Particle Swarm Optimization

Simulation in Greenfoot

Particle Swarm Optimization (PSO) , developed by Eberhart and Kennedy in 1995, is a form of swarm intelligence in which the behavior of a biological social system like a flock of birds or a school of fish is simulated. When a swarm looks for food, its individuals will spread in the environment and move around independently. Each individual has a degree of freedom or randomness in its movements which enables it to find food accumulations. So, sooner or later, one of them will find something digestible and, being social, announces this to its neighbors [1].

PSO is a robust stochastic optimization technique based on the movement and intelligence of swarms. It applies the concept of social interaction to problem solving.[2]

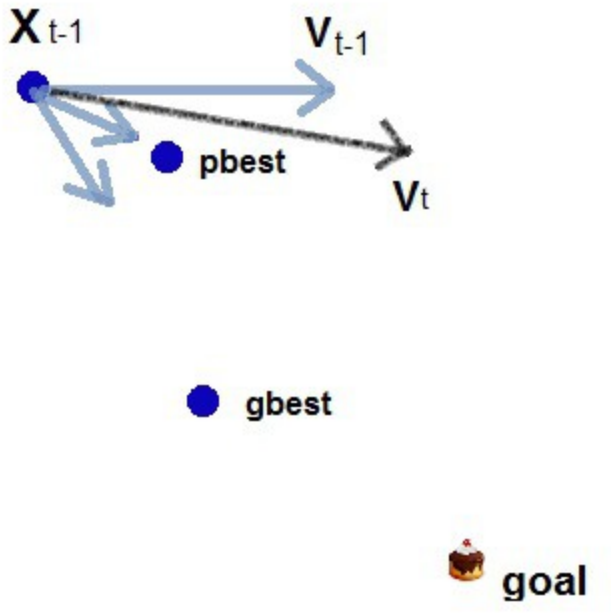
The first practical application of PSO was in the field of neural network training and was reported together with the algorithm itself (Kennedy and Eberhart 1995). Many more areas of application have been explored ever since, including telecommunications, control, data mining, design, combinatorial optimization, power systems, signal processing, and many others. To date, there are hundreds of publications reporting applications of particle swarm optimization algorithms. For a review, see (Poli 2008). Although PSO has been used mainly to solve unconstrained, single-objective optimization problems, PSO algorithms have been developed to solve constrained problems, multi-objective optimization problems, problems with dynamically changing landscapes, and to find multiple solutions.[3]

PSO uses a particle population, each particle having a position in a N-dimensional space and a travel velocity. Each particle has a memory in which it retains its best position and the best position achieved by a particle of swarms. So the future position of each particle is influenced by the previous velocity (inertia), the best personal position (pbest) and the best

position of a particle of the swarm (gbest).

Pseudocode:

- population initialization (random particle positions)
- fitness function calculation
- selection of gbest and pbest
- calculating new particle positions
- check stop condition



X – Particle position , V – Particle velocity

Updating the position is done using the following formula:

$Vt = c1*rand()*V_{t-1} + c2*rand()*(pbest-X_{t-1}) + c3*rand()*(gbest-X_{t-1})$

$Xt = X_{t-1} + V_t$

where Vt - speed at time t; Xt - position at time t;

c1 - constant representing the acceleration factor

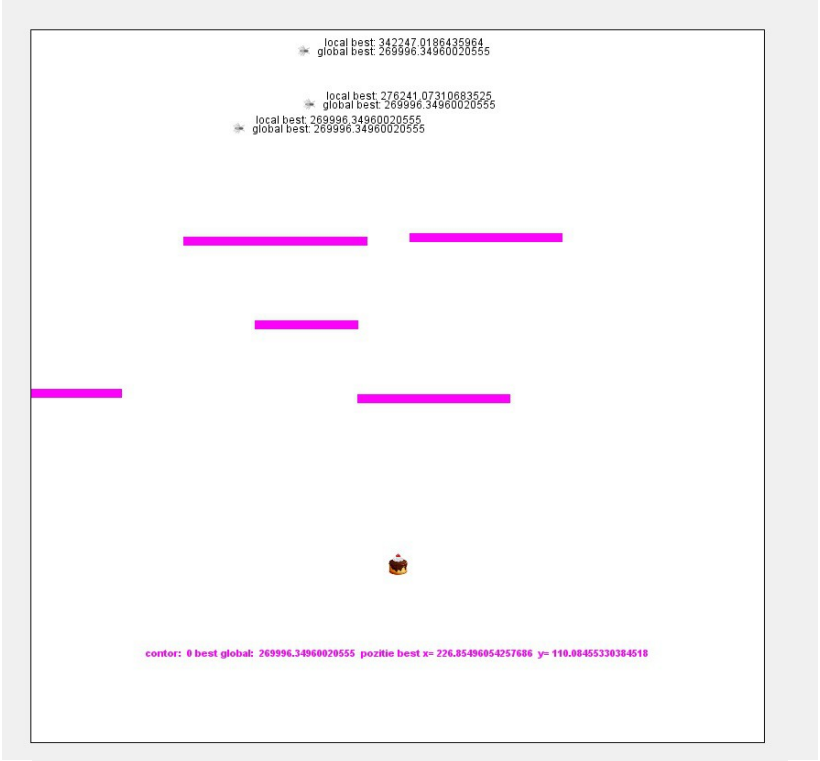
c2 – constant representing the learning factor

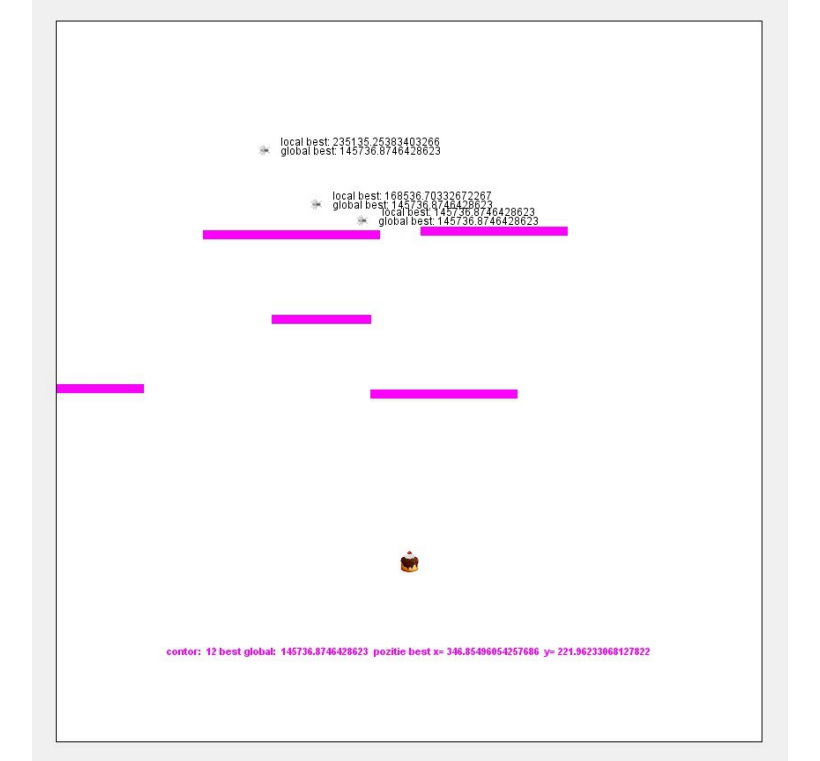
c3 – constant representing the social factor

Implementation in Greenfoot

For graphical visualization we used only 2 dimensions for the particle position, the fitness function - the Cartesian distance.

In a first variant we only used 3 particles and we showed the best fitness function achieved by the particle and swarms.





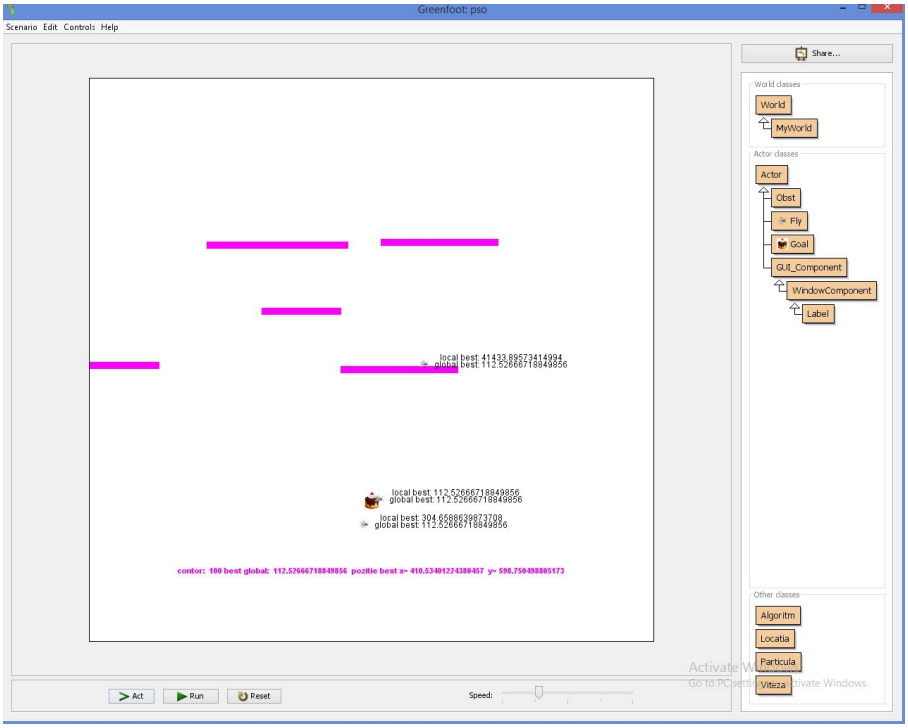


Illustration 1: contor : 100

In the second variant I used a population of 100 particles:

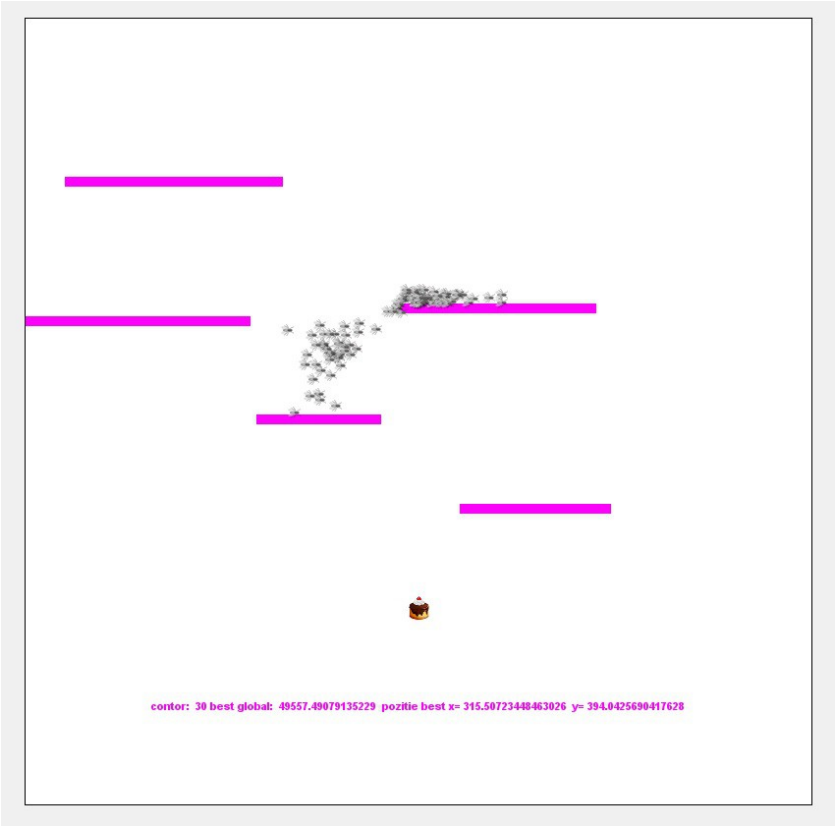


Illustration 2: contor: 30

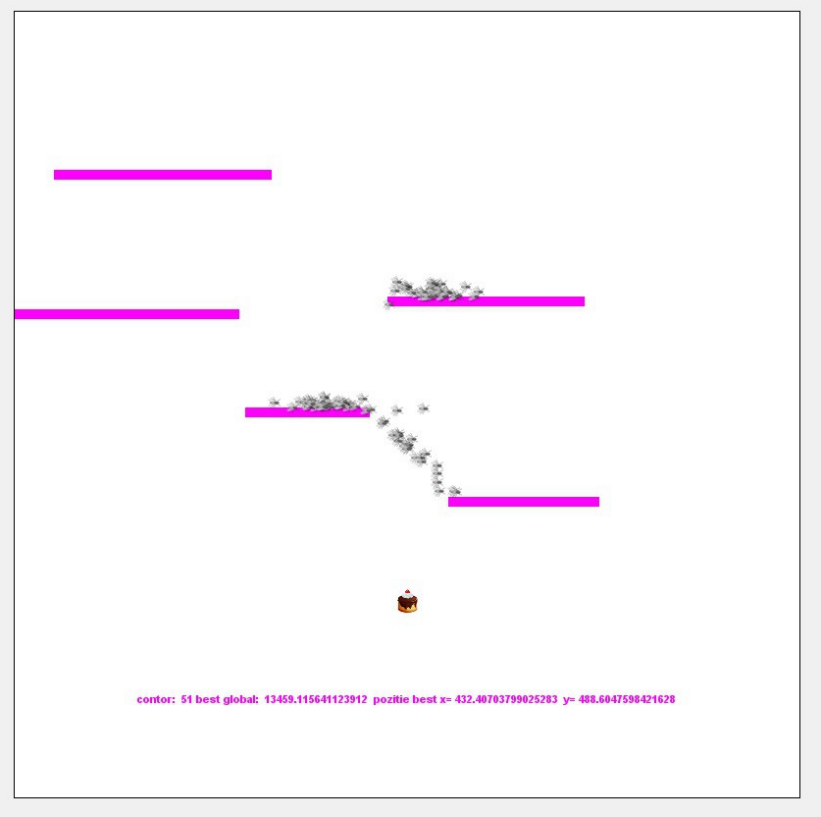


Illustration 3: contor: 51

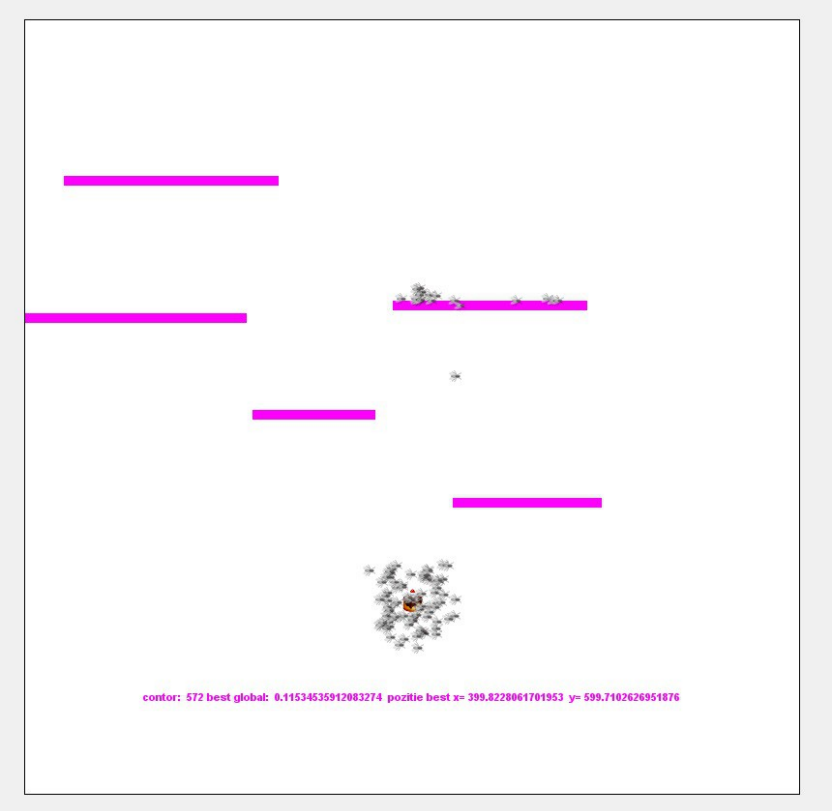


Illustration 4: contor 572

[1] Global Optimization Algorithms – Theory and Application – Thomas Weise , http://www.it-weise.de/projects/book.pdf.

[2] Swarm intelligence - http://slideplayer.com/slide/11340780.

[3] http://www.scholarpedia.org/article/Particle_swarm_optimization.