## Modelling of Grey Wolf Optimization Algorithm Using 2D P Colonies

Daniel Valenta<sup>1</sup>, Lucie Ciencialová<sup>1,2</sup>, Miroslav Langer<sup>1,2</sup>, and Luděk Cienciala<sup>1,2</sup>

<sup>1</sup> Institute of Computer Science, Silesian University in Opava, Czech Republic

<sup>2</sup> Research Institute of the IT4Innovations Centre of Excellence, Silesian University in Opava, Czech Republic {daniel.valenta, lucie.ciencialova, miroslav.langer, ludek.cienciala }@fpf.slu.cz

Abstract: The P colonies are a well-established version of the P systems, the computational device based on membrane computing. One branch of the research of the P colonies focuses on the possibility to consider the twodimensional environment, in which the agents act, and the 2D P colonies were introduced. 2D P colonies showed to be suitable for the simulations of various (not only) multi-agent systems, and natural phenomena, like the flash floods.

However, the agents of the 2D P colony are able to communicate via the environment, by leaving some special symbols in it, this may not be sufficient for simulating some more complex communities of the agents, for example the hunting pack of the grey wolves.

In this paper, we introduce a formal model of the 2D P colonies with the blackboard as an extension of the 2D P colonies. We also allow the agents to use simple relational operations to compare real numbers that can be stored inside special symbols. We use this model to simulate the Grey wolf algorithm.

Keywords: pack algorithm, P system, 2D P colony, optimization, multi-agent system, blackboard.

#### 1 Introduction

2D P colony (see [1]) is a variant of P colonies (see [2]), a community of agents set in the two dimensional environment. It is a very simple membrane system originally derived from the P systems (see [3]).

2D P colony consists of a finite number of agents living in a shared environment. The environment of 2D P colony is represented by a 2D grid of square cells. In each cell, there is a multiset of objects. Each agent is represented by a finite collection of objects enclosed with a membrane. The agent has programs consisting of rules. These rules are of three types: they may change the objects of the agents and they can be used for interacting with the joint shared environment of the agents and movement rule. The direction of the movement of the agent is determined by the contents of cells surrounding the cell in which the agent is placed. The program can contain at most one motion rule. There is one more condition set to achieve the greatest simplicity in agent's behaviour. If the agent moves, it cannot communicate with the environment. So, if the program contains a

motion rule, then the other rule is an evolution rule. The number of objects inside each agent is set by definition and it is usually a very small; one, two or three.

The agent itself has information neither about its position nor about the position of the other agents in the environment. The communication between the agents is also possible only via the environment by leaving special symbols in it. These factors are limiting the 2D P colonies in simulating more advanced communities of agents. For the remote information exchange, we add the agents the possibility to store and read the information from a blackboard. The blackboard is a read/write device with an unchangeable structure given by definition. The agents can change only defined parts of the blackboard.

#### **Grey Wolf Optimization Algorithm** 2

Grey wolf optimization algorithm (GWO) (see [4]) is already well-established meta-heuristic optimization technology. It is inspired by social dynamics found in packs of grey wolves and by their ability to dynamically create hierarchies in which every member has a clearly defined role. Each wolf can fulfil one of the following role:

- Alfa is the dominant pair and the pack follows their lead for example during hunts or while locating a place to sleep.
- Beta wolves support and respect the Alpha pair during its decisions.
- Delta wolves are subservient to Alpha and Beta wolves, follow their orders, and control Omega wolves. We divide them into scouts - they observe the surrounding area and warn the pack, sentinels they protect the pack when endangered and caretakers - they provide aid to old and sick wolves.
- Omega wolves help to filter the aggression of the pack and frustrations by serving as scapegoats.

The primary goal of the wolves is to find and hunt down prey in their environment. The prey equals the optimal solution to the given problem. The environment is represented by a mathematical fitness function characterizing the problem. The value of the function at the current position of the particular wolf represents the highest-quality prey. The wolf with the best value is ranked as Alpha, the second one as Beta, third as Delta, and all the other are

Copyright ©2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

#### Omegas.

The hunting technique of a wolf pack can be divided into 5 steps:

- *Search for the prey* wolves are attempting to find the most valuable prey with respect to the effort required to hunt it successfully.
- *Exploitation of the prey* wolves are attempting to draw attention to themselves and to separate the prey from its herd.
- *Encircling prey* the attempt to push the prey into a situation from which it cannot escape.
- The prey is surrounded it can no longer escape.
- *The attack* wolves attack the weak spots of the prey (belly, legs, snout) until it succumbs to fatigue. Afterwards, they bring it down and crush its windpipe.

The algorithm is inspired by this process and smoothly transitions between scouting and hunting phases. In the scouting phase, the pack extensively scouts its environment through many random movements so that the algorithm does not get stuck in a local extreme, while in the hunting phase, the influence of random movements is slowly reduced and pack members draw progressively closer to the discovered extreme. To maintain the divergence between those phases, each wolf is assigned vectors  $\vec{A}$  and  $\vec{C}$ .

 $\vec{A}$  is a vector with components rand(-1,1)\*a, where rand(-1,1), generates a random number between -1 and 1 and where

$$a = 2 - \left(\frac{2i}{i_{max}}\right),$$

while *i* is the algorithm's current iteration, and  $i_{max}$  is the maximum number of iterations.

Vector  $\vec{A}$  is random value between -2 and 2. We can see its impact in the Fig. 1. With growing iterations, it is more likely that its value will be between -1 and 1. That makes it more likely for a wolf to be hunting. Another



Figure 1: Vector  $\vec{A}$  and its impact in 1D

component supporting the scouting phase is vector  $\vec{C} = rand(0,2),$ 

where rand(0,2), generates a random number between 0 and 2. Vector  $\vec{C}$  is like vector  $\vec{A}$ , but iterations don't influence it. It helps the wolves behave more naturally. Analogously, in nature, wolves encounter various obstacles which prevent them from approaching prey comfortably. Vectors  $\vec{A}$  and  $\vec{C}$  encourages wolves to prefer scouting, or hunting, and so to avoid local optima regardless of the algorithm's current iteration.

Wolves' positions within the environment are updated based upon the estimated location of the prey using Alpha, Beta, and Delta wolves as guides.

Let  $\vec{X}_j(i)$  be a position vector of wolf *j* in *i*-th iteration. The position vector of wolf *j* is updated as follows:

$$\vec{X}_{j}(i+1) = \frac{\vec{X}_{1} + \vec{X}_{2} + \vec{X}_{3}}{3},$$

where *i* is the current iteration of the algorithms, and  $\vec{X}_1$ ,  $\vec{X}_2$ ,  $\vec{X}_3$  are new potential position vectors of Alpha, Beta, and Delta wolves obtained from following formulas:

$$\begin{array}{rcl} \vec{X}_1 &=& \vec{X}_{\alpha}\left(i\right) - \vec{A}_1 * \vec{D}_{\alpha} \\ \vec{X}_2 &=& \vec{X}_{\beta}\left(i\right) - \vec{A}_2 * \vec{D}_{\beta} \\ \vec{X}_3 &=& \vec{X}_{\delta}\left(i\right) - \vec{A}_3 * \vec{D}_{\delta} \end{array}$$

where  $\vec{X}_{\alpha}(i)$ ,  $\vec{X}_{\beta}(i)$ ,  $\vec{X}_{\delta}(i)$  are the position vectors of Alpha, Beta, and Delta wolves, they are representing the positions in the environment that are closest to the optimum in *i*-th iteration. The vectors  $\vec{A}_1, \vec{A}_2, \vec{A}_3$  are calculated in the same way as vector  $\vec{A}$ . The vectors  $\vec{D}_{\alpha}, \vec{D}_{\beta}, \vec{D}_{\delta}$  are defining the distance of the wolf *j* position from the prey as follows:

$$\begin{array}{lll} \vec{D}_{\alpha} & = & \left| \vec{C}_{1} \ast \vec{X}_{\alpha} \left( i \right) - \vec{X}_{j} \left( i \right) \right| \\ \vec{D}_{\beta} & = & \left| \vec{C}_{2} \ast \vec{X}_{\beta} \left( i \right) - \vec{X}_{j} \left( i \right) \right| \\ \vec{D}_{\delta} & = & \left| \vec{C}_{3} \ast \vec{X}_{\delta} \left( i \right) - \vec{X}_{j} \left( i \right) \right| \end{array}$$

where  $|\vec{X}|$  is the vector whose components are the absolute values of the components of  $\vec{X}$ .

The vectors  $\vec{C}_1, \vec{C}_2, \vec{C}_3$  are computed in the same way as vector  $\vec{C}$ , and they influence the weight of the estimated position of the prey  $\vec{X}_{\alpha}, \vec{X}_{\beta}, \vec{X}_{\delta}$ , increasing or decreasing it.

We can see in Fig. 2 that thanks to this principle wolves have the tendency to move closer towards prey and encircle it (wolves approach from different directions).



Figure 2: Positional updates of Omega wolves as it is described by the mathematical formula [4]

### 2.1 Algorithm pseudocode

In this subsection we describe the algorithm in pseudocode. The inputs of the algorithm are dimensions of the environment of the problem, boundaries of the environment of the problem, fitness function characterizing the problem, size of the pack (number of wolves/agents), number of iterations of the algorithm, termination criteria and criteria of the fitness function.



Figure 3: Algorithm steps

The algorithms pseudocode follows:

- In the first step, agents (wolves) are randomly spread out across the environment.
- In each iteration *i*:
  - calculate the fitness value of each agent and determine the social hierarchy Fig. 3. part 1. The agent with the best value (closest to the optimum) is Alpha, second best is Beta, third best is Delta, and all others are Omega.
  - calculate the best solution found thus far by Alpha, Beta and Delta  $(\vec{X}_{\alpha}(i), \vec{X}_{\beta}(i), \vec{X}_{\delta}(i))$  and average it Fig. 3. part 2,
  - update positions of all the wolves  $X_j(i+1)$ , while random vectors  $\vec{A}$  and  $\vec{C}$  are updated for each one – Fig. 3. part 3,
  - check the termination criterion Fig. 3. part 4. Iterations terminate when fitness function value reaches a preset value.

## 3 2D P Colonies

In this section, we recall the definition of 2D P colonies and other terms related to them.

#### **Definition 1.** A 2D P colony is a construct

where

- $\Pi = (A, e, Env, B_1, \ldots, B_k, f), k \geq 1,$
- A is an alphabet of the colony, its elements are called objects,

- *e* ∈ *A* is the basic environmental object of the 2D P colony,
- Env is a pair (m×n,w<sub>E</sub>), where m×n,m,n ∈ N is the size of the environment and w<sub>E</sub> is the initial contents of the environment, it is a matrix of size m×n of multisets of objects over A - {e}.
- $B_i$ ,  $1 \le i \le k$ , are agents, each agent is a construct  $B_i = (o_i, P_i, [o, p]), 0 \le o < m, 0 \le p < n$ , where
  - $o_i$  is a multiset over A, it determines the initial state (contents) of the agent,  $|o_i| = 2$ ,
  - $P_i = \{p_{i,1}, \dots, p_{i,l_i}\}, l \ge 1, 1 \le i \le k \text{ is a finite set of programs, where each program contains exactly 2 rules, which are in one of the following forms each:$ 
    - \*  $a \rightarrow b$ , called the evolution rule,  $a, b \in A$ ,
    - \*  $c \leftrightarrow d$ , called the communication rule,  $c, d \in A$ ,
    - \*  $[a_{q,r}] \rightarrow s, 0 \le q, r \le 2, s \in \{\Leftarrow, \Rightarrow, \Uparrow, \Downarrow\},$ called the motion rule,
- $f \in A$  is the final object of the colony.

A configuration of the 2D P colony is given by the state of the environment - matrix of type  $m \times n$  with multisets of objects over  $A - \{e\}$  as its elements, and by the state of all agents - pairs of objects from alphabet A and the coordinates of the agents. An initial configuration is given by the definition of the 2D P colony.

A computational step consists of three parts. The first part lies in determining the set of applicable programs according to the current configuration of the 2D P colony. In the second part, we have to select from this set one program for each agent, in such a way that there is no collision between the communication rules belonging to different programs. The third part is the execution of the chosen programs.

A change of the configuration is triggered by the execution of programs and it involves changing the state of the environment, contents and placement of the agents.

A computation is non-deterministic and maximally parallel. The computation ends by halting when there is at least one agent that has no applicable program.

The result of the computation is the number of copies of the final object placed in the environment at the end of the computation.

The aim of introducing 2D P colonies is not studying their computational power but monitoring their behaviour during the computation.

## 4 Modelling of Grey wolf optimization algorithm using 2D P colonies

As for the modelling of GWO, some similarities as well as a few differences have been found. Both are inspired by the nature, usable for solving optimization problems, and both are multi-agent system models. For comparison see the differences / problems:

- Environmental problem,
- Communication problem,
- Randomness problem.

These problems are described in the Table 1.

 Table 1: Differences between Grey wolf algorithm and P colony

Difference /	Grey wolf algo-	2D P colony	
System	rithm		
Environmental	The environment	The environment	
problem	is represented by	is represented by	
	a mathematical	a multiset of sym-	
	fitness function.	bols in each cell	
		of 2D grid.	
Communication	The agents have	They are commu-	
problem	the knowledge of	nities of simple	
	their global posi-	reactive agents	
	tion in the envi-	independently	
	ronment.	living and acting	
		in a joint shared	
		environment.	
Randomness	Random vectors	Each rule is	
problem	A and C influence	deterministic,	
	the movement of	the only way	
	wolves in the en-	to implement	
	vironment.	randomness is	
		to randomize	
		the choosing	
		rule for identical	
		configurations.	

The definition of the 2D P colony needs to be adjusted to meet described requirements. Proposed solutions for those problems are described in the following subsections.

#### 4.1 Environmental problem solution

The environment will be a pair of matrix  $m \times n$  and fitness function  $f_E$ , where  $m \times n$ ,  $m, n \in \mathbb{N}$  is the size of the environment and  $f_E$  is a mathematical function with the initial contents of environment. Alphabet *V* will contains objects that can store real numbers and common objects,  $V = \{ [...], [...]', [...]', ... \} \cup \{a, b, c, d, e, f, g, ... \}$ , The rules of the programs, which guide the agents, will compare the number values of the objects using operators greater than ">" and greater or equal to " $\geq$ ". The agent  $A_i$  will be defined as  $A_i = (o_i, P_i, [o, p])$ ,  $|o_i| = 2$ . For example, environment can be defined as  $Env = (4 \times 4, w_E, f_E)$ , with object *a* in each cell, and fitness function  $f_E$  is the Ackermann function

$$f_E(x,y) = \begin{cases} y+1 & \text{for } x = 0\\ f_E(x-1,1) & \text{for } m > 0 \text{ and } n = 0\\ f_E(x-1,f_E(x,y-1)) & \text{otherwise} \end{cases}$$

1 a	2 a	3 a	4 a
2 a	3 a	4 a	5 a
3 a	5 a	7 a	9 a
5 a	13 a	29 a	61 <i>a</i>

Figure 4: Graphical representation of the environment

#### 4.2 Communication problem solution

We extend the 2D P colony by adding the blackboard that saves the agents' best fitness values and is always accessible to read and write by all agents. To simulate GWO agents cannot decide how to continue with hunting without knowledge of their position in the environment, because the fitness value is not enough for calculating the prey's estimated position. That is why the approximate position is stored in the blackboard. The position of each agent is computed by a function using moving receivers (see Section 6. for more details) We will use the blackboard as a



Figure 5: Simple Blackboard example

means of giving feedback to agents. Agents do not need to know their position in the environment, all agents who can contribute to the search will send their solution to the blackboard points called receivers, and estimation of prey position is calculated as an average of distances collected by blackboard points from wolves Alpha, Beta and Delta. Omega wolves can ping the blackboard if changing position and get their distance from the prey. If the distance would decrease compared to the original distance, then the wolf will move.



Figure 6: Blackboard structure

#### 4.3 Randomness problem solution

The omega wolves movement is based not only on position of the prey but also on random vectors  $\vec{A}$  and  $\vec{C}$ . In 2D P Colony model the randomness is replaced by non-deterministic choice between several applicable rules.

## 5 Model of numerical 2D P colony with the blackboard

Let us consider that there is not only multiset of objects in each environmental cell, but there is also one (natural or real) number computed by a function (called an environmental function) depending on the position of the cell and time. This number can be read by the agent, but it can be changed only by the environmental function.

The number can be stored inside the agent inside special kind of objects (box-objects). To read an environmental number the agent needs a special reading rule. Using this rule agent rewrites an object inside of it into the special object that contains the number that equals the environmental number. This rule is in the form:  $a \rightarrow x$ , where  $a \in V$  is an object and x is the environmental number in the cell, where the agent executing the rule is placed. The environmental number can be read by more than one agent at the same time, and each agent can use the reading rule multiple times in one program. For example, let ab is the content of agent with program  $p : \langle a \rightarrow x \rangle$ , and 5 is the number stored in the environmental cell where the agent is placed. After execution of the program p the content of agent is [5].

If there are two objects with numbers inside the agent, an execution of particular program can be restricted by these numbers. Agent can compare the numbers, and distinguish which rule will be applied to which object. The programs with such condition look like follows:  $\langle x > y : r_1, r_2 \rangle$  or  $\langle x \ge y : r_1, r_2 \rangle$ , where  $x, y \in \mathbb{R}$  and  $r_1, r_2$  are rules that work with objects with numbers. These box-objects can be of different kind but both must store a number.

The memory of 2D P colony is extended by a blackboard. It is a pair  $BB = (\vec{f}_{BB}, BM)$ , where  $\vec{f}_{BB}$  is a vector of *i* functions  $f_i$ , and BM is a matrix of size  $i \times j$ ,  $i, j \in \mathbb{N}$ .  $f_i$  is a function that fills the members of corresponding row of matrix BM with values. The function can update only such members that are not updated by agents.

To access and affect the blackboard, the agents have the rules for reading and writing a value placed in a specified part of the blackboard. These rules use functions Get(BB[ad],b) and Update(b,BB[ad]), where BB[ad] is an address of value on the blackboard which have to read or written, and  $b \in V$  is the box-object. The get rule is in a form  $a \rightarrow Get(BB[ad],b)$  and the update rule consist only from the function Update(b,BB[ad]), where  $a, b \in V$ . When the rule  $a \rightarrow Get(BB[2,1], [])$  is executed, the object a inside the agent is evolved into the object [], and this object is filled in with the number stored

on the blackboard at position [2,1]. By execution of rule Update(15'', BB[0,4]), the agent with object 15'' places number 15 into the fifth member in the first row of the blackboard. Only one agent can read or update value in one position at the same time. If two agents have applicable programs that affect the same position in blackboard, then only one of them can execute its program. The other agent must use another applicable program, or be inactive if another applicable program does not exist. The agent able to write on the blackboard is chosen non-deterministically.

All newly introduced rules can be combined in the programs with any kind of rules, except the movement rule, that can be paired only with the rewriting rule.

**Definition 2.** A numerical 2D P colony with blackboard is a construct

 $\Pi = (V, e, Env, A_1, ..., A_k, BB, f), k \ge 1$ , where

- *V* is an alphabet of the colony, its elements are called objects, there are special objects that can contain an arbitrary number,
- *e* ∈ *V* is the basic environmental object of the 2D *P* colony,
- Env is a triplet  $(m \times n, w_E, f_E)$ , where  $m \times n, m, n \in N$ is the size of the environment, and  $w_E$  is the initial contents of the environment. It is a matrix of size  $m \times$ n of multisets of objects over  $V - \{e\}$ , and  $f_E$  is an environmental function.
- $A_i$ ,  $1 \le i \le k$ , are agents. Each agent is a construct  $A_i = (o_i, P_i, [o, p]), \ 0 \le o \le m, \ 0 \le p \le n$ , where
  - $o_i$  is a multiset over V, it determines the initial state (contents) of the agent,  $|o_i| = 2$ ,
  - $P_i = \{p_{i,1}, \dots, p_{i,l_i}\}, l \ge 1, 1 \le i \le k \text{ is a finite set of programs, where each program contains exactly 2 rules. Each rule is in the following form:$ 
    - \*  $a \rightarrow b$ , the evolution rule,  $a, b \in V$ ,
    - \*  $c \leftrightarrow d$ , the communication rule,  $c, d \in V$ ,
    - \*  $[a_{q,r}] \rightarrow s, a_{q,r} \in V, 0 \le q, r \le 2, s \in \{\Leftarrow, \Rightarrow, \uparrow, \downarrow\}, the motion rule,$
    - \*  $a \rightarrow x$ ,  $x \in \mathbb{R}, a, x \in V$ , the reading rule

If the program contains evolution or communication rule  $r_1$ ,  $r_2$  that each works with objects with numbers, it can be extended by a condition:  $\langle x > y : r_1, r_2 \rangle$ ,  $\langle x \ge y : r_1, r_2 \rangle$ ,

-  $[o, p], 1 \le o \le m, 1 \le p \le n$ , is an initial position of agent  $A_i$  in the 2D environment,

#### • $f \in V$ is the final object of the colony.

A configuration of the 2D P colony is given by the state of the environment - matrix of type  $m \times n$  with pairs - multiset of objects over  $V - \{e\}$ , and a number - as its elements, and by the states of all agents - pairs of objects from the alphabet V, and the coordinates of the agents. An initial configuration is given by the definition of the 2D P colony.

A computational step consists of three phases. In the first step, the set of the applicable programs is determined according to the current configuration of the 2D P colony. In the second step, one program from the set is chosen for each agent, in such a way that there is no collision between the communication rules belonging to different programs. In the third step, chosen programs are executed, the values of the environment and on the blackboard are updated.

A change of the configuration is triggered by the execution of programs, and updating values by functions. It involves changing the state of the environment, contents and placement of the agents.

A computation is non-deterministic and maximally parallel. The computation ends by halting when there is no agent that has an applicable program.

The result of the computation is the number of copies of the final object placed in the environment at the end of the computation.

# 5.1 Numerical 2D P Colony with the Blackboard for GWO

Because of a restricted number of pages of this contribution we show an idea of the construction of a numerical 2D P colony with the blackboard that models GWO in finding extreme of a function with two variables in this section.

 $P_{gw} = (V, e, Env, A_1, A_2, \dots, A_k, BB, f), k \ge 1$ , where:

- $V = \{ [., [.]', [.]'', [.]'', [.]^{iv}, [.]^{v}, [.]^{vi} \} \cup \\ \cup \{e, f, a', b, c, d, h, h', h'', m_{OK}, m_{KO}, m'_{KO}, m'_{KO} \} \cup \\ \cup \{n, A, B, D\} \cup \{l, l', l'', l''', l^{iv} \mid Y \in \{\Leftarrow, \Rightarrow, \Uparrow, \Downarrow\} \} \cup \\ \cup \{k_{z_1 z_2 z_3 z_4} \mid z_i \in \{\Leftarrow, \Rightarrow, \Uparrow, \Downarrow\} \land k, i \in \{1, 2, 3, 4\} \}$
- $e \in V$  is the basic environmental object,
- *Env* is a triplet  $(i \times j, w_e, f(x, y))$ , where  $i, j \in \mathbb{N}$ ,  $w_E = |a_{r,s}|, a_{r,s} = \varepsilon, 1 \ge r \ge i, 1 \ge s \ge j$ ,
- $A_1, A_2, \dots, A_k$  are the agents,  $A_i = (O_i, P_i, [r_x, r_y])$ , where:
  - $-|o_i|=2,$
  - $P_1 = P_2 = \cdots = P_k$ ,  $P_i$  rules are defined below,
  - [*r*,*s*] are the initial coordinates,
- *BB* is the blackboard, to the definition of the blackboard is devoted a separate subsection of this chapter,
- f is the final object,  $f \in V$ .

The initial agent's configuration is: ee and its position is [r,s].

Programs  $P_i$  associated with i-th agent are:

- 1.  $\langle e \rightarrow x'; e \rightarrow Get(BB[alpha, ]) \rangle x \in \mathbb{R}$  is number placed in the environmental cell [r, s], *alpha* is address of current value *y* of alpha wolf in blackboard; This program is to read the number from the environmental cell and the value of current alpha wolf in blackboard.
- 2. The following set of programs *Compare* is to compare two numbers stored inside the agent:
  - (a)  $\langle x > y : [x] \to [x], [y]' \to A \rangle I$  am new Alpha,
  - (b)  $\langle x \ge y : x' \to b, x \to x \rangle$
  - (c)  $\langle x \to x, b \leftarrow Get(BB[beta], []') \rangle$ ,
  - (d)  $\langle x > y : [x] \to [x], [y]'' \to B \rangle$  I am new Beta,
  - (e)  $\langle x \ge y : x'' \to c, x \to x \rangle$
  - (f)  $\langle x \rightarrow x, c \leftarrow Get(BB[BB[gamma], ]'''], \rangle \rangle$ ,
  - (g)  $\langle x > y : [x] \to [x], [y]''' \to C \rangle$  I am new Delta,
  - (h)  $\langle x \ge y : x | '' \to d, x \to x | i^{\nu} \rangle$  I am Omega,
- 3. If there is *A*, *B* or *C* inside the agent, the agent updates the blackboard using programs:
  - (a)  $\langle Update([x], BB[alpha]), A \to a' \rangle$ ,
  - (b)  $\langle Update([x], BB[beta]), B \to a' \rangle$ ,
  - (c)  $\langle Update([x], BB[gamma]), C \rightarrow a' \rangle$ ,
  - (d)  $\langle x \to e, a' \to e \rangle$ ,
- 4. If the agent is an omega wolf, the agent is supposed to move so as to approach the prey. The agent reads its distance from the prey computed by the function and placed on the blackboard, and it moves in a random direction. The direction is generated in such a way, that the agent creates the object 1 with a low index formed from four directions in random order. 1 means that the agent will move in the first direction.
  - (a)  $\langle d \to 1_w, [x]^{i\nu} \to Get(BB[dist. from prey], []^{\nu}) \rangle$ ,  $w = z_1 z_2 z_3 z_4, \quad z_i \in \{\Rightarrow, \Leftarrow, \Uparrow, \Downarrow\},$   $z_1 \neq z_2 \neq z_3 \neq z_4$ (b)  $\langle 1_w \leftrightarrow e, [x]^{\nu} \to [x]^{\nu} \rangle$ .

(b) 
$$\langle \mathbf{I}_{W} \leftrightarrow e, \underline{x} \rangle \rightarrow \underline{x} \rangle$$
,  
(c)  $\left\langle \begin{bmatrix} e & e & e \\ e & X_{z_{1}z_{2}z_{3}z_{4}} & e \\ e & e & e \end{bmatrix} \rightarrow z_{X}, e \rightarrow z_{X}$   
 $X \in \{1, 2, 3, 4\}, z_{X} \in \{\Rightarrow, \Leftarrow, \uparrow, \Downarrow\}$ 

Then the agent puts the object with movement sequence into environmental cell and reads its distance from the prey from new location.

(a)  $\langle z_X \to z'_X, x^{\nu} \leftrightarrow e \rangle$ ,

(b) 
$$\langle z'_X \to z''_X, e \to h \rangle$$
,  
(c)  $\langle z''_X \leftrightarrow \underline{x}^{\nu}, h \to Get(BB[dist. from prey], \underline{}^{\nu i}) \rangle$ ,

5. If the new value is smaller then value from previous location, the agent consumes object corresponding to direction and object with movement sequence and rewrite its content to *ee*.

(a) 
$$\left\langle x \ge y : x^{\nu} \to m_{OK}, y^{\nu i} \to h' \right\rangle$$
,  
(b)  $\left\langle x > y : x^{\nu i} \to m_{KO}, y^{\nu} \to h' \right\rangle$ ,

(c) 
$$\langle m_{OK} \to m_{OK}, h' \leftrightarrow z_X'' \rangle$$
,

(d)  $\left\langle \begin{bmatrix} e & e & e \\ e & e & e \\ e & e & e \end{bmatrix} \rightarrow \overline{z}_X, z_X'' \rightarrow z_X''' \right\rangle$ ,  $\overline{z}_X$  is opposite movement to  $z_X$ ,

(e) 
$$\langle m_{OK} \leftrightarrow X_w, z_X'' \to z_X^{i\nu} \rangle$$
,  
(f)  $\left\langle \begin{bmatrix} e & e & e \\ e & e & e \\ e & e & e \end{bmatrix} \to z_X, z_X^{i\nu} \to h'' \right\rangle$   
(g)  $\langle X_w \to e, h'' \to e \rangle$ ,

- 6. If the new distance is greater then old one, the agent consumes object corresponding to the direction and moves to original location and rewrite the object 1 with movement sequence into object 2 with the same movement sequence, and the agent continues with investigation. If the agent moves back and there is object 4 with movement sequence, it did not find a smaller distance to the prey and it stops working.
  - (a)  $\langle m_{KO} \rightarrow m_{KO}, h' \leftrightarrow z_X'' \rangle$ ,
  - (b)  $\langle m_{KO} \rightarrow m'_{KO}, z_X''' \rightarrow e \rangle$ ,
  - (c)  $\langle m'_{KO} \rightarrow m'_{KO}, e \leftrightarrow X_w \rangle$ ,
  - (d)  $\langle m'_{KO} \rightarrow m''_{KO}, X_w \rightarrow (X+1)_w \rangle, X \in \{1,2,3\}$
  - (e)  $\langle m''_{KO} \to Get(BB[dist. from prey], []^{v}), X_{w} \leftrightarrow e \rangle, X \in \{2, 3, 4\},$
  - (f)  $\langle m'_{KO} \to f, e \leftrightarrow 4_w \rangle$ ,

The computation is possibly non-halting because of the three agents (alpha, beta and gamma) can always find applicable program. The positions of these three agents are important to estimate prey position by the Blackboard (more in Section 5.2).

### 5.2 Blackboard

The blackboard for GWO is a structure defined as follows:

$$BB_{GWO} = (\vec{fnc}, [\vec{v1}, \vec{v2}]),$$
 where:

dimension of both vectors v1, v2 is j = max(7,k), k ≥ 1 is the number of agents, in this case, a matrix of type i × j, i, j ∈ N, is represented by a vector of these vectors.

- v1 is a vector with elements that can be named AlphaValue, BetaValue, DeltaValue, AlphaPosition, BetaPosition, DeltaPosition, preyPosition.
  If j > 7, then elements with index greater than 7 are without a name, they are addressed by its position. The first three elements are serviced by the agents, so blackboard function for the first row only copy their values, if they are not updated by agents in current step of computation.
  - initial content of  $\vec{v1}$  is 0 in each element.
- $\vec{v2}$  is a vector with elements named  $A'_0sDistanceFromPrey$ ,  $A'_1sDistanceFromPrey$ , ...,  $A'_ksDistanceFromPrey$ , k is number of agents (wolves), the elements without name can be addressed only by its position in blackboard matrix.

- initial content of  $\vec{v2}$  is 0 in each element.

• fcn is a vector of functions (fnc1(i), fnc2(i)) for manipulating the vectors  $\vec{v1}$  and  $\vec{v2}$ , where fnc1(i)updates i - th element of vector  $\vec{v1}$  and fnc2(i) updates i - th element of vector  $\vec{v2}$ ,  $0 \le i \le j$ , j is a dimension of vectors  $\vec{v1}$  and  $\vec{v2}$ :

$$i fnc1(i) = \begin{cases} identity & i = \{0, 1, 2\}, \\ B_{Position} & i = \{3, 4, 5\}, \\ \frac{fnc1(3) + fnc1(4) + fnc1(5)}{3} & i = 6. \end{cases}$$

ii 
$$fnc2(i) = |fnc1(preyPosition) - B_{Position}|$$
, for  $i = \text{index of agent } A_i$ .

Auxiliary function  $B_{Position}$  is described in the section Receivers.

# 6 Receivers - from theoretical model into real life

A communication between agents and blackboard is realized by receivers. We equip our model with two receivers that are listening signals coming from the agents. The abilities of receivers are crucial for functioning of the functions of the blackboard, because they are providers of values of  $B_{position}$  function. We can assume, that receivers can "see" position of each agent but for wide areas it is not very realistic. For our model we choose another approach. We introduce time into our model - it takes some time to signal from agents to reach receivers.

• rcv - the blackboard has two receivers. Both receivers are located in the environment. Their initial position are on the opposite sides of the boundary points of the environment Env (positions [0,0] and [m-1,n-1], where  $m \times n, m, n \in \mathbb{N}$ , is the size of the environment). Receivers' positions are updated in each derivation step and receivers circulate around the environment as follows:

- 
$$x < 1, y < n : x = x, y = y + 1,$$



Figure 7: Blackboard in use

- x < m, y > n 1 : x = x + 1, y = y,
- x > m 1, y > 0: x = x, y = y 1,
- x > 0, y < n+1 : x = x-1, y = y.

Primary task of the *rcv* is to collect data (messages) from agents.

- Messages have a given structure: *msg*: (*contents, agent's index, timestamp*). *contents* is request of an agent - function *Get, Update* or it is an empty string, *timestamp* corresponds to the time, when request was sent.

Receivers are listening to agents' signal. If both receivers receive the same message from the agent, received message is being processed in the following sequence: computation of auxiliary function  $B_{Position}$ , execution of *contents* part of message.

-  $B_{Position} = x \in R_1 \cap R_2$ , where  $R_1$ ,  $R_2$  are circles with centre at receivers' position and radius r = now - sent, where now is time of receiving the message, and *sent* equals to *timestamp*, *x* is chosen randomly in the intersect area. The intersection shapes are changed over time due to the movements of the receivers.

At this point, it is important to focus on the use of the blackboard by the agents. Agents can use blackboard's functions above.

If the agent concludes that it is Alpha, it rewrites field  $\vec{v1}[0]$  using communication program 3 in Fig. 7 on the left side. In the same way, Beta and Delta wolves can rewrite field  $\vec{v1}[1]$  using the same function. On the right side of Fig. 7, the agent concludes that it is Omega, and it will try to move with blackboard's assistance using communication program 1.

## 7 Conclusion

The 2D P colonies as defined are not able to simulate the behaviour of the complex multi-agent systems. The agents of the 2D P colony are not able to communicate in other

way than via the environment. They also have no information about their own position and position of others in the environment. These deficiencies can be solved by adding the universal communication device, the blackboard, into the definition of the 2D P colony.

The Grey wolf optimization algorithm is meta-heuristic optimization technology inspired by the social dynamics of the packs of grey wolves. The looking for the extreme of a function is inspired by hunting the prey by the pack of the wolves.

We introduced the formal definition of the 2D P colony equipped with a blackboard and presented the ability of this formal model to simulate the Grey wolf algorithm. The blackboard serves not only as a communication device, but also as a device capable to set the particular roles of the agents simulating the wolves, and successfully find the extreme of the function represented by the discrete environment of the 2D P colony.

Acknowledgements. This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science - LQ1602".

Research was also supported by the SGS/11/2019 Project of the Silesian University in Opava.

## References

- Cienciala, L., Ciencialová, L., Perdek, M.: 2D P colonies. In: Csuhaj-Varjú E., Gheorghe M., Rozenberg G., Salomaa A., Vaszil Gy. (eds) Membrane Computing. CMC 2012. Lecture Notes in Computer Science, vol 7762. Springer, Berlin, Heidelberg, pp. 161–172 (2012)
- [2] Kelemen, J., Kelemenová, A., Păun, Gh.: Preview of P colonies: A biochemically inspired computing model. In: Workshop and Tutorial Proceedings. Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX). pp. 82–86. Boston, Massachusetts, USA (September 12-15 2004)
- [3] Păun, Gh.: Computing with membranes. J. Comput. Syst. Sci. 61(1), 108–143 (2000)

[4] Mirjalilia, S., Mirjalilib, S.M., Lewisa, A.: Grey Wolf Optimizer. Advances in Engineering Software 69, 46– 61(2014)