Gamesourcing: Perspectives and Implementations

Ivan Zelinka, Martin Němec and Roman Šenkeřík

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71703

Abstract

This chapter discusses game-based methods of problem solution and data processing, analysis, and information mining. Attention is mainly focused on swarm algorithm and principles used in the task of complex problem solving using computer games. We intensively discuss the interdisciplinary intersection between swarm systems dynamics and computer science, including a variety of data sources and examples. Possibilities of the new approach based on swarm algorithm used in a game or using principles of swarm algorithms to solve the problem are demonstrated here. More precisely, this chapter discusses modern methods of calculation and crowd use, so-called gamesourcing, i.e., game-driven crowdsourcing, from various points of view such as history, motivation, or paradigm and presents several examples of contemporary projects of this kind. Ideas, results, and methodologies reported and mentioned here are based on our previous results and experiments that are fully reported here for detailed study in the case of reader's interest. Therefore, this chapter is an overview survey of our research.

Keywords: swarm algorithm, swarm intelligence, dynamics, data, analysis, gamesourcing, game

1. Crowdsourcing and gamesourcing: a brief overview

Gamesourcing is a new term created by combining a pair of English word games and crowdsourcing. The game is a game, but the explanation of crowdsourcing is somewhat more extensive. It is a method where a job (often very difficult for a computer but relatively easy for a person) is divided among a large number of people and depending on the nature of the assignment, either by common forces or by each one of them. In the first case, all participants will cooperate, with only the best of the other being selected. By joining with the game, we offer a means of making crowdsourcing on a slightly different basis. That is, players do not even *need to know that they are doing any crowdsourcing at all*. While they still enjoy it.

The history and development of crowdsourcing over time and the gradual emergence of gamesourcing are quite new and young discipline. In the past decade, the new phenomenon—crowdsourcing—or the use of a large group of people to engage in some creative activity—is beginning to spread through the Internet, e.g., drawing ideas, ideas, content, or contributions, whether financial or professional. Usage can be virtually any: education (Wikipedia), research, health, transport (accident reporting), marketing and advertising, donation, volunteering, etc. However, even in the political sphere, the wisdom of the crowd has already found its application. After 2008, when Iceland was bankrupt due to the economic crisis, the creation of a new constitution took place, and its proposal was made available online for a wide-ranging discussion. By debating and commenting through social networks, Icelandic citizens could influence and correct the final form and text of the constitutional charter [1, 2].

The term crowdsourcing was first used in 2006 [3], which could lead (due to the Internet boom) to the erroneous assumption that it is exclusively an online affair. However, the process of using the masses to cooperate on a task has been working since time immemorial, starting with prehistory [4]. It was then only the adoption of advice and recommendations by the Chieftain from his subordinates, but in some way, it was some collective involvement in achieving the goal (e.g., how to survive the winter). So, a certain form of crowdsourcing existed for tens of thousands of years before the emergence of the all-knowing medium, so basically it is not a new thing.

Crowdsourcing has been used for some time already in architecture. In 1955, Prime Minister of the State of New South Wales, Joseph Cahill, announced a £ 5000 competition to design an opera house on the Gulf Coast in Sydney. A total of 233 proposals were sent from 32 different countries. The winning design of the Danish architect Jør Utzon was the beginning of one of the most innovative and innovative buildings of the present era. This type of architectural competitions continues to be widely used [5].

In 2006, Jeff Howe first used the term "crowdsourcing" [6] in his article and had since become more and more popular. The first purely crowdsourcing projects such as DesignCrowd (crowdsourcing of graphic designs, logos, websites, etc.) or Digg (crowdsourcing aggregator of news) were launched. Also, crowdfunding has begun to develop, contributing more individuals with smaller amounts to the target amount to fund an interesting project or product.

With the growing trend in the gaming industry, its potential for crowdsourcing has been properly estimated. The author professor Luis von Ahn (Carnegie Mellon University) launched it [7] under the auspices of Google. It was about describing images in a fun and catchy form.

The number of similar acts has been slowly rising since then, and awareness of the gameplay has begun to be gained among the wider public. For example, in 2016 the Sea Hero Quest [8] was also commercialized on domestic TV stations. It is a mobile device-based game to aid in the study of dementia. There has not been a year since the release, and the authors have been honored with excellent results.

Crowdsourcing is so slowly becoming a part of everyone's life slowly, without being aware of it. Its special forms are called gamesourcing, where the means of collective cooperation is playing games. It seems that gamesourcing is on the rise. It has already proved to be an excellent tool in several cases. However, its potential has not been fully utilized by far.

2. Swarm intelligence in computation

Swarm algorithms (called also swarm intelligence) have been very popular lately due to the features that are characteristic of this class. However, before we begin to discuss their application, it is appropriate to mention selected algorithms in this area, which are (or can be) further elaborated in research reported in this chapter. They are:

- Ant colony optimization (ACO) [9],
- Artificial immune system [10],
- Self-organizing migrating algorithm (SOMA) [11],
- Memetic algorithms [12],
- Grey wolf [13],
- Particle swarm [14],
- Artificial bee colony [15], and
- Fire fly [16],

among the others. These algorithms can be used to solve very different problems. Many more versions and strategies than it is mentioned above exist there. Given that their description would go beyond the scope of this text, it is only necessary to refer the applicant to the relevant literature. The SOMA [11] and ACO [9] have been used for our experiments, reported further.

Firstly discussed is the intro to crowdsourcing and gamesourcing. Then we use SOMA as a swarm intelligence in two computer games as a prelude to real gamesourcing that is represented by our experiments in the game called Labyrinth. This game has compared ACO against the human player crowd in the maze, which is equivalent to traveling salesman problem (TSP). At the end, we introduce a new version of Labyrinth for 3D that is dedicated to the new experiments of this kind.

3. Case studies

Here, in a few examples, our income on the field of swarm intelligence principles in computer games and gamesourcing is mentioned. Some of them were already published in conference and journals; so for more details, it is recommended to follow the references in the text.

First of all, we will discuss the use of swarm algorithm SOMA in Tic-Tac-Toe game [17], which is released on Google Play store [7], and anyone can easily download and play this game. Swarm intelligence is a counter player against human one in this game.

The second one is utilizing SOMA algorithm in the famous game StarCraft: Brood War. Swarm intelligence was controlling a combat unit strategy of movement in extraterrestrials wars, as explained in [17, 18].

The real gamesourcing is in the next section, where a human crowd plays a game Labyrinth, that is equivalent to a traveling salesman problem (TSP) problem [19, 20]. The results are then compared with ACO algorithm applied on the same problem. At the end is introduced our new platform for gamesourcing Labyrinth 3D for more extensive experiments.

3.1. SOMA Tic-Tac-Toe

This application is focused on swarm intelligence techniques and their practical use in a computer game. The aim is to show how a game player (based on swarm algorithms, in this case, SOMA) can replace a man in computer games. This provides an opportunity for effective, coordinated movement in the game fitness landscape. The implementation of our experiments uses classic techniques of artificial intelligence environments, as well as unconventional techniques, such as swarm intelligence. Research reported here has shown the potential benefit of evolutionary computation in the field of strategy games.

SOMA is a stochastic optimization algorithm that is modeled based on the social behavior of competitive-cooperating individuals [11]. It was chosen because it has been proved that this algorithm can converge usually toward the global optimum of the given problem [11]. SOMA works on a population that consists of the possible candidate solutions in iterations. They are called migration loops. The population is initialized, as usual for this kind of algorithms, by uniform random distribution, i.e., over the search space at the beginning of the evolutionary search process. In each migration loop, the population is evaluated, and the individual with the lowest cost value (the best fitness) becomes the leader. Apart from the leader, in one migration loop, all individuals will traverse the space of possible solutions in the direction of the leader. Mutation, the random perturbation of individuals, is an important operation for SOMA. It ensures diversity among all the individuals, and it also provides a means to restore lost information in a population. The mutation process is different in SOMA as compared with other evolutionary algorithms. A parameter, called PRT, is used to achieve mutations and perturbations. This parameter has in SOMA the same effect as mutation for other EAs. The PRT vector defines the final movement of an active individual in the search space. It is a randomly generated binary perturbation vector, which controls the allowed dimensions for an active individual. If an element of the perturbation vector is set to zero, then the individual is not allowed to change its position in the corresponding dimension. An individual will travel over a certain distance (PathLength) toward the leader in finite steps on PathLength. If the PathLength is chosen to be greater than one, then the individual overshoot the leader. This path is perturbed randomly. The main principle of SOMA, i.e., traveling of an individual over space of possible solutions, is depicted in **Figure 1**. The initial parameter sets of SOMA for our experiments can be set in the Android program, as demonstrated in **Figures 2** and **3**. For more details about SOMA, see [11]. SOMA can be considered as a member of swarm intelligence class algorithms.

It is true that all experiments are influenced by the player itself and algorithm setting. The screenshot from one game is depicted in **Figures 4** and **5**. If SOMA parameters would be set differently in a nonoptimal way, then, of course, the performance would be different. Thus, the performance of the SOMA depends on two factors: human skills and SOMA setting. For more details and game description, it is recommended to refer [17] and game on Google Play store [7].

http://dx.doi.org/10.5772/intechopen.71703

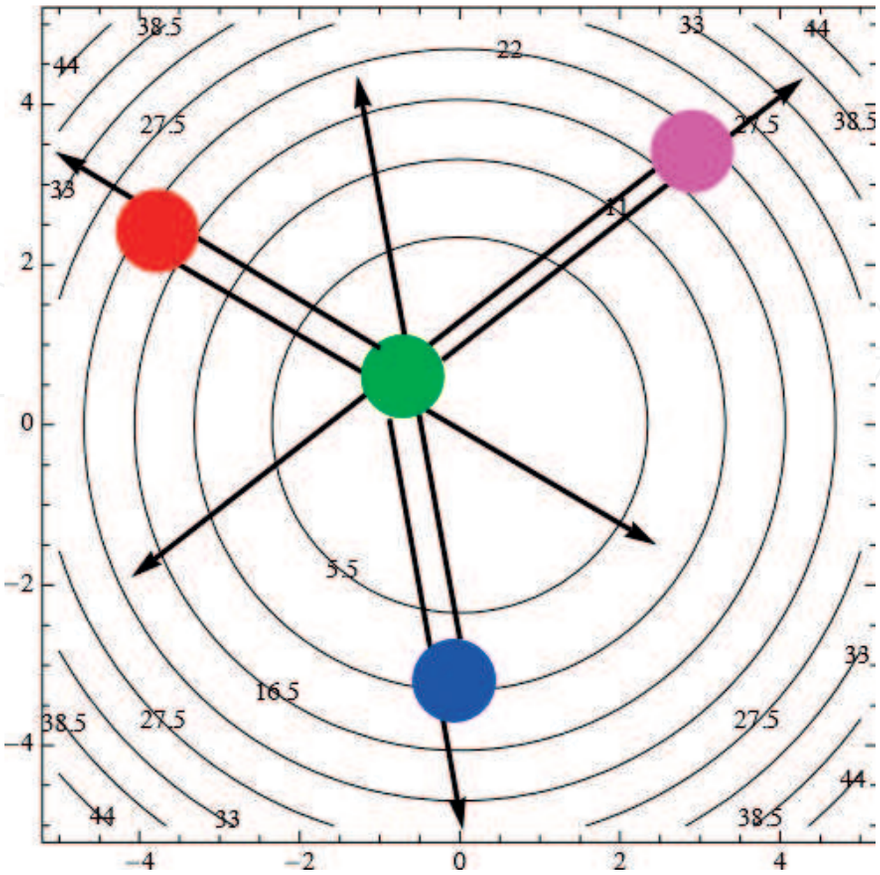

Figure 1. The SOMA principle.

3.2. SOMA StarCraft: Brood War

StarCraft: Brood War is a well-known real-time strategy game in which swarm intelligence was used again. In [17, 18], SOMA implementation is focused on applications and practical utilization in combat units control. The goal of the war (and its results are partially reported here) is to implement computer player replacing a human in a real-time strategy StarCraft: Brood War. The computer player behavior is provided by the decision-making tree together with SOMA and is used to remote movement of combat units. A particular implementation of SOMA algorithm provides an opportunity for efficient, coordinated movement of combat units over the map. The work and its results reported in [17, 18] has shown great benefit of evolutionary techniques in the field of real-time strategy games.

Similar to other strategic games, it is also crucial to provide acceptable amount of resource materials for creating an army to win. Materials are located on several places as well as at the starting position. The game allows a choice of three races: the **Protoss** are strong extraterrestrial beings with an expensive units. The **Zergs** seems to be primitive and overgrown kind of insects. Their units have a low price and are weaker than Protoss units. The last race is **Terrans**, a race of human-like being, a compromise between mentioned races that have balanced prices and efficiency of theirs combat units. For the use of the SOMA swarm intelligence, a *Zergs race has been chosen*. The techniques such as decision-making tree or unconventional techniques in the

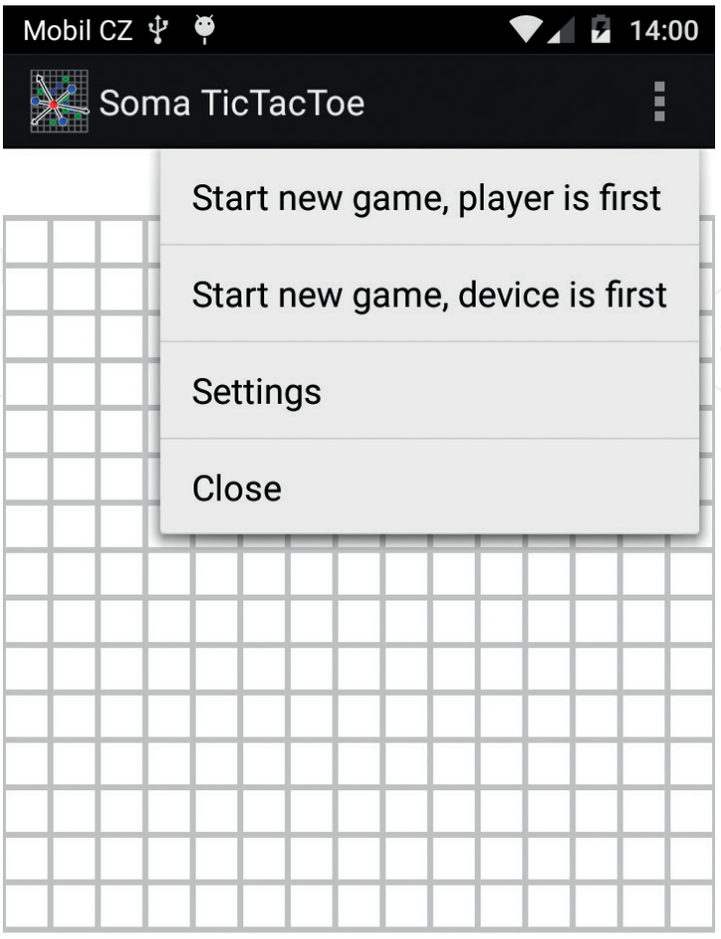

Figure 2. SOMA setting in Tic-Tac-Toe.

form of evolutionary algorithms (SOMA algorithm) have been used. The algorithm controlled a movement of the combat units on the landscape. The game also further do not allow random placement of a population on the landscape. Starting positions of the units are on purpose near the structures used for production of the units.

In the case of the incorrect objective function, we would be challenging problems with the speed of finding the enemy. The contact with the enemy would stabilize the process of searching so that searching units would be attracted to this place. It can be solved by the right setting of the parameters of SOMA algorithm and modified implementation of the objective function. Other problems could not be solved this way. Parameter PathLength [11] was set to a value 1 for the units to stay on the position of the leader. Experiments have shown that various numerical settings of the parameter Step were not so beneficial, and small numerical

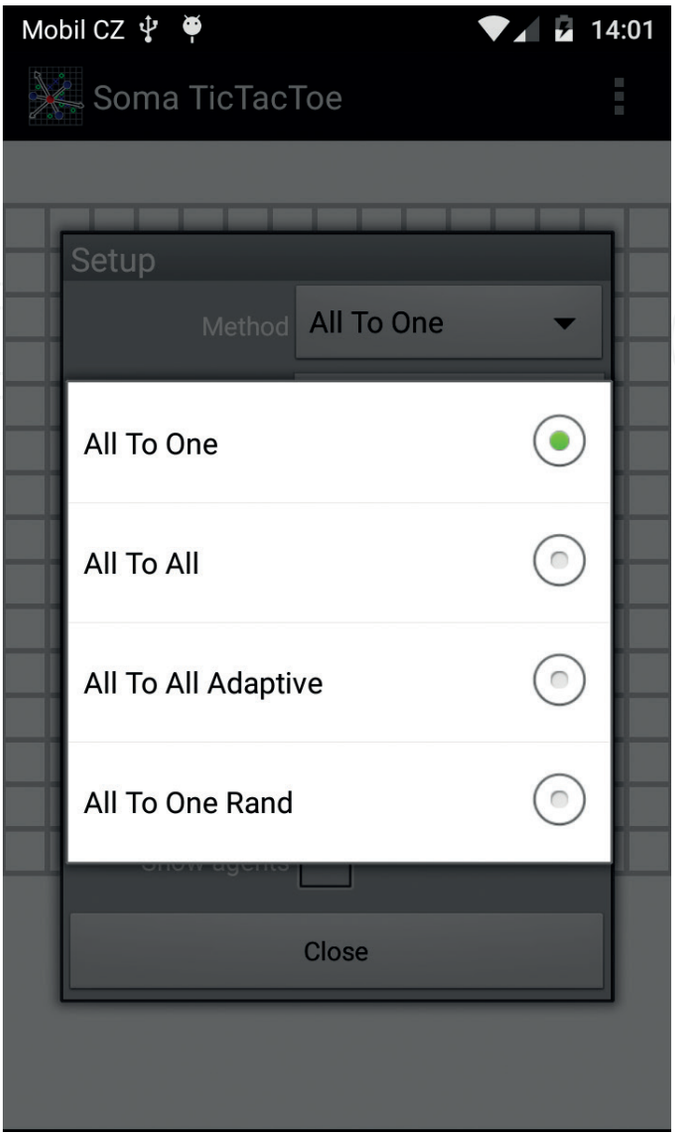

Figure 3. SOMA setting in Tic-Tac-Toe—strategy setting.

values had no significant impact on the combat unit performance. The unit was particularly searching by Step long as its own; it was set to Step = 0.5. Parameter PRT was selected near 1 so that the units would not deviate from each other so much, and at least some random searching came to pass. PRT was set to value 0.8. Dimension was set to 2 since we were searching in two-dimensional map given by coordinates x and y. PopSize is an adjustable parameter. Each combat unit was added to the population until its death. All the SOMA stop criterion, i.e., Migrations & MinDiv were ignored—SOMA was working through whole game until any unit was present. The positions of the enemy units were a part of the fitness function as well as important locations on the map. SOMA algorithm thus works on the dynamic function in order to maximize success of combat units to win.

The largest value was associated with the enemy units. The biggest fitness, the more enemies are present there. The closer position of the individual to enemy position guarantee the bigger fitness. Starting position itself was set to 0. The significant problem in this implementation was

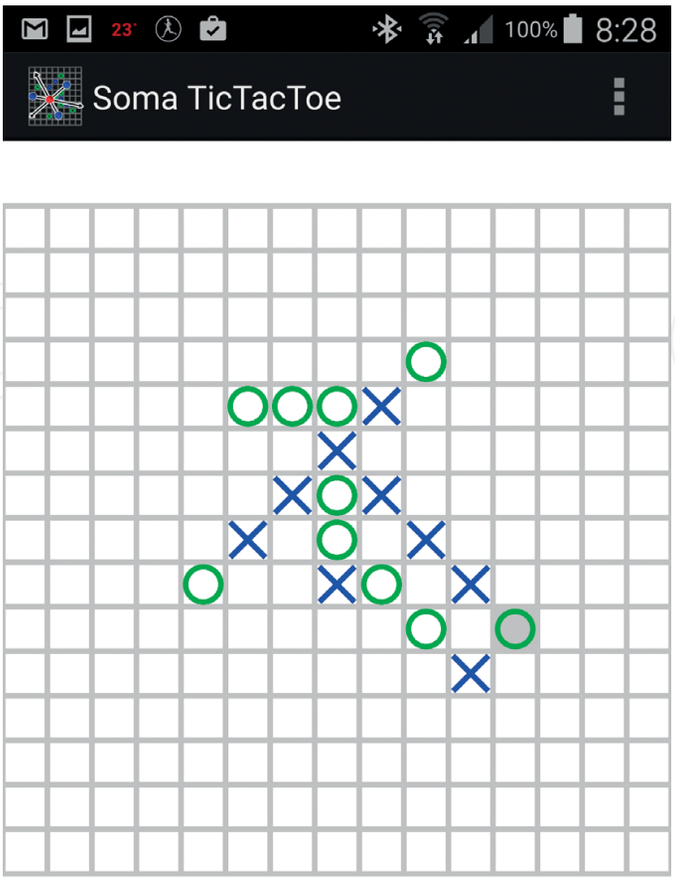

Figure 4. Tic-Tac-Toe screenshot. The game in the process: blue cross—human player, green circles—SOMA.

a nonrandom placement of the population as well as the inefficiency of the random search on the landscape. The position of the headquarters of the enemy was known. It is one of the starting positions. These problems were solved by the division of the population into two subpopulations: static and dynamic. A dynamic subpopulation consists of a classical individual who yields to recombination and mutation. It is migrating individual in the terminology of SOMA algorithm. Static individuals, on the other hand, are not migrating. They are located in the important positions on the map like locations with materials and starting positions. They can also be placed on the position of each existing structure. Their role is important because the static individual, at the starting position of the enemy, is calculated the biggest fitness. Moreover, they attract all of the units albeit from only one position. If they encounter the enemy on their route, this kind of individual is chosen as a leader [11] (regarding the highest value of the enemy's hit points). If any of our structures are attacked by the enemy, this structure is activated to be a leader (there is a static

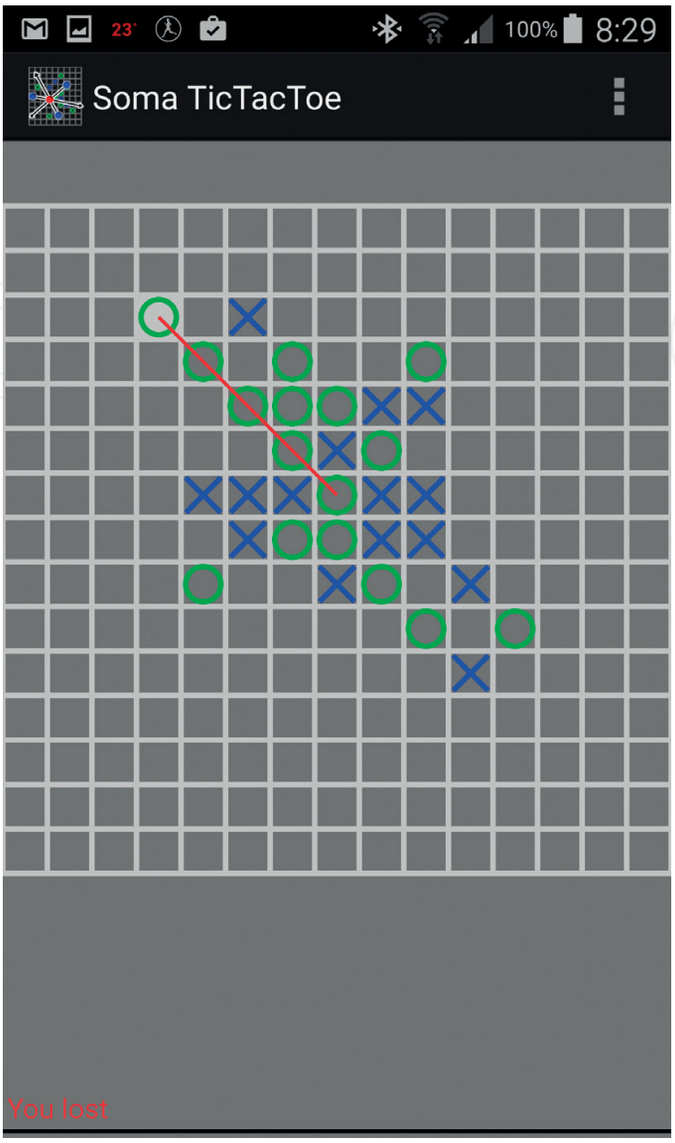

Figure 5. Game over.

individual). In this case, the leader will attract friendly combat units. These individuals also create a kind of alarm system. The screenshot from many battle situations is shown in **Figure 6**, and the results of the fights are reported in [18]. All experiments were done with one human player against four SOMA strategies (AllToOne, AllToOnenRand, AllToAll, and AllToAllAdaptive) in 100 experiments in total (i.e., 25 experiments per strategy).

3.3. Traveling salesman, gamesourcing versus swarm intelligence

In this experiment, real gamesourcing experiment with real human players in the game Labyrinth that is equivalent of TSP is reported. Labyrinth is a web game embedded in an existing game portal environment [21]. The problem of a traveling salesman is transformed into a maze, and players, by collectively crafting, mimic the behavior of the ant algorithm [9]—the swarm technique that is currently, widely, and successfully applied to this problem. The aim is

Figure 6. Zerg units are driven by SOMA in combat action in [18].

to collect data about players' moves, from which it will then be possible to read their strategies and tactics that could lead to the improvement of ant algorithms.

For gamesourcing, it is crucial to be able to combine the problem we want to address with the game. There must be very close ties between them. It is necessary for the game by its execution to lead the player exactly to what we want to achieve, i.e., to solve the problem, so that they will not be able to understand the problem.

In this case, we have decided to solve the problem of a traveling salesman and get inspired by ant colony optimization (ACO) algorithms. Individual ants are represented by humans. This could help to find new variants and adjustments to these algorithms, thus making them more efficient.

Traveling salesman problem, used in a game, is a simple idea. The salesman needed to make a trip to certain cities (each to visit only once) and to go back home. For high efficiency, the trip has to be planned to be as short as possible. This is a NP-heavy problem [19]. That is, all NP problems are convertible to it in polynomial time, and it is not in the NP itself.

If we describe this role as a graph, then its vertices (nodes) represent the starting point and the places that traveling salesman has to visit. Each of the two vertices is linked to the graph by an edge, which is evaluated by the distance that is needed to offset the trader from the place represented by the first node to the point representing the second node. This is a nonoriented complete graph. The aim is then to find the shortest Hamiltonian circle (cycle).

All the possible Hamiltonian circles in the graph (at the starting point does not matter) depends on the number of cities, so the number of possibilities needed for exploration is growing very (exponentially) fast. There are several exact algorithms (Brute-force, Branch & Bound, Cutting Plane) that can solve the TSP but only in 40–80 cities [22] in a reasonable time. For this reason, heuristic algorithms have found abundant use, which does not guarantee the correctness of the result, but in most cases, the solution found is of sufficient quality and is achieved in an acceptable time. One of those algorithms is the ant colony optimization.

Optimization using ant colonies is a group of algorithms inspired by the real behavior of ants in nature. Every ant leaves pheromones in their path, which then attract others. The stronger the pheromone trace is, the more ants follow it. The heavily used paths keep constantly high levels of pheromones, while the pheromones finally evaporate from the unused ones.

Thus, in general, there is an iteration of the ACO algorithm. Various versions of ant algorithms then have different ways of making decision rules and different rules for pheromones.

3.3.1. Proposal

Due to the mission and focus of the experiment, its appearance is obvious. For the sake of easy access and the ability to play as many players as possible, it must essentially be playable on the Internet. At the same time, it is quite intuitive that the best illustration of a TSP is a maze. The graph nodes will be represented as rooms (or city in TSP) and edges as corridors. The weight of each edge can then be specified as some form of obstacle. The Labyrinth crawling is also popular since time immemorial, and according to the game developer's focus, these games continue to thrill our heads.

3.3.2. Used platform

There are plenty of games that try to make the most of the human brain's potential to achieve something useful. People still surpass computers in many tasks, such as recognizing objects such as galaxies (Galaxy Zoo). Meanwhile, none of these projects have even reached popularity, say, Minecraft, which has already published 107.86 million copies since its release (2009), and this has grown by another 53,000 [23] each day. With such a huge shot, gamesourcing would be interesting things, but unfortunately, such a popular crowdsourcing game did not make it so popular.

So, to design the experiment, it was approached in this way: do not try to think of a game that would solve a problem and then try to make it as fun and popular as possible, instead try to take a game that has already gained and edited its popularity, so that our problem can be solved.

Just imagine—could a combination of all those players who play shooters like Call of Duty or strategic games like StarCraft help in curing cancer? Could a "farm" of gold, which is so crucial in World of Warcraft or Farmville-type social games, to do something useful? Many developers would say that not playing is fun because it is not a job. However, there are elements in these games that are computerized, but easy for a computer? If scientists have been able to answer this question, they should have a powerful tool to solve great problems. It should be noted that even regarding financial profit, it would certainly be an interesting thing.

For example, such elements are already-mentioned in the game Minecraft. The player is pumping materials for the construction of tools and buildings so that he have where to spend nights full of dangerous creatures. It is a simple concept, but there are countless variants and modes. Everything is made from cubes, which directly calls for easy creation, just like mazes. I can imagine the mode where it would be necessary to build Foldit-style proteins or build a rocket and then fly it through an asteroid-like universe as in Genes in Space. The potential regarding gamesourcing is enormous and, in our opinion, it is only a matter of time, when someone sees this idea. So far, only attempts have been made to use it for educational purposes [24].

For our purposes, online browser game Immortal Fighters has been used. This is a portal available at http://www.immortalfighters.net/, which allows you to play game Dragon Lair online. Dragon's Lair is a legendary Czech hero game inspired by the Dungeons & Dragons system. The game is led by the Lord or the Lady of the Cave (abbreviated **PJ**). This adventure narrator knows the rules of the game and in his imagination, he creates the world in which other players play through his fanciful characters. The game is usually run by PJ describing what is going on, and the players will tell how their character responds. PJ is governed by the mechanisms described in the rules, and even when he has the main word in the game, he often throws dice and uses data (strength, dexterity, intelligence, charisma, etc.). Each character has different depending on race (man, Dwarf, etc.) and vocations (warrior, sorcerer, border guard, etc.).

Dragon's Lair is a role-playing game (RPG): a hero game. Role-play means that a player lives in a fanciful character and plays for her as if he were alive. Players often play the characters in their characters, but it is important to realize that the player is not a character and that it is necessary to separate them. While playing for his character, he has to act not only with his attributes (especially intelligence and charisma) but also to separate his knowledge from his character's knowledge (for example, he cannot know that someone is a thief).

The best players are not those whose characters kill the most monsters, but those who play the best RP (role-playing). While the classic Dragon Lane requires players and PJ to come together physically and tell the story aloud, the online version is based on writing. Offline personal contact and game are missing from the offline version.

The algorithm behind our modification of the Labyrinth is called human ant system (HAS) for obvious reason. Each player represents one ant. The main differences from the above-mentioned alternative algorithms are:

1. A person decides about the transfer of an ant. This is the biggest and most fundamental difference. In the game, players are provided statistical information about the available edges, but the final decision depends on and only on the player. The reasons for player's decision can be very complex and are not limited to comparing the calculated probability with a randomly generated number from 0 to 1, as is the case with the classical ACO.

2. Pheromones. Pheromones as such are not present in the game. Each player can, however, look at any route of any player and arrange it accordingly. It is to be assumed that mainly the better routes will be observed; on the contrary, the worst will not be interested in almost anyone. This simulates the addition and evaporation of pheromones. However, if a player decides not to do so at all, nobody will be against it.

3. Starting point. The choice of the starting point depends on the player's consideration. This does not have to be a purely random matter, as with other algorithms.

4. Does not engage in the local extreme. With game mechanics, when duplicate paths are not rewarded, players are constantly trying to find new variants of Labyrinth passages and want to avoid going the same way as anyone before them. This way, the algorithm can never get stuck in local extrema and obey.

5. Parallelism. Ants (players) move in this form of the algorithm unbounded. They do not have to wait for each other. Moreover, it may well happen that one already goes through the chart 10 times while the other only once. This makes it impossible to talk about iterations throughout the colony. Every human-ant has a free will. He can only go through the graph after others have "provided" the right way. However, waiting at the same time, he risks being overtaken by someone. This brings to the last point.

6. Motivation. Unlike the artificial ants, the human drives the desire to win. It is a feature that each of us have to a varying degree. Trying to overcome others and become the best—it cannot replicate exactly with your computer.

3.3.3. Results and comparison

The HAS performance has been compared with the classical ACO algorithms, which were used in all cases with 10 ants (the SW limitation), and in the beginning, in all cases, a $1/n$ pheromone was placed on all edges, where n is the number of nodes in the graph. The number of iterations was 10,000 or until the right solution was found. Each variant of the algorithm run five times. With the average of the results of each variation, the comparison with HAS was done. Since the Labyrinth with HAS does not work synchronously (the players can play when they want), the comparison was carried out according to the number of ant moves (AM). For cases where an algorithm has been converted to the local minimum, the quality of the solution means the ratio of the ideal distance to the best found. HAS and ACO were tested on a TSP test problems and information on how the algorithm has dealt with them is listed here.

1. Burma14. It is a map of 14 cities in Burma. The shortest possible distance for TSP is 3323 km. In total, 37 people were involved in playing the Labyrinth and solved the TSP on this map in 64 hours, 2 minutes, and 52 seconds.

2. Ulysses16. Map of the Odysseus path that leads through 16 places. The ideal circular route is 6859 km long. Here, the game was played by eight players for only 7 hours, 3 minutes, and 52 seconds.

3. Gr17. Problem with 17 cities from Groetschel. The requested distance is 2085 km. At the time of writing this chapter, this problem has not yet been resolved by the Labyrinth. A total of 28 people were involved in the solution, and in 11 days, they made a total of 1035 ant movements. It can be assumed that for 100,000 AM (10,000 iterations with 10 ants) that other algorithms have available in this case, players will find the ideal path to find, and again much faster.

From the choice of ACO algorithms, only the classic and elitist variants have always managed to find the right solution for up to 10,000 iterations. Ranked, best-worst, and min-max

alternatives quickly converged, and in 28 cases out of 45, they landed at the local minimum (best-worst tackled every time). The ant colony system did not show convergence, but in 12 of 15 attempts, it failed to reach the right solution in 10,000 iterations. It is possible, however, that it would be achieved in subsequent iterations. The human ant system on 14 and 16 peaks far outperformed all the others, and on a graph of the Gr17 problem, it managed to find a very satisfactory solution with a much lower number of ant movements than the competitors did. Results are visually summarized in **Figure 7**.

At this point, it cannot be explained what makes the HAS so successful. The answer lies in the data that were taken during the game. Each player's movement was recorded along with the time stamp and current distance at that time. However, analyzing these data is not easy, and it is not finished yet. Likely human intelligence, intuition, and total freedom in decision-making have played a crucial role in those results. Certain similarities in watching the course of play at each maze have been noticed. Once someone who has found a way to a much better score than the best ones has come up, it has begun a massive improvement of all the following results, and the ideal solution has quickly been reached. This is probably the consequence of our human property to want to be the best. Moreover, this is probably the most important driving force of the entire Labyrinth.

3.4. LabyrinTS: a new platform for experimentation

The previous version of the Labyrinth was sufficient for initial experiments, but it had its limitations, for example, graphics (the game was mainly based on the imagination and imagination of the player) or the number of players. To enhance the game, increasing its attractiveness and enhancing its graphical interface, a new and separate version is created.

Currently, we are working on an experimental multiplayer maze application (unlimited number of players and no PJ presented), **Figures 8** and **9**, where users aim to collect as many items as possible in the maze. The goal of the game is to investigate the behavior of people who can see the

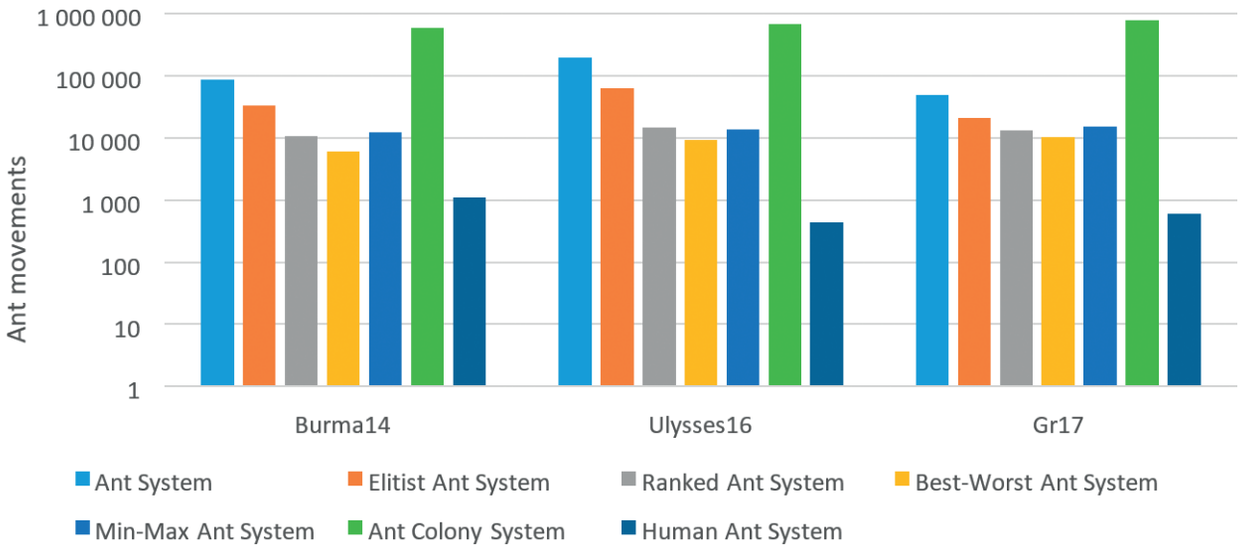

Figure 7. Comparison of ACO algorithms regarding required number of ant moves (logarithmic vertical axis).

http://dx.doi.org/10.5772/intechopen.71703

Figure 8. Screenshot of our prototype 3D maze application.

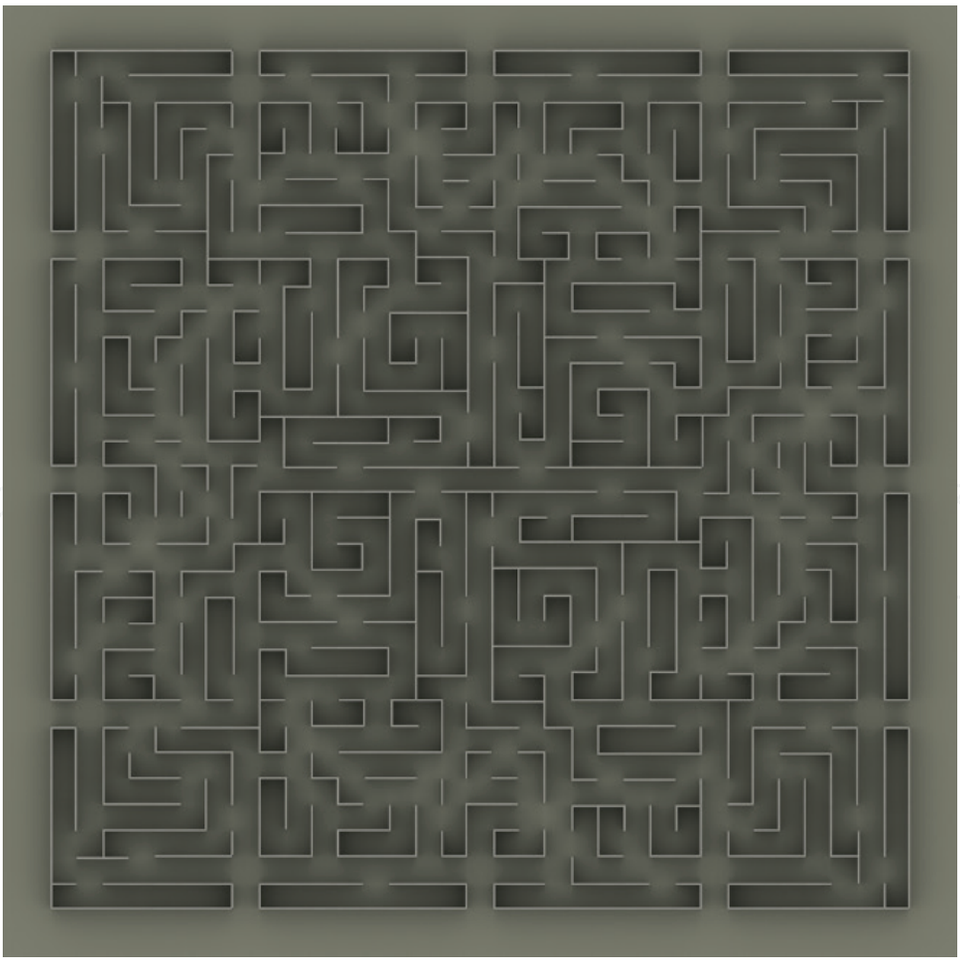

Figure 9. An example of a static maze created in blender.

score and position of other players. This is the same principle as in the previous version. All this can be modified. The whole application is designed so that players start randomly on the outer edge of the maze and start the maze and begin searching for and collecting items inside the maze.

Within each application, each player is represented by his avatar, through which the current 3D maze runs. The player follows the character of his avatar from a third person perspective (**Figure 8**), so that he can watch what his avatar has before him and where he can move.

The primary objective of the player is to collect as many items as possible in the scene for which points are awarded. As a result, the player with the highest number of points wins. After the game is started, players will accidentally appear on the margins (perimeter) of the maze, from where players can step in and start browsing the maze and collecting items.

In the current version, the maze is created for static experiments in different sizes (**Figure 10**). In the future, the maze should be generated randomly and automatically, according to the specified parameters. It would also be possible to create a 3D maze and move between levels through stairs or teleports.

To collect objects, two basic objects, a gold ring and a diamond can be found in the scene (**Figure 10**). The gold ring is rated one credit, and the diamond is rated ten credits. Regarding deployment, golden rings are randomly placed within the scene when creating a game, while diamonds are generated only in the narrow neighborhood in the center of the maze. The presence of diamonds in the center of the maze motivates players to try to get to the center of the maze as quickly as possible. In the future, it is possible either to expand the range of subjects or to modify the possibilities of occurrence of objects with different credit ratings and to test how these changes affect the behavior of the players.

In addition to collecting items, users can throw bombs, which are important when two players meet inside a maze, which happens in experimental applications, especially when players try to get diamonds in the middle of a maze. Each player has a 100% of his life at the beginning,

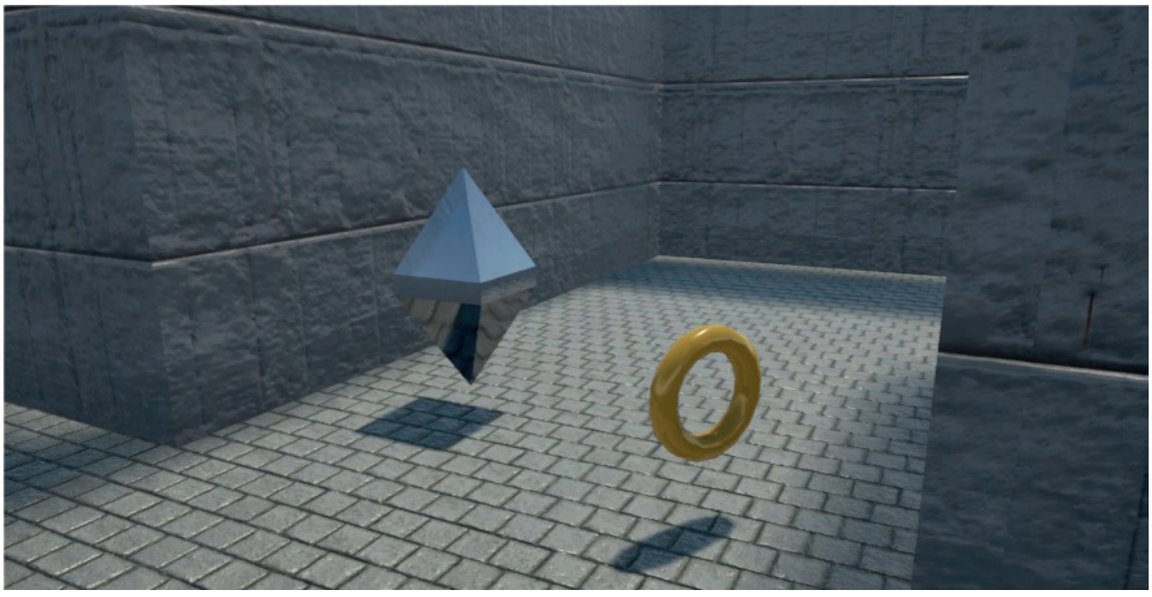

Figure 10. A sample of objects that users search inside a maze.

and his opponent can use a bomb against him. When the bomb explodes, the life is reduced. If his life drops below zero, the user will be relocated to a randomly generated location on the maze, where he can re-enter and continue collecting items.

The movement of all players during a maze crawl is automatically logged into the log file. These resulting log files of all players allow us to perform subsequent analysis of individual games. This allows us to look for different patterns of individual player's behavior.

- 10:10:46|5.8.2017;-2891.54;-1536.56;418.151;23;not picked;25;Alive;NotThrown
- 10:10:46|5.8.2017;-2891.54;-1536.56;418.151;23;not picked;25;Alive;NotThrown
- 10:10:46|5.8.2017;-2891.54;-1536.56;418.151;23;not picked;25;Alive;NotThrown
- 10:10:46|5.8.2017;-2891.54;-1536.56;418.151;23;not picked;25;Alive;NotThrown
- 10:10:46|5.8.2017;-2891.54;-1536.56;418.151;23;Coin Picked;25;Alive;NotThrown
- …

The entire application is developed in Unreal Engine 4 [25], one of the best 3D game and visualization designer tools. Using Unreal Engine enables us to use state-of-the-art algorithms and rendering techniques for real-time 3D games and applications. The static mazes themselves were created using the Blender modeling tool [26] and subsequently imported into the UE4. The application source code was created in C++ or using the blueprint (**Figure 11**), which is a visual scripting system in Unreal Engine.

This game is going to be an updated version of Labyrinth, described in the previous section. The game allows us to record movements of all players and consequent visualizations and analysis. As an example of possible outputs can serve **Figures 12–16**. **Figure 12** visualize a 2D trajectory in the Labyrinth (**Figure 9**), **Figure 13** shows all trajectories of all eight simultaneously playing players and **Figure 14** shows trajectories of first and sixth players in 2D coordinates. Those trajectories are less more similar and shows that there was a mutual influence of movements between players, as can be better observed from **Figure 16**, which captures the same moves but only for x coordinate. **Figure 15** shows trajectories of all eight players for x coordinate. It is clear that massive statistical analysis is needed to capture relevant information about swarm behavior of the players. This is the open research on which our group is working now. The entire application is currently under development and experimental testing.

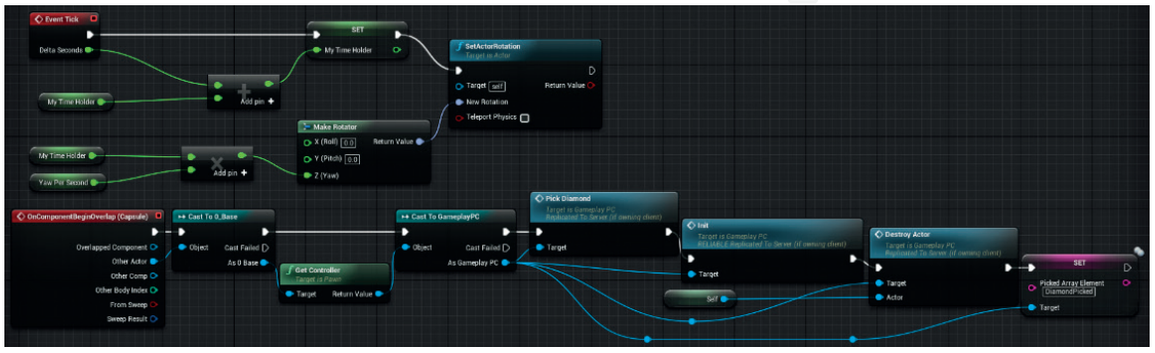

Figure 11. An example of creating an application for blueprint editor in UE4.

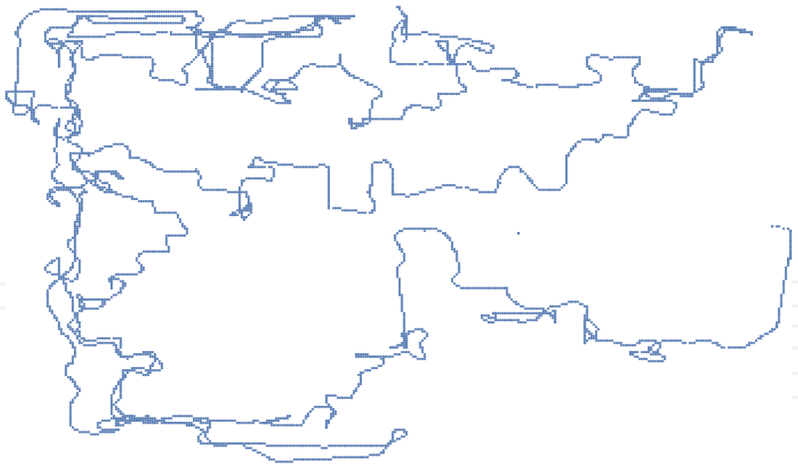

Figure 12. The 2D trajectory in the LabyrinTS.

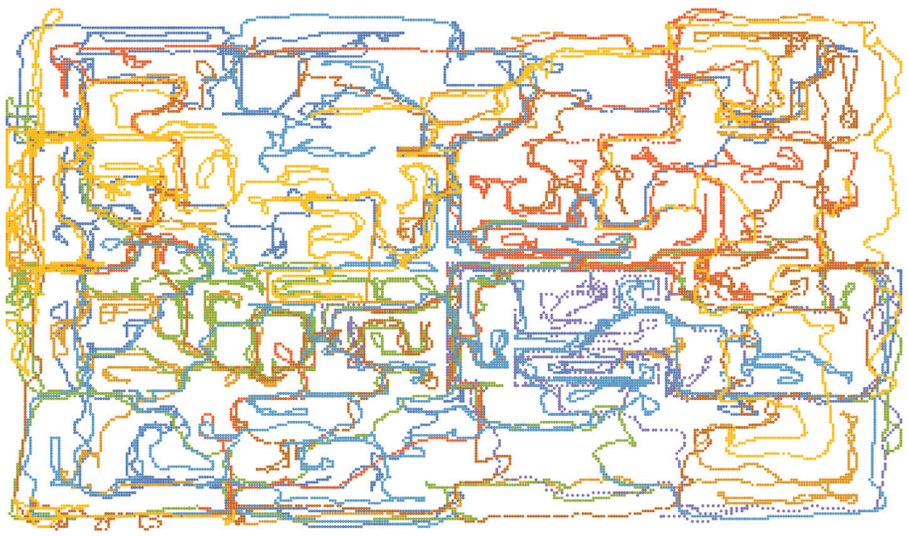

Figure 13. All eight players' 2D trajectories in the LabyrinTS.

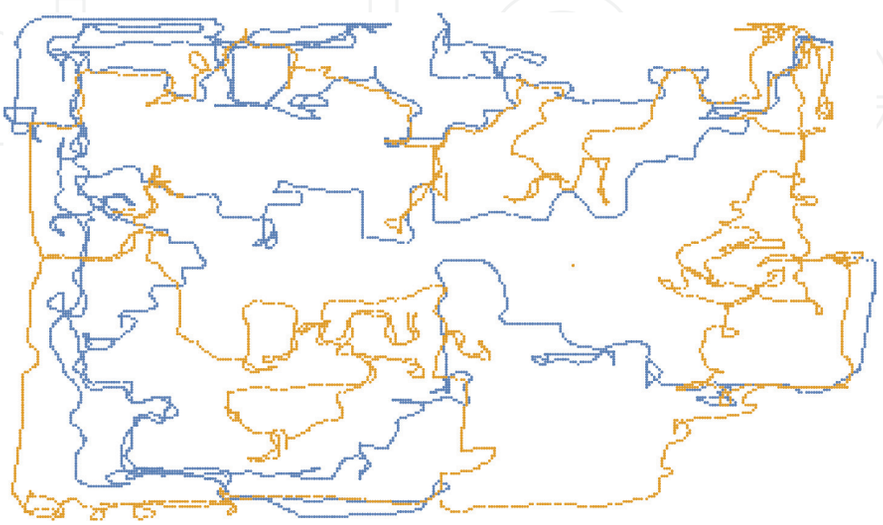

Figure 14. The second and sixth player trajectories in 2D.

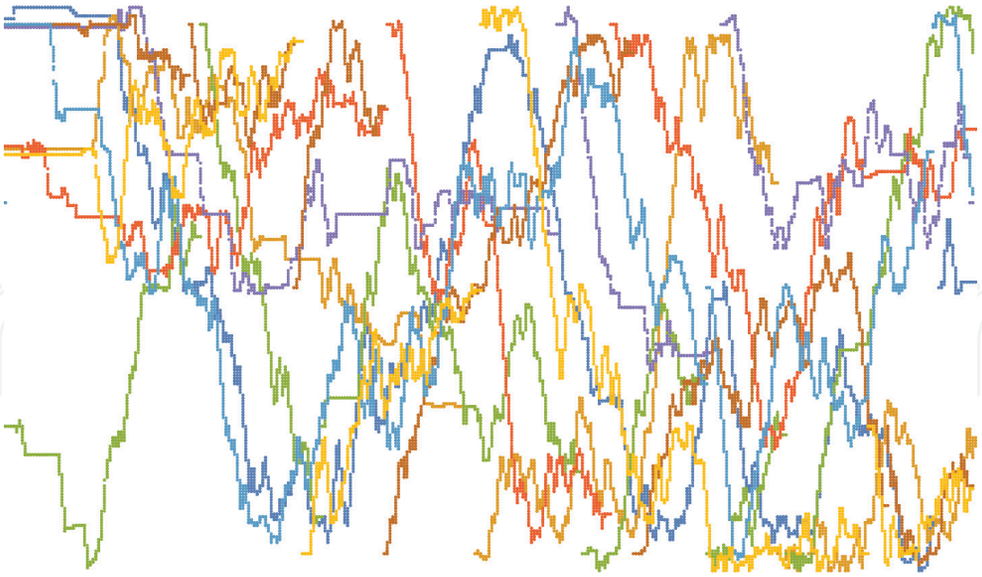

Figure 15. All eight players' x coordinate-based trajectories in the LabyrinTS.

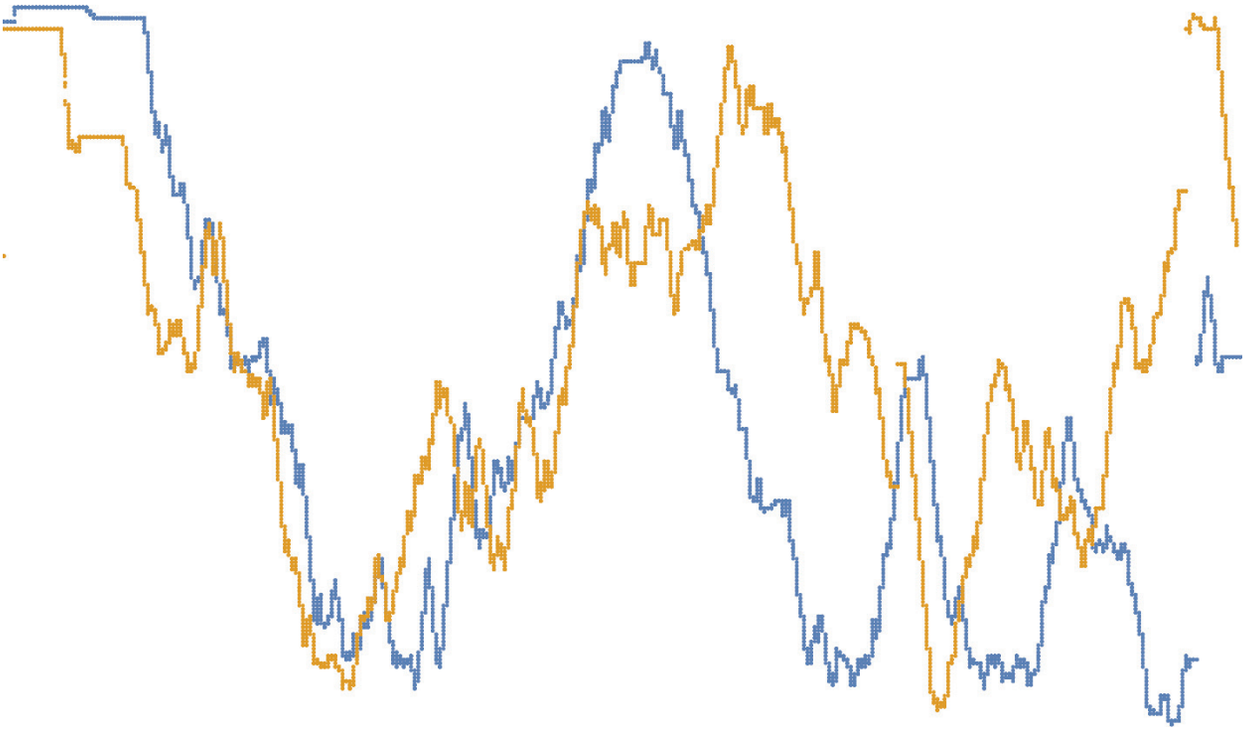

Figure 16. The second and sixth players' x coordinate-based trajectories in the LabyrinTS.

4. Conclusion

In this paper, we have demonstrated the use of swarm class algorithm in hybridization with computer games in three applications. The first two demonstrate the use of swarm algorithms as the human player counterpart in the game. The last one is the most important. It shows

how can a crowd of human players be used to solve the problem via computer game environment. This is a real application of gamesourcing.

The first application refers to SOMA [11] used on Tic-Tac-Toe [17] game and the second to our previous application on strategic game StarCraft: Brood War [18]. Concerning to SOMA use, comparing to our previous results from [11], where SOMA dynamics have been modifying for game purpose and used to control movement of the combat units in order to win battle, here Tic-Tac-Toe was used and all experiments were done with one human player against four SOMA strategies in 100 experiments in total (i.e., 25 experiments per strategy). Our implementation and obtained results have shown (remember this is not a mathematical proof but a numerical demonstration) that EAs and swarm algorithms (in this case, SOMA) are applicable for such class of games with acceptable success ratio.

The most important experiment, the third one, was focused on real gamesourcing. In this work, a game that allows solving the symmetric variation of the optimization problem of a TSP has been created and tested; i.e., the shortest Hamiltonian circle in an undirected graph in the form of a collective walk through the maze was done. It was necessary to define the game to include barriers and principles that force players to less more indirectly cooperate.

The performance of experiment with different ACO variants on three TSP instances has been evaluated. The use of HAS in Labyrinth has exhibited a much lower number of steps that AM needed to resolve the problem. The reason is certainly the very human component HAS in connection with the structure of the algorithm that the players performed through the game. Also, very important point is that HAS consisted of intelligent agents (players) with free will, intuition, and intelligent thinking, while ACO does not. This is different from standard ACO [9].

When playing the Labyrinth, data about players' actions were logged. In their correct analysis lies the key to revealing patterns of behavior that would provide understanding crowd behavior when the problem is solved. This can be regarded as the scope extension of this work. To process those data will not be a trivial task. It is clear that also classical disciplines outside of traditional computer science such as psychology or neurology will be needed.

The work [22] contains a complete description of the entire development process of this game-based application, which is called the Labyrinth, including all implementation details. The Labyrinth [21] has been warmly welcomed by the community, and according to the number of players playing, who are still on the same level, it can be concluded that the concept is good and the game is also fun for players.

It is clear that computer games are promising field for unconventional solution of the given problem and its design on this "purpose" [27] becomes to be more important.

Acknowledgements

The following grants are acknowledged for the financial support provided for this research: SGS SP2017/61 grant agency of the Czech Republic—GACR P103/15/06700S, grant of No. SGS

2017/134, VSB-Technical University of Ostrava. The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science -- LQ1602"; further by NPU I No. MSMT-7778/2014 by the Ministry of Education of the Czech Republic and by the European Regional Development Fund under the project CEBIA-Tech no. CZ.1.05/2.1.00/03.0089, and COST Action CA15140.

Author details

Ivan Zelinka[1]*, Martin Němec[1] and Roman Šenkeřík[2]

*Address all correspondence to: ivan.zelinka@vsb.cz

1 Department of Computer Science, Faculty of Electrical Engineering and Computer Science VŠB-TUO, Ostrava-Poruba, Czech Republic

2 Department of Informatics and Artificial Intelligence, Faculty of Applied Informatics, Tomas Bata University in Zlín, Zlín, Czech Republic

References

[1] Kuipers S. Citizen Crowds Can Make a Big Impact on Elections. Daily Crowdsource [Internet]. Available from: http://dailycrowdsource.com/content/open-innovation/1286-crowd-leader-shelley-kuipers-citizencrowds-can-make-a-big-impact-on-elections

[2] Fish LP. Island Create a New Constitution, Czech ed. Aktuálně.cz [Internet]. 14 July 2011. Available from: http://zpravy.aktualne.cz/ekonomika/technika/island-tvori-novou-ustavu-on-line/r~i:article:703969/

[3] Marie F. Crowdsourcing. Wikisofia [Internet]. 10 January 2015. Available from: https://wikisofia.cz/index.php/Crowdsourcing

[4] Handl Jan .Crowdsourcing is not new, Czech ed. Lupa.cz [Internet]. 14 August 2010. Available from: http://www.lupa.cz/clanky/crowdsourcing-neni-novinka/

[5] Crowdsourcing Is Not New—The History of Crowdsourcing (1714 to 2010). DesignCrowd [Internet]. 28 October 2010. Available from: http://blog.designcrowd.com/article/202/crowdsourcing-is-not-new--the-history-ofcrowdsourcing-1714-to-2010

[6] Jeff H. The Rise of Crowdsourcing. Wired [Internet]. June 2006. Available from: https://www.wired.com/2006/06/crowds/

[7] Google Play. SOMA TicTacToe [Internet]. Available from: https://play.google.com/store/apps/details?id=cz.bukacek.soma_tictactoe [Accessed: 22 August 2016]

[8] Sea Hero Quest [Internet]. Available from: http://www.seaheroquest.com/cs/faq [Accessed: 6 February 2017]

[9] Marco D, Birattari M, Stutzle T. Ant colony optimization. IEEE computational intelligence magazine. 2016;**1**(4):28-39

[10] DasGupta D. An overview of artificial immune systems and their applications. In: Artificial Immune Systems And Their Applications. Berlin Heidelberg: Springer; 1993. pp. 3-21

[11] Davendra D, Zelinka I, editors. Self-Organizing Migrating Algorithm, New Optimization Techniques in Engineering. 1st ed. Heidelberg: Springer; 2016

[12] Moscato P, Cotta C, Mendes A. Memetic algorithms. In New optimization techniques in engineering. Berlin Heidelberg: Springer; 2004. pp. 53-85

[13] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Advances in Engineering Software. 2014;**69**:46-61

[14] Kennedy J. Particle swarm optimization. In: Encyclopedia of Machine Learning. US: Springer; 2011. pp. 760-766

[15] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. Journal of Global Optimization. 2007;**39**(3): 459-471

[16] Yang XS. Firefly algorithm, stochastic test functions and design optimisation. International Journal of Bio-Inspired Computation. 2010;**2**(2):78-84

[17] Zelinka I, Bukacek M. SOMA swarm algorithm in computer games. In: International Conference on Artificial Intelligence and Soft Computing. Vol. June 2016. Zakopane, Poland: Springer; 2016

[18] Zelinka I, Sikora L. StarCraft: Brood War—Strategy powered by the SOMA swarm algorithm. IEEE Conference on Computational Intelligence and Games (CIG), August 2015. pp. 511-516

[19] Puget Jean Francois. No, The TSP Isn't NP Complete [Internet]. 29 December 2013. Available from: https://www.ibm.com/developerworks/community/blogs/jfp/entry/no_the_tsp_isn_t_np_complete?lang=en

[20] Solving the Travelling Salesman Problem Using the Ant Colony Optimization [Internet]. 23 September 2011. Available from: http://www.ef.uns.ac.rs/mis/archive-pdf/2011%20%20No4/MIS2011_4_2.pdf

[21] Immortal Fighters—Online Web Game [Internet]. Available from: http://www.immortalfighters.net/

[22] Jiri A. Computer games, gamesourcing and swarm intelligence in problem solution [Master thesis]. Ostrava: VSB-TU; 2017

[23] Chloi R. Minecraft Sales Surpass 100 Million Copies. IGN [Internet]. 2 June 2016. Available from: http://www.ign.com/articles/2016/06/02/minecraft-sales-surpass-100-million-copies

[24] The Minecraft Phenomenon Will also Find Work in Schools, Czech ed. Novinky.cz [Internet]. 2 November 2016. Available from: https://www.novinky.cz/internet-a-pc/hry-a-herni-systemy/419424-fenomen-minecraft-najdeuplatneni-i-ve-skolach.html

[25] Epic Games Inc.: Unreal Engine 4 [Internet]. Available from: https://www.unrealengine.com

[26] Blender Foundation: Blender [Internet]. Available from: https://www.blender.org

[27] Von Ahn L, Dabbish L. Designing Games With A Purpose. Carnegie Mellon University [Internet]. June 2008. Available from: http://www.cs.cmu.edu/~biglou/GWAP_CACM.pdf

IntechOpen