ITC 1/46
Journal of Information Technology and Control
Vol. 46 / No. 1 / 2017
pp. 150-164
DOI 10.5755/j01.itc.46.1.13872

Application of Bio-Inspired Methods in Intelligent Gaming Systems

Received 2016/12/23 | Accepted after revision 2017/01/19

http://dx.doi.org/10.5755/j01.itc.46.1.13872

Application of Bio-Inspired Methods in Intelligent Gaming Systems

Marcin Woźniak, Dawid Połap

Silesian University of Technology, Institute of Mathematics, Kaszubska 23. 44-100, Gliwice, Poland
e-mails: marcin.wozniak@polsl.pl, dawid.polap@gmail.com

Christian Napoli, Emiliano Tramontana

Department of Mathematics and Informatics, University of Catania,Viale A. Doria 6, 95125 Catania, Italye-mail: napoli@dmi.unict.it, tramontana@dmi.unict.it

Corresponding author: marcin.wozniak@polsl.pl

Creation of the layout for a board given to a player is usually achieved by means of complex algorithm that is detailing position of walls, passages, etc. We propose a novel solution based on bio-inspired algorithms adapted to easily create and then reorganize layouts of board games. The resulting layout presents a maze that is being automatically generated with given entry and exit point. This structures, because of proposed methodology, can be recomposed during the game. We show execution time of proposed algorithms and discuss efficiency in composition of mazes in various dimensions.

KEYWORDS: computational intelligence, heuristic algorithm, automatic control, board games.

Introduction

Bio-Inspired Methods (BIMs) are algorithms that can simulate various real life behaviors for optimization and control purposes. The advantage of this type of simulation is that the implementation of BIMs gives a powerful tool to solve complex problems. Using them we can implement a method where potential solution can be verified by many agents working on the problem. In the nature fish, birds and other move and

live in schools where the individuals join together to increase chances for optimal breeding and life conditions. Therefore similar approach implemented in each step of the algorithm gives many responses on the problem, among which we can choose the most appropriate one. By performing such operations, the attempted solutions are improved with each new step, similarly to the evolution among populations. Real organisms communicate with each other to take advantage of the experience of others. In BIMs we implement this communication by devoted modeling of implemented algorithms. Generally, all bio-inspired aspects of computer algorithms are achieved by an appropriate implementation, where with each new step BIMs can improve subsequently proposed solutions to precisely fit given criterions and solve complex tasks.

Developments in research on possible applications of BIMs have given various algorithms and attempted solutions. Behavior of various species of animals has been modeled to fit optimization and control purposes. Hetmaniok et al. proposed devoted versions of BIMs to simulate inverse problem of metal purification and casting processes [9], [10]. Similarly Brociek and Słota gave an example of application of these type of algorithms to fractional heat conduction problem [3] and [4]. Cpałka et al. discussed application of evolutionary algorithms to control processes [5], where similarly to BIMs an algorithm was inspired by evolution of real life organisms. Other various techniques of Computational Intelligence (CI) are also inspired by some features of real life organisms. These techniques enable more efficient data processing, as proposed by Rutkowski [23]. Ant population modeling and swarm techniques were presented to optimize computation of control systems. Dziwiński et al discussed fully controllable model of ant colony implemented to cluster text data [7] and identify significant operating points of simulated objects [8], while Okulewicz and Mandziuk verified application of BIMs into dynamic routing problems [19]. Kapuściński et al. proposed devoted application to engineering [28]. Napoli et al. have presented the application of an CI approach to improve quality of service in distributed cloud systems constructed for bit torrent services [18]. These techniques found very efficient implementations for image processing. Damaševičius and Ziberkas gave an example of improved energy consumption for image processing [6], what was presented in devoted for mobile devices form by Toldinas et al. [26] and Ignasius and Damaševičius [22], while Korytkowski et al. proposed innovative image classification by application of boosting fuzzy approach [13]. As presented due to the development of BIMs several alternative solutions to various scientific problems have been proposed, where BIMs high precision and easy implementation influenced efficiency. In the following sections of this article, we present an application of BIMs approach to automatically build game scenarios and game management system.

Related works

Computer games are part of commonly used entertainment, where dedicated technologies help to improve expression and develop several skills like hand-eye coordination. Playability is one of the important aspects for each game. Nonlinearity and complexity of levels and action scenarios make the game more interesting for larger audiences. To achieve such aspects it is necessary to develop a very large scenario or implement an intelligent and automatic management system. Lucas and Kendall [15] similarly to Yannakakis et al. [31] presented various aspects of games modeling and implementations by application of various CI techniques, while Lucas discussed potential innovations for further development of games strategies and scenarios modeled by usage of CI [15]. Wooldridge and Dunne discussed one of vary important aspect of any computer game: computational complexity that can significantly influence computer resources need for playing [27]. Among all aspects of games there are some of paramount importance: gamification and real-time management, since these two are conditional for users. Asteriskis and Damaševičius presented gamification patterns in various applications [1], among them tower games can efficiently be improved by CI patterns as discussed by Avery et al. [2] and security games as discussed by Karkowski and Mańdziuk [12]. Real-time strategies and management approaches were presented by Lara-Cabrera et al. [14].

This article presents a novel approach to develop a system that will actively assist both board game creation and online management. In general, a very good example of 2D board game is maze. We can define it as an environment having various paths, where the

majority leads to a blind end. Most mazes have one entrance and one exit, however it is possible to create mazes with several entrances and exits. The solution we present here is based on novel ideas proposed by Połap et al. application of BIMs for maze construction [21] and innovative management systems [20]. In this article we provide a solution for the automatic creation and management for one entrance and one exit maze type with improved BIMs devoted to ad-hoc maze composition. Maze generators very often apply graph theory algorithms like Prim's as discussed by Hirao et al. [11] and Kruskal's algorithms as discussed by Najman et al. [17]. In this article, we propose an alternative solution based on tailored BIMs, where some similarities can be found in approaches proposed by Świechowski and Mańdziuk for self-adapting game strategies [24] and automatically personalized contents creation aspects proposed by Togelius et al. [25]. The proposed novel BIMs-based automatic system is more efficient. The application of swarm intelligence made it possible to easily adapt maze shape to conditions revealed in real-time. Moreover, a novel management of the game is proposed to support remote resources what makes it faster and flexibly adjust to various scenarios.

Developed bio-inspired methods for 2D maze composition

We can define Bio-Inspired Methods (BIM) as a tailored algorithmic solutions that implement some habits common in nature into computer methods applicable in modeling and control purposes. In such methods, the movements of swarm particles are modeled to optimize particular function. Modeling is based on communication, which is unique for each one of the species of animals, insects or plants. All particles pass information to the others. In this way population adapt to natural conditions. We can name this behavior a swarm intelligence, since all the particles exchange information and learn from each other. In this article we present developed application of two BIMs versions to create 2D mazes in comparison to approaches proposed by Połap et al. [20-21] and a dedicated model for *control agent* based on swarm intelligence applied to game processing and recomposition of game boards. To solve the maze, players must find road from the entrance to the exit, however cannot cross any wall. In the developed Game Management System (GMS), an intelligent agent can change the maze in real-time, while players move from entrance to exit. This active control solution makes the game more challenging without any special plot engines.

In the following sections, we present a model of game management system and dedicated algorithms designed to build 2D mazes, where the maze structure is developed by an agent using Flower Pollination Algorithm proposed by Yang [29] or Artificial Bat Algorithm also proposed by Yang [30]. Each of them returns an array of values representing maze. For this purpose, each value in the array should be evaluated in accordance with

$$\Lambda(y)=\begin{cases}\text{empty space} & \text{if} \quad y\in[0,\kappa]\\ \text{walls} & \text{if} \quad y\in(\kappa,1]\end{cases}, \tag{1}$$

where κ is constant limit value, which decides on placing the wall.

Artificial Bat Algorithm for maze composition

Artificial Bat Algorithm (ABA) is one of the new heuristic algorithms [30]. ABA is inspired by echolocation used by bats to trace prey and scan surroundings while hunting. This algorithm is based on some initial assumptions that help to implement bat behavior:

- Each bat is able to judge the distance and recognize whether an object is an obstacle or prey using echolocation.
- All bats are flying with fixed velocity v_i and have sound frequency f_{min}, which is used in the phenomenon of echolocation.
- Each bat is able to adapt periodicity of sending pulse as r ∈ [0, 1].

Bats use plants not only as food but also as landmarks in a field allowing them to find feeding grounds. In order to create 2D mazes, we assume that the field where the bats move is a forest with two feeding grounds (which represent maze exit and entrance). Bats are searching for the path between the two feeding grounds. While each bat is looking for a way to another feeding ground, the trajectory of its movements is recorded. A field consists of a grid of cells. For the developed ABA version, the value of each cell is increased when crossed by a bat, i.e. 0.25 is added to the array representing the path of each bat. At the

beginning of the algorithm, each cell is marked with a random value between 0 to 0.2. Then, after the simulation, if two neighboring cells have value greater or equal to 0.5 then the wall between them is removed.

The initial bat population in ABA is created at random by assigning position, velocity $v_i \in [0, 1]$ and sound frequency $f_i \in [f_{min}, f_{max}]$. In each iteration, the position of the bat i in iteration t is updated by

$$x_i^t = x_i^{t-1} + v_i^t, \quad (2)$$

where velocity v_i is calculated by

$$v_i^t = v_i^{t-1} + (x_i^t - x_{feed})(f_{min} + \beta(f_{max} - f_{min})), \quad (3)$$

where $\beta \in [0, 1]$ is a random value and x_{feed} means one of the feeding grounds. In order to adapt the ABA algorithm to create 2D mazes, we introduced a change to (2) that

$$x_i^{new} = x_i^{old} + \eta\, Avg(v_i^t), \quad (4)$$

where $Avg(v_i^t)$ is the average velocity of all bats in current iteration and η value is chosen based on a random value of μ according to

$$\begin{cases} \eta = -1 & \text{if} \quad \mu \le 0.25 \\ \eta = 0 & \text{if} \quad \mu \in (0.25, 0.75) \cdot \\ \eta = 1 & \text{if} \quad \mu \ge 0.75 \end{cases} \quad (5)$$

The value of the pulse for each bat r_i^t is updated in each iteration t by

$$r_i^t = r_i^{t-1}\,(1 - \frac{1}{e^{\gamma t}}), \quad (6)$$

where γ is constant. As a fitness function, we use the distance from the feeding grounds (exits) calculated by Cartesian metric

$$L_{ji} = \| x_j^{exit} - x_i^t \| = \sqrt{\sum_{k=1}^{2} (x_{j,k}^{exit} - x_{i,k}^t)^2}\ . \quad (7)$$

Formula (7) allows for the assessment of each individual in the population - the smaller the metric value, the output is closer. The more individuals with better adaptation means the more convoluted maze in final stage.

Algorithm 1: ABA to create 2D mazes

Define coefficients: n - size of bat population, m - number of chosen best bats,
Create position array with random value between (0, 0.2) and two different exits,
Create population composed of n bats, initial frequency f and velocity v,
  While there is no passage through the maze **do**
  Move bats according to (4),
  Update velocity with (3) and position array for each bat,
    If $rand > r_i^t$ **then**
      Generate a local solution around the selected bat by (4),
    End If
    Take the best m bats ranking them according to how close they are to the exit using (7),
    To keep the number of bates in population constant create for rest *new position* at random,
    If rand>0.8 or *f(new position)* < *f(current position)* **then**
      Replace *current position* with *new position*,
      Change value r_i^t using (6),
    End If
  End While
Recalculate value in position array according to (1),
Return array.

Flower Pollination Algorithm for maze composition

Xin-She Yang proposed an algorithm inspired by the pollination process of flowers [29]. Modeling of pollination process adopts assumptions:

- The phenomenon of biotic and cross-pollination is understood as a global pollination. It is a process during which, pollinators are transferred by Levy flights.
- Abiotic pollination and self-pollination are interpreted as local pollination.
- The process of global and local pollination is controlled by value of p ∈ [0, 1]. Controllability is understood as various factors, such as wind.
- Every plant has exactly one flower, which produce only one flower.

At the beginning of the Flower Pollination Algorithm (FPA) an array of values representing the maze is created. Each cell of the maze is given a random value in range (0,0.2). In order to enhance the visibility of exit fields, their values are increased up to 0.7. The array is interpreted as a meadow on which pollen are sprayed.

FPA consists of two stages: global and local pollination. In case of global pollination, pollen is transported by insects which allow to enlarge the analyzed area. This solution allows for the reproduction of the fittest flowers and is modeled as

$$x_i^{t+1} = x_i^t + L(x_i^t - g^*), \tag{8}$$

where x_i^t is pollen transported by insects in iteration, g^* means the nearest exit of the maze and L is the strength of the pollination represented by Levy flights described as

$$L(x,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \frac{\exp[-\frac{\gamma}{2(x-\mu)}]}{(x-\mu)^{\frac{3}{2}}}, 0 < \mu < \infty, \\ 0, \text{other} \end{cases} \tag{9}$$

where μ>0 is minimal step length and γ is scaling parameter. Again local pollination occurs by using the following equation

$$x_i^{t+1} = x_i^t + \varepsilon(x_j^t - x_k^t), \tag{10}$$

where x_j and x_k is a pollen from different flowers of the same plant selected in random way and $\varepsilon \in [0, 1]$ is a step of random walk.

Algorithm 2: FPA to create 2D mazes

Define coefficients: n - size of flower population, m - number of chosen best flowers, p – global pollination value,
Create position array with random value between (0,0.2) and two different exits,
Find the best solution g^* in initial population (which represents an exit),
While there is no passage through the maze **do**
If *rand*>*p* **then**
Perform global pollination by (8) for each flower,
Else
Choose at random two indexes of flowers from population,
Perform local pollination by (10),
End If
Evaluate population according to how close they are to the exit using (7),
Take best m flowers to next iteration,
Generate at random n-m *new flowers*,
Evaluate these *new flowers* according to how close they are to the exit using (7),
If *L(new flower)* < *L(old flower)* **then**
Replace *old flower* with *new flower*,
For the rest of flowers in the population increase value of created *new flower* by 0.2 and decrease value of *old flower* by 0.05,
End If
Decrease value of each field by 0.05,
End While
Recalculate value in position array according to (1),
Return array.

Game Management System

Proposed Game Management System (GMS) is a software module that models actions like create initial maze, control the game in real-time, and support other features. Information, after its retrieval, is sent back to the user. Therefore, it is possible to use the gaming system even on devices having low processing power. We are considering further improvements for energy saving with appropriate modeling of computing systems that can efficiently increase quality of service and support front-end interfaces.

Presented ABA and FPA have been implemented to create various initial mazes, and their results give a maze representation. Besides maze creation, another procedures are responsible to transform maze representation into a graphical form that can be presented to player. When a player starts the game, agents based on using dedicated swarm intelligence automatically executes to control game progress. An initial population of board control agents is placed in maze. After each player move these control agents start to change the board to make the game more difficult. GMS continues maze modifications until the player reaches the exit or looses the game trapped in the maze. Maze real-time recomposition is based on a dedicated version of Cuckoo Search Algorithm.

Cuckoo Search Algorithm for real-time automatic game management

Cuckoo Search Algorithm (CSA) simulates cuckoos while tossing their eggs to nests of other birds. CSA for maze recomposition (i.e., changes on the possible paths that can be followed by the player) has some as-

sumptions to increase coherence:

- Cuckoos can drop only one egg to randomly chosen location in each iteration.
- Number of locations remains unchanged.
- Dropped eggs can be detected by hosts with probability $p \in [0, 1]$, for this reason we place new cuckoo at random position in the maze.

Cuckoos are moving randomly, since the location of the new nest they visit is chosen at random. This situation was modeled with Levy flights. These are peculiar examples of random walks, which are stochastic processes representing movements in random directions by equation

$$L(x,\gamma,\mu)=\begin{cases}\sqrt{\frac{\gamma}{2\pi}}\frac{\exp[-\frac{\gamma}{2(x-\mu)}]}{(x-\mu)^{\frac{3}{2}}}, 0<\mu<x<\infty,\\ 0, \text{other}\end{cases} \quad (11)$$

where μ is minimum step length and γ is scaling parameter. Levy flight (10) is applied to simulate random movements according to equation

$$x_i^{t+1} = x_i^t + \mu\, L(x, \gamma, \mu), \quad (12)$$

where t is the number of iteration. The decision to remove it is taken by the host which discovers tossed egg. Such a situation is described by

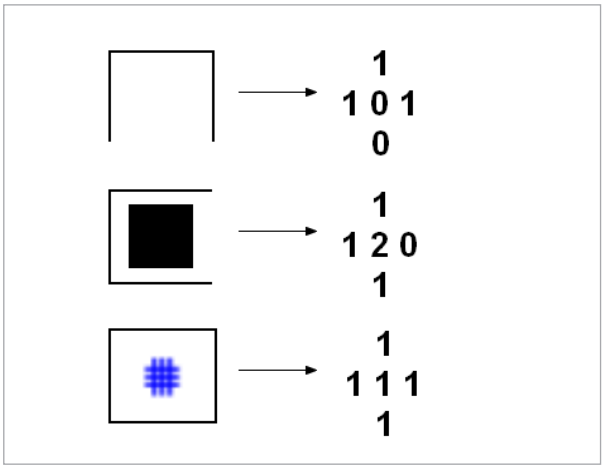
$$H(x_i^t)=\begin{cases}1-p, \text{remove the egg}\\ p, \text{leave the egg}\end{cases}, \quad (13)$$

where p represents hosts random decision.

Cuckoo Search Algorithm (CSA) has been applied to automatically manage board games created by ABA and FPA. We have developed a dedicated version of CSA tailored for being managed in distributed systems. Cuckoos are made to move over the maze represented in two-dimensional arrays created by ABA or FPA. As before, i.e. during maze creation, a maze consists of cells, which are held in memory by means of a representation. Each cell is described by five values. The first four values represent the walls of the cell (north, south, west and east, respectively), if there is a wall then we have value 1 on the respective position, whereas if there is no wall then we have 0. A player who happens to be positioned in the cell is given by the fifth number. If a player is in the cell, presence is coded as value 2, whereas if in the cell there is a control cuckoo, then the value is 3, otherwise if cell is empty then the value is 0. The proposed notation is shown in Figure 1.

Figure 1

Digital representation of a sample cell in a board maze: empty cell (at the top), player occupied cell (in the middle), cuckoo occupied cell (at the bottom)



CSA for active game control and management has been developed to modify maze construction in real-time. The actions of the control cuckoo depend on player position. Decisions of cuckoo depend on the distance from the player and some random coefficients, which are modeled with a dedicated fitness function

$$\Gamma(x_i^t, x_j^t)=\begin{cases}L_{ij}^t+\nu & \text{if} \quad L_{ij}^t<\sqrt{S}\\ e^{-\alpha/\nu} & \text{if} \quad L_{ij}^t>\sqrt{S}\end{cases}, \quad (14)$$

where S is maze cell value, α and $\nu \in [0, 1]$ are scaling parameters, L_{ij}^t is Cartesian distance between positions x_i (cuckoo) and x_j (player) calculated by (7). Cuckoos move over the maze. Their function is to automatically change composition of fields. Therefore cuckoos control situation in real-time, i.e. while player moves over the maze, cuckoos modify it to make the game more challenging. Additionally in our system, if

player comes across cuckoo then player must play a subplot game, hence cuckoos are given more time to change the maze again.

Algorithm 3: CSA to manage game in real-time

Define coefficients: c – size of population, μ – minimum step length, γ – scaling parameter, p – hosts random decision, r – number of random alleys, α – the maximum value of the disposal and created walls,
Create initial population at random positions,
While the game is not finished **do**
If $t>1$ **then**
Generate two random numbers a, b ranging from 0 to α that represent number of walls to be removed,
If $adaptation<5$ **then**
Remove a random walls within 5 fields,
Create b random walls within 5 fields,
Else
$i=0$,
While $i<r$ **do**
Generate random number $rand \in [0, 1]$,
If $rand>0.5$ **then**
Remove randomly walls within 5 fields,
End If
End While
End If
End If
Move cuckoos to inspect maze according to (11) and (12),
Hosts decide about situation according to (13),
Evaluate positioned cuckoos according to (14),
Best cuckoos go to next round and the rest of cuckoos are positioned at random over the maze,
t++,
End While

Real-time game board controlling

An intelligent control procedure is used, which handles special stop conditions to decide if there is still need to change paths for the player connected to the developed GMS. This procedure is also controlling the way from the entrance to the exit. In the GMS after each iteration we check the passage entrance-exit through the maze. If the passage exists, a board is complete and can be presented to the player. Otherwise, supervising procedure returns the board to CSA for additional changes. Stop conditions are based on Cartesian metric defined in (7), which show the way between next two cells i and j in the maze on the way from entrance to exit. If $L_{ij} = 1$, it means that supervising agent (and also player) can move from one cell to another only horizontally or vertically, excluding diagonal passages across the maze board. Control procedure is presented in Algorithm 4 for which applied supervising agents based model to verify the composition is run as presented in Algorithm 5, while schematic system composition is presented in Figure 2.

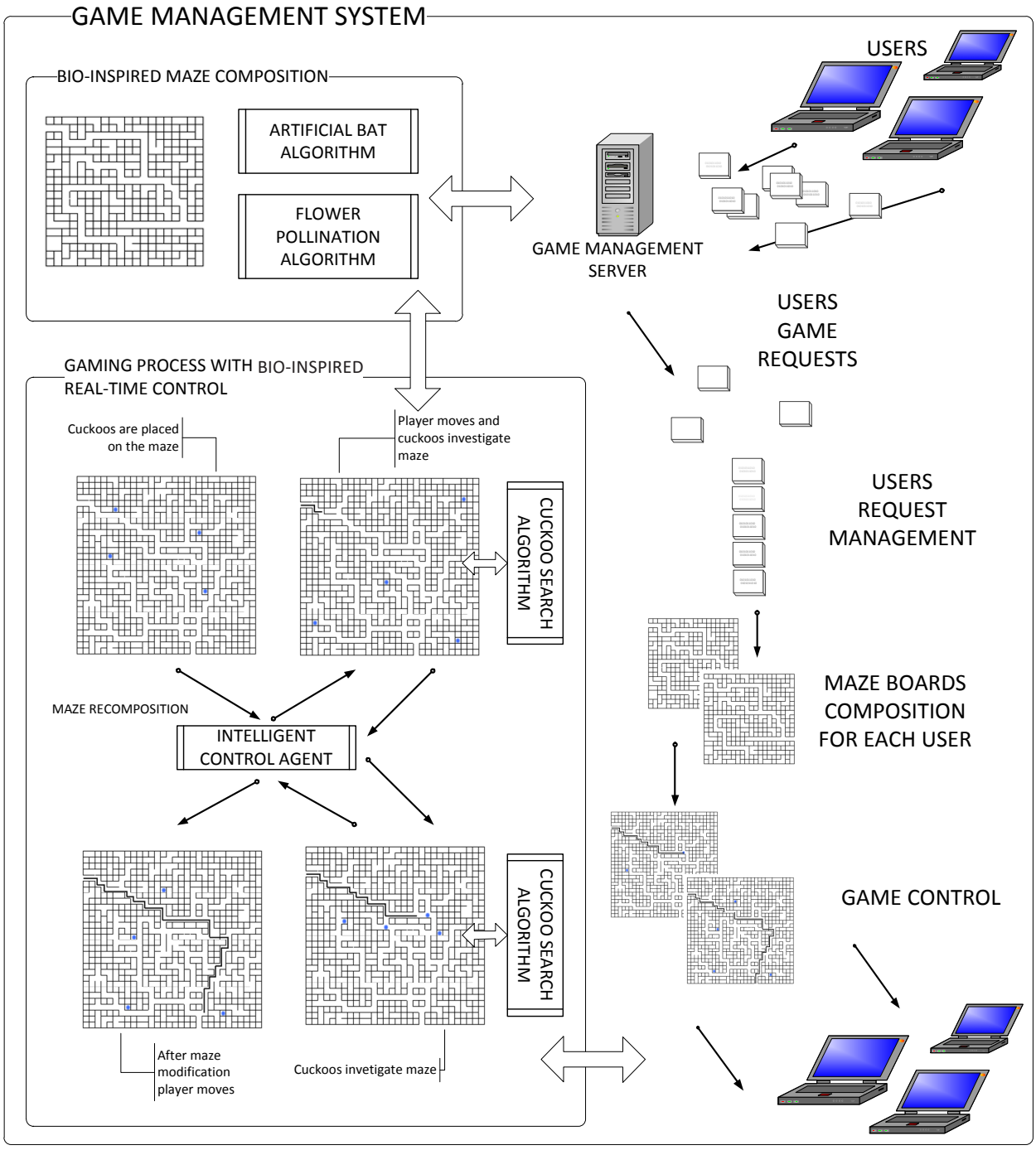
Algorithm 4: Control agent decision in real-time

Call algorithms: ABA or FPA to create an array of digital numbers representing 2D maze,
Create bitmap to present to the player,
For all values v in array **do**
If v is 1 **then**
If $r>0$ **then**
If random value is lower than 0.5 **then**
Create a wall,
End If
Else
Create wall,
End If
Else
For all neighbor of v **do**
If neighbor is 1 **then**
Create a wall,
End If
End For
End If
End For

Algorithm 5: Supervisors verification of the passage
For all entrances **do**
While no movement is possible **then**
Remove fields in the row that were supervised in previous step,
For all neighbor fields **do**
If $L_{ij} = 1$ is not true or there is a wall **then**
Delete a field,
End If
End For
Select randomly a movement,
End While
If last field is entrance **then**
There is way out,
Else
There is no way out,
End If
End For

Figure 2

Model of the Game Management System with implemented Bio-Inspired Methods to create mazes and recompose them in real-time



Experimental results

Proposed GMS was implemented to compose various mazes with an automatic control procedures. In benchmark tests we have compared solutions presented in Połap et al. [21] with ABA and FPA. The parameters used for the development are as follows (their definitions are given in the previous sections).

- ABA: n=30, m=10, r=25,
- FPA: n=30, m=10, p=0.4, r=25,
- CSA: n=4, $\mu = 0.7$, $\gamma = 0.6$, $\nu = 0.4$, $p = 0.3$.

Developed methods were applied to compose various resolutions and combinations of 2D mazes. As part of the tests for the proposed method, several mazes were generated for different sizes for each of the examined algorithms. Results of the timing for examined procedures are presented in Table 1.

Figure 3 presents comparison of creation times benchmark tests for 2D maze boards when using ABA and FPA in comparison to AACA and ABCA. Analyzing benchmark tests results we can see that for board composed of up to 3000 cells ABA is similar to AACA and about 100 seconds slower than ABCA and FPA, however FPA outperforms all other methods. All four of examined solutions are similarly efficient for boards of about 3000 fields. Between 3000 to 6000 fields AACA composes mazes faster

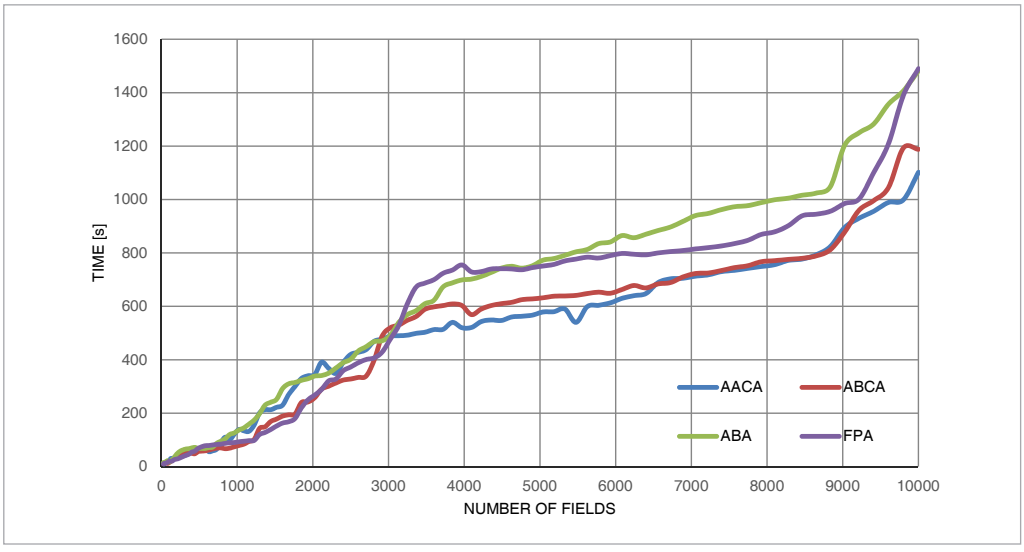
Table 1

Comparison of experimental results for implemented methods of maze composition

Fields	Artificial Ant Colony Algorithm (AACA)	Artificial Bee Colony Algorithm (ABCA)	Artificial Bat Algorithm (ABA)	Flower Pollination Algorithm (FPA)
25	12	11	15	10
500	52	50	70	54
1000	139	79	137	93
1500	230	189	292	163
2000	345	258	339	268
2500	420	328	402	373
3000	489	520	492	478
3500	513	598	623	700
4000	521	569	702	729
4500	547	610	743	741
6000	630	663	865	798
6500	689	684	884	800
7000	713	723	940	815
7500	735	745	973	835
8000	756	771	999	879
8500	777	780	1016	939
9000	895	878	1203	985
9500	989	1045	1357	1209
10000	1102	1188	1482	1490

Figure 3

Comparison chart of composition time for various dimensions of boards composed by ABA and FPA



than all of the methods (about 200 seconds faster than FPA and ABA), however between 6000 to 10000 cells AACA and ABCA are the fastest methods that outperformed FPA, and ABA altogether of at least 200 seconds. For very big board of about 10000 fields we see that AACA is more efficient and about 10% faster than ABCA and about 20% faster than ABA and FPA. FPA is most efficient for small board perfect for portable electronic devices, what gives proof to proposed methodology.

Proposed CSA control has been applied to control changes of these mazes in real-time due to player actions. Management system has been tested empirically for each of generated solutions. A small number of cuckoos (4 control agents) makes the GMS perform all ad-hoc changes in composition of boards while minimizing changes to current labyrinth form. Distance from player is also taken into account, i.e. the smaller the distance from the player the more modifications agents do. Two game scenarios are presented in Figure 4 for 30x30 fields and in Figure 5 for 30x94 fields. The while loops in Algorithms 1 and 2 ensure creation of the maze with full passage – however, for a very large size, generation time will be correspondingly longer independently of applied parameters.

Figure 4

Sample Game Management System work for 30x94 fields maze: in the first row random positioning of cuckoos over the maze while players movement, in the second row after maze recomposition cuckoos are approaching player to investigate the board, in the third row player came across a cuckoo investigating the maze, in the four row player is finishing the maze reaching the exit

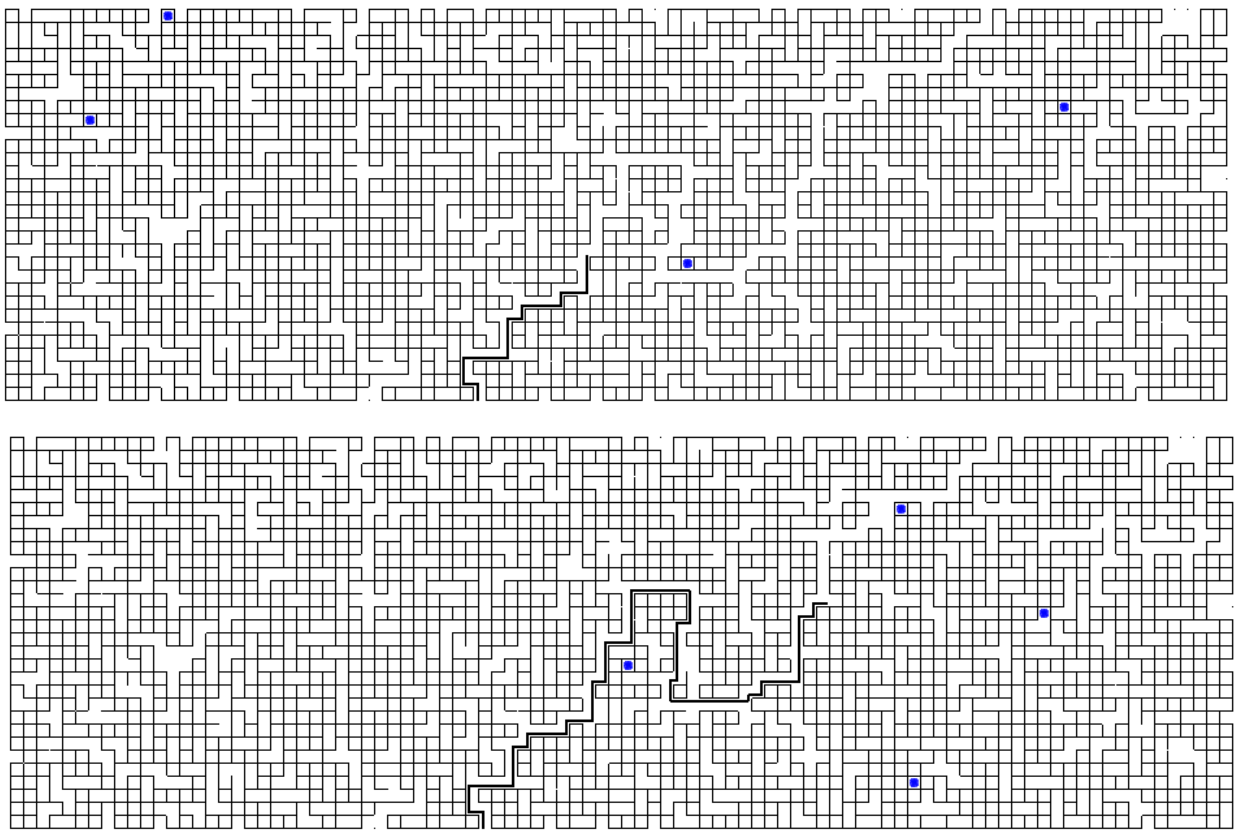


Figure 4 (continuation)

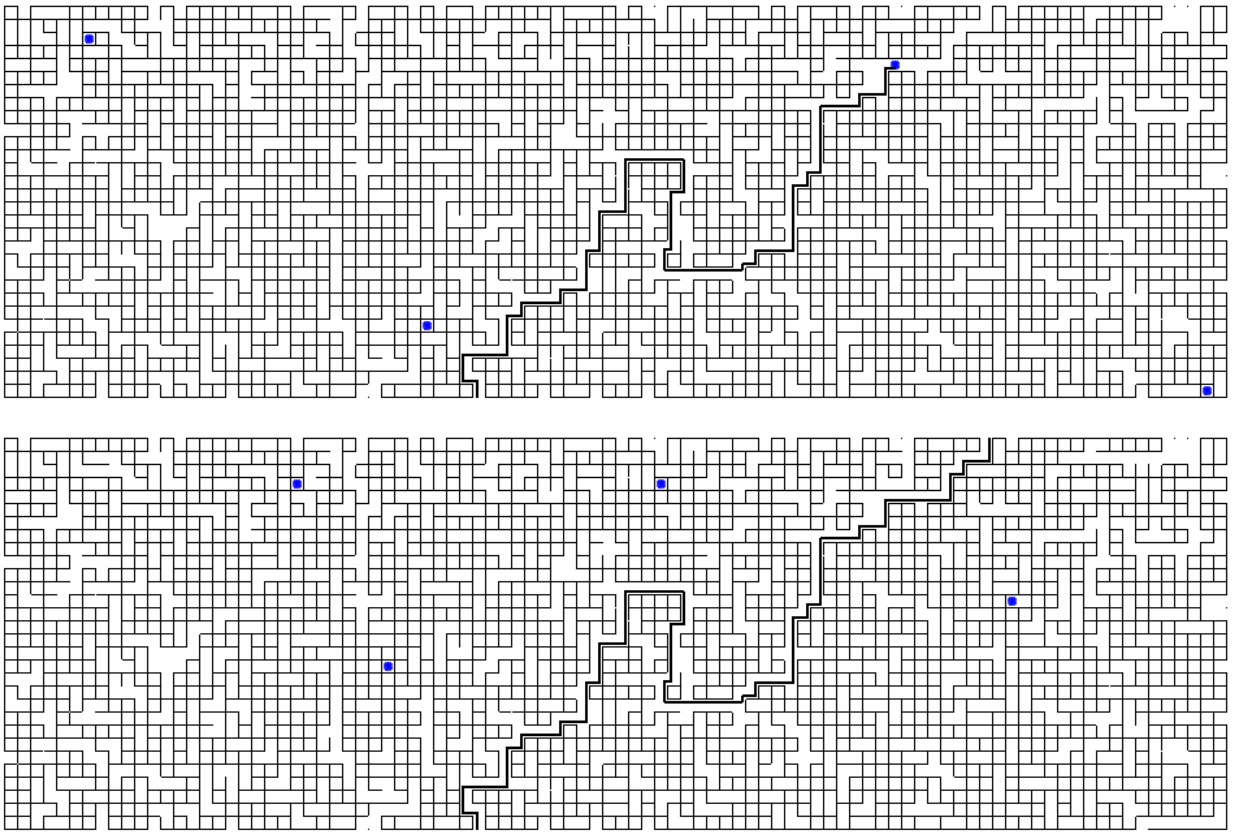


Figure 5

Sample Game Management System work for 30x30 fields maze: in the first row from left to right 1) random positioning of cuckoos over the maze, 2) player movement, 3) cuckoos fly to investigate the maze and an intelligent agent use these investigations results to decide on board modifications, in the second row from left to right: 4) player moves over changed board, 5) cuckoos fly to investigate the board, 6) the intelligent agent use investigations to modify the maze and the player encounters a cuckoo in the move, in the last row from left to right: 7) cuckoos fly to investigate maze, 8) the intelligent agent decides on changes and player moves, 9) cuckoos fly to investigate and after modification of the board the player reaches exit in final movement

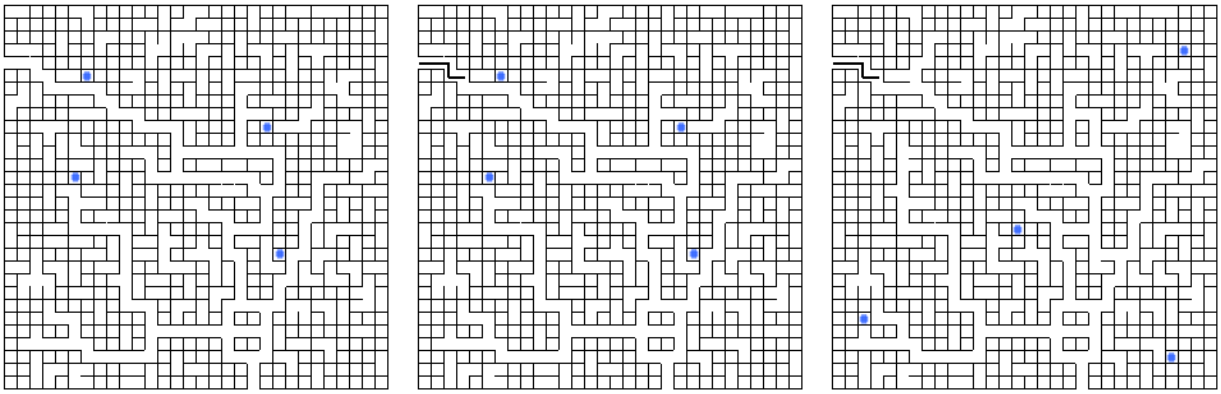
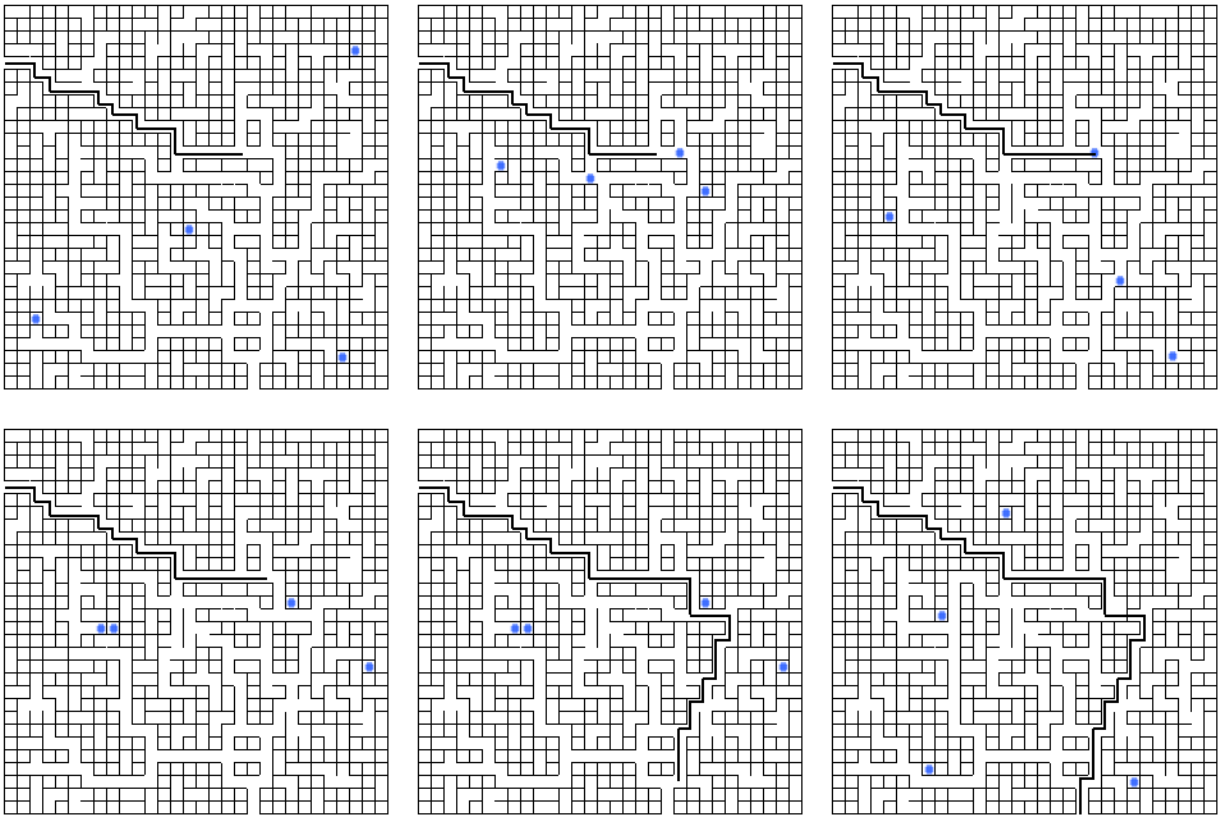


Figure 5 (continuation)



Conclusions

The presented solution gives opportunity to modify linearity of the storyline in real-time. Applied Bio-Inspired Methods are simple to implement and enable easy calculations. Presented GMS has many advantages, including ability to dynamically compose various quests for players. Because of adaptation of intelligent control agents (based on Cuckoo Search Algorithm) the system can frequently modify storyline in real-time, hence realizing ad-hoc interactions with player.

In two-dimensional situations, where player has an opportunity to watch entire game board - the bigger the maze, the player's attention is smaller. This solution costs more time to prepare it (see Figure 3). In contrast, in case of three-dimensional, where player sees the board in terms of its position – the labyrinth may be suitably small. Game management system with a proper calibration can looping path for unaware player. What is more, the situation where a small maze is copied in real time is also possible, but more difficult to manage – the population in management system must be restarted while the maze is copied. The player can repeatedly play the game, and each time a new and interactive game can be presented to him/her what is an advantage of the solution. From our benchmark tests we can conclude that FPA is most efficient for mazes up to 3000 cells, and above this limit AACA has shown best results, but for big boards ABCA can show similar efficiency as AACA. Maze generation is a complex process however presented methods are efficient and created boards are unique. Moreover, application of Bio-Inspired Methods enable us to implement a more detailed graphics, since efficiency of maze composition and control al-

gorithms keep the processor available for graphically-related tasks. Since these results are promising we hope to develop dedicated versions for 3D mazes, however this task is much more demanding. For this type of games we need very efficient methods to compose 3D boards.

Final remarks

In this paper we have examined Bio-Inspired Methods to compose and manage 2D mazes in real-time. Implemented algorithms were used to create boards of various mazes, where in particular we were interested in application of Flower Pollination Algorithm and Artificial Ant Colony Algorithm in comparison to other proposed methods.

In presented system we have developed real-time control agents. These are also using Bio-Inspired Method to control maze and recompose it ad-hoc to make the game more demanding for player. Applied Cuckoo Search Algorithm helped to implement intelligent agents that move over maze board. These agents are able to control situation and recompose the structure of the maze while player moves from entrance to exit.

While performing dynamic creation of mazes, proposed solution works in real-time. Presented Game Management System is using swarm intelligence to efficiently process data, which result in simple calculations. Therefore player can use the system, which is ad-hoc reacting to the action. Moreover, simple calculation means faster response, and hence less processing power is needed.

Acknowledgments

This work has been partially supported by project PRIME funded by the Italian Ministry of University and Research within POR FESR Sicilia 2007-2013 framework and Operational Programme: Knowledge Education Development financed by the European Social Fund under grant application POWR.03.03.00-00-P001/15.

References

1. D. Aseriskis, R. Damasevicius. Gamification Patterns for Gamification Applications, International Conference on Intelligent Human Computer Interaction – IHCI'2014, Evry, France, 2014, 83-90, DOI: 10.1016/j.procs.2014.11.013. https://doi.org/10.1016/j.procs.2014.11.013
2. P. Avery, J. Togelius, E. Alistar, R. van Leeuwen. Computational intelligence and tower defence games. IEEE Congress of Evolutionary Computation, New Orleans, USA, 2011, 1084-1091, DOI: 10.1109/CEC.2011.5949738. https://doi.org/10.1109/CEC.2011.5949738
3. R. Brociek, D. Słota. Application and comparison of intelligent algorithms to solve the fractional heat conduction inverse problem. Information Technology and Control, 2016, 45(2), 184-194, DOI: 10.5755/j01.itc.45.2.13716. https://doi.org/10.5755/j01.itc.45.2.13716
4. R. Brociek, D. Słota. Application of intelligent algorithm to solve the fractional heat conduction inverse problem. Communications in Computer and Information Science – ICIST'2016, 2016, 639, 369-379, DOI: 10.1007/978-3-319-46254-7_29. https://doi.org/10.1007/978-3-319-46254-7_29
5. K. Cpałka, K. Łapa, A. Przybyl. A new approach to design of control systems using genetic programming. Information Technology and Control, 2015, 44(4), 433-442, DOI: 10.5755/j01.itc.44.4.10214. https://doi.org/10.5755/j01.itc.44.4.10214
6. R. Damaševičius, G. Ziberkas. Energy Consumption and Quality of Approximate Image Transformation. Elektronika Ir Elektrotechnika, 2012, 4, 79-82, DOI: 10.5755/j01.eee.120.4.1459. https://doi.org/10.5755/j01.eee.120.4.1459
7. P. Dziwiński, L. Bartczuk, J. T. Starczewski. Fully Controllable Ant Colony System for Text Data Clustering. Lecture Notes in Computer Science – ICAISC'2012, 2012, 7269, 199-205, DOI: 10.1007/978-3-642-29353-5. https://doi.org/10.1007/978-3-642-29353-5
8. P. Dziwiński, L. Bartczuk, A. Przybyl, E. Avedyan. A New Algorithm for Identification of Significant Operat-

ing Points Using Swarm Intelligence. Lecture Notes in Artificial Intelligence – ICAISC'2014, 2014, 8468, 349-362, DOI: 10.1007/978-3-319-07176-3_31. https://doi.org/10.1007/978-3-319-07176-3_31

9. E. Hetmaniok, D. Słota, A. Zielonka. Experimental Verification of Immune Recruitment Mechanism and Clonal Selection Algorithm Applied for Solving the Inverse Problems of Pure Metal Solidification. Int. Comm. Heat & Mass Transf., 2013, 47, 7-14, DOI: j.icheatmasstransfer.2013.07.009.
10. E. Hetmaniok, D. Słota, A. Zielonka. Solution of the Inverse Continuous Casting Problem with the Aid of Modified Harmony Search Algorithm. Lecture Notes in Computer Science – PPAM'2014, 2014, 8384, 402-411, DOI: 10.1007/978-3-642-55224-3_38. https://doi.org/10.1007/978-3-642-55224-3_38
11. A. Hirao, Y. Nomura, H. Yonezu, H. Takeshita. Prim's algorithm based p2mp energy-saving routing design for midori. IEEE COIN 2012 – IEEE International Conference on Optical Internet, Yokohama, Kanagawa, 2012, 29-31, 86-93.
12. M. Swiechowski, J. Mandziuk. Fast interpreter for logical reasoning in general game playing. J. Log. Comput., 2016, 26(5), 1697-1727, DOI 10.1093/logcom/exu058. https://doi.org/10.1093/logcom/exu058
13. M. Korytkowski, L. Rutkowski, R. Scherer. Fast image classification by boosting fuzzy classifiers. Information Sciences, 2016, 327, 175-182, DOI: 10.1016/j.ins.2015.08.030. https://doi.org/10.1016/j.ins.2015.08.030
14. R. Lara-Cabrera, C. Cotta, A. Fernández-Leiva. A review of computational intelligence in RTS games. IEEE Symposium on Foundations of Computational Intelligence, Singapore, 2013, 16-19, 114-121, DOI: 10.1109/FOCI.2013.6602463. https://doi.org/10.1109/FOCI.2013.6602463
15. S. Lucas, G. Kendall. Evolutionary computation and games. IEEE Computational Intelligence Magazine, 2006, 1(1), 10-18, DOI: 10.1109/MCI.2006.1597057. https://doi.org/10.1109/MCI.2006.1597057
16. S. Lucas. Computational intelligence and games: Challenges and opportunities. International Journal of Automation and Computing, 2008, 5(1), 45-57, DOI: 10.1007/s11633-008-0045-8. https://doi.org/10.1007/s11633-008-0045-8
17. L. Najman, J. Cousty, B. Perret. Playing with kruskal: algorithms for morphological trees in edge-weighted graphs. Lecture Notes in Computer Science – MMASIP'2013, 2013, 7883, 135-146.
18. C. Napoli, G. Pappalardo, E. Tramontana. A mathematical model for file fragment diffusion and a neural predictor to manage priority queues over BitTorrent. Applied Mathematics and Computer Science, 2016, 26(1), 147-160, DOI: 10.1515/amcs-2016-0010. https://doi.org/10.1515/amcs-2016-0010
19. J. Mandziuk, A. Zychowski. A memetic approach to vehicle routing problem with dynamic requests. Appl. Soft Comput., 2016, 48, 522-534, DOI 10.1016/j.asoc.2016.06.032. https://doi.org/10.1016/j.asoc.2016.06.032
20. D. Połap, M. Woźniak, C. Napoli, E. Tramontana. Real-Time Cloud-based Game Management System via Cuckoo Search Algorithm. International Journal of Electronics and Telecommunications, 2015, 61(4), 333-338, DOI: 10.1515/eletel-2015-0043. https://doi.org/10.1515/eletel-2015-0043
21. D. Połap, M. Woźniak, C. Napoli, E. Tramontana. Is swarm intelligence able to create mazes? International Journal of Electronics and Telecommunications, 2015, 61(4), 305-310, DOI: 10.1515/eletel-2015-0039. https://doi.org/10.1515/eletel-2015-0039
22. I. Martisius, R. Damaševičius. A Prototype SSVEP Based Real Time BCI Gaming System. Comp. Int. and Neurosc., 2016, DOI: 10.1155/2016/3861425. https://doi.org/10.1155/2016/3861425
23. L. Rutkowski, M. Jaworski, L. Pietruczuk, P. Duda. A New Method for Data Stream Mining Based on the Misclassification Error. IEEE Trans. Neural Netw. Learning Syst., 2015, 26(5), 1048-1059, DOI: 10.1109/TNNLS.2014.2333557. https://doi.org/10.1109/TNNLS.2014.2333557
24. M. Swiechowski, J. Mandziuk, Yew Soon Ong: Specialization of a UCT-Based General Game Playing Program to Single-Player Games. IEEE Trans. Comput. Intellig. and AI in Games, 2016, 8(3), 218-228, DOI 10.1109/TCIAIG.2015.2391232. https://doi.org/10.1109/TCIAIG.2015.2391232
25. J. Togelius, R. Nardi, S. Lucas. Towards automatic personalised content creation for racing games. IEEE Symposium on Computational Intelligence and Games, Honolulu, Hawaii, 2007, 1-5, 252-259, DOI: 10.1109/CIG.2007.368106. https://doi.org/10.1109/CIG.2007.368106
26. J. Toldinas, R. Damaševičius, A. Venckauskas, T. Blazauskas, J. Ceponis. Energy Consumption of Cryp-

tographic Algorithms in Mobile Devices. Elektronika Ir Elektrotechnika, 2014, 20(5), 158-161, DOI: 10.5755/j01.eee.20.5.7118. https://doi.org/10.5755/j01.eee.20.5.7118

27. M. Wooldridge, P. Dunne. On the computational complexity of coalitional resource games. Artificial Intelligence, 2006, 170(10), 835–871, DOI: 10.1016/j.artint.2006.03.003. https://doi.org/10.1016/j.artint.2006.03.003
28. T. Kapuścinski, R. K. Nowicki, C. Napoli: Application of Genetic Algorithms in the Construction of Invertible Substitution Boxes. Lecture Notes in Artificial Intelligence – ICAISC'2016, 2016, 380-391, DOI: 10.1007/978-3-319-39378-0_33. https://doi.org/10.1007/978-3-319-39378-0_33
29. X-S. Yang. Flower pollination algorithm for global optimization. Unconventional computation and natural computation, Springer, Berlin-Heidelberg, 2012, 240-249. https://doi.org/10.1007/978-3-642-32894-7_27
30. X-S. Yang. A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010), Springer Berlin Heidelberg, 2010, 65-74. https://doi.org/10.1007/978-3-642-12538-6_6
31. G. Yannakakis, J. Togelius. A Panorama of Artificial and Computational Intelligence in Games. IEEE Trans. Comput. Intellig. and AI in Games, 2015, 7(4), 317-335, DOI: 10.1109/TCIAIG.2014.2339221. https://doi.org/10.1109/TCIAIG.2014.2339221

Summary / Santrauka

Creation of the layout for a board given to a player is usually achieved by means of complex algorithm that is detailing position of walls, passages, etc. We propose a novel solution based on bio-inspired algorithms adapted to easily create and then reorganize layouts of board games. The resulting layout presents a maze that is being automatically generated with given entry and exit point. This structures, because of proposed methodology, can be recomposed during the game. We show execution time of proposed algorithms and discuss efficiency in composition of mazes in various dimensions.

Žaidėjų naudojamų stalo žaidimų lentų išplanavimas dažniausiai kuriamas naudojant kompleksinį algoritmą, kuris detalizuoja sienų, praėjimų ir kt., poziciją. Mes siūlome naują sprendimą, pagrįstą bio-įkvėptais algoritmais, kuriuos pritaikius galima nesunkiai sukurti ir reorganizuoti stalo žaidimų lentų išplanavimą. Gaunamas išplanavimas turi automatiškai sugeneruotą labirintą su įėjimu ir išėjimu. Siūlomos metodikos dėka, šios struktūros gali būti reorganizuojamos žaidimo metu. Straipsnyje parodome siūlomų algoritmų vykdymo laiką ir aptariame įvairių dimensijų labirintų struktūros efektyvumą.