Particle Swarm Optimization

Summary of a paper by
James Kennedy and Russell Eberhart

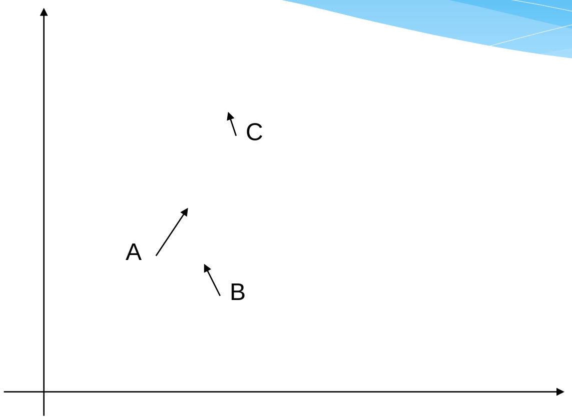
Introduction

- Particle Swarm Optimization (PSO)
- A method for optimizing continuous non-linear functions
- Has connections to A-life, flocking, and evolutionary computing

Structure of the System

* 2D
* Agents:
  * Randomly placed in 2D
  * Random initial X, Y velocities
* During each iteration, we determine the nearest neighbor and take its velocity vector.

Structure of the System



“Craziness”

* Very simple rule.
* Lead to a situation where the system converged to a situation where all agents took on the same, non-changing direction.
* To correct this behavior, the velocity vectors were changed by a small, randomly determined factor.
* This resulted in a rather artificial system.

Optimization

Optimization

* The purpose of this system is to find an optimum.
* To accomplish this, every agent determines the value of the function to be optimized at their current location.
* Additionally, the maximum of each agent is stored.
* We store the individual maxima in an array **pbest[]** and their X and Y positions in **pbestx[]** and **pbesty[]**. The length of the array corresponds to the number of agents.

Use of the Local Optima

* Agents explore their world.
* If an agent is to the right of its maximum, its velocity decreases as follows:

  if (presentx[] > pbestx[])
  *then vx[] = vx[ ] – rand()*p_increment*
* **p_increment** is a tunable parameter
* If an agent is to the left of its maximum, its velocity increases correspondingly.
* Similarly for the vertical axis

Global Maximum

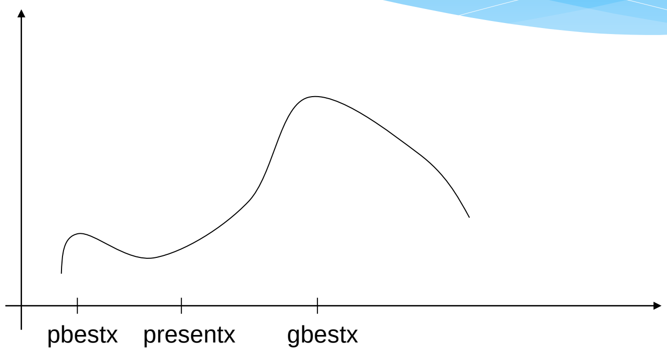
- So far, all agents act independently.
- In addition to the local maxima, we store the global maximum.
- The index of the agent that produced the global maximum is stored in the variable **gbest**.
- It is available to all agents.
- The velocity of each agent is now determined as follows (for both sides and dimensions):

  if (presentx[] > pbestx[gbest])
  then vx[] = *vx[ ] – rand()*g_increment*

Options

- Through experiments, it was determined that high values of **g_increment** tend to have the agents convert quickly towards the global maximum.
- With small values of **g_increment** the agents tend to take their time, thereby exploring the space more thoroughly.
- Combine with local update rule or not.
- Limited flocking, as each agent only cares about its own value and the overall best value.

Visualization of influence of **pbest** and **gbest**



Improvements

* In order to improve the exploration of the space, the velocity should change relative to the distance to the individual maximum:

  if (presentx[] > pbestx[gbest])

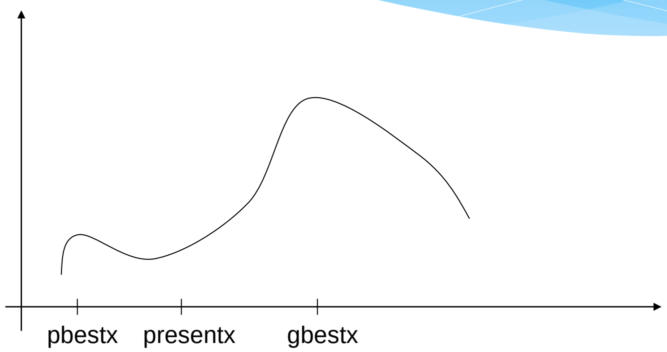
  then *vx[] = vx[] - rand() * p_increment * (pbestx[gbest] – presentx[])*
* Initially, we take large steps which get smaller as we approach the local maximum.

Improvements

* In higher-dimensional spaces, it is hard to estimate whether **p_increment** or **g_increment** should be larger.
* Instead, the random value will be multiplied by 2 to give it a mean of 1.
* Through this modification, agents will overshoot the global maximum half the time.
* The expression is now:

  vx[] = vx[] + 2 * rand() * (pbestx[] – presentx[]) +
  2 * rand() * (pbest[gbest] – presentx[])

Visualization of Improvements



What is PSO?

- The authors state that "*conceptually, it seems to lie somewhere between genetic algorithms and evolutionary programming.*"
- They additionally state that "*much of the success of PS seems to lie in the agents' tendencies to hurtle past their target.*"