologies, but also solutions that increase playability. One of such aspects is the way the world interacts with the user. The effect of this is the technique of opponents action, which enforces a certain degree of difficulty on the player and is constantly driven and motivates to pass the game faster and more efficiently. In this work, I present the strategy of reaching a certain point on the board by the computer through the use of a modified heuristic algorithm, ie a Cuckoo Search Algorithm. The proposed solution has been described, tested and discussed due to the advantages and disadvantages of this solution.**

I. Introduction

The computer market develops through a lot of competition, as can be seen from the AMD and NVIDIA companies that produce graphics cards. Releasing a product of one of these companies, forces a lot of pressure on the other one. It results in the release of another card within a few months. Graphics cards are one of the basic requirements of computer games so the pressure to produce more efficient and cheaper cards is also on the side of game producers. Developers put more requirements for hardware due to improvement realism and computer graphics used in their products. An additional burden for computers are engines, which are based on the methods of artificial intelligence. The goal is to increase playability and quality of products

Creating games is about applying and modifying advanced solutions to achieve the greatest effectiveness. Effectiveness is understood as the time of action, computing power, the quality of the effects obtained and the impact on playability. One of the basic elements due to the programmers is designing the world, or maps on which the user can move. In [6], the authors presented procedural approach to generating game maps. Again in [3], the idea of prediction player's move is analyzed by the use of neural networks. This solution can improve, especially, the playability of games by blocking the execution of reflex movements. An interesting aspect is the analysis of information for a given game and use for a similar purpose as before [16]. In [15] presented the famous algorithm that beat the World Champion in GO. Artificial intelligence techniques have found application in protecting the lives of players while using phones in augmented reality [11]. Not only expanded but virtual reality is developed using these algorithms [1], [2].

Heuristic algorithms find an increasing application in game theory due to their advantages. Not the predictability achieved by random movements, as well as low computing power. In [9], game balancing was achieved by the use of one of these algorithms. Moreover, it is interesting to create hybrids of various classic techniques in games such as decision trees with fuzzy logic [8] or heuristics [14]. Of course, for large parameter values, each algorithm will need more power. For this reason, parallelization and use of the full available processor power can be used what has been shown in [12].

II. Heuristic algorithm

Cuckoo Search Algorithm's (CSA) [21] name derives from the model of cuckoo's behavior. During the breeding period cuckoos do not hatch their eggs by themselves, but search for another nests belonging to others birds and leave their eggs in there. They look for nests where the probability of being and hatching their offspring is greatest.

The natural behavior of these birds resulted in the creation of a model originally used as a minimization of continuous functions. Algorithm assumes that cuckoo is interpreted as a point (x, y) on the solution space. In these case it is a board of size $w \times h$. Additionally, some assumptions must be done to adapt the algorithm to a new operating environment

- potential nests are random points on the board,
- each cuckoo has only one egg to lay,
- total number of cuckoos is constant in each iteration.
- best nests means the points located closest to the goal,
- hosts may detect thrown eggs egg with probability $p \in \langle 0, 1\rangle$. In this case the egg is thrown out of the nest and new cuckoo is placed randomly on the board.

At the beginning of the models, some parameters of CSA must be defined as number of cuckoos and number of *iterations* (which can be interpreted as number of cuckoo's moves). The initial population of birds are chosen at random on the whole solution space. In each iteration, the best nest is chosen after some steps. The first one is the movement of each birds according to

$$x = x \pm L(x, d, c), \tag{1}$$

where addition or subtraction is selected at random with the equal probability 50% and $L(\cdot)$ is modified Levy distribution

burdened with the ceiling function so the obtained values are integers. This action is enforced by the solution space, which is indexes of the height and width of the board. This function can be formulated as

$$L(x,\zeta,\eta) = \left\lceil \sqrt{\frac{\eta}{2\pi}} \frac{e^{-2\eta(x-\zeta)}}{\sqrt{(x-\zeta)^3}} \right\rceil \tag{2}$$

where the variables $\zeta, \eta \in \mathbb{R}$ are randomly generated and x is a spatial coordinate of the mother-cuckoo.

The second step of the algorithms is the host decision. After, the cuckoo toss the egg, the host need to decide whether egg will stay or not in the nest. Of course, there is a chance that host will not notice tossed egg. Both situation are modeled by one condition as

$$\begin{cases} p & \text{drop the egg} \\ 1-p & \text{leave the egg} \end{cases}. \tag{3}$$

The algorithm returns the best cuckoo in the last iteration. To make this possible, the cuckoos must be compared. This is possible by introducing the fitness function that will clearly define which one is the best. It is made by the following formula

$$g(x) = d(x,t) \pm \sqrt{w+h}, \tag{4}$$

where t is the nearest goal in the neighborhood. The sign of action depends on whether the cuckoo's position is the goal of the game. If so, the value is subtracted, otherwise added. This action forces you to move towards the goal you have found. The used function $d(\cdot)$ is Euclidean metric defined as

$$d(x,t) = \sqrt{\sum_{i=0}^{2} (x_i - t_i)^2}, \tag{5}$$

where x and t are points in two dimensional space. The pseudo-code of these version is shown in Alg. 1

Algorithm 1 Modified Cuckoo Search Algorithm

1: Start,
2: Define all required coefficient, number of cuckoos n, fitness function $g(\cdot)$, number of iterations and static position of enemies,
3: Create initial population of n cuckoos at random,
4: t:=0,
5: **while** $t \leq T$ **do**
6: Drop the egg according to Eq. (1),
7: Make the host decision using Eq. 3,
8: $t++$,
9: **end while**
10: Find the cuckoo with the best adaptation according to fitness condition,
11: Return the best cuckoo,
12: Stop.

III. Game strategy based on heuristic algorithm

A. Initial settings

Let us assume that a computer walks around a certain board of size $w \times h$. His task is to catch all opponents in as few moves as possible, so in the shortest possible time. Enemies are scattered on the board in a random way. In the case when two players play the game, its task is to collect at least half of all points (for one enemies is one point) to win the game. Of course, each player is placed in the opposite corner of the board. The player can make one move in one step, alternately.

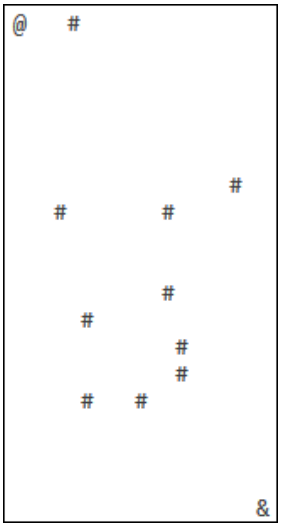


Figure 1: Created board game with enemies "#" and two players marked as "&" and "@".

In the game, several signs have been used. "@" as a player, "%" as a computer, and "#" as an enemy. The visualisation of these game is presented in Fig. 1.

B. Game strategy

Each player can take only one step in each round. The step can be made in one of four directions (north/south/west/east) but only when it is possible. The move will be impossible to do if the player wants to move outside the board.

The proposed strategy consists of two stages. The first step is to check the nearest neighbors. As a neighborhood, we understood a matrix (which the maximum size is 3×3), where the player is in its center. If the target is achievable after taking one movement, it is performed without the slightest hesitation. If the cost of obtaining a goal is a maximum of 3 moves, movement towards it is carried out. The cost is calculated by the Euclidean metric described in Eq. (5).

The second step is based on the existence of a case when the target is not in the player's neighborhood. And here, the Cuckoo Search Algorithm is used. A random population of cuckoos is spread on the board and shifted during a certain number of iterations. Individuals are looking for a goal using the fitness function. At the end of algorithm's performance, the best adapted cuckoo is returned relative to the current position of the player. The player makes one move towards the position

Algorithm 2 Proposed Game Strategy

Start,
Define the position of player,
neighbor:= $false$,
for each direction in set {north, south, west, east} **do**
  if the target position is a neighbor **then**
    neighbor:= $true$,
    break the loop,
  end if
end for
if neighbor is $true$ **then**
  Make a move towards the goal,
else
  for each position in the neighborhood of size 3×3 **do**
    if the target position is a neighbor **then**
      Make a move towards the goal,
      neighbor:= $true$,
      break the loop,
    end if
  end for
end if
if neighbor is $false$ **then**
  Use Cuckoo Search Algorithm described in Alg. 1 to find the goal,
  Make a move towards the target found by the cuckoos,
end if
Stop.

found by the cuckoo. The use of heuristic allows to search the whole board, not to move randomly when the target is not nearby and the board is large. The main idea of the proposition is presented in Alg. 2.

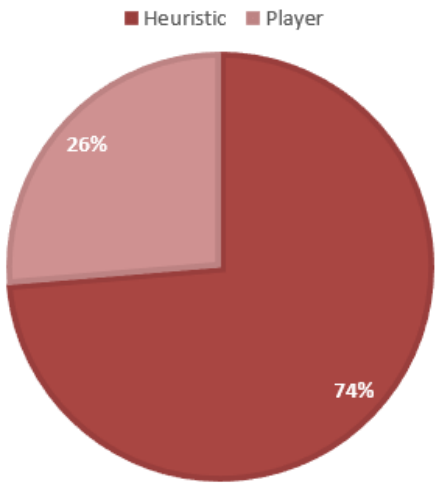


Figure 2: Average number of wins.

IV. Experiments

The proposed solution has been implemented in C# and tested in terms of the duration of the performance and number of moves needed to complete the game or to win. All obtained results were averaged in the following way

$$v_{avg} = \frac{\sum_{i=0}^{n} v_i}{n}, \tag{6}$$

where v_{avg} is the average value based on n trials (where the results are marked as v_i). Based on 100 games, the average percentage of the player's winnings and the proposed technique was calculated. The result is shown in Fig. 2. In the conducted experiments, the player and the computer algorithm participated together in the game, which directly allowed to compare the game's cope. It is easy to see that the computer still has an major advantage over the player, although it is slightly less than the intended effect of 100% winnings.

The next parameter was the time needed to win as opposed to the other player (see Fig. 3). It is clear that the computer is doing better because it always ends the game in almost twice as fast compared to the human opponent. It was also noticed that the results of the algorithm are more convergent than the player, which allows to assess the accuracy of the algorithms as sufficient for conducting subsequent experiments. Applying linear regression on the obtained results shows some stability of the proposed technique – the time of which increases very slowly in relation to the increase the number of movements. For people, the average results are quite chaotically distributed, which may be due to the randomness of the points arrangement. The mentioned randomness turns out to be insignificant for the algorithm with the average number of samples. In Fig. 4 illustrated the average movement results in relation to the number of points in the game obtained for described method. The computer needs an average of one second to make a move based on the assumption that the heuristics operates on 100 individuals and moves them during 1000 iterations. Such a result is satisfactory considering the amount of calculations.

V. Conclusion

In this paper, the use of a heuristic algorithm as a help in controlling the player on large boards and the pursuit of randomly set points has been presented. This type of solution is characterized by high randomness, albeit with a small amount of calculations. The obtained results indicate a great potential in the use of heuristics as a gaming support technique. Especially in the action of the opponent or the second player. The level of difficulty can be adapted to the number of iterations/individuals in the population.

References

[1] B. Burgh and K. Johnsen. Effects of tracking scale on user performance in virtual reality games. In *Everyday Virtual Reality (WEVR), 2017 IEEE 3rd Workshop on*, pages 1–4. IEEE, 2017.

[2] P. Gamito, J. Oliveira, C. Coelho, D. Morais, P. Lopes, J. Pacheco, R. Brito, F. Soares, N. Santos, and A. F. Barata. Cognitive training on stroke patients via virtual reality-based serious games. *Disability and rehabilitation*, 39(4):385–388, 2017.

[3] C. Gao, R. B. Hayward, and M. Müller. Move prediction using deep convolutional neural networks in hex. *IEEE Transactions on Games*, 2017.

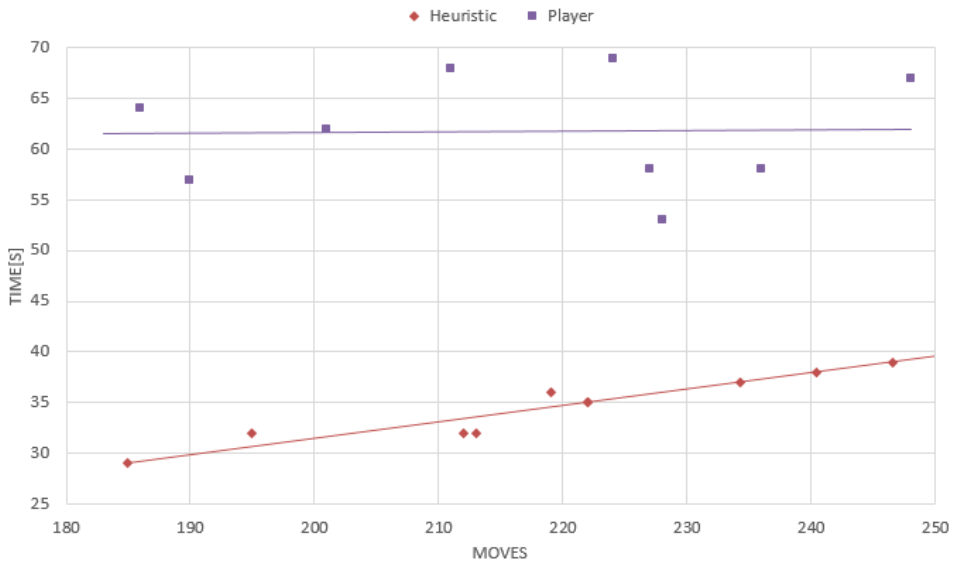


Figure 3: A graph of time dependence on the average number of movements needed to win the game for a man and the proposed algorithm.

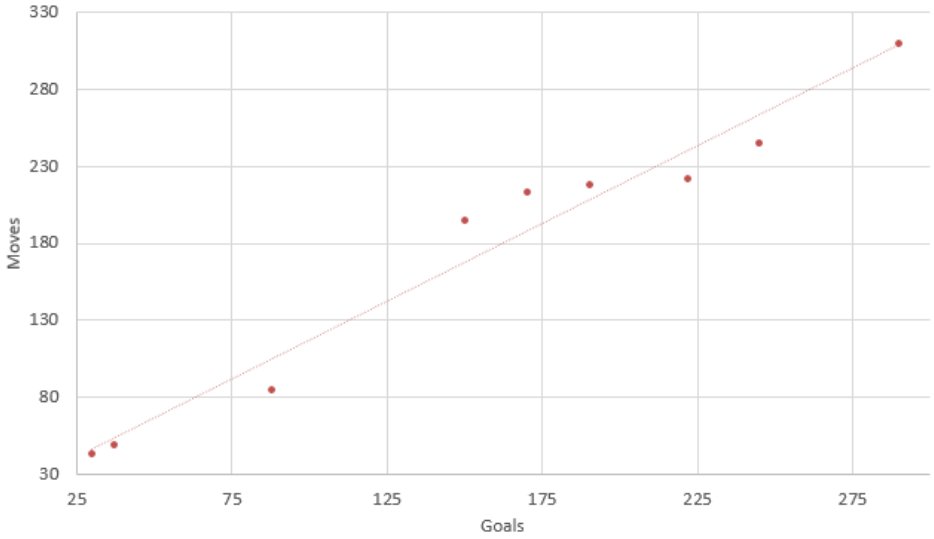


Figure 4: Graph of the number of movements in relation to time.

[4] G. Graditi, M. L. Di Silvestre, R. Gallea, and E. R. Sanseverino. Heuristic-based shiftable loads optimal management in smart micro-grids. *IEEE Transactions on Industrial Informatics*, 11(1):271–280, 2015.

[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] L. H. Lelis, W. M. Reis, et al. Procedural generation of game maps with human-in-the-loop algorithms. *IEEE Transactions on Games*, 2017.

[7] N. Liu, Q. Chen, J. Liu, X. Lu, P. Li, J. Lei, and J. Zhang. A heuristic operation strategy for commercial building microgrids containing evs and pv system. *IEEE Transactions on Industrial Electronics*, 62(4):2560–2570, 2015.

[8] M. Mohammadi, R. Tavakkoli-Moghaddam, A. Siadat, and Y. Rahimi. A game-based meta-heuristic for a fuzzy bi-objective reliable hub location problem. *Engineering Applications of Artificial Intelligence*, 50:1–19, 2016.

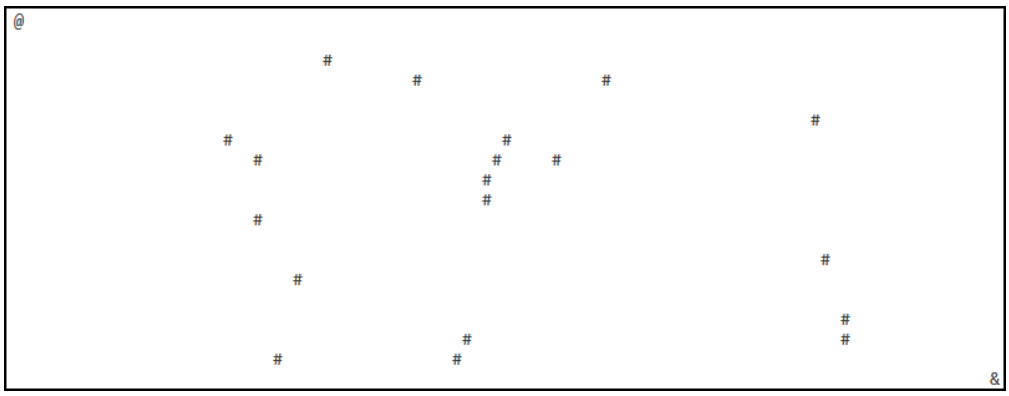
[9] M. Morosan and R. Poli. Automated game balancing in ms pacman and starcraft using evolutionary algorithms. In *European Conference on the Applications of Evolutionary Computation*, pages 377–392. Springer, 2017.

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

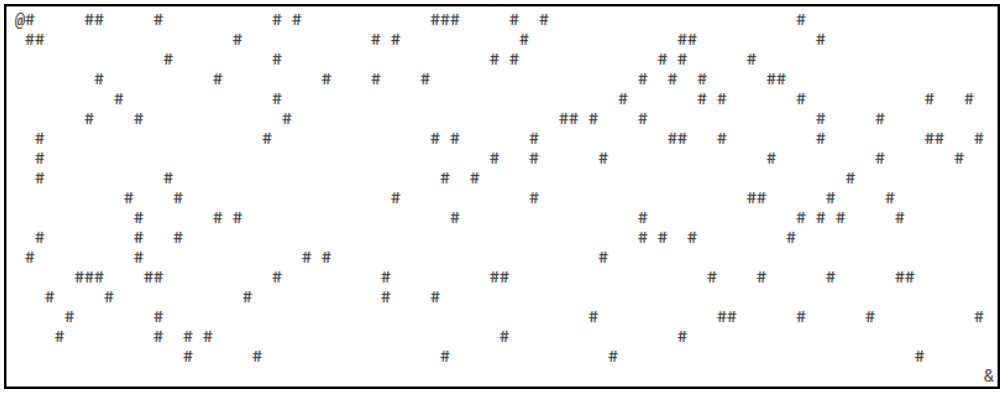
[11] D. Połap, K. Kęsik, K. Książek, and M. Woźniak. Obstacle detection as a safety alert in augmented reality models by the use of deep learning techniques. *Sensors*, 17(12):2803, 2017.

[12] D. Połap, K. Kęsik, M. Woźniak, and R. Damaševičius. Parallel technique for the metaheuristic algorithms using devoted local search and manipulating the solutions space. *Applied Sciences*, 8(2):293, 2018.

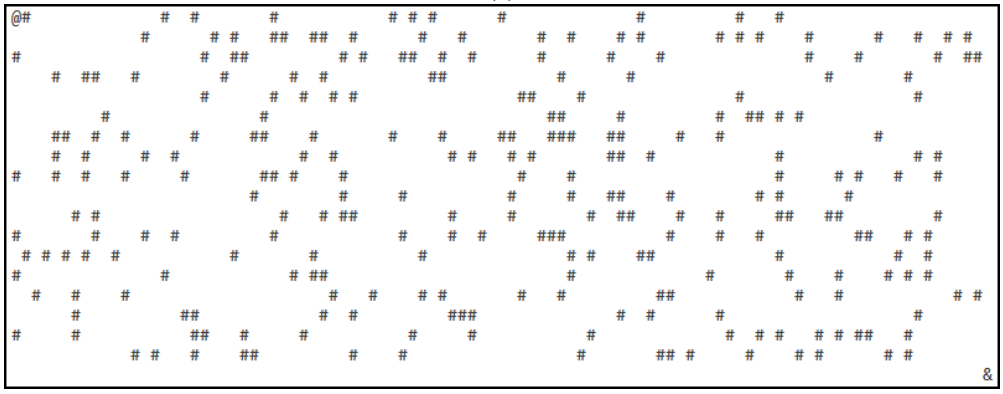
[13] D. Połap, M. Woźniak, C. Napoli, and E. Tramontana. Real-time cloud-based game management system via cuckoo search algorithm. *International Journal of Electronics and Telecommunications*, 61(4):333–338,



(a)



(b)



(c)

Figure 5: Tested boards for different amounts of goals.

2015.

[14] N. Sephton, P. I. Cowling, E. Powley, and N. H. Slaven. Heuristic move pruning in monte carlo tree search for the strategic card game lords of war. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, pages 1–7. IEEE, 2014.

[15] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[16] I. J. Sledge and J. C. Príncipe. Analysis of agent expertise in ms. pac-man using value-of-information-based policies. *IEEE Transactions on Games*, 2018.

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

[18] 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.

[19] 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.

[20] M. Woźniak, 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.

[21] X.-S. Yang and S. Deb. Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE, 2009.