ELTE | FACULTY OF INFORMATICS

Swarm intelligence

Introduction to Machine Learning – Lecture 9
Computer Science BSc Course, ELTE Faculty of Informatics

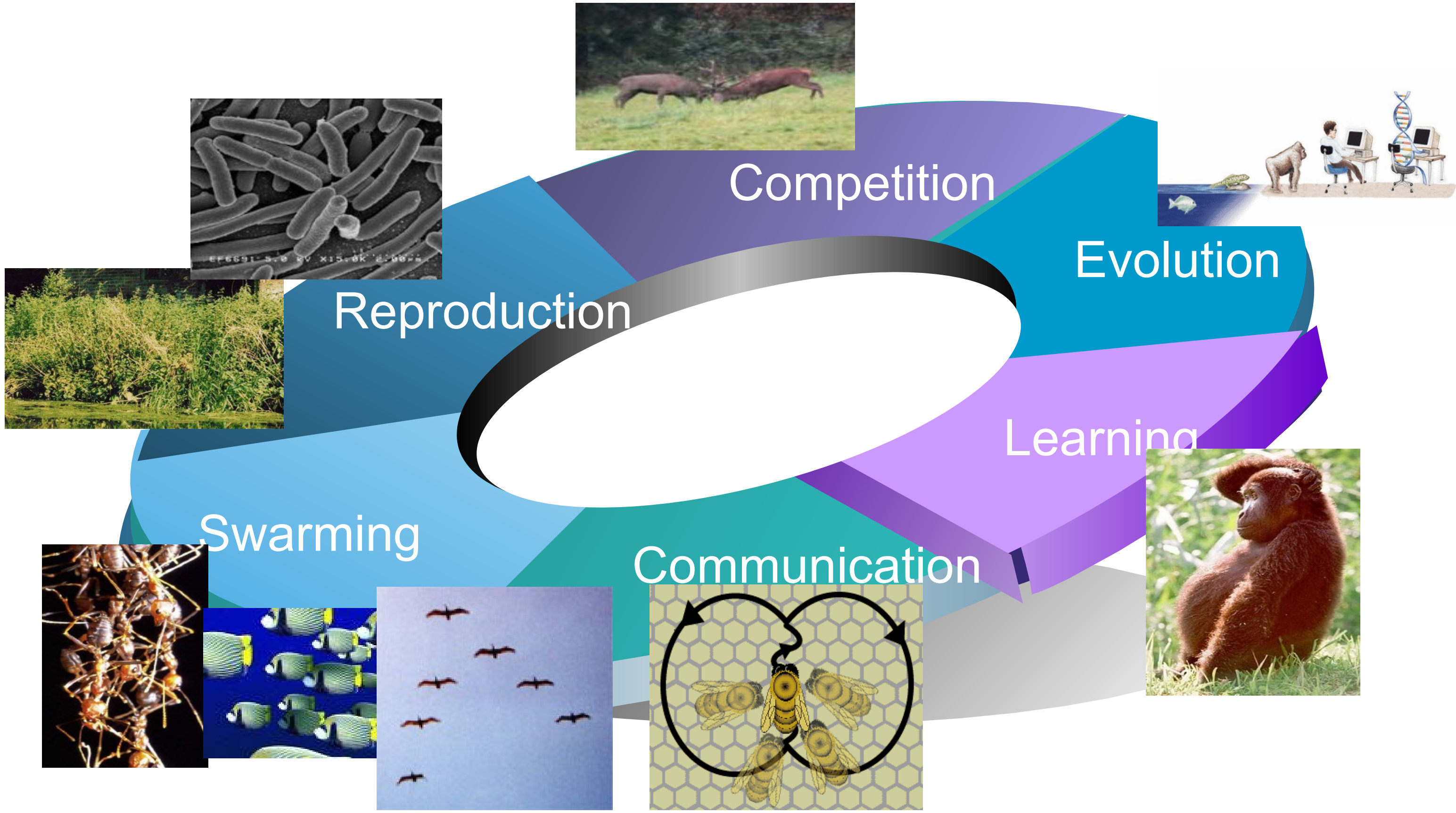
János Botzheim

Associate Professor
Department of Artificial Intelligence
botzheim@inf.elte.hu

ELTE | IK
DEPARTMENT OF ARTIFICIAL INTELLIGENCE

Introduction, motivations

Elements of intelligence in biological systems



Swarm intelligence

- A collective system capable of performing complex tasks in a dynamic and changing environment without any external control or central coordination
- Capable of achieving a collective performance that cannot normally be achieved by the organism alone
- On this basis, we can build a natural model that is suitable for distributed problem solving

Swarm intelligence

Swarm intelligence

Swarm intelligence

Swarm intelligence

Particle Swarm Optimization

Particle Swarm Optimization (PSO)

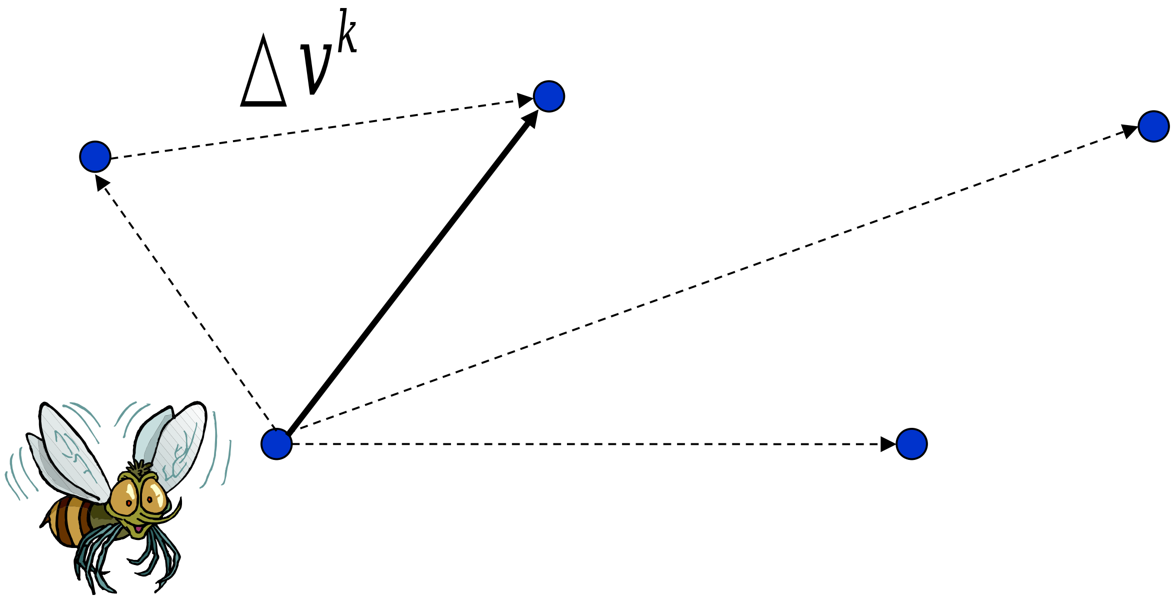
- PSO applies the concept of social interaction to problem solving
- It uses a number of so-called particles that move in swarms in the search space in search of the best solution
- It treats each particle as a spatial point, which adjusts its "flight" based on its own flight experience as well as the flight experience of other particles
- James Kennedy and Russell Eberhart, 1995

Particle flight model

- **pbest**: the best solution achieved by the particle so far
- **gbest**: best solution achieved by the whole swarm
- The principle of PSO is that individual particles accelerate towards the **pbest** and **gbest** sites, each time with a randomly weighted acceleration

Particle flight model

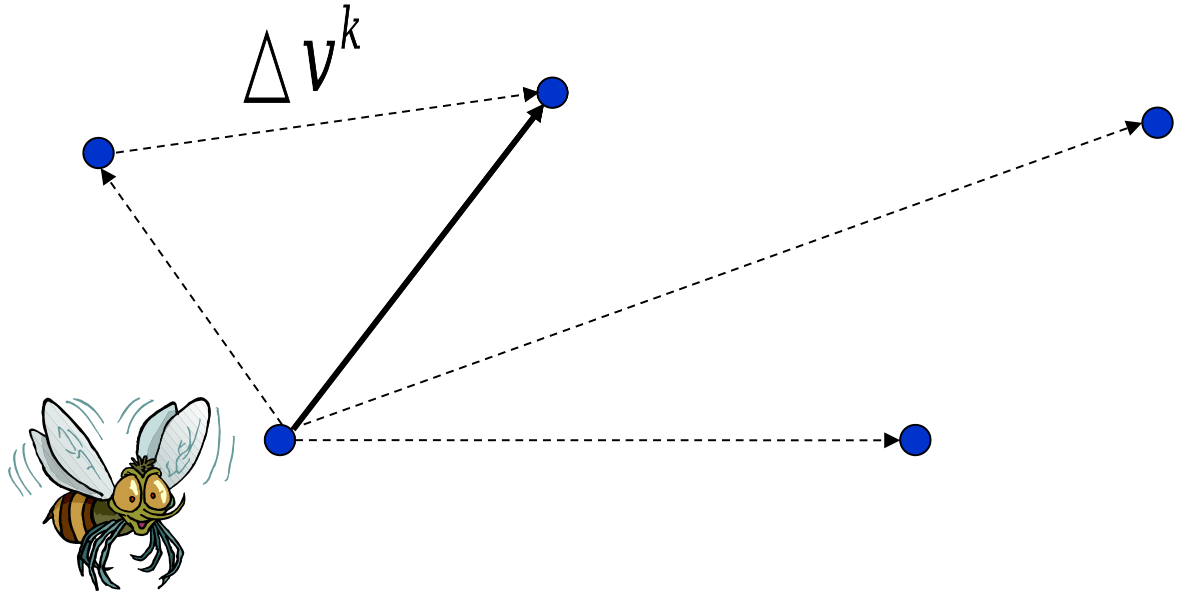
$$\Delta v^k = w_1 d^{pbest^k} + w_2 d^{gbest^k}$$



Particle flight model

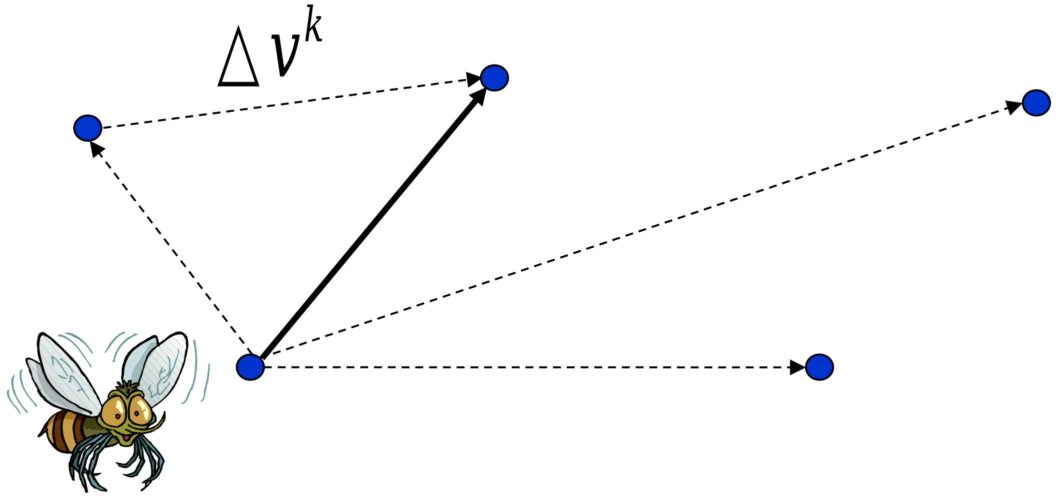
- Each particle changes its position based on the following information:
  - the current position
  - the current velocity
  - the distance between the current position and the pbest
  - the distance between the current position and the gbest

$$\Delta v^k = w_1 d^{pbest^k} + w_2 d^{gbest^k}$$



Particle flight model

$$\Delta v^k = w_1 d^{pbest^k} + w_2 d^{gbest^k}$$



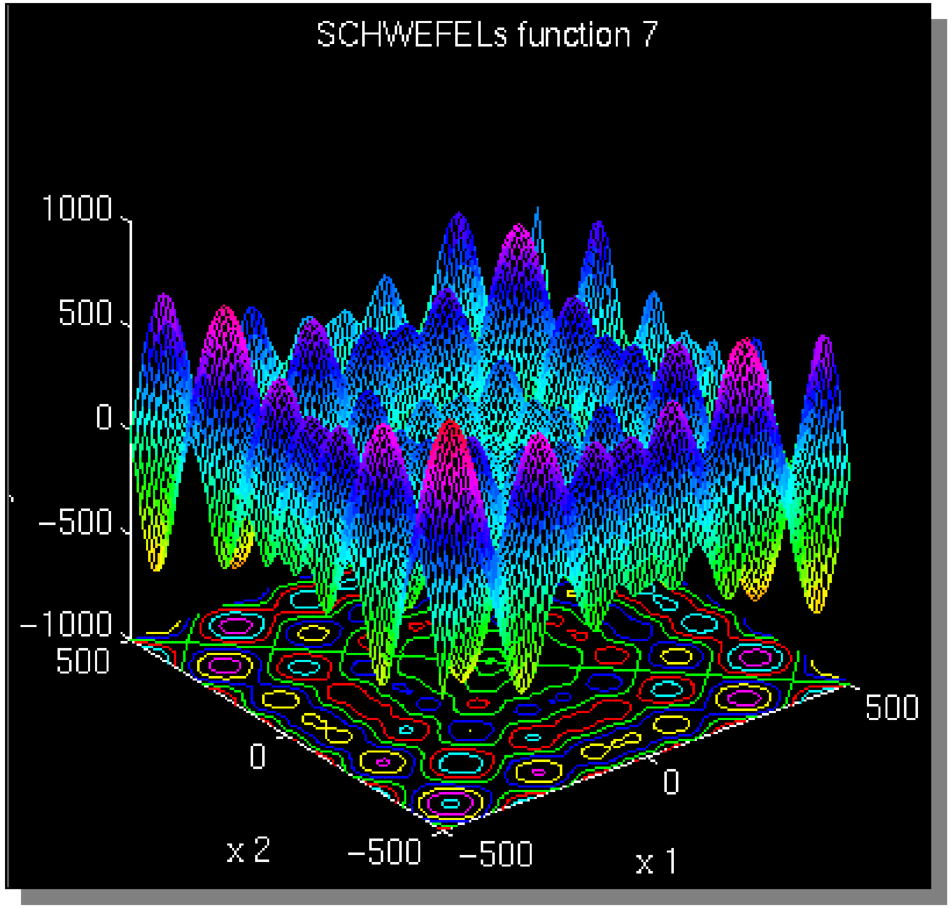
PSO algorithm

```
For each particle
   Initialize particle
END

Do
  For each particle
      Calculate fitness value
      If the fitness value is better than the best fitness value (pbest) in history
         set current value as the new pbest
  End

  Choose the particle with the best fitness value of all the particles as the gbest
  For each particle
      Calculate particle velocity according equation (*)
      Update particle position according equation (**)
  End
While maximum iterations or minimum error criteria is not attained
```

(*) $$v_i^{k+1} = v_i^k + \Delta v_i^k$$

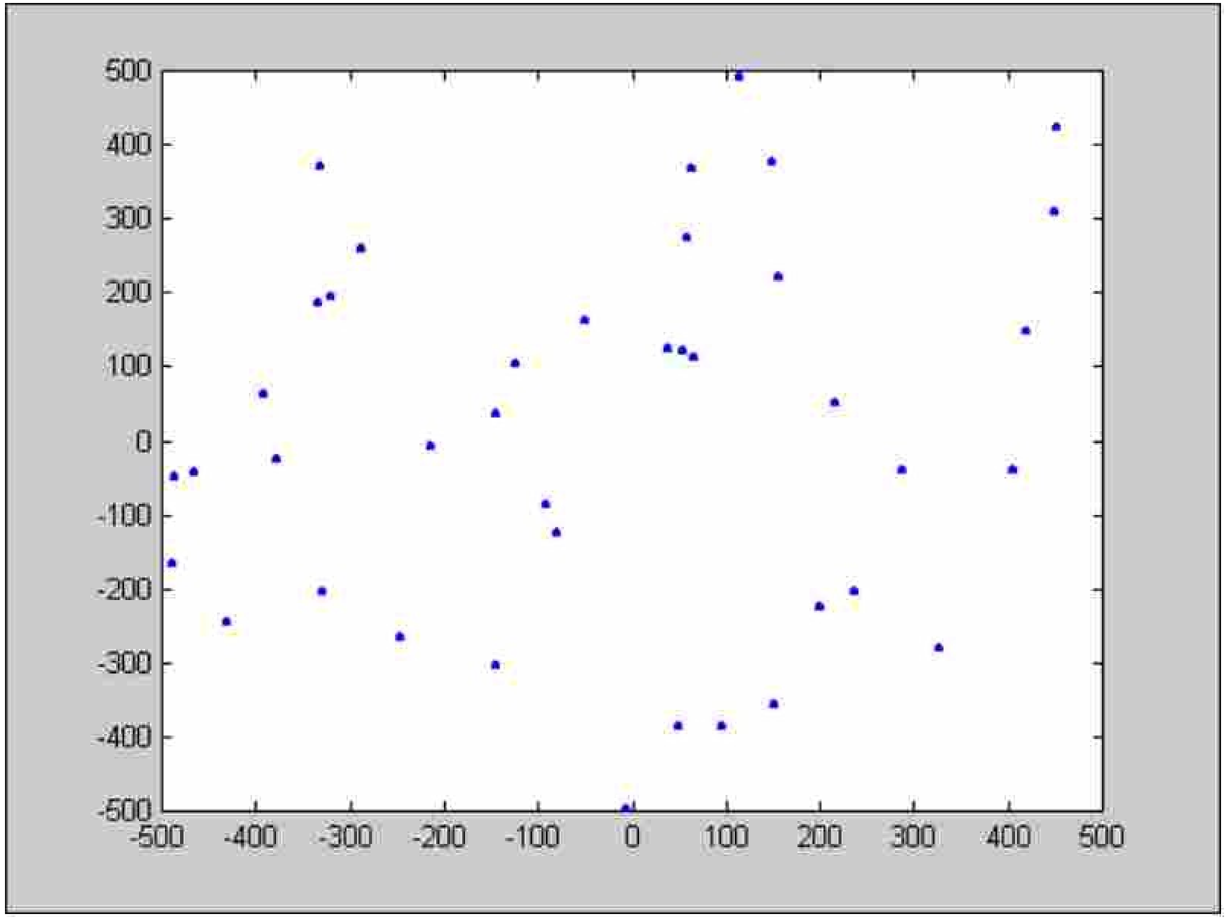
(**) $$s_i^{k+1} = s_i^k + v_i^{k+1}$$

PSO example

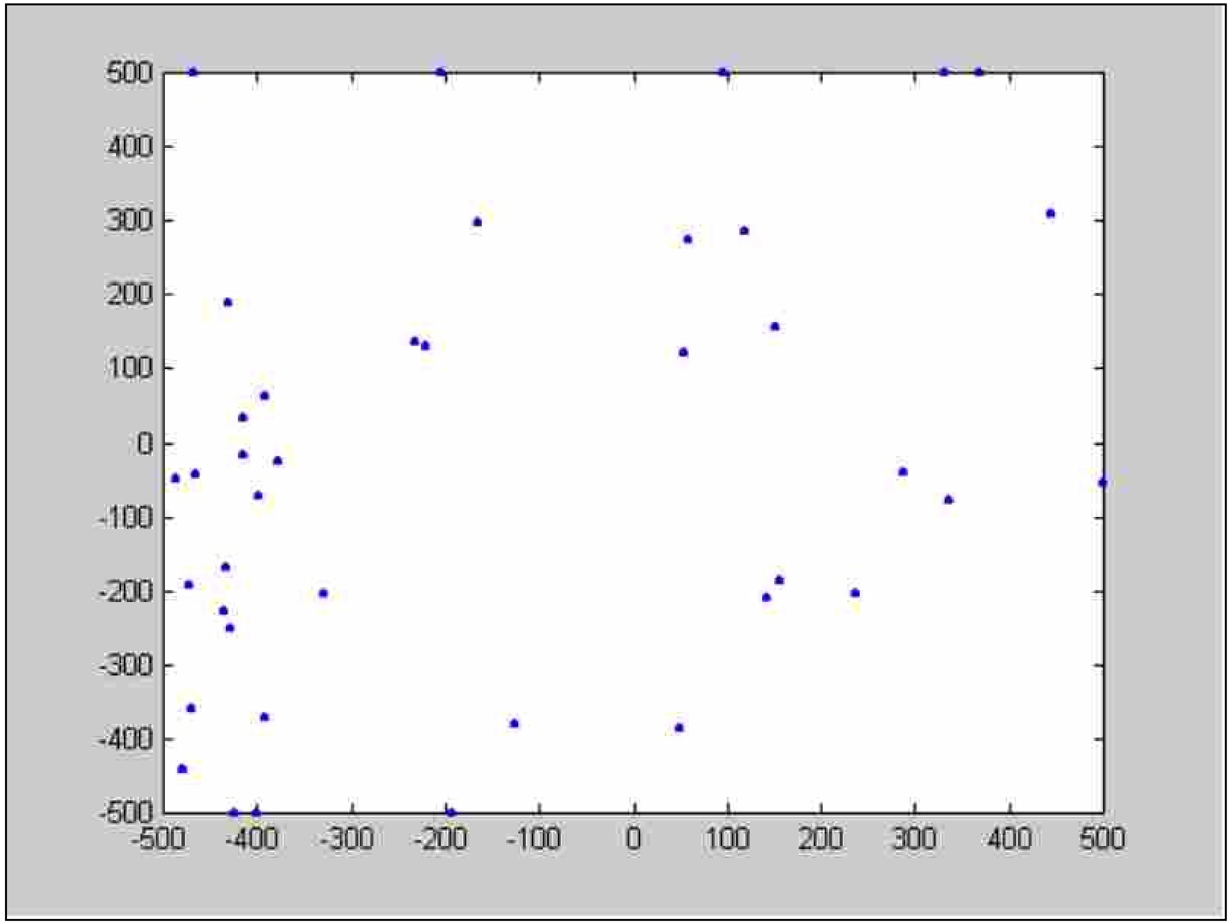


Global minimum:

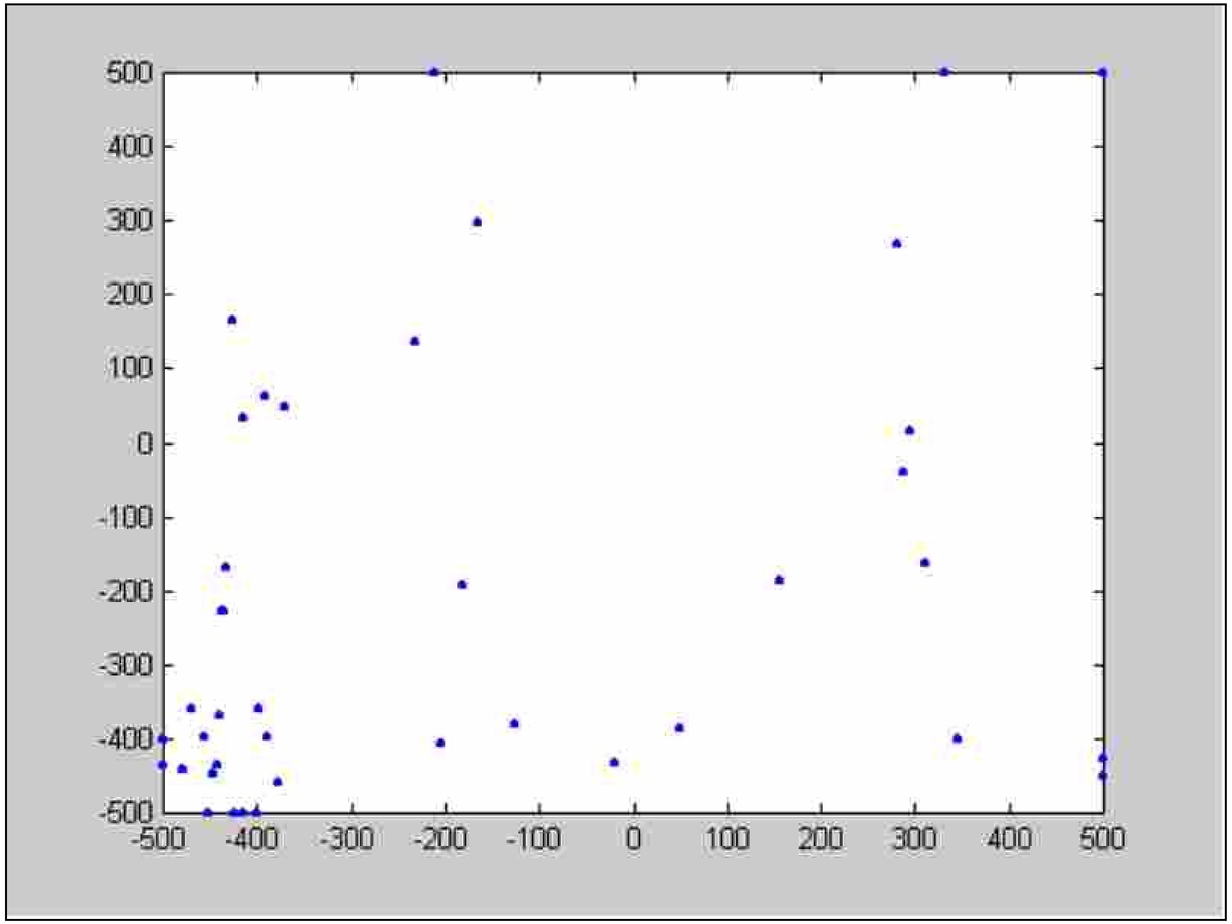
PSO example - Initialization



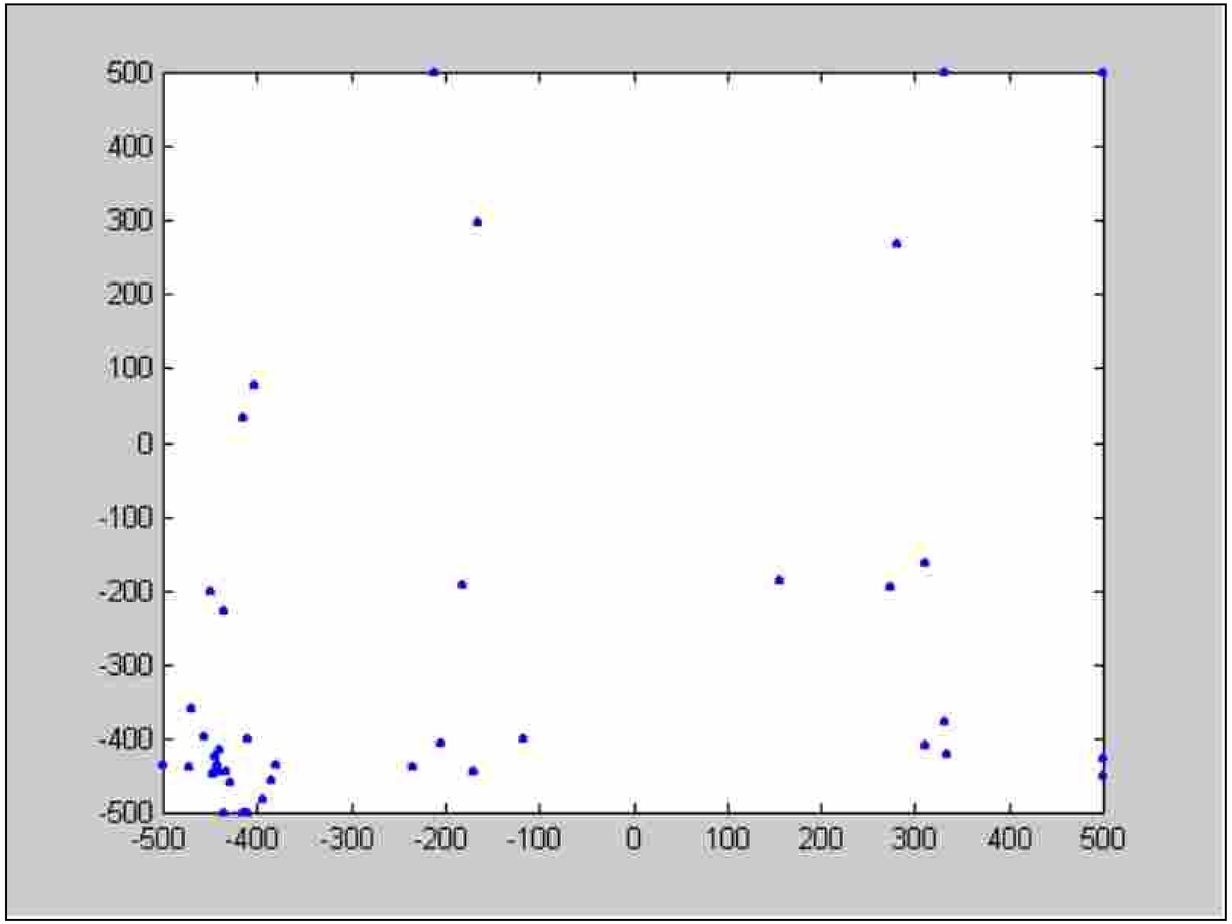
PSO example – after 5 generations



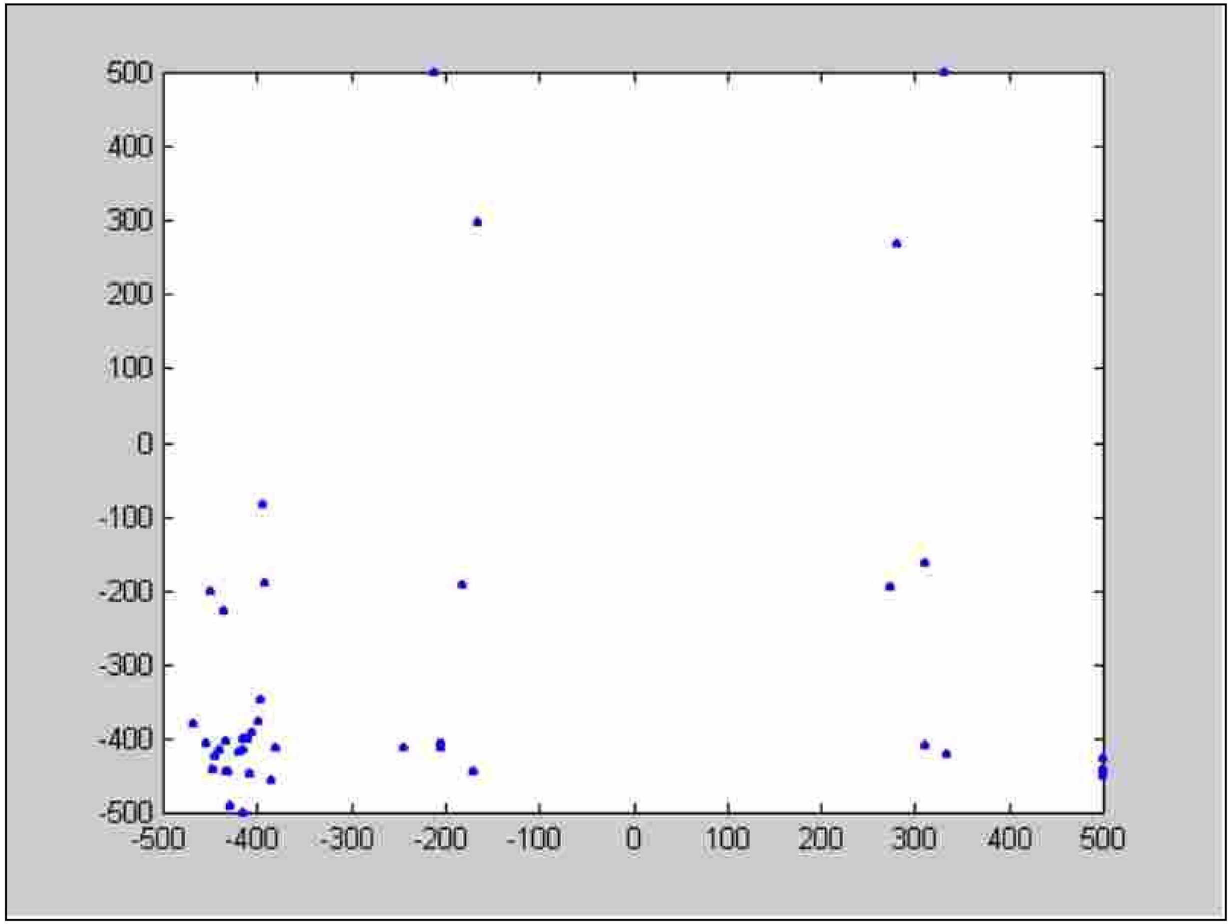
PSO example – after 10 generations



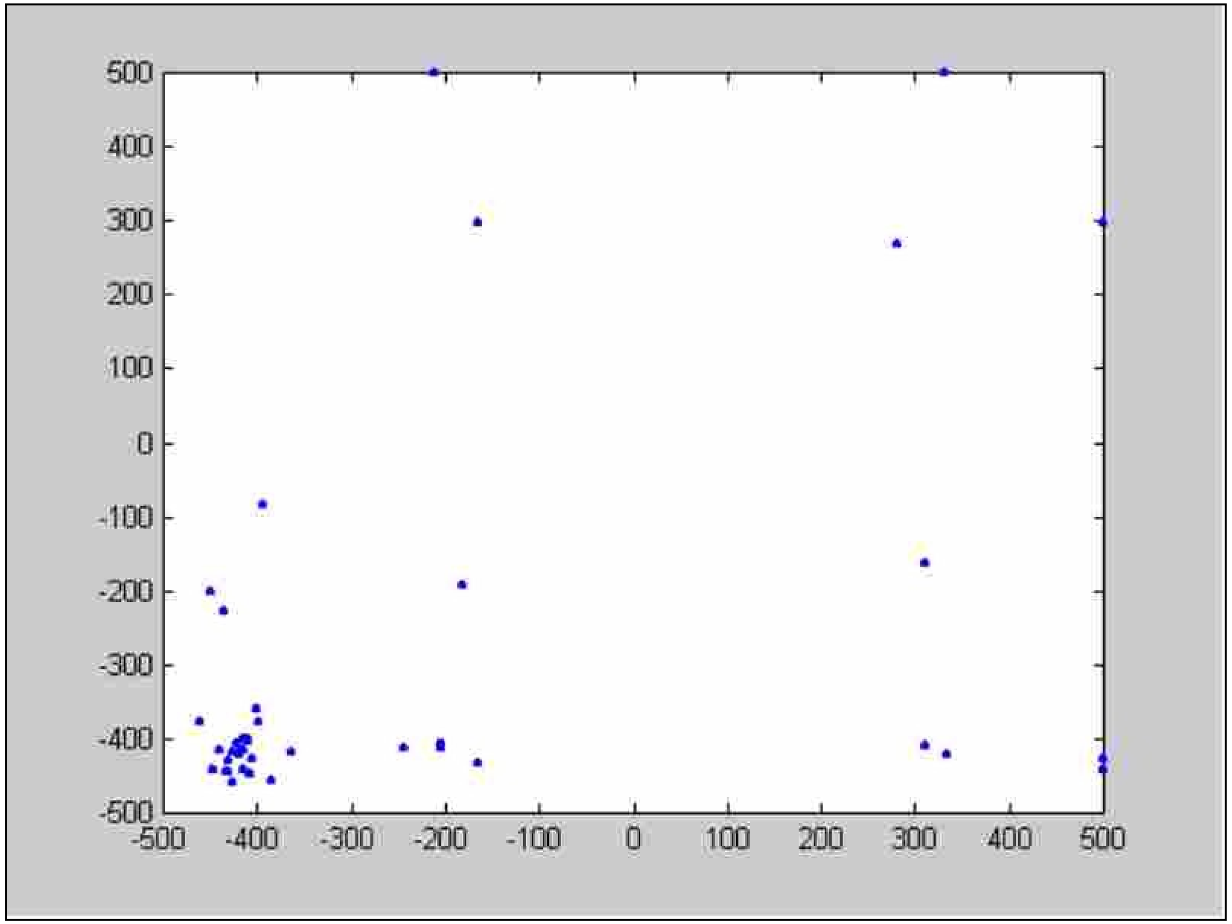
PSO example – after 15 generations



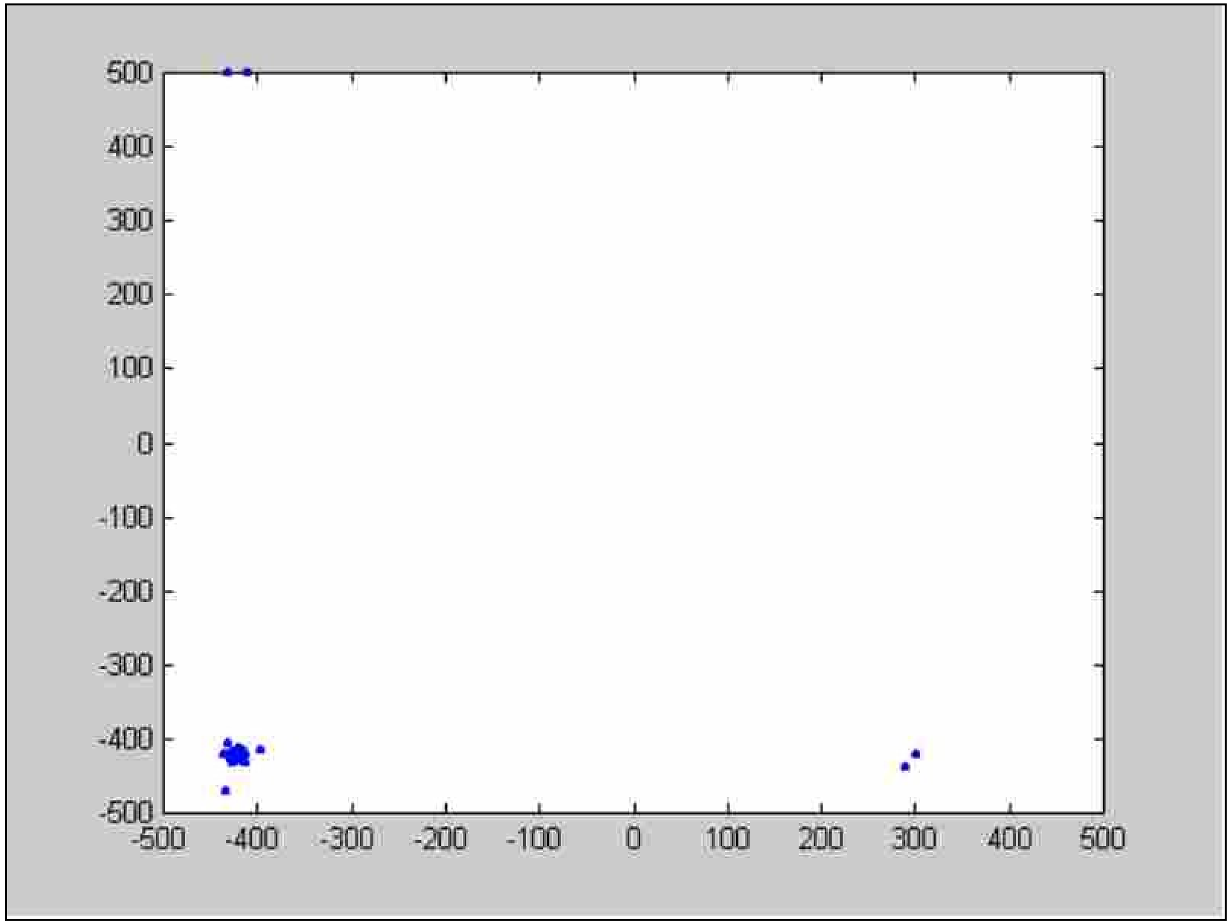
PSO example – after 20 generations



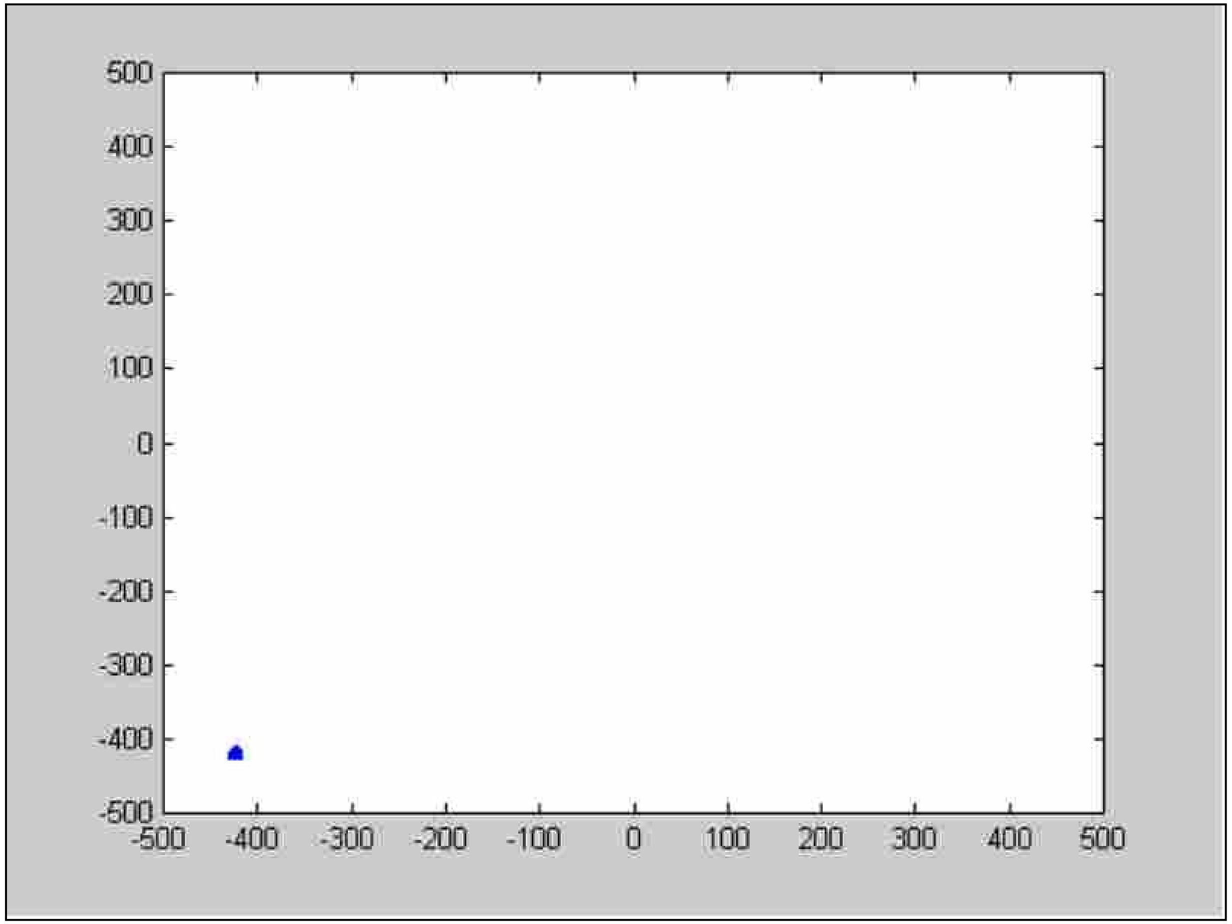
PSO example – after 25 generations



PSO example – after 100 generations

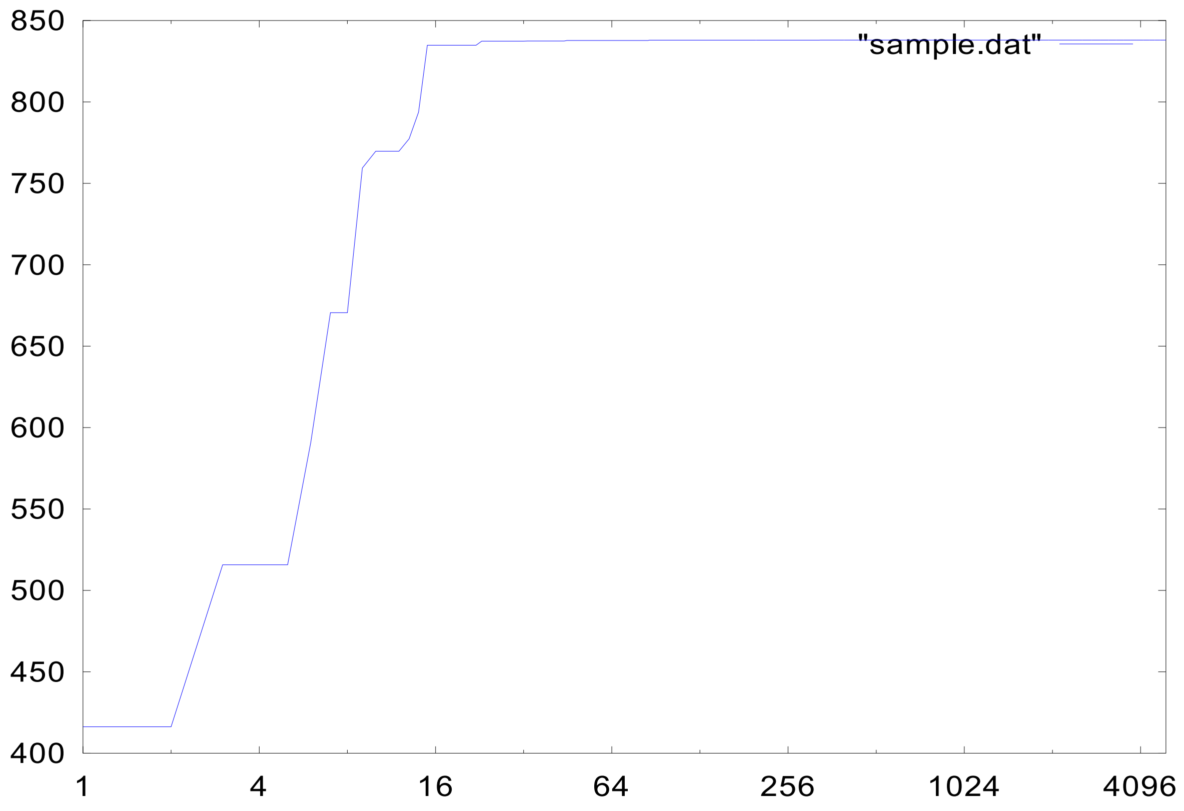


PSO example – after 500 generations



PSO example - result

iteration	gBest
0	416.245599
5	515.748796
10	759.404006
15	793.732019
20	834.813763
100	837.911535
5000	837.965771
optimum	837.9658



Simplified Swarm Optimization (SSO)

SSO

- The fundamental concept of SSO is that each selected variable value may be generated from the current solution, pBest, gBest, or a random number according to the specifics in SSO. Based upon the above concept, the update mechanism of SSO is:

$$x_{ij}^{t+1} = \begin{cases} x_{ij}^t, & \text{if } \rho \in [0, C_w = c_w) \\ p_{ij}, & \text{if } \rho \in [C_w, C_p = C_w + c_p) \\ g_j, & \text{if } \rho \in [C_p, C_g = C_p + c_g) \\ x, & \text{if } \rho \in [C_g, 1) \end{cases}$$

- where is a uniform random number generated in the interval and is a random number between the lower and upper bounds of the *j*th variable; represent the probabilities of the new variable value generated from the current solution, , , and a random number in SSO, respectively.

Firefly Algorithm

About Fireflies – General

- One of the family of insects
- Live in tropical environment
- Have wings
- Produce «cold light» chemically
- Yellow, green, pale-red lights
- Their larvae called glowworm
- ~2000 species
- Flightless females

About Fireflies - Video

- vidzsió

About Fireflies - Behavior

- Their purpose of flashing:
  - Attract mating partners (communication)
  - Attract potential prey
  - Protective warning mechanism

- They have unique flashing pattern
- In some species, females can mimic «mating pattern» to hunt other species

- They have limited light intensity

The Algorithm

- Like Particle Swarm Optimization
- Inspired by the behavior of fireflies
- Developer of the algorithm is Dr. Xin-She Yang
- Three main assumptions:
  1. All fireflies are unisex
  2. Attractiveness $\propto$ Brigtness & Attractiveness $\propto$ 1/Distance
  3. Brightness is determined by objective function

Formulas - Attractiveness

- Suppose absolute darkness
- Light intensity of each firefly is proportional to quality of solution
- Each firefly needs to move towards the brighter fireflies
- Light intensity reduction abides the law:
  - is the light intensity at zero distance
  - *d* is the observer's distance from source
- If we take absorbtion coefficient into account:
  - Attractiveness (,) =

Formulas - Distance

- In our case, *d* is going to be the Euclidean distance

$$d_{i,j} = \mathrm{Distance}(\mathbf{x}^i, \mathbf{x}^j) = \sqrt{\sum_{k=1}^{n} (x_k^i - x_k^j)^2}$$

Formulas - Movement

- Movement consists of two elements
  - Approach to better solutions
  - Move randomly

$$
\begin{aligned}
\mathbf{x}^i &= \mathbf{x}^i + \text{Attractiveness}(I_0, \gamma, \text{Distance}(\mathbf{x}^i, \mathbf{x}^j)) \cdot (\mathbf{x}^j - \mathbf{x}^i) + \alpha \cdot (Random() - \frac{1}{2}) \\
&= \mathbf{x}^i + \text{Attractiveness}(I_0, \gamma, d_{i,j}) \cdot (\mathbf{x}^j - \mathbf{x}^i) + \alpha \cdot (Random() - \frac{1}{2}) \\
&= \mathbf{x}^i + I_0 e^{-\gamma d_{i,j}^2} \cdot (\mathbf{x}^j - \mathbf{x}^i) + \alpha \cdot (Random() - \frac{1}{2}) \\
&= \mathbf{x}^i + \beta \cdot (\mathbf{x}^j - \mathbf{x}^i) + \alpha \cdot (Random() - \frac{1}{2}) \\
&= (1 - \beta) \cdot \mathbf{x}^i + \beta \cdot \mathbf{x}^j + \alpha \cdot (Random() - \frac{1}{2}),
\end{aligned}
$$

- determines the maximum radius of the random step
- determines the step size towards the better solution

Special Cases

- if $\gamma = 0$, then = =
  - Absolutely clear air
  - No light dispersion
  - Each fireflies can see each other
  - Exploration-exploitation is out of balance
- if $\gamma = \infty$, then = =
  - Foggy air
  - Extreme light dispersion
  - Fireflies can’t see each other
  - Random walk
  - Exploration-exploitation is out of balance

Pseudo Code

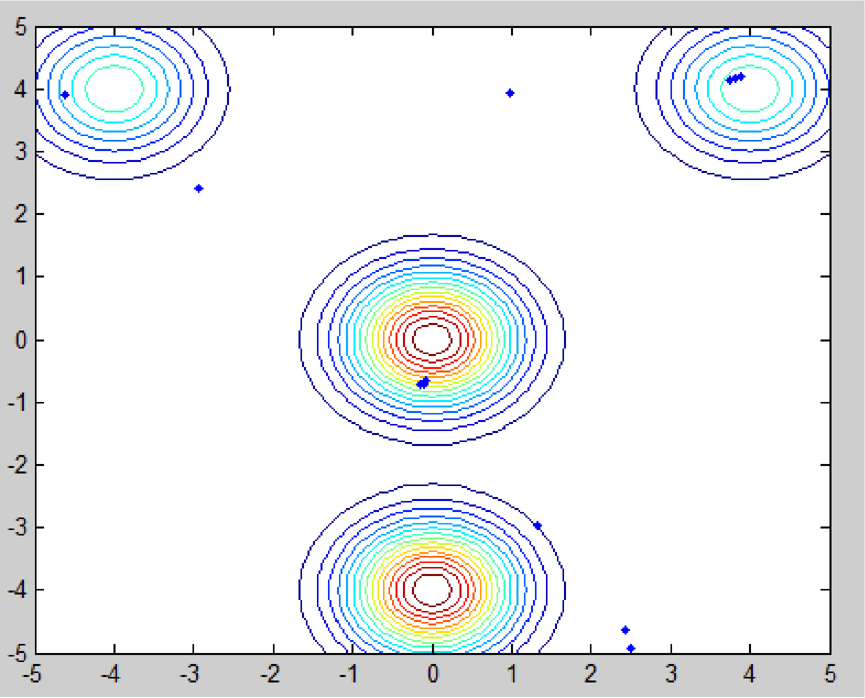
Objective function $f(\mathbf{x})$, $\quad \mathbf{x} = (x_1, ..., x_d)^T$
Generate initial population of fireflies $\mathbf{x}_i$ $(i = 1, 2, ..., n)$
Light intensity I_i *at* $\mathbf{x}_i$ *is determined by* $f(\mathbf{x}_i)$
Define light absorption coefficient γ
while $(t < MaxGeneration)$
for $i = 1 : n$ *all* n *fireflies*
  for $j = 1 : i$ *all* n *fireflies*
    if $(I_j > I_i)$, *Move firefly* i *towards* j *in d-dimension;* **end if**
    Attractiveness varies with distance r *via* $\exp[-\gamma r]$
    Evaluate new solutions and update light intensity
  end for j
end for i
Rank the fireflies and find the current best
end while
Postprocess results and visualization

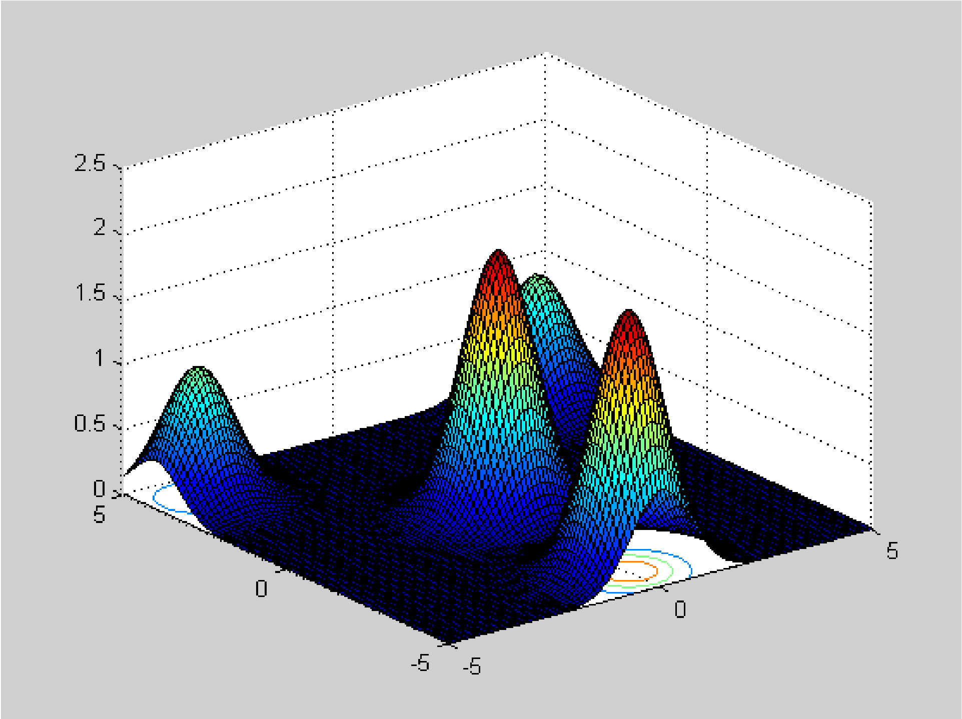
Applications

- Digital Image Compression and Image Processing
- Feature Selection and Fault Detection
- Antenna Design
- Structural Design
- Scheduling
- Semantic Web Composition
- Chemical Phase Equilibrium
- Clustering
- Dynamic Problems
- Rigid Image Registration Problems

Fireflies in Use

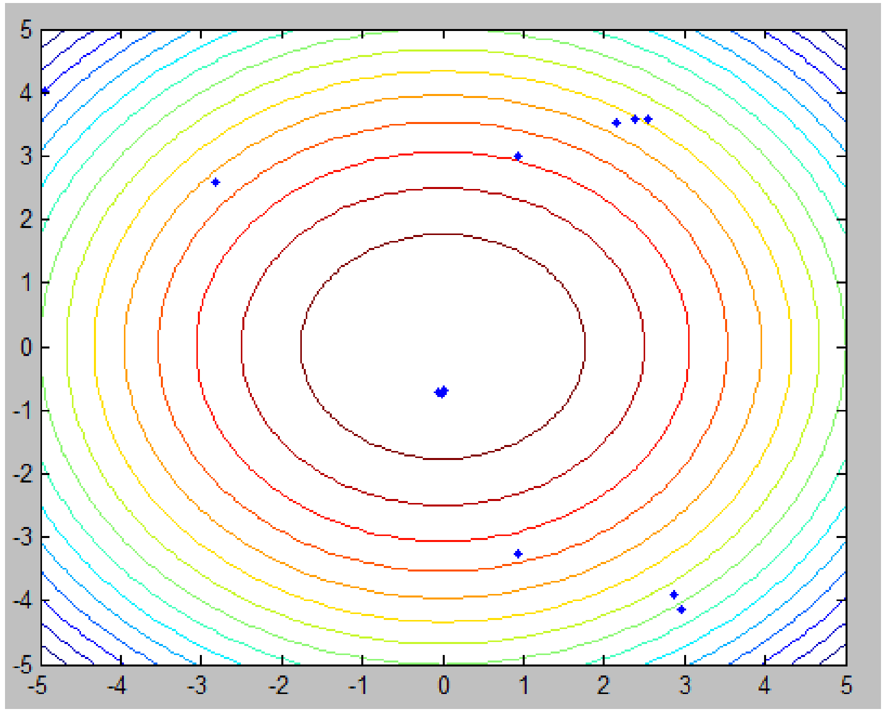
- 1. Four Peak Function

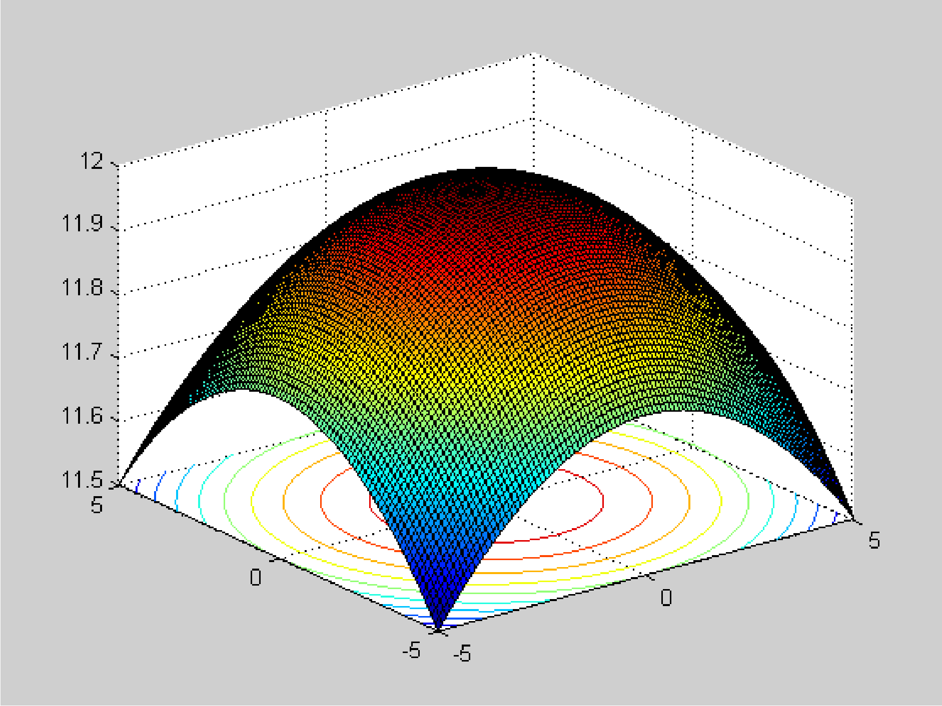




Fireflies in Use

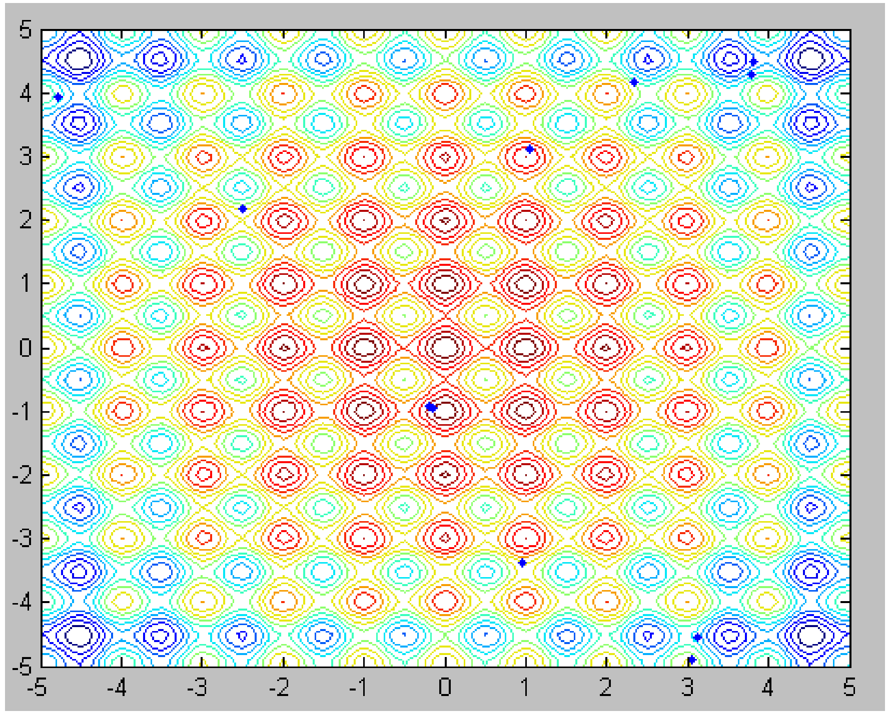
- 2. Parabolic Function

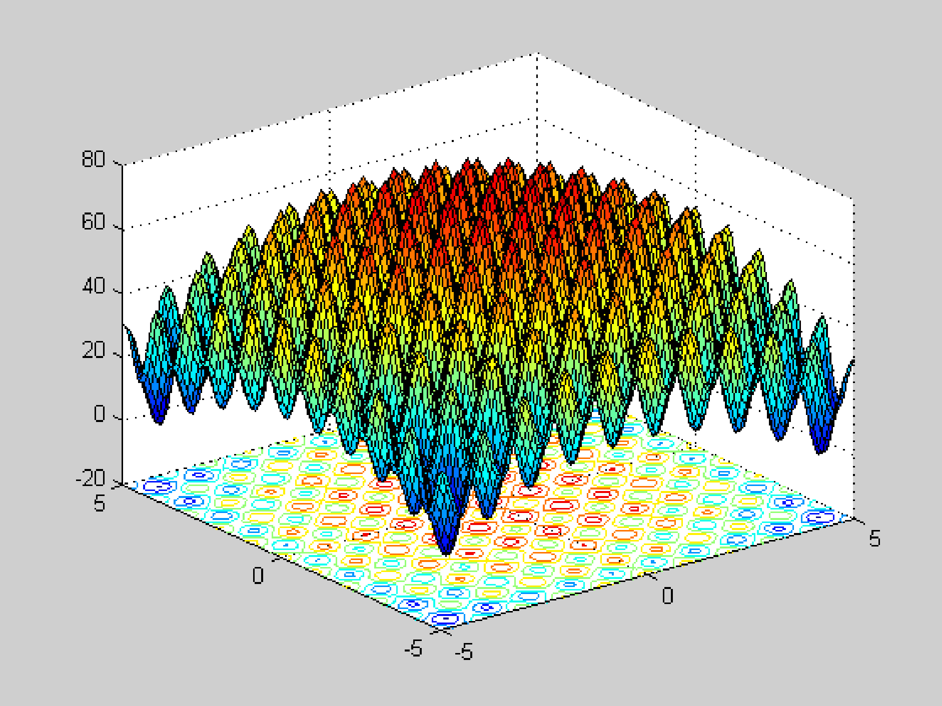




Fireflies in Use

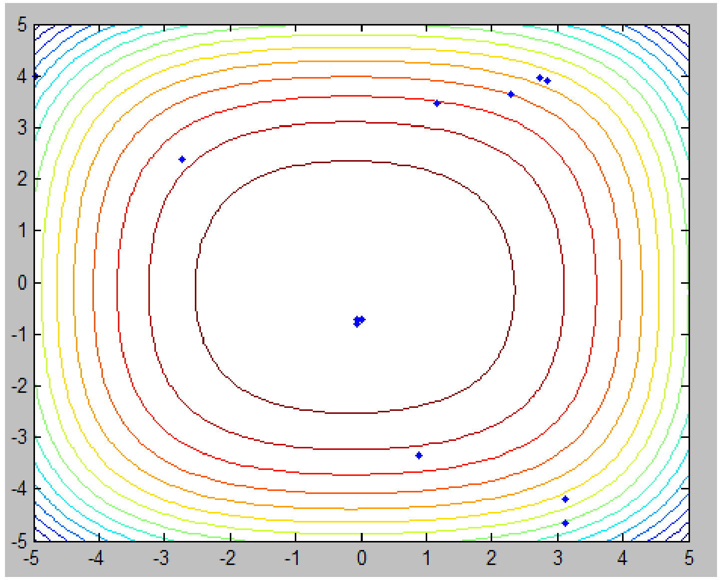
3. Rastrigin function

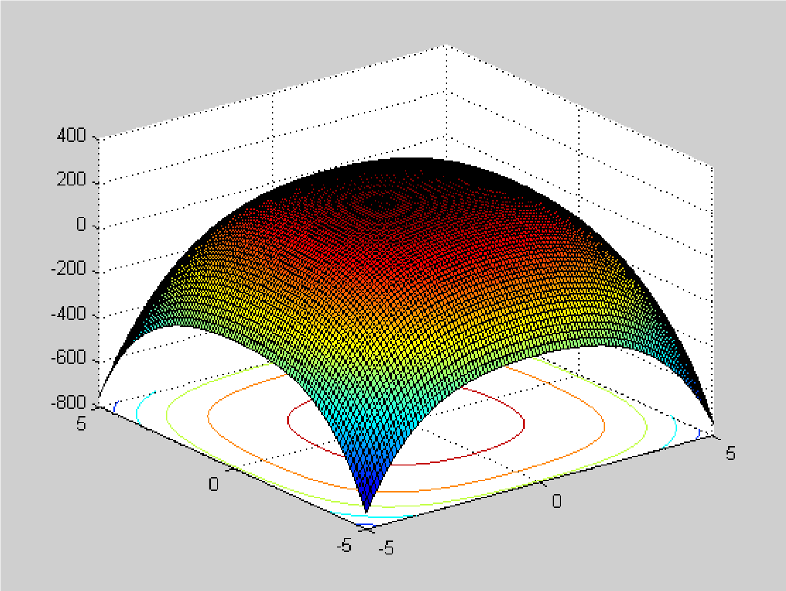




Fireflies in Use

4. Styblinski function





Comparison with PSO

Function	N	PSO	Firefly Algorithm
Four-peak	15	1,5356	1,4840
	20	2,0135	1,9326
	25	2,4959	2,3652
Parabolic	15	1,5482	1,5039
	20	2,0884	1,9296
	25	2,6466	2,3534
Rastrigin	15	9,6761	9,5298
	20	12,6412	12,5404
	25	15,6878	15,5457
Styblinski	15	1,6444	1,5478
	20	2,1504	2,0725
	25	2,6144	2,5323

Performance Comparison

	Genetic Algorithm	Particle Swarm	Firefly Algorithm
Michalewicz	95%	98%	99%
Rosenbrock	90%	98%	99%
De Jong	100%	100%	100%
Schwefel	95%	97%	100%
Ackley	90%	92%	100%
Rastrigin	77%	90%	100%
Easom	92%	90%	100%
Griewank	90%	92%	100%
Shubert (18 min)	89%	92%	100%
Yang	83%	90%	100%

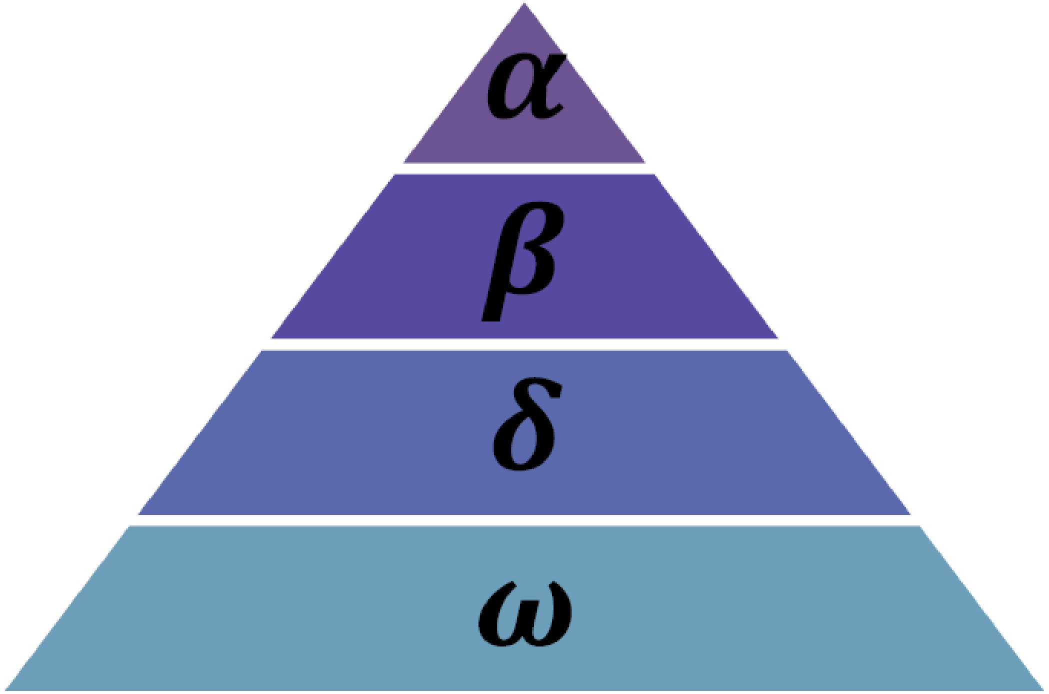
Grey Wolf Optimizer

Grey Wolf Optimizer

- Grey wolf optimizer (GWO) is a population based meta-heuristics algorithm simulating the leadership hierarchy and hunting mechanism of grey wolves in nature
- Grey wolves are considered as apex predators, which means they are at the top of the food chain.
- Grey wolves prefer to live in a groups (packs), each group contains 5-12 members on average.
- All the members in the group have a very strict social dominant hierarchy as shown in the following figure.

Hierarchy of grey wolf

- The social hierarchy consists of four levels as follow.
- Dominance decreases from top down
- The first level is called Alpha (α). The alpha wolves are the leaders of the pack and they are a male and a female.



Hierarchy of grey wolf

- Alpha is responsible for making decisions about hunting, time to walk, sleeping place and so on.
- The pack members have to dictate the alpha decisions and they acknowledge the alpha by holding their tails down.
- The alpha wolf is considered the dominant wolf in the pack and all his/her orders should be followed by the pack members.

Hierarchy of grey wolf

- The second level is called Beta (β).
- The betas are subordinate wolves, which help the alpha in decision making.
- The beta wolf can be either male or female and it considers the best candidate to be the alpha when the alpha passes away or becomes very old.
- The beta reinforces the alpha’s commands throughout the pack and gives the feedback to alpha.

Hierarchy of grey wolf

- The third level is called Delta (δ)
- The delta wolves are not alpha or beta wolves and they are called subordinates.
- Delta wolves have to submit to the alpha and beta but they dominate the omega (the lowest level in wolves social hierarchy).
- There are different categories of delta as follows:
  - Scouts: The scout wolves are responsible for watching the boundaries of the territory and warning the pack in case of any danger.
  - Sentinels: The sentinel wolves are responsible for protecting the pack.

Hierarchy of grey wolf

- Elders: The elder wolves are the experienced wolves who used to be alpha or beta.
- Hunters: The hunter wolves are responsible for helping the alpha and beta wolves in hunting and providing food for the pack.
- Caretakers: The caretakers are responsible for caring for the ill, weak and wounded wolves in the pack.

- The fourth (lowest) level is called Omega (ω)
- The omega wolves are considered the scapegoat in the pack, they have to submit to all the other dominant wolves.
- They may seem not important individuals in the pack and they are the last allowed wolves to eat.
- The whole pack is fighting in the case of losing the omega.

Hierarchy of grey wolf

- In the grey wolf optimizer (GWO), we consider the fittest solution as the alpha, and the second and the third fittest solutions are named beta and delta, respectively.
- The rest of the solutions are considered omega
- In GWO algorithm, the hunting is guided by α, β, and δ
- The ω solutions follow these three wolves.

Grey wolf hunting phases

- The main phases of grey wolf hunting are as follows:
  - Tracking, chasing, and approaching the prey. (A)
  - Pursuing, encircling, and harassing the prey until it stops moving. (B,C,D)
  - Attack towards the prey. (E)

Grey wolf encircling prey

- During the hunting, the grey wolves encircle prey.
- The mathematical model of the encircling behavior is presented in the following equations:

$$\vec{D} = |\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)|$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D}$$

- where t is the current iteration, A and C are coefficient vectors, X_p is the position vector of the prey, and X indicates the position vector of a grey wolf.
- The vectors A and C are calculated as follows: $\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a}$

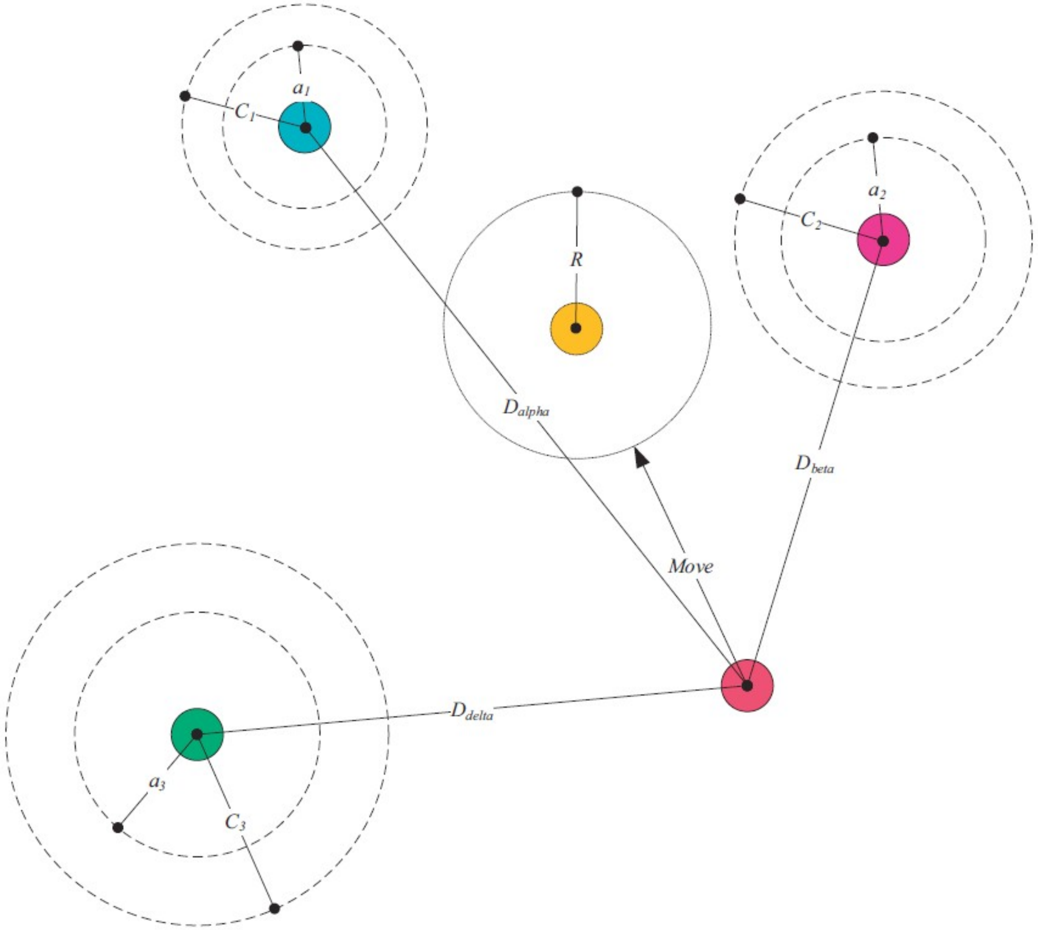
$$\vec{C} = 2 \cdot \vec{r}_2$$

- where components of a are linearly decreased from 2 to 0 over the course of iterations and r_1, r_2 are random vectors in [0, 1]

Grey wolf hunting

- The hunting operation is usually guided by the alpha.
- The beta and delta might participate in hunting occasionally.
- In the mathematical model of hunting behavior of grey wolves, we assumed the alpha, beta, and delta have better knowledge about the potential location of prey.
- The first three best solutions are saved and the other agent are obliged to update their positions according to the position of the best search agents as shown in the following equations.

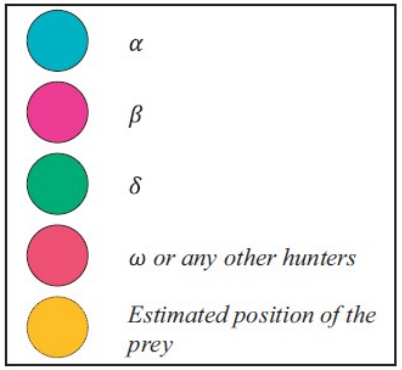
Grey wolf hunting (position update)



$$\vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}|, \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}|, \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}|$$

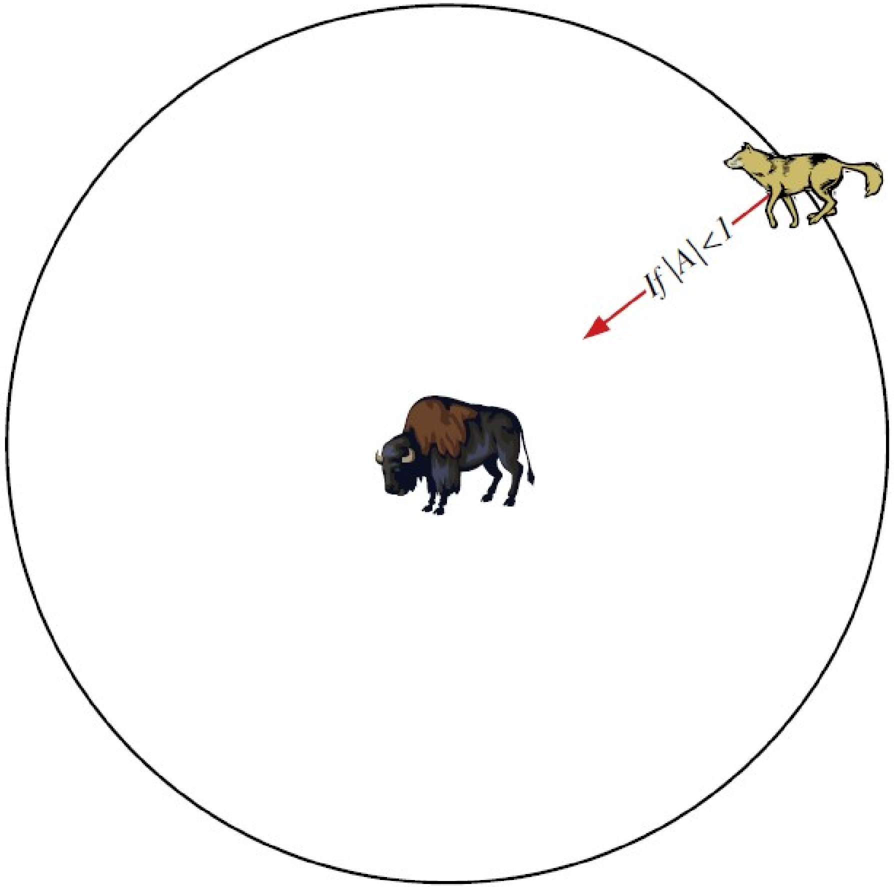
$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot (\vec{D}_\beta), \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta)$$

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}$$



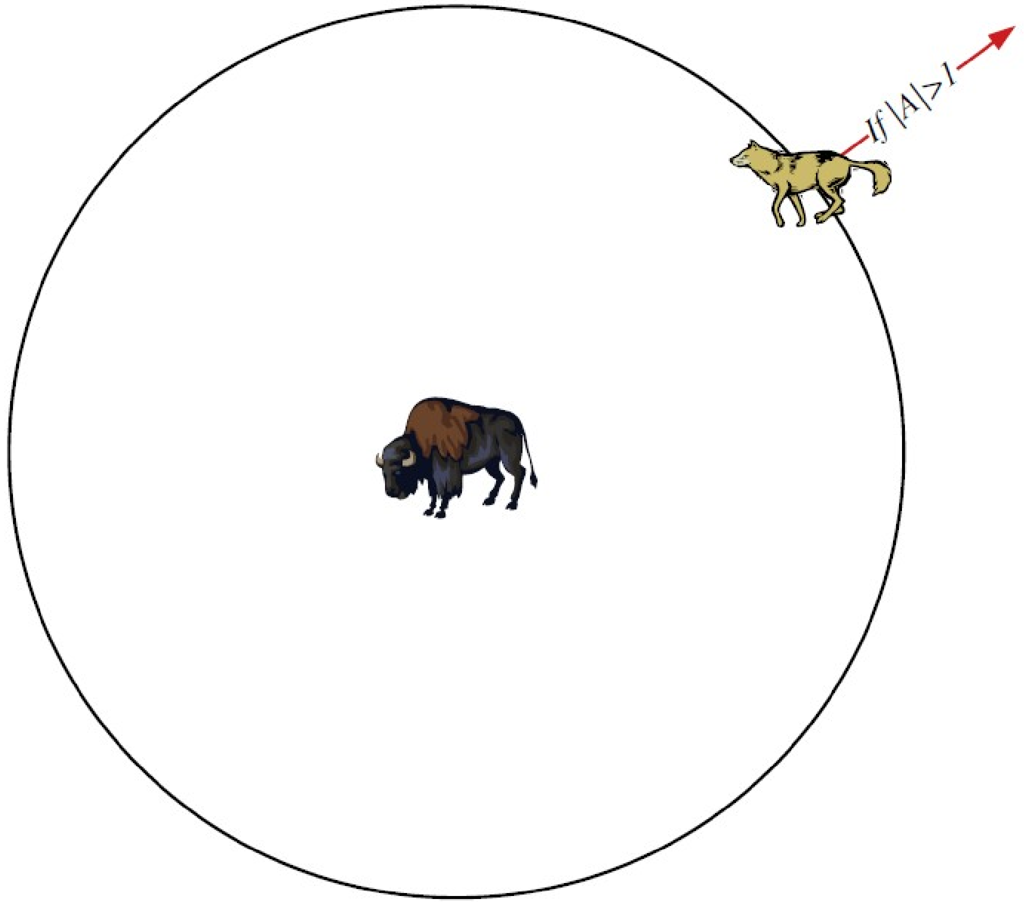
Attacking prey (exploitation)

- The grey wolf finishes the hunt by attacking the prey when it stops moving.
- The vector A is a random value in interval [-2a, 2a], where a is decreased from 2 to 0 over the course of iterations.
- When $|A| < 1$, the wolves attack towards the prey, which represents an exploitation process.



Search for prey (exploration)

- The exploration process in GWO is applied according to the position, and that diverges from each other to search for prey and converges to attack prey.
- The exploration process is modeled mathematically by utilizing *A* with random values greater than 1 or less than -1 to oblige the search agent to diverge from the prey.
- When $|A| > 1$, the wolves are forced to diverge from the prey to find a fitter prey.



GWO algorithm

Initialize the grey wolf population X_i (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
X_α=the best search agent
X_β=the second best search agent
X_δ=the third best search agent
***while** (t < Max number of iterations)*
    ***for** each search agent*
        Update the position of the current search agent
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update X_α, X_β, and X_δ
    t=t+1
end while
return X_α