Modeling of Grey wolf algorithm using membrane system agents

Mgr. Daniel Valenta
RNDr. Lucie Ciencialová, Ph.D
Doc. RNDr. Luděk Cienciala, Ph.D

2020

Presentation outline

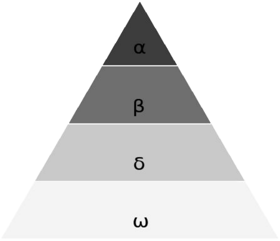
1. *Grey wolf algorithm*

2. *Relationship with P systems*

1.
Grey wolf algorithm

Grey wolf algorithm

- inspired by the 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,
- primarily used for solving optimisation-based problems,
- they have already found use in a variety of fields.



Alfa	Beta	Delta	Omega
Dominant pair, the pack follows their lead during hunts, while locating a place to sleep, ...	They support and respect the Alpha pair during its decisions and provide feedback.	Scouts – they observe the surrounding area and warn the pack, Sentinels – they protect the pack when endangered, Caretakers – they provide aid to old and sick wolves.	They help to filter the packs aggression and frustrations by serving as scapegoats.

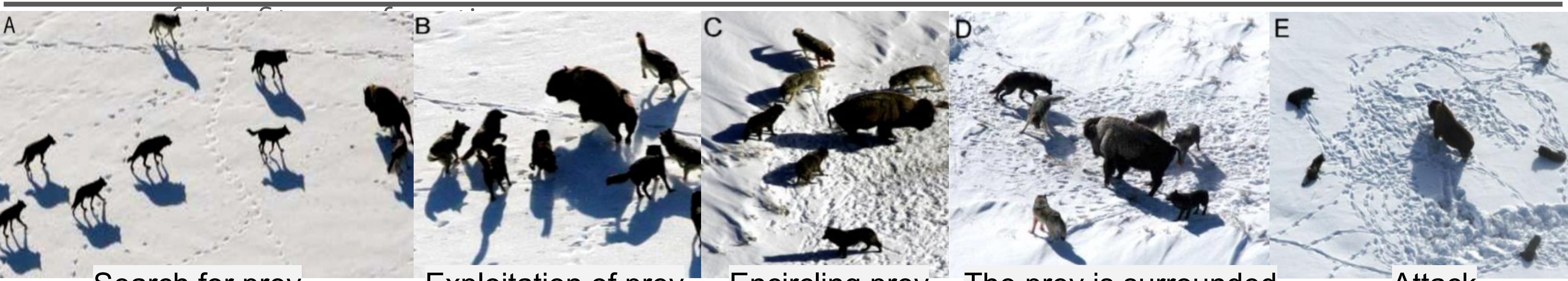
The environment and the wolf

- Wolf’s primary goal in its environment is to find and hunt down prey, which in our case equals finding the optimal solution to the given problem (the coordinates in which the fitness function reaches its criterium,
- The environment is represented by
  - dimensions of the problems environment (2D, 3D, …),
  - boundaries of the problems environment,
  - fitness function characterising the problem.
- The value of the fitness function at the wolfs current position will be metaphorically referred to as the highest-quality prey located,
  - the wolf with the best (lowest) fitness value is ranked as Alpha, the second best as Beta, third best as Delta, and all the other as Omega.

Hunting technique

The algorithm smoothly transitions between two phases:

- **Scouting phase** – the pack extensively scouts its environment through many random movements so that the algorithm does not get stuck in a local minimum.
- **Hunting phase** – the influence of random movements is slowly reduced and pack members draw progressively closer to the discovered extreme (minimum)



Search for prey | Exploitation of prey | Encircling prey | The prey is surrounded | Attack

$\vec{A}$ with components $rand(-1,1) * a$,

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

Scouting and tracking prey

- Wolves positions within the environment are updated based upon the estimated location of the prey using Alpha, Beta, and Delta wolves as guides,
- in order to maintain divergence between scouting and the actual hunt, each wolf is assigned a vector A,
- another component supporting the scouting phase is vector C = rand (0,2).

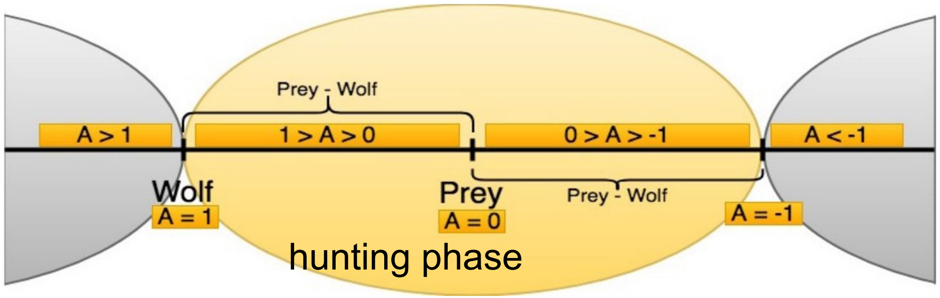


Fig. 1. Vector A and its impact in 1D space

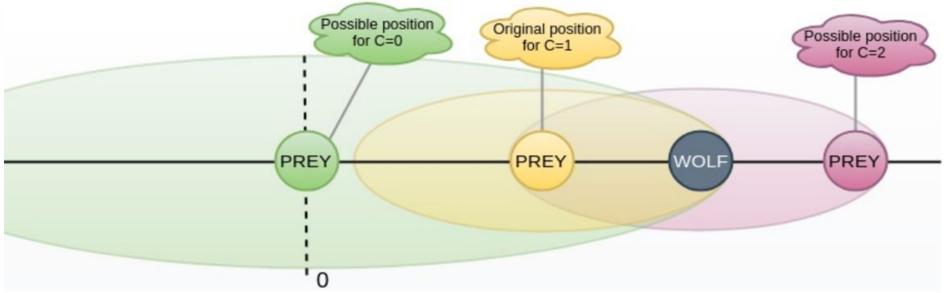


Fig. 2. Vector C and its impact in 1D space

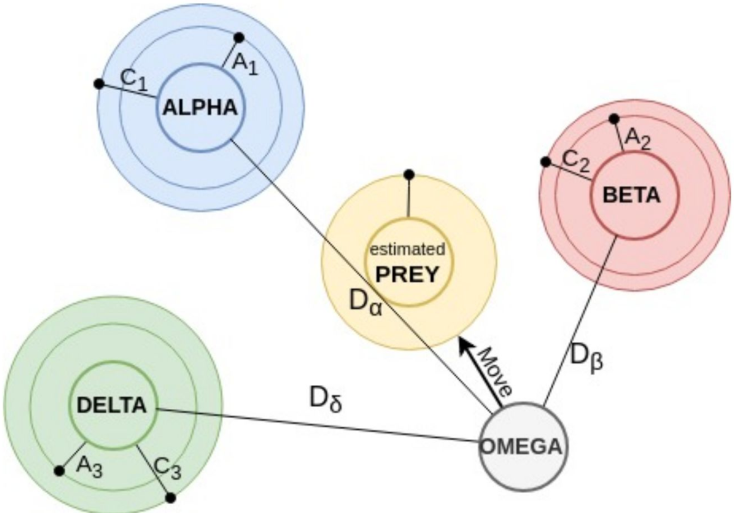
Encircling prey

Positional vectors of individual wolves $\overrightarrow{X_j}$, where j is the wolfs index, are updated according to the following formula:

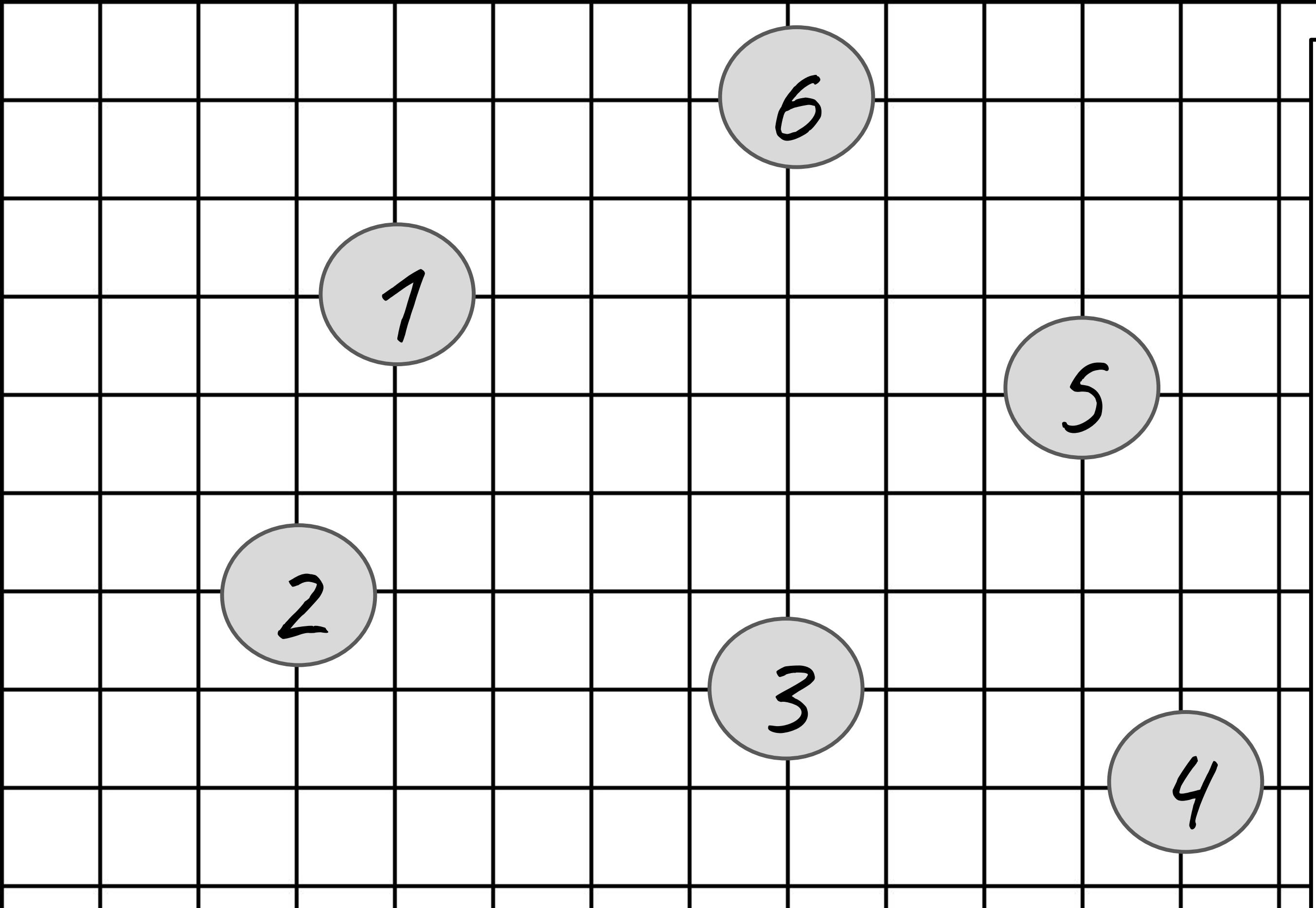
$$\overrightarrow{X_j}(i+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3},$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha}(i) - \overrightarrow{A_1} * \overrightarrow{D_\alpha},$$

$$\overrightarrow{D_\alpha} = \left| \overrightarrow{C_1} \cdot \overrightarrow{X_\alpha}(i) - \overrightarrow{X_j}(i) \right|$$

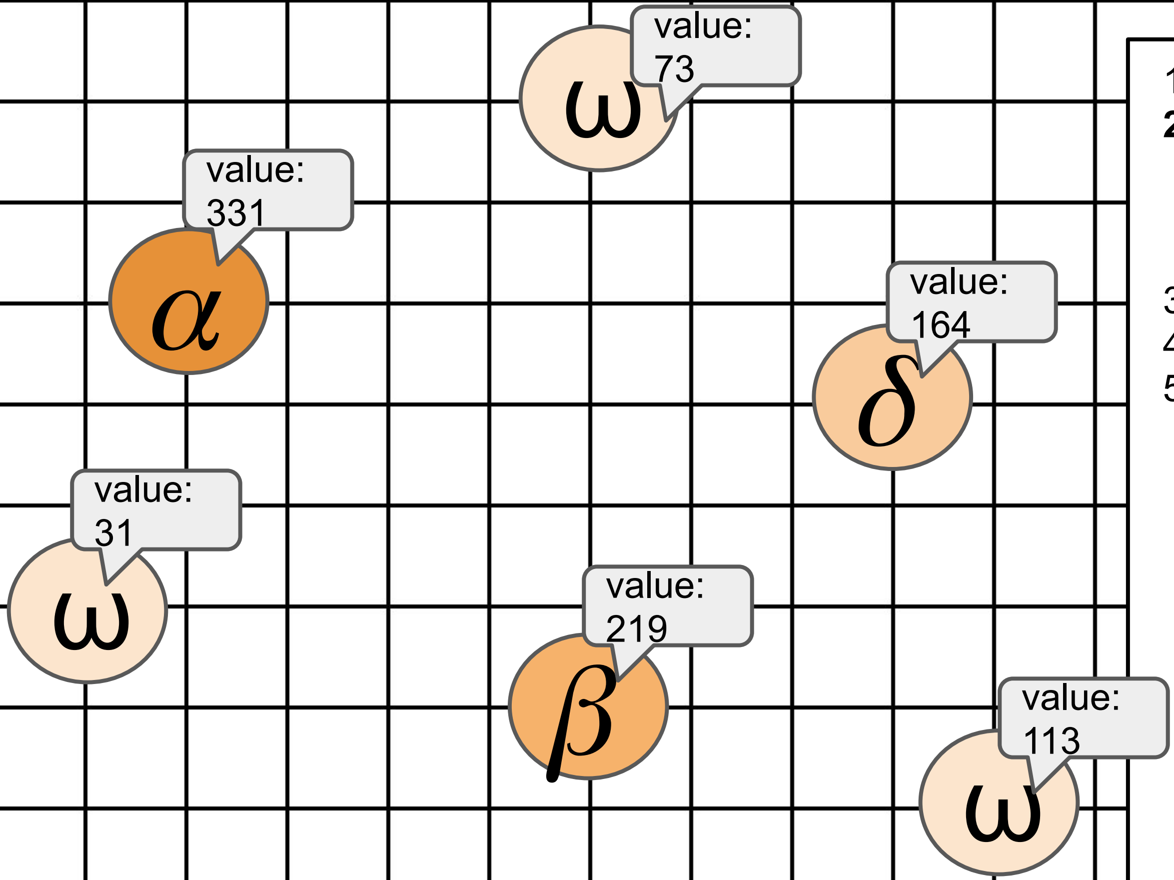


Wolves have the tendency to move closer towards prey and encircle it (wolves approach from various directions).



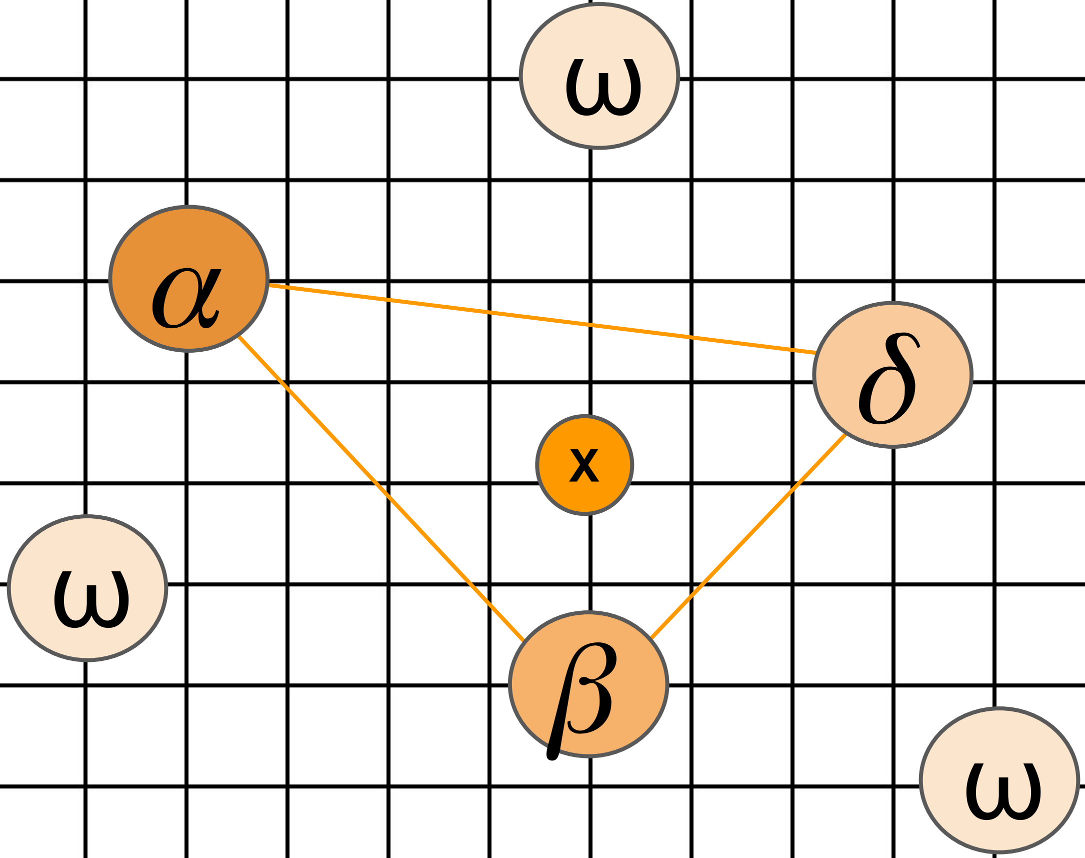
1. **Initialize agents (wolfs)**
2.
3.
4.
5.

Describing the algorithm



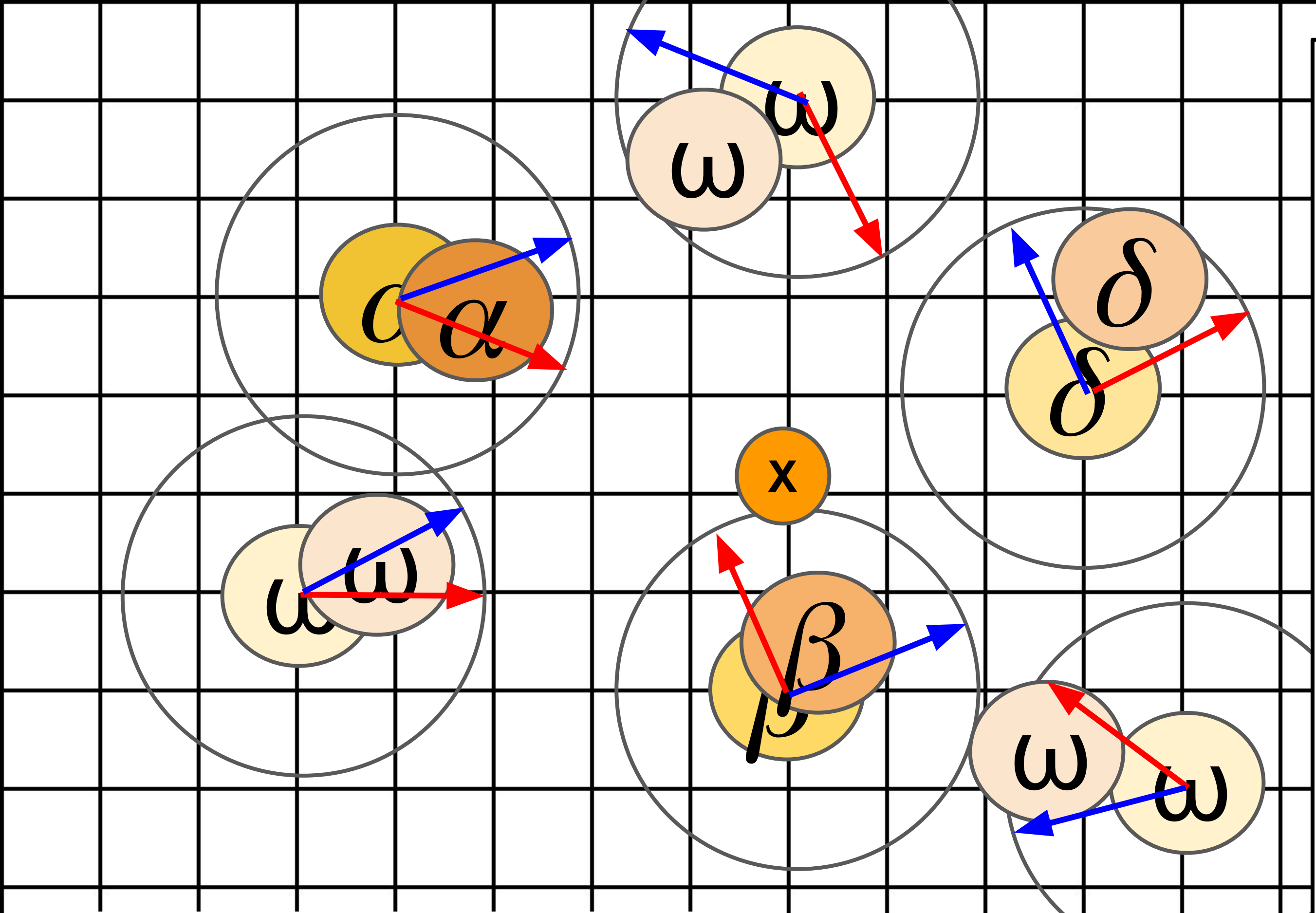
1. Initialize agents (wolfs)
2. **Calculate fitness of each agents and determine the social hierarchy**
3.
4.
5.

Describing the algorithm



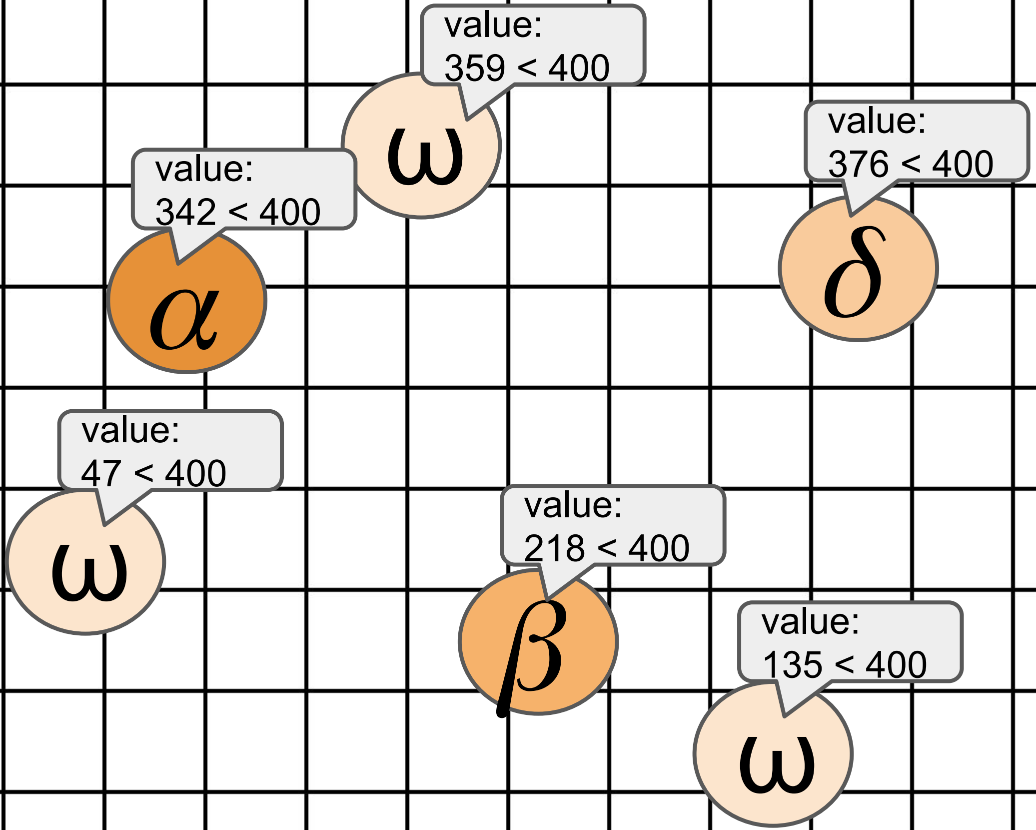
1. Initialize agents (wolfs)
2. Calculate fitness of each agents and determine the social hierarchy
3. **Calculate the best solution found thus far X_α (i), X_β (i) , X_δ (i)**
4.
5.

Describing the algorithm



Describing the algorithm

1. Initialize agents (wolfs)
2. Calculate fitness of each agents and determine the social hierarchy
3. Calculate the best solution found thus far $X_\alpha(i)$, $X_\beta(i)$, $X_\delta(i)$
4. **Update positions of all wolves $X_j(i+1)$, while vectors A (red arrow) , C (blue arrow) , are updated for each one**
5.



1. Initialize agents (wolfs)
2. Calculate fitness of each agents and determine the social hierarchy
3. Calculate the best solution found thus far Xα (i), Xβ (i) , Xδ (i)
4. Update positions of all wolves X_j (i+1), while vectors A, C, are updated for each one
5. **Check for the termination criteria**

Go to step 2

Describing the algorithm

2.
Relationship with P systems

Modeling of Grey wolf algorithm using membrane system agents

Inspired by nature

Usable for solving optimization problems

Multi-agent system model

?

Problem with randomness

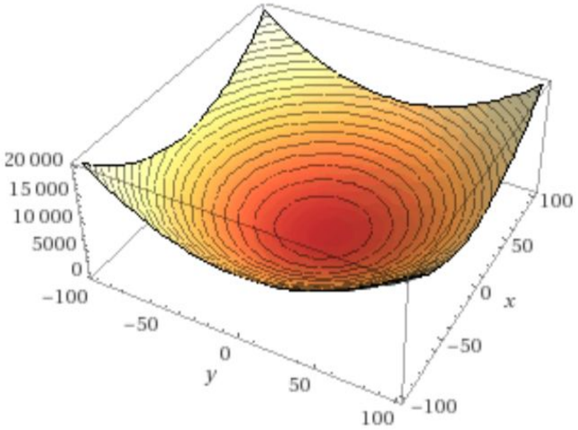
Environmental problem

Communication problem

Environmental problem

Grey wolf algorithm

- represented by a mathematical fitness function



Example 1: Grey wolf algorithm fitness function
$F = x^2 + y^2 : x, y \in (-100, +100)$.

P colonies

- represented by multiset of symbols / objects

$$Env = (6 \times 6, w_E),$$

$$w_E = \begin{bmatrix} D & D & D & D & D & D \\ D & S & S & D & D & D \\ D & S & S & D & D & D \\ D & D & D & S & S & D \\ D & D & D & S & S & D \\ D & D & D & D & D & D \end{bmatrix},$$

Example 2: 2D P colony environment

Environment problem solution

Proposed solution:

- *Env* is a pair (m × n, *f(x)*) , where m × n, m, n ∈ N is the size of the environment and *f(x)* is the initial contents of environment,
  - A = {ℝ (real numbers)} ∪ e (environmental symbol)
- Agent's program rules will compare the number values of objects using operators "<" (or ">"),
  - $B_i = o_i$, P_i, $[r_i, s_i]$, o = 2,
  - example: $o_1 = 12$, $o_2 = 31$, env = x ∈ A,
    $P_i = (o_1 < o_2, x)$: an action rule

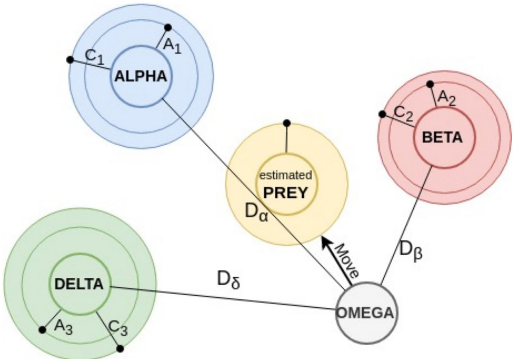
Example:

$$Env = (6\times6, f(x)),\ f(x) = \begin{pmatrix} 3 & 12 & 17 & 22 & 14 & 4 \\ 8 & 24 & 28 & 31 & 19 & 11 \\ 14 & 27 & 33 & 37 & 31 & 15 \\ 21 & 25 & 41 & 48 & 30 & 18 \\ 15 & 21 & 32 & 33 & 26 & 10 \\ 7 & 11 & 19 & 21 & 9 & -2 \end{pmatrix}$$

Communication problem

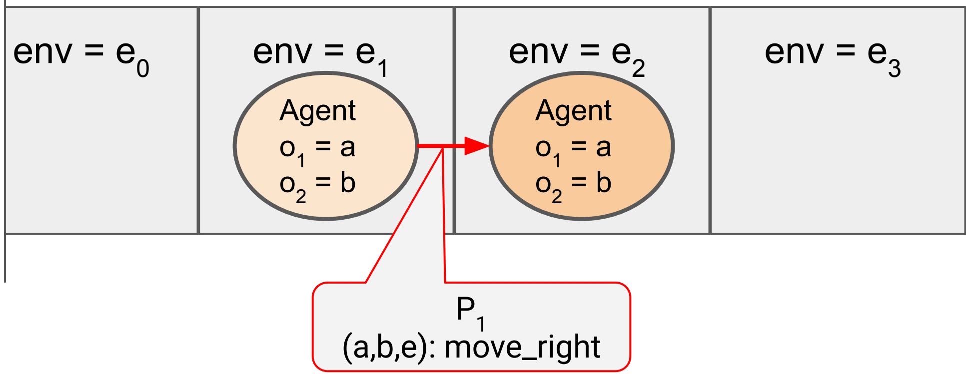
Grey wolf algorithm

- Agents (wolves) have the knowledge of their global position in the environment,
- wolves positions are updated based upon the estimates created by Alpha, Beta, or Delta wolves.



P colonies

- Communities of simple reactive agents independently living and acting in a joint shared environment,
- indirect communication through the special objects in environment

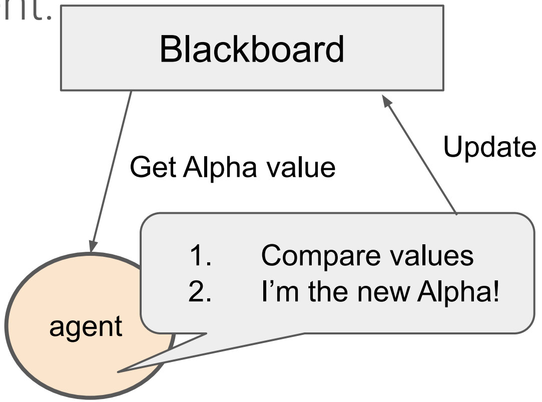


Communication problem solution

Proposed solution:

- Extending the P system by adding the Blackboard that:
  - saves the agents' best fitness values,
  - is always accessible to read and write by all agents,
- Agents must know their position in the environment.

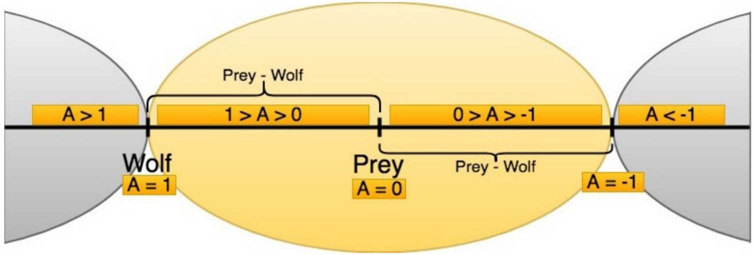
	BlackBoard		
Index	0	1	2
Agent	B_{Alpha}	B_{Beta}	B_{Delta}
Position	x_1, y_1	x_2, y_2	x_3, y_3
Value	Best value	2nd best value	3rd Best value



The problem with randomness

Grey wolf algorithm

- Random vectors A and C influence the movement of wolves in the environment.



Iterations influence the random values

P colonies

- Each program rule is deterministic,
- the rules can be chosen in a non-deterministic manner.

P_1:

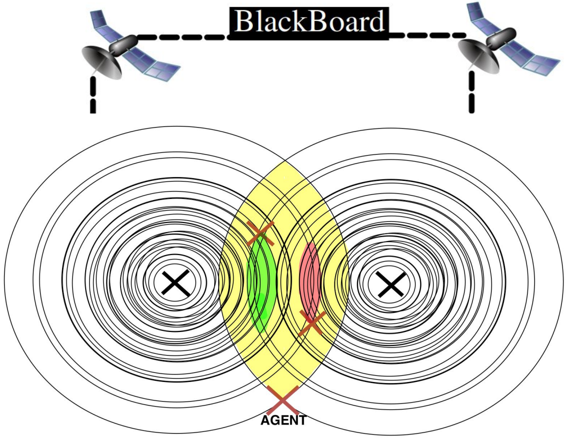
1. (a, a, e): action_1
2. (a, a, e): action_2

?

Randomness problem solution

Proposed solution:

- Agents don't need to know their position in the environment,
- all agents who can contribute to the search will send solutions to the blackboard points (receiver),
- Estimation of prey position is calculated as average of distances collected by blackboard points from wolves Alpha, Beta, Delta,
- Omega wolves can ping the blackboard if changing position.

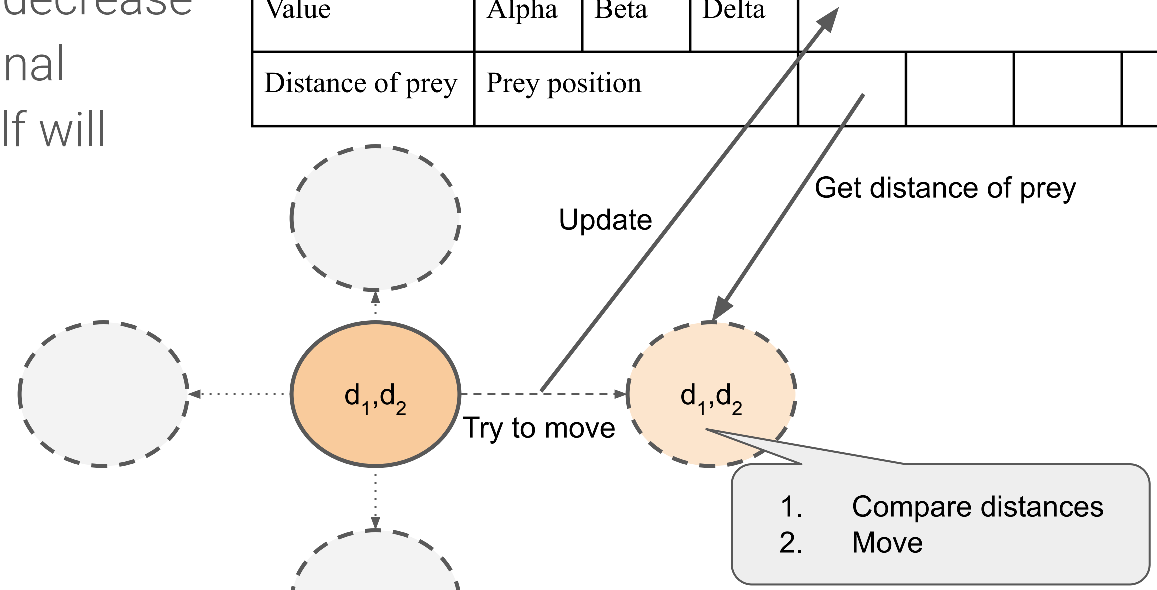


	BlackBoard						
Index	0	1	2	3	4	...	5
Agent	B_{Alpha}	B_{Beta}	B_{Delta}	B_1	B_2	…	B_n
Value	Best value	2nd best value	3rd Best value				
Distance	Estimation of prey position (calculated by the BlackBoard)			distance of prey	distance of prey	…	distance of prey

Blackboard

- Omega wolf ping the blackboard before changing position and get its distance from the prey.
- If the distance would decrease compared to the original distance, then the wolf will move.

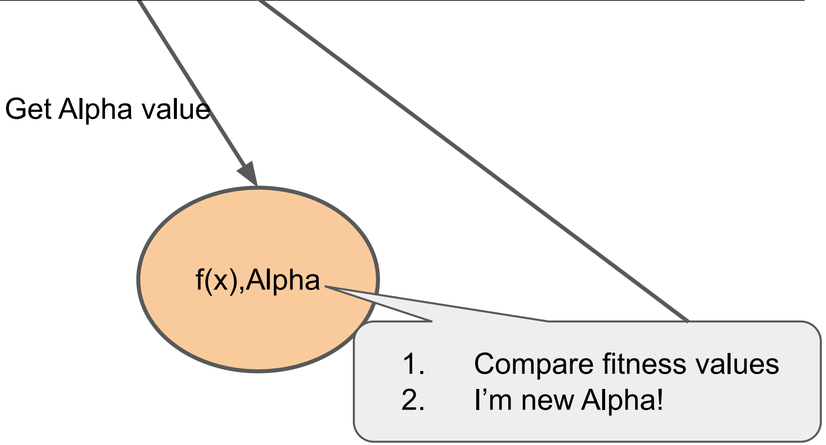
	BlackBoard						
Index	0	1	2	3	4	5	6
Value	Alpha	Beta	Delta				
Distance of prey	Prey position						



Blackboard

- the agent compares its fitness value to the Alpha. If this agent has the better fitness value, it updates the blackboard.

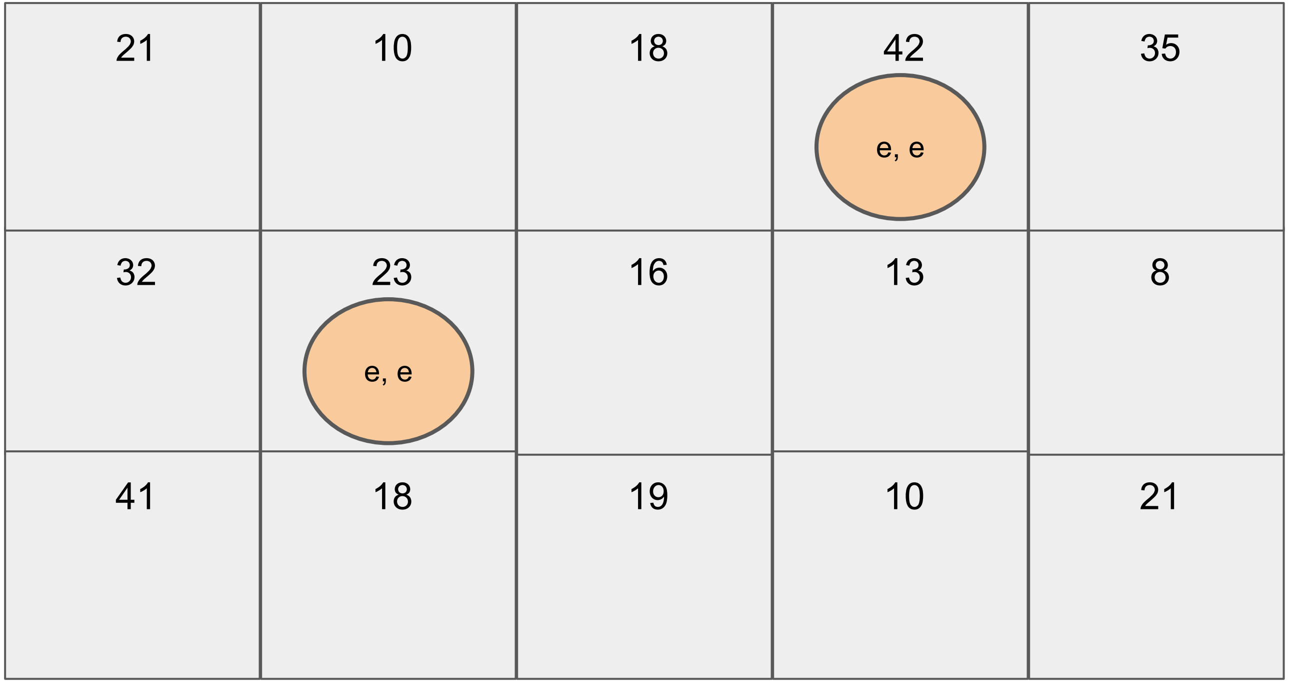
	BlackBoard						
Index	0	1	2	3	4	5	6
Value	Alpha	Beta	Delta				
Distance of prey	Prey position						



Initialization

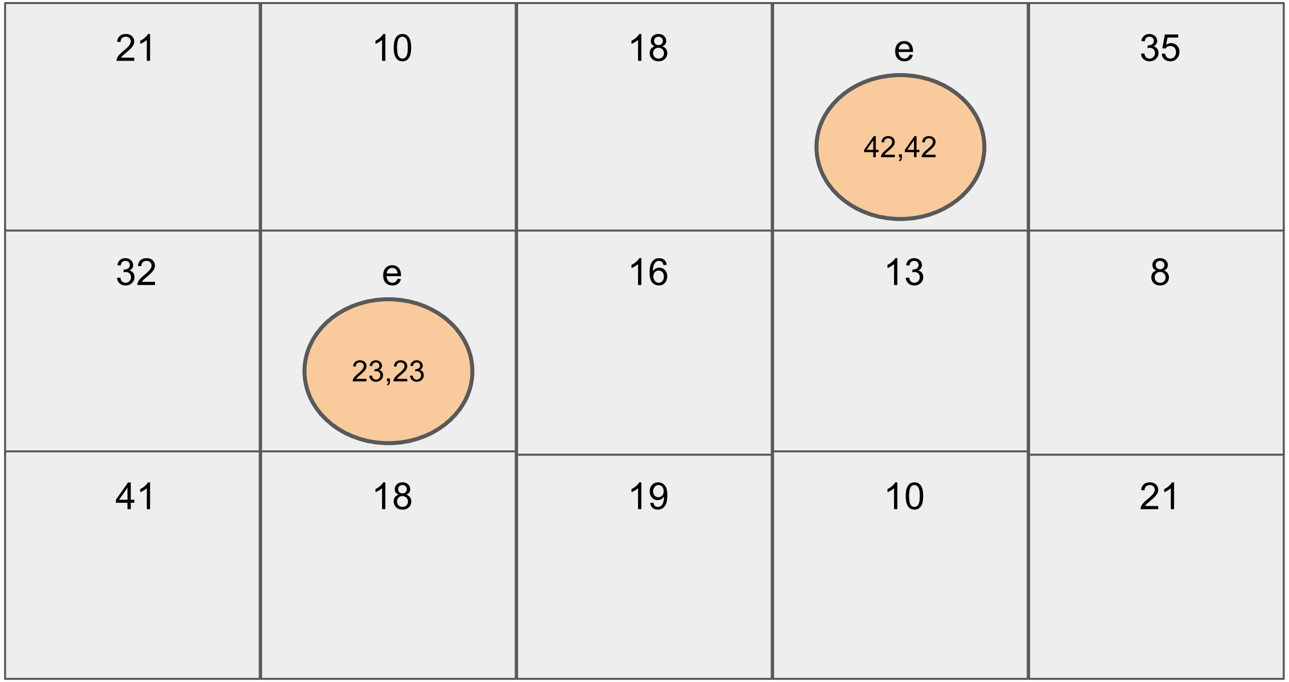
	BlackBoard				
Index	0	1	2	3	4
Value	Alpha	Beta	Delta		
Distance of prey	Prey position				

$$(e_1,e_2,x): e_1, e_2 \leftrightarrow x; x \in \mathbb{R}$$



	BlackBoard				
Index	0	1	2	3	4
Value	Alpha	Beta	Delta		
Distance of prey	Prey position				

Iteration 1



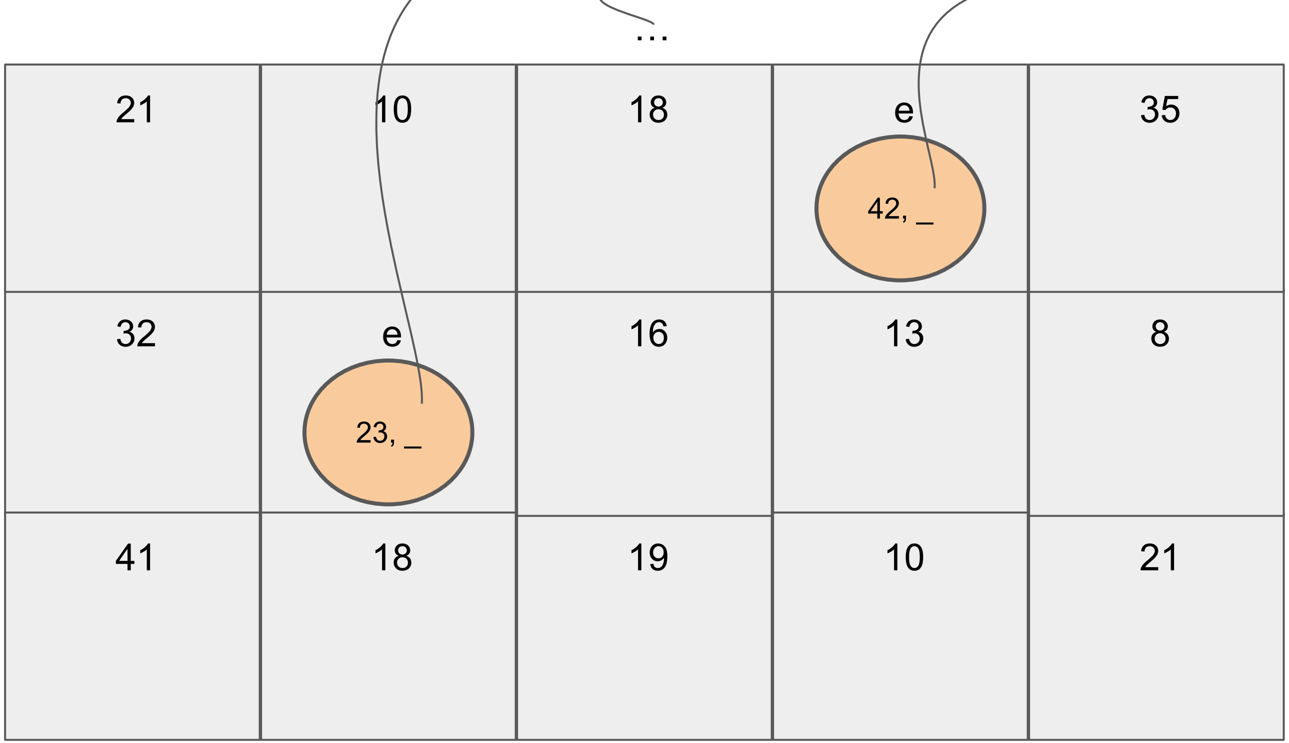
(e_1, e_2, x)**: $e_1, e_2 \leftrightarrow x; x \in \mathbb{R}$**

$(x, y, e); x, y \in \mathbb{R}$:

    y ← get(BB[Alpha])
    compare(x,y):
        x > y: I'm new Alpha!
            update(BB[Alpha])
        x < y: y ← get(BB[Beta])
            compare (x, y):
                x > y: I'm new Beta
                ....

Iteration 1

	BlackBoard				
Index	0	1	2	3	4
Value	Alpha	Beta	Delta		
Distance of prey	Prey position				



(e_1,e_2,x): e_1, $e_2 \leftrightarrow x$; $x \in \mathbb{R}$

(x, y, e); x, y ∈ ℝ:

  y ← get(BB[Alpha])

  compare(x,y):

    x > y: **I’m new Alpha!**

      update(BB[Alpha])

    x < y: y ← get(BB[Beta])

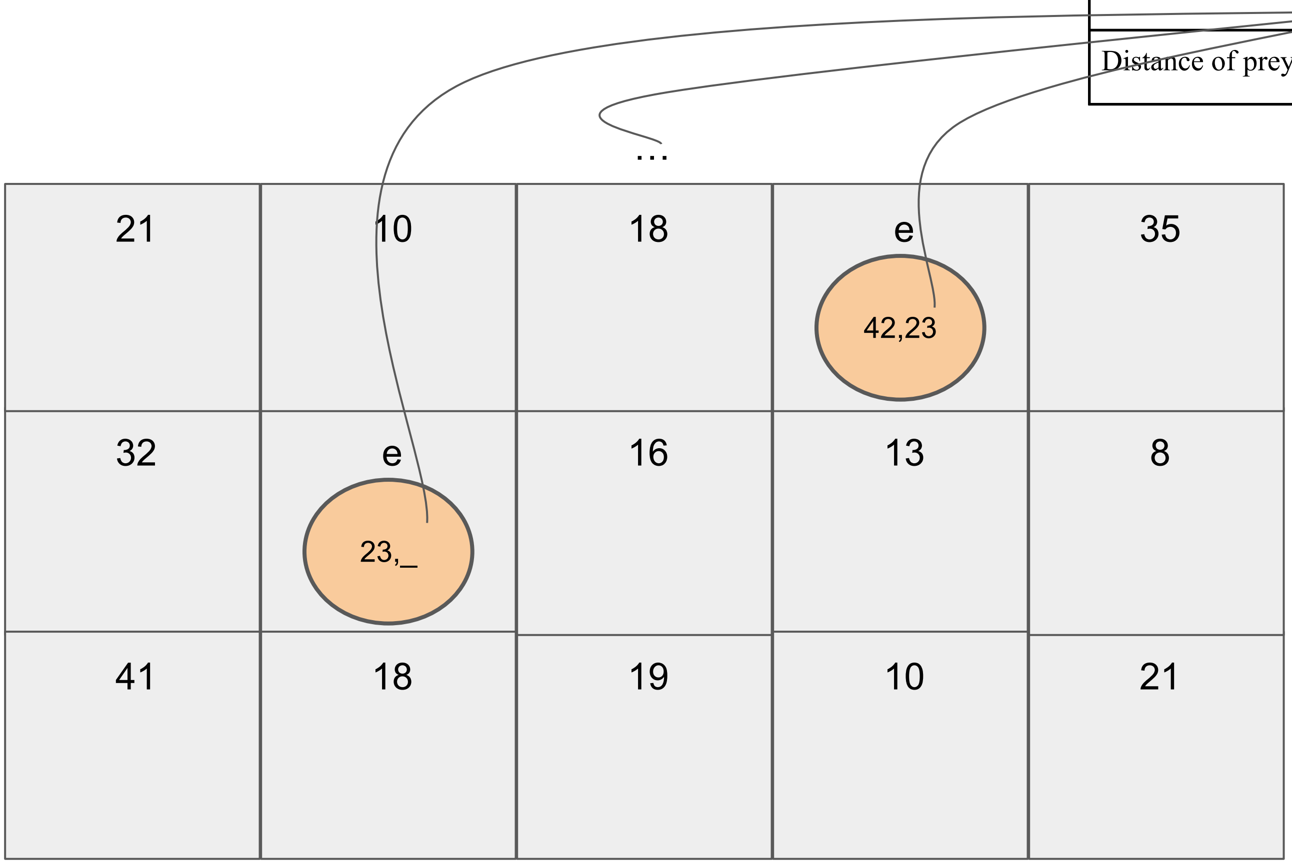
      compare (x, y):

        x > y: I’m new Beta

        ....

Iteration 2

	BlackBoard				
Index	0	1	2	3	4
Value	23	Beta	Delta		
Distance of prey	Prey position				



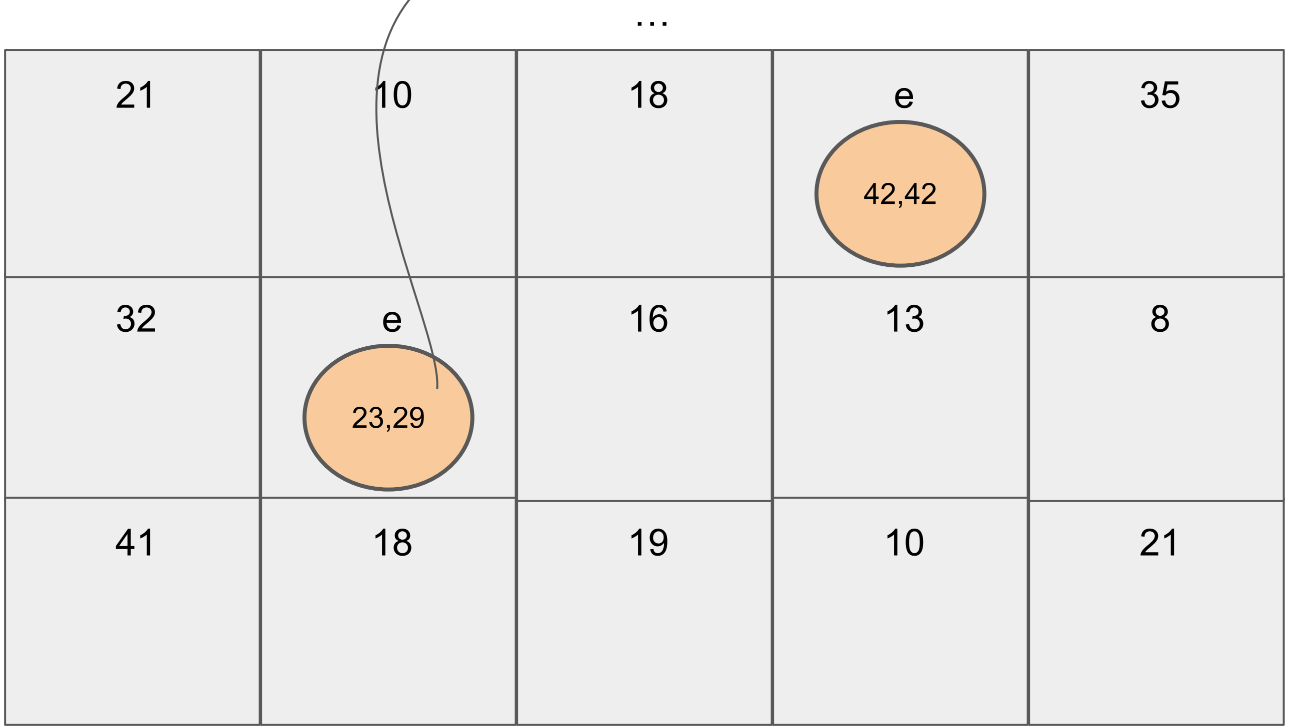
(e_1,e_2,x): $e_1, e_2 \leftrightarrow x$; $x \in \mathbb{R}$

(x, y, e); $x, y \in \mathbb{R}$:

    y ← get(BB[Alpha])
    compare(x,y):
        x > y: **I'm new Alpha!**
            update(BB[Alpha])
        x < y: y ← get(BB[Beta])
            compare (x, y):
                x > y: **I'm new Beta**
                ....
                    x > y: **I'm new Delta**

Iteration n

	BlackBoard				
Index	0	1	2	3	4
Value	42	36	29		
Distance of prey	Prey position				



(x, y, e); x, y ∈ ℝ:

...

I'm Omega!

y ↔ e

e → m

m ↔ y

(x, y, m); x, y ∈ ℝ:

Ping BB[i]

x ← get(BB(i))

Ping+mv_1 BB[i]; mv_1 = rand (⇐,⇑,⇒,⇓)

y ←get(BB(i))

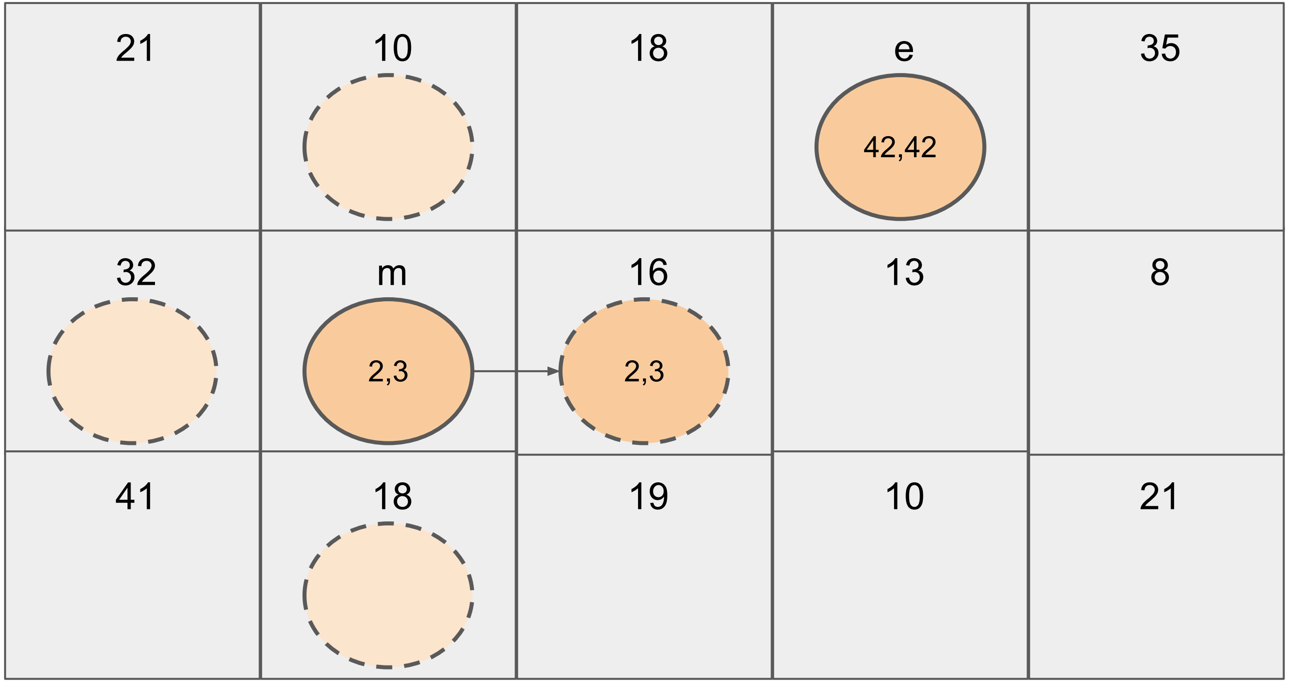
compare (x,y)

x > y: do mv_1

Iteration n+1

	BlackBoard				
Index	0	1	2	3	4
Value	42	36	29		
Distance of prey	Prey position				

…



(x, y, e); x, y ∈ ℝ:

…

I'm Omega!

y ↔ e

e → m

m ↔ y

(x, y, m); x, y ∈ ℝ:

Ping BB[i]

x ← get(BB(i))

Ping+mv_1 BB[i]; **mv_1 = rand (⇐,⇑,⇒,⇓)**

y ←get(BB(i))

compare (x,y):

x > y: do mv_1

Model of Grey wolf algorithm using membrane system agents

P_{gw} = (A, e, env, B_1, B_2, …, B_n, X, f), where:

- A = {$\mathbb{R}$} ∪ {e,m,f},
- e ∈ A is the basic environmentální object,
- f is the final object, f ∈ A,
- Env is a pair (m × n, f(x)) , where m × n,
  m, n ∈ $\mathbb{N}$, is the size of the environment
  and f(x), is the initial contents of environment,
- X is the blackboard,
- B_1, …, B_n are the agents, B_i = (O_i, P_i, [r_x,s_y]),
  O_i = 2,
  $P_1 = P_2 = \ldots = P_n$,
  R_x, S_y are the initial coordinates,
- Initial agents' configuration: (O_1[e], o_2[e], env[i]), i ∈ $\mathbb{R}$.

$P_i = \{$

1. (e_1, e_2, x): $e_1, e_2 \leftrightarrow x$; $x \in \mathbb{R}$
2. (x, y, e); $x, y \in \mathbb{R}$:
   A. $y \leftarrow$ get(BB[Alpha])
   B. compare(x,y):
      a. x > y: I'm new Alpha!
         - update(BB[Alpha])
      b. x < y: y ← get(BB[Beta])
         - compare (x, y):
            - x > y: I'm new Beta!
               i. update(BB[Beta])
            - x < y: y ← get (BB[Delta])
               i. x > y: I'm new Delta!
                  ➢ Update(BB[Delta])
               ii. x < y: I'm Omega:
                  ➢ $y \leftrightarrow e$
                  ➢ $e \rightarrow m$
                  ➢ $m \leftrightarrow y$

$\longleftarrow$
get from blackboard
$\longrightarrow$
rewrite agent's object
$\longleftrightarrow$
change agent's object with environment object

3. (x, y, m); $x, y \in \mathbb{R}$:
   A. Ping BB[i]
   B. x ← get(BB(i))
   C. Ping+mv_1 BB[i]; mv_1 = rand (⇐,⇑,⇒,⇓)
   D. y ←get(BB(i))
   E. compare (x,y):
      a. x > y: do mv_1
      b. x < y:
         - Ping+mv_2 BB[i];
           mv_2 = rand(⇐,⇑,⇒,⇓) - mv_1
         - y ←get(BB(i))
         - x > y: do mv_2
         - x < y: ...
            - Can't move:
               i. $y \leftrightarrow m$
               ii. $m \rightarrow f$
               iii. $f \leftrightarrow m$
4. (x, y, f); $x, y \in \mathbb{R}$: stop the agent

$\}$

Thanks for your attention

Daniel Valenta
Silesian University in Opava
Faculty of Philosophy and Science
Institute of Computer Science
F180337@fpf.slu.cz