VOLUME: 4 | ISSUE: 3 | 2020 | September

People vs Differential Evolution in Search of The Shortest Path

*Michal BUKÁČEK**

VSB - Technical University of Ostrava, Department of Computer Science, 17. listopadu 15, Ostrava-Poruba, Czech Republic

*Corresponding Author: Michal BUKÁČEK (Email: michal@bukacek.cz)

(Received: 27-May-2020; accepted: 8-Sep-2020; published: 30-Sep-2020)
DOI: http://dx.doi.org/10.25073/jaec.202043.292

Abstract. *The most common aim of computer game optimisation is to find the shortest path within a game or to solve a problem of a travelling salesman within a small group of cities. This article deals with the possibilities of comparing the ascertained solutions of a given problem of human intelligence and evolutionary algorithms. Human intelligence is represented by mobile game players programmed for the Android operating system, by their conduct during playing the game, and by the achieved the results. Evolutionary algorithms are represented by differential evolution. The best possible parameter estimation will be sought and compared with the player's results. The goal is to find parameter estimation of an equal or better quality in comparison with results of human players. Another task is to verify whether this setting is suitable for all mazes and whether people or the differential evolution are better at searching.*

Keywords

Travelling salesman problem, differential evolution, android, parallel algorithms.

1. Introduction

You will hardly meet a person who has never played a computer game whether on his or her phone or on a computer. People enjoy filling the blank spaces in their free time by resting while playing computer games and, what's more, they even organise tournaments in this activity. According to (Syracuse University's online MBA program, 2019 [10]), it becomes increasingly popular every year to monitor computer game players and certain areas are even becoming as popular as watching the classic sports tournaments. Playing computer games is actually discussed to be possibly added among the new sports disciplines of the eSports category (Chris Beer, 2019 [12]) at the Summer Olympic Games in 2024 which will take place in Paris (International Olympic Committee, 2019 [11]).

The most common issue of computer games as well as the general information sciences is to look for the shortest path through a field of obstacles. This problem could be generalised as searching for the shortest path through a maze where we need to find our way from point A to point B. The solution comprises various algorithms based on various principles (D. Green et al., 2017 [4]). The simplest solution is to use brute force where the area is searched through widthwise. This solution can be optimised by trying to make a widthwise search but in the assumed direction. The most sophisticated and complex solution is using evolutionary algorithms (Marrow, 2000 [5]). There are various options and the limiting factor is always the required accuracy of results and how time-demanding is the search. In this article, we will be comparing the conduct of people in a computer game with the use of the differential evolution algorithm. We will set collections of solutions of the shortest paths that can be found by computer game players and compare them with the solutions ascertained by the differential evolution - DE. The aim is to find a suitable setting for DE which could lead to a relatively easy provision of a solution of the highest quality comparable with the solution provided by real people playing the games, who can practically see the solutions in front of them within small mazes. We will verify whether the same DE setting can be applied to various kinds of maze. At the same time, we will determine whether real people or DE can provide better research outcomes (Price, Storn, Lampinen, 2005 [6]).

2. Motivation

The motivation is to compare a group of people playing the same game with an evolutionary algorithm. In the case of an evolutionary algorithm, different results can be achieved due to a change in the parameter settings. The goal is to compare human intelligence with the intelligence of an evolutionary algorithm also with regard to the size and complexity of the maze. Another goal is to verify the possibility of the existence of one DE parameter setting, that would be optimal for different types of games.

3. Experiment design

3.1. Solution design

The entire work is divided into two tasks. The first one is to programme a game application for users of mobile phones with the Android OS with a server part that should be recording the individual resulting paths and their lengths (Ahn, Dabbish, 2008 [8]). The second task is to create a solution by using DE. The settings of the input DE parameters can be changed and a solution will be sought, that would be better compared to the previous ones, and thanks to this feedback, the DE setting will be optimized (Zelinka, Vasek, Lampinen, 2001 [3]).

a) User game

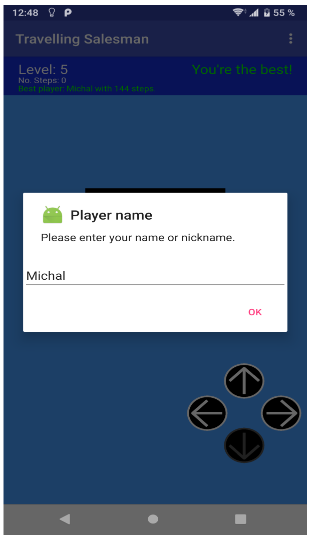
In order to have it more popularised and available, the game "Travelling Salesman" has been programmed better for mobile phones with the Android operating system, the icon of which is displayed as Fig. 1. Android developer page was used actively for creating the app (Android developer page, 2020 [9]). App "Travelling Salesman" can be installed from Google Play server here: https://play.google.com/store/apps/details?id=cz.bukacek.travellingsalesman

Fig. 1: App Icon of the app "Travelling Salesman" Source: Custom graphic processing.

Another option is to install with Google Play directly from Android device and find the app using the "Travelling Salesman" name.

Upon the installation and initiation of the game, the players enter their nickname Fig. 2. The reason why a nickname is required is the possibility of subsequent comparison of the results of a given player with others. The results of each player are recorded in the logs at the server page. Players do not see the paths of other players, but only the number of steps of the currently shortest path, compared to the parallel solution of the SOMA algorithm (Zelinka, Bukacek, 2016 [7]).

b) Description of communication in Game



12:48 55 %

Travelling Salesman

Level: 5

No. Steps: 0

Best player: Michal with 144 steps.

You're the best!

Player name

Please enter your name or nickname.

Michal

OK

Fig. 2: Running app "Travelling Salesman"-settings of Player name Source: Custom graphic processing.

The game consists of 10 levels with progressively increasing difficulty as the searched area keeps enlarging. Figure 3 compares graphically the first and the last level generated.

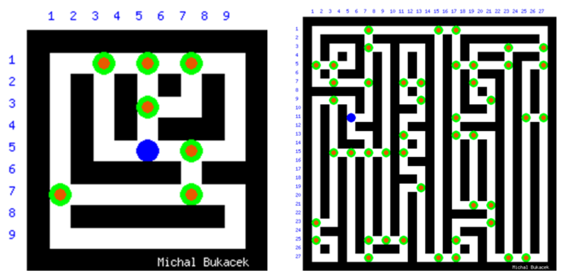
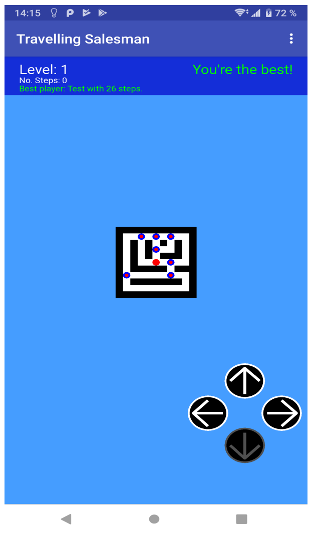


Fig. 3: Comparison-Level 1 Source: Custom graphic processing.

Upon starting the game on an Android device, Fig. 4 the device is connected to the application server which is indicated in Fig. 5 by the orange arrow, and the maps for given game tasks are downloaded along with the highest scores attained for the given levels.

The players will see this way the highest score attained which they either attempt to exceed or at least attain the same score. The course of the game is saved into the log and sent to the server



14:15 72 %

Travelling Salesman

Level: 1

No. Steps: 0

Best player: Test with 26 steps.

You're the best!

Fig. 4: Running app "Travelling Salesman–Level 1 Source: Custom graphic processing.

upon completion of the given round where it is saved. In Fig. 5 this is indicated by the blue arrow.

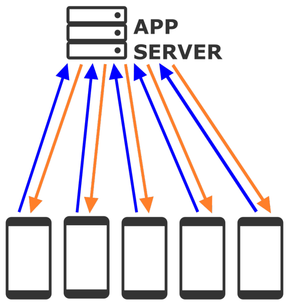


Fig. 5: Communication scheme Source: Custom graphic processing.

3.2. Algorithm of differential evolution

The description of the classic differential evolution (Price and Storn, 1995 [1]) is as follows.

The initial population P_0 with elements N from the population of possible solutions P is determined by an equal selection from space S. The value of the objective function is calculated for all points of the initial population.

a) Pseudocode DE

```
**Initial population generation** $P_0 = (x_1, x_2, ..., x_n)$
For all elements $P_0$ calculate the values of the purpose functions $F$;
**G=0**;    //Current generation
***while (G<G_no) begin*** // G_no Number of generations
    $Q_G$ =**P(G)**;
    ***i=0***;
    ***while i<N do***
     Random selection of individual $x_i$ from population $P$;
     Random selection of three individuals ($r_1$, $r_2$ and $r_3$) from population $P$;
     $u = r_1 + F(r_2 - r_3)$;
     Create individuals $y$ by crossing $u$ and $x_i$;
     **if** *Rand[0,1]*$\leq$***CR*** **then**
       ***y=u***;
     ***else***
       ***y***=$r_1$;
     ***end if***
     Calculate the value of the objective function *F(y)*;
     ***if*** $F(y) \leq F(x_i)$ **then**
     Into $Q_G$ inserts point y in place of point $x_i$;
     ***end if***
     **i=i+1**;
    ***end while***
    **P(G+1)**=$Q_G$
    ***G=G+1***
***end while***
```

A new possible solution Q_G enters the new population y, the value of which is given by the result of the binomial crossing and the value is either the result of the mutation u or the first randomly selected solution r_1 from the previous population. The intersection takes place in such a way that if the randomly selected number from the interval (0-1) is less than or equal to the intersection constant *CR* with the value from interval (0-1), the value y is formed by the mutated solution u. Otherwise by a random solution r_1. The mutated solution u arises using a basic random mutation of three randomly selected solutions r_1, r_2, r_3 from the previous generation according to the formula $u = r_1 + F(r_2 - r_3)$, where F is a mutation parameter with a value of (0-1). This new solution y obtained by mutation and crossing will enter a new population if its functional value, in our case the path length is less than or equal to the path length of the solution from the previous generation. Otherwise, it enters a new population Q_G solution x_i, where i is the order of the solution. As a result, only better or the same solutions enter new populations than in previous populations.

b) Differential Evolution in the game

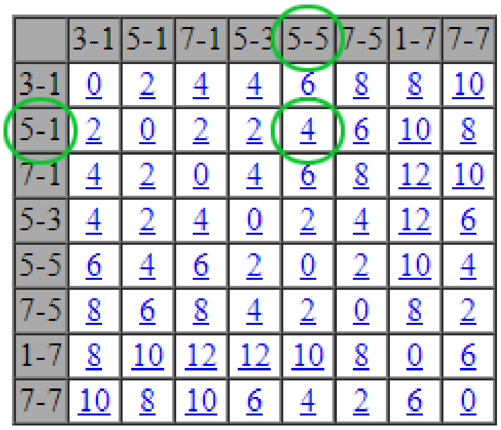
As for programming the differential evolution searching the maze, we will need to deal with the key issue, i.e. the fact that the classic algorithm of differential evolution works in a continuous environment rather than in a discrete one. Within the maze, crossroads (TSP cities) are given where the algorithm must stop, yet it must visit all of them in order to prevent the solution from penalization for taking the incorrect path. Each city follows a certain group of cities where the cities are not situated next to each other.

In order to compare the decision-making of DE and of people, we must take into account the fact that entire paths are compared with DE, which is contrary to human decision-making (Yannakakis, Togelius 2018 [2]). As for the latter, every individual attempts to go through the maze from one point to another and if there is an unvisited place on the way and it is possible to visit it, the individual decides to take this detour to optimize better the efforts made in order to take this path and to assess it better. A person considers a longer path at each crossroads of the given maze; DE does not make such considerations. The solution within the first generation is determined by DE as all paths which can be passed without considering the individual places through which it has travelled already or which it has not visited. The resulting paths are created by these partial paths between the cities regardless of any paths taken already. However, there are always better solutions to be found from which other generations produce smart individuals, i.e. new solutions, which then traverse the paths and, in the end, the last generations of DE take the correct shortest paths.

In order to combine the paths within the maze and set a resulting path through the maze every time, an entire assessment must be first made between the shortest points within the maze. In this case, the best solution is to create a database (a table) where the lengths of the paths between the individual paths can be found easily. The database is created before the DE initiation by means of an algorithm of brute force which finds the shortest distance for each city to the other cities by searching widthwise.

Figure 6 displays a table of the shortest paths within the maze. The coordinates of two specific cities are marked green with coordinates 5-5 and 5-1, and the distance between them is 4 steps. The shortest path for these two cities within the maze itself is displayed in Fig. 7.

In Fig. 8 of the same table of the shortest paths, the path of 8 steps between the cities 7-5 and 7-1 is highlighted and it is displayed in the maze of Fig. 9.



	3-1	5-1	7-1	5-3	5-5	7-5	1-7	7-7
3-1	0	2	4	4	6	8	8	10
5-1	2	0	2	2	4	6	10	8
7-1	4	2	0	4	6	8	12	10
5-3	4	2	4	0	2	4	12	6
5-5	6	4	6	2	0	2	10	4
7-5	8	6	8	4	2	0	8	2
1-7	8	10	12	12	10	8	0	6
7-7	10	8	10	6	4	2	6	0

Fig. 6: Shortest travel table (shown between cities 5.5 and 5.1) Source: Custom processing-app screenshot.

4. Results

4.1. Measured values

Humans are represented by a group of 10 people aged 7-60. Each person plays each level at least once.

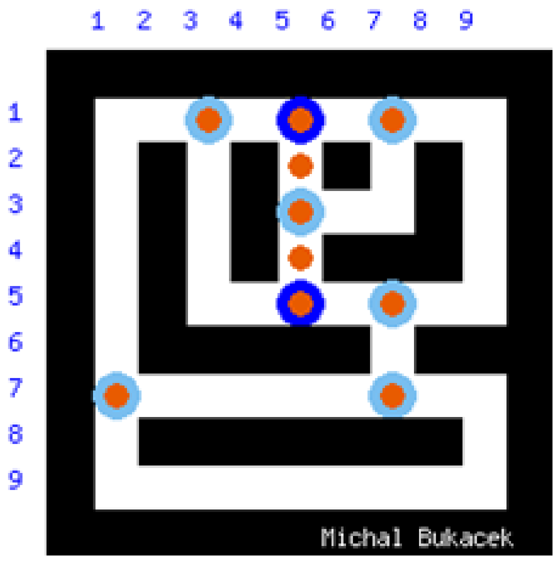
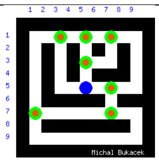
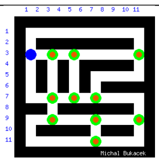
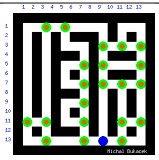
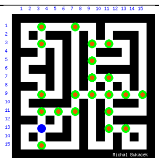
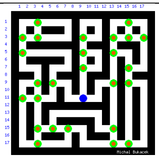
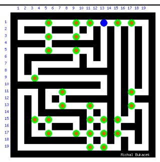
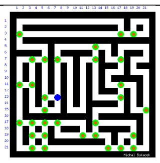
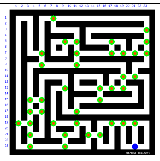
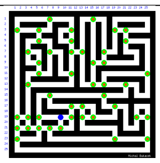
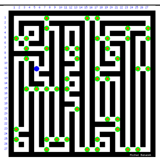


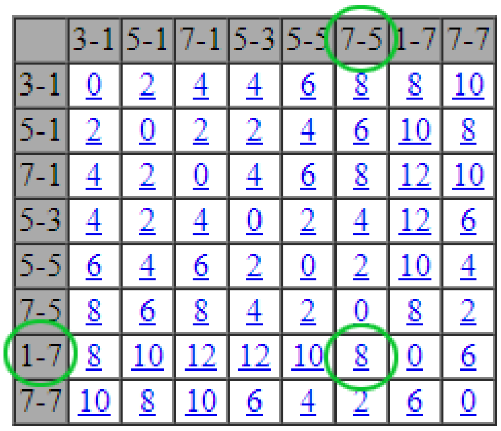
Fig. 7: Show the shortest route on the map (between cities 5.5 and 5.1) Source: Custom processing-app screenshot.

Tab. 1: Scoreboard (Source: Custom processing).

Level		People		DE	
		Path length	Number of games	Path length (F and CR)	Number of games
1.		26	20	26 F=1 CR=50 26 F=0.75 CR=75 26 F=0.5 CR=50 26 F=0.25 CR=50 26 F=0.25 CR=75	200 populations 50 generations
2.		36	20	36 F=1 CR=50 36 F=0.5 CR=50 36 F=0.25 CR=50 36 F=0.75 CR=50 36 F=0.75 CR=80	200 populations 50 generations
3.		66	20	62 F=0.25 CR=50 62 F=0.5 CR=50 62 F=0.75 CR=50 64 F=1 CR=50 66 F=0.5 CR=80	200 populations 50 generations
4.		91	15	74 F=0.5 CR=50 74 F=0.75 CR=50 74 F=0.5 CR=80 76 F=0.25 CR=50 78 F=1 CR=50	200 populations 50 generations
5.		120	15	116 F=1 CR=75 118 F=1 CR=90 128 F=1 CR=50 124 F=0.5 CR=75 130 F=1 CR=50	200 populations 50 generations

Tab. 1: Scoreboard (Source: Custom processing).

Level		People		DE	
		Path length	Number of games	Path length (F and CR)	Number of games
6.		121	15	110 F=0.75 CR=50 114 F=0.5 CR=50 114 F=0.5 CR=75 118 F=0.25 CR=75 118 F=0.75 CR=80	200 populations 50 generations
7.		158	15	158 F=0.5 CR=80 178 F=0.5 CR=90 178 F=0.5 CR=30 186 F=0.5 CR=75 190 F=1 CR=50	200 populations 50 generations
8.		154	15	158 F=0.5 CR=80 178 F=0.5 CR=90 178 F=0.5 CR=30 186 F=0.5 CR=75 190 F=1 CR=50	200 populations 50 generations
9.		234	15	242 F=0.2 CR=80 244 F=0.5 CR=80 248 F=1 CR=50 256 F=0.75 CR=80 272 F=1 CR=50	200 populations 50 generations
10.		254	15	308 F=0.2 CR 80 326 F=0.2 CR 80 336 F=0.25 CR 80 348 F=0.5 CR 80 362 F=1 CR 50	200 populations 50 generations



	3-1	5-1	7-1	5-3	5-5	7-5	1-7	7-7
3-1	0	2	4	4	6	8	8	10
5-1	2	0	2	2	4	6	10	8
7-1	4	2	0	4	6	8	12	10
5-3	4	2	4	0	2	4	12	6
5-5	6	4	6	2	0	2	10	4
7-5	8	6	8	4	2	0	8	2
1-7	8	10	12	12	10	8	0	6
7-7	10	8	10	6	4	2	6	0

Fig. 8: Shortest travel table (shown between cities 7.5 and 7.1) Source: Custom processing–app screenshot.

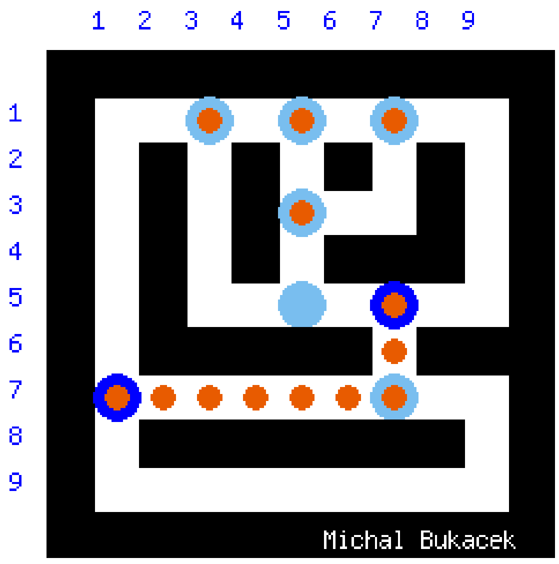


Fig. 9: Show the shortest route on the map (between cities 7.5 and 7.1) Source: Custom processing–app screenshot.

4.2. Findings

a) Conduct of humans vs. DE

The result of comparison of human conduct on the same local path with the DE conduct is also worth mentioning. DE is set in such a way that the final path length, containing the already visited places, is penalised proportionally to the number of such repeated visits. For instance, if we are supposed to get from point A to point B, as displayed in Figs. 10 or 11, and we wish to visit points 1, 2 and 3 on the way as well, we have two options to do that: the path following Figs. 10 or 11. DE, affected by the traversing and assessment, will reach the best path following Fig. 10, as the path following Fig. 11 is penalised for visiting city 2 repeatedly. As opposed to that, humans tend to select the variant following Fig. 11. In fact, it does not matter which variant is selected, as both paths are of the same length, both figures display a part from city 1 to city 2 of the same length and two identical parts between cities 2 and 3.

b) Blind alleys

Despite the fact that the generated graphs do not contain any blind corridors and the crossroads always have at least 3 paths within the maze, there are blinds alleys contained. As displayed in Fig. 12, a blind alley is such a part of the maze with only one path leading to it and this path must also be used to return from this area. A blind alley contains several cities. If there are more such alleys within the maze, it is more difficult for DE to combine this path; a locally blind alley is passed correctly by DE evolution.

A blind alley can be created even if the area can be accessed from two ends and one of these ends was used for a subsidiary local search. If we use this path again, the resulting path is prolonged by it. If we enter the area through a path we have not used before, when exiting the area, we will need to take one of the paths we have already used in both cases. It is best to keep such an area till the end of searching and enter the area through a path we have not used before, or more precisely, through a city we have not visited before.

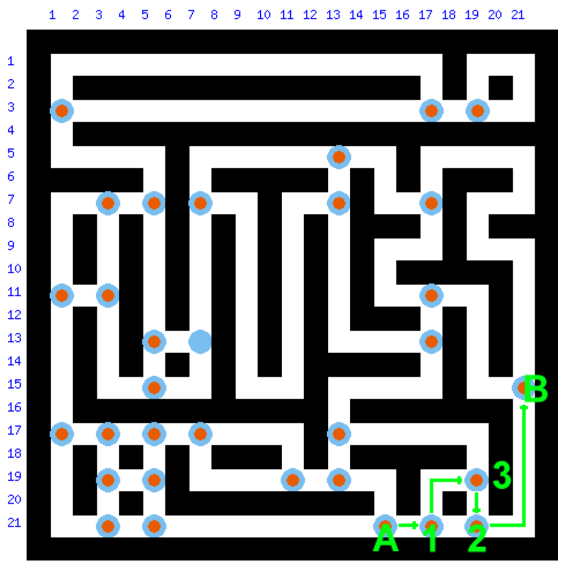


Fig. 10: Variant A) Local path from point A to point B Source: Custom processing – app screenshot.

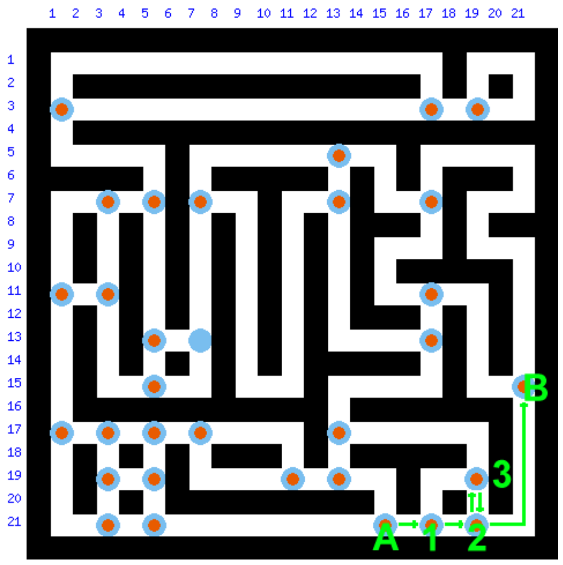


Fig. 11: Variant B) Local path from point A to point B – with repetition (return to an already visited place.) Source: Custom processing – app screenshot

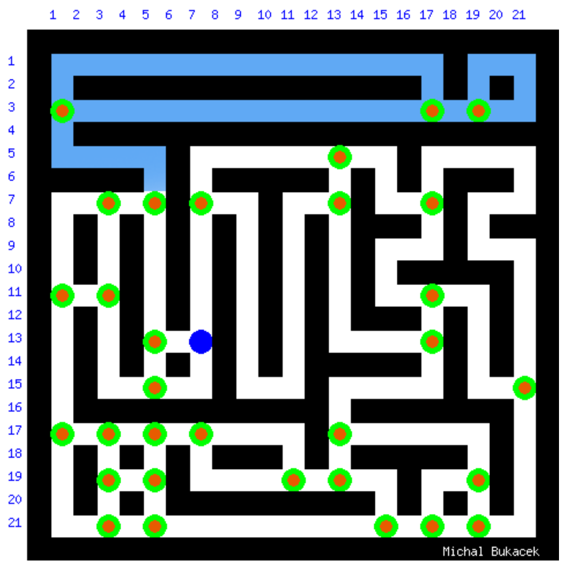


Fig. 12: Blind alley highlighted blue in the 7th level of the game. Source: Custom processing – app screenshot.

5. Conclusions

We may say that the first human solution is not bad at all, because people use their common sense when being at a crossroads and they do not select the paths between the cities randomly, as displayed in Figs. 13 and 14. The given area can be passed in different ways. A thinking person will select subconsciously the shortest local path and solve the passage through the entire map as a compound of the shortest paths.

From the comparisons, it emerges that the humans mostly use an algorithm to find the shortest connection which they attempt to pass in such an order that will make the total path as short as possible. The green intuitive human solution is rational, correct and fast with small mazes.

Our gaming levels can be divided into three types depending on their size. Low levels (small mazes of level 1 and 2), medium levels (3-7) and high levels (8-10). As for the low levels, the results of people and DE are identical, both categories found short paths within the maze of the same length. DE evolution found more combination shapes with the same length, but needs

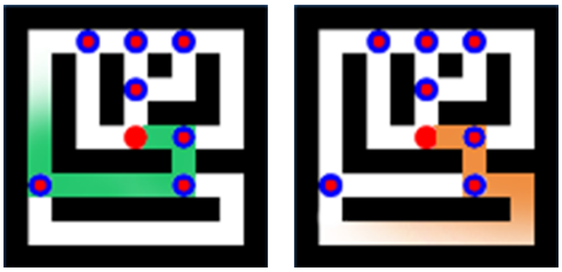


Fig. 13: Green local path is shorter than the orange one Source: Custom processing-app screenshot.

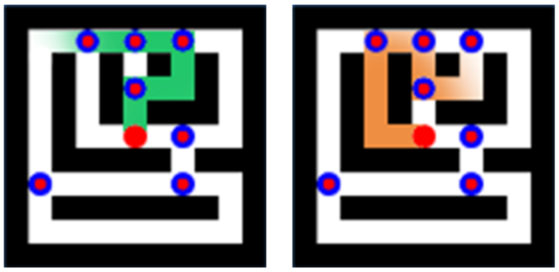


Fig. 14: Green local path is shorter than the orange one Source: Custom processing-app screenshot.

a relatively large number of evolution cycles, a large population to offer a verifiable and the shortest path. We may say that people come to the same or even better conclusions than DE with less effort. As for the medium levels, the results of DE are better than the ones of people. We can see that the maps contain some blind alleys with only one entrance, but there are not as many of those, so the DE with the same settings as for the low levels did well and found the best combination. DE is better at these levels. As opposed to that, people are obvious winners when passing through large mazes. There are more blind alleys and it is necessary to combine them better. DE starts connecting the blind alleys which are not always neighbouring, but sometimes in a leap through the entire game plan and so often visited some places repeatedly. It emerges from the measured values and observing the algorithm that it is always better to set a higher CR value (0-100), i.e. approximately 80%, as it leads to a higher traversal number and parameter F (0-1) within 0.5-1. Thanks to these settings, the algorithm will try to pass more paths and more combinations of cities. People are able to keep in mind the entire path and to orient themselves, so when there is a place, they have not visited next to the place they have just passed, they visit this place as well.

There is not a single common DE setting for all kinds of mazes, as it depends on the size of the maze, the number of cities, the number of blind alleys and on the alleys neighbouring to the given field. Furthermore, the element of a chance plays a part in DE at all times. A good solution can be created even in the first generations and all solutions derived therefrom may not be necessarily as good.

In the next work, it would be good to add the DE optimization so that when generating random local roads, another city already passed on the given road would be recorded and taken into account. The DE algorithm does not take into account the situation when, for example, we have three cities marked A, B, C, which lies on the line [AC], where between points A and C lies point B. Then of course the shortest path is given by the sequence A, B, C, but the algorithm randomly generates as one of the first solutions the path A, C, B, while in order to get from A to C, he had to visit city B, but as if he was not there, he returns to city B after passing city C. This path is of course longer than the variant A, B, C, but it is cultivated in the next generations due to mutations and crosses to the correct result.

Acknowledgments

The following grants are acknowledged for the financial support provided for this research: Grant of SGS No. SP2020/78, VSB Technical University of Ostrava.

References

[1] Rainer, S., & Price, K. (1995). Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, vol. 3. Berkeley: ICSI.

[2] Yannakakis, G. N., & Togelius, J. (2018). Artificial intelligence and games, vol. 2.

New York: Springer.

[3] Ivan, Z., Vasek, V., & Lampinen, J. (2001). New Algorithms of Global Optimization. Automatizace (Journal of Automatization, Czech Ed.), 10 (1), 628-634.

[4] Green, D., Aleti, A., & Garcia, J. (2017). The nature of nature: Why nature-inspired algorithms work. In Nature-Inspired Computing and Optimization, Springer, Cham, 1-27.

[5] Marrow, P. (2000). Nature-inspired computing technology and applications. BT Technology Journal, 18(4), 13-23.

[6] Price, K., Storn, R. M., & Lampinen, J. A. (2006). Differential evolution: a practical approach to global optimization. Springer Science & Business Media.

[7] Zelinka, I., & Bukacek, M. (2016). SOMA swarm algorithm in computer games. In International Conference on Artificial Intelligence and Soft Computing, Springer, Cham, 395-406.

[8] Von Ahn, L., & Dabbish, L. (2008). Designing games with a purpose. Communications of the ACM, 51(8), 58-67.

[9] Android developer page (2020) https://developer.android.com/

[10] Syracuse University's online MBA program (2019) https://onlinebusiness.syr.edu/blog/esports-to-compete-with-traditional-sports/

[11] Intenational Olympic Committee (2019) https://www.olympic.org/news/declaration-of-the-8th-olympic-summit

[12] Chris Beer (2019). 2019 in Review: Esports Comes of Age. In Globalwebindex https://blog.globalwebindex.com/trends/2019-review-esports/

About Authors

Michal BUKÁČEK He works as a software developer. He likes to program apps for Android OS, likes to use swarm algorithms. In free time solves a TSP problem.