SWARM INTELLIGENCE - II

Today’s Lecture

- Recap
- Particle Swarm Optimization
  - History
  - Continuous PSO
  - Variants
  - Binary PSO

Swarm Intelligence

- Any attempt to design algorithms or distributed problem solving devices inspired by the collective behaviour of social insect colonies and other animal societies

(Bonabeau et al., 1999)

Swarm Intelligence Algorithms

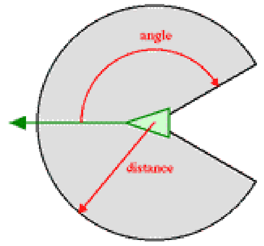
- Ant Colony Optimization
- Particle Swarm Optimization
- Stochastic Diffusion Search
- The Bees Algorithm

ACO Demo

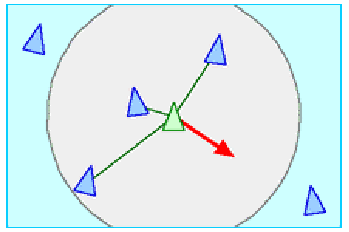
- ACO Demonstration - Christian Borgelt
- http://www.borgelt.net/acopt.html

Particle Swarm Optimization

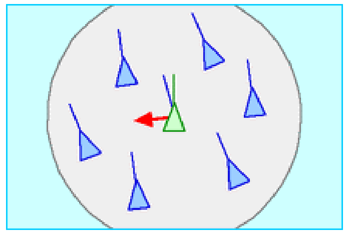
History



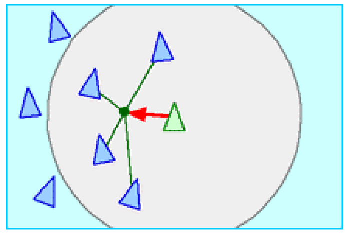
In 1986 Reynolds created model of coordinated animal motion in which the agents (named boids) obeyed three simple local rules which lead to surprisingly realistic behaviour:



Separation: steer to avoid crowding local flockmates



Alignment: steer towards the average heading of local flockmates



Cohesion: steer to move toward the average position of local flockmates

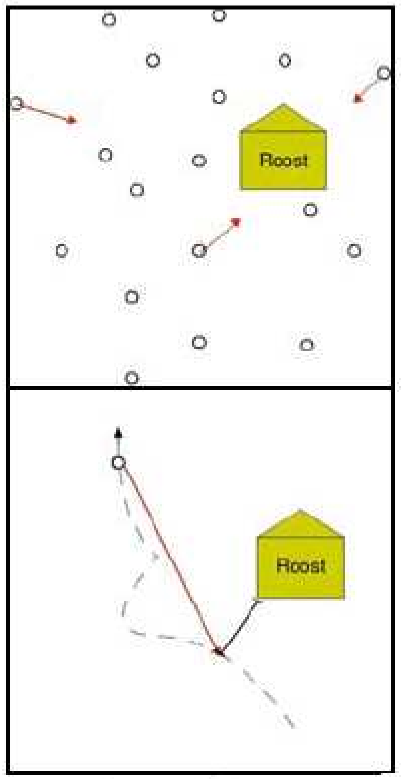
http://www.red3d.com/cwr/boids/

DEMO: Boids

- http://www.vergenet.net/~conrad/boids/
- http://www.siggraph.org/education/materials/HyperGraph/animation/art_life/video/3cr.mov

History

- Heppner included a ‘roost’ (cornfield) that the birds flocked around before landing there.
  - Birds are attracted to a focal point, or roost; the closer they get to it, the stronger the attraction.
  - Birds are attracted to each other, but become repelled if they get too close.
  - Birds want to maintain a fixed velocity.
  - Flight paths can be altered by random inputs such as wind gusts, rainfall, appearance of a predator.
- Grenander translated Heppner's rules into a mathematical language



Particle Swarm Optimization

Hypothesis: Individuals profit from the discoveries and previous experience of all other members of the swarm

- Use this to optimize continuous functions

Particle Swarm Optimization

- Developed by Russell Eberhart and James Kennedy in 1995

- “particle swarm algorithm imitates human (or insects) social behaviour. Individuals interact with one another while learning from their own experience, and gradually the population members move into better regions of the problem space”.

Kennedy and Eberhart

The Optimization Problem

In general, any optimization problem *P* can be described as a triple (*S*,*Ω*, *f*), where

1. *S* is the search space defined over a finite set of decision variables X_i, i = 1, . . . , *D*. *Ω* is a set of constraints among the variables;
2. $f : S \rightarrow \Re^n$ is the objective function that assigns a positive cost value to each element (or solution) of *S*.

Goal is to find a solution $s' \in S$ such that $f(s') \leq f(s)$, $\forall s \in S$ (in case we want to minimize the objective function), or $f(s') \geq f(s)$, $\forall s \in S$ (in case the objective function must be maximized).

The Original PSO

- Each individual in the particle swarm is composed of three D-dimensional vectors, where D is the dimensionality of the search space. These are
  - the current position $\mathbf{x}_i$ - *a solution*
  - the previous personal best position $\mathbf{p}_i$, and
  - the velocity $\mathbf{v}_i$.
- New positions are chosen by adding $\mathbf{v}_i$ to the coordinates $\mathbf{x}_i$ which can be seen as a step size
- For each individual, the *fitness* value of the previous personal best position is stored in a variable that can be called ***pbest***$_i$ *(*for previous best*)*

Original PSO: Maximization

```
Randomly generate an initial swarm
repeat
    for each particle $i$ do
            if $f(\mathbf{x}_i) > f(\mathbf{p}_i)$ then $\mathbf{p}_i = \mathbf{x}_i$ and $pbest_i = f(\mathbf{x}_i)$;
            $\mathbf{p}_g = \max(\mathbf{p}_{\text{neighbours}})$;
            UpdateVelocity;
            UpdatePosition;
    end for
until termination criterion is met
```

$pbest_i$

Original PSO

- Velocity Update & Position Update

$$\mathbf{v}_i = \mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i)$$

$$\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i$$

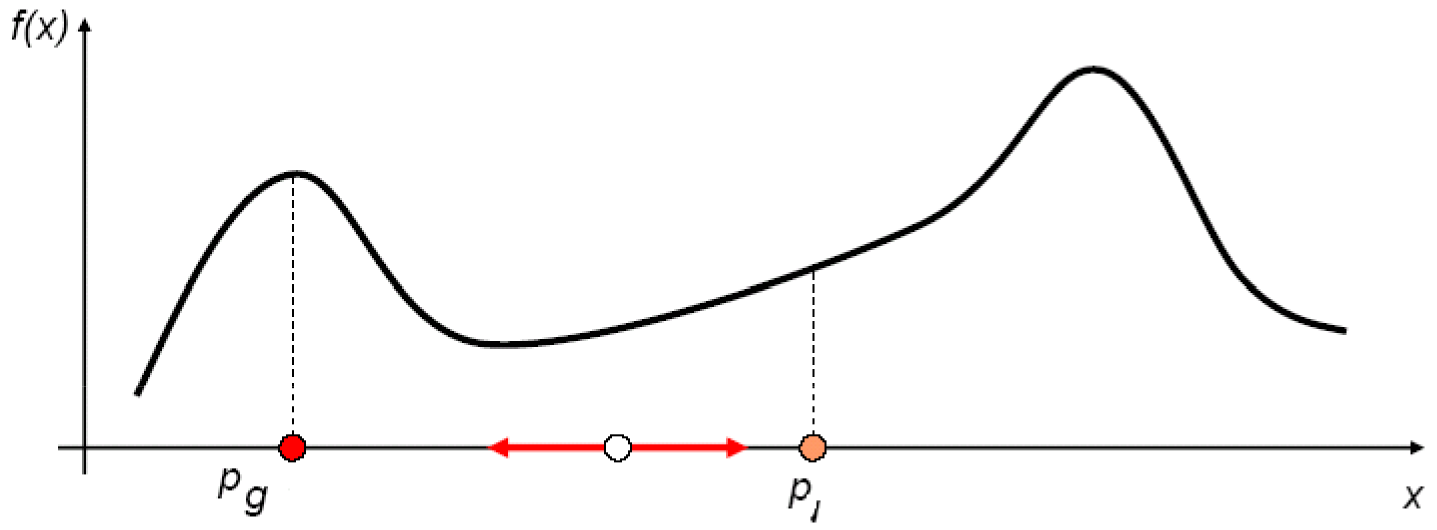
  - $\otimes$ - pointwise vector multiplication
  - $\varphi_1 = c_1\mathbf{r}_1$ and $\varphi_2 = c_2\mathbf{r}_2$ where
  - $\mathbf{r}_1$ and $\mathbf{r}_2$ are two vectors of random numbers uniformly chosen from [0, 1];
  - c_1 and c_2 are acceleration coefficients

The Velocity

$$\mathbf{v}_i = \overbrace{\mathbf{v}_i}^{\text{momentum}} + \overbrace{\boxed{\varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i)}}^{\text{Cognitive component}} + \overbrace{\boxed{\varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i)}}^{\text{Social component}}$$

- Three components of velocity (rate of change)
  - **Momentum** - previous velocity term to carry the particle in the direction it has travelled so far
  - **Cognitive** Component - tendency of the particle to return to the best position it has visited so far
  - **Social** component - tendency of the particle to be attracted towards the position of the best position found by the entire swarm.

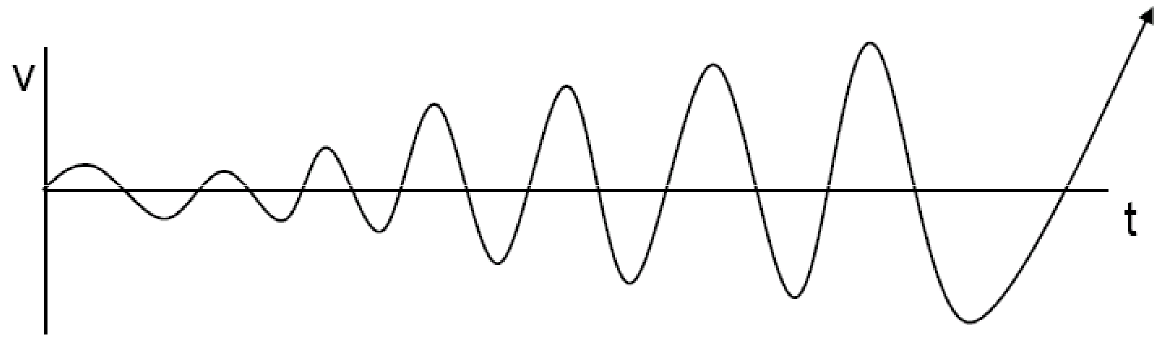
The Acceleration Coefficients



- c_1 and c_2 determine the magnitude of the random forces in the directions of $\mathbf{p}_i$ and $\mathbf{p}_g$
- The behaviour of the PSO can change drastically with c_1 and c_2.
- The system can become more or less responsive or even unstable.

Acceleration Coefficients

- Higher acceleration coefficients result in less stable systems in which the velocity has a tendency to explode



- To fix this, the velocity $\mathbf{v}_i$ is usually kept within the range $[-\mathbf{v}_{max}, \mathbf{v}_{max}]$
- The optimal value of $\mathbf{v}_{max}$ is problem specific and no reasonable rule of thumb is available
- Moreover, limiting the velocity does not necessarily prevent particles from leaving the search space, nor does it help to guarantee convergence

Acceleration Coefficients

$c_1>0, c_2=0$	particles are independent hill-climbers
$c_1=0, c_2>0$	swarm is one stochastic hill-climber
$c_1=c_2>0$	particles are attracted to the average
$c_2>c1$	more beneficial for unimodal problems
$c_1>c_2$	more beneficial for multimodal problems
low c_1, c_2	smooth particle trajectories
high c_1, c_2	more acceleration, abrupt movements

- Adaptive acceleration coefficients have also been proposed. For example to have c_1 and c_2 decreased over time

$$\mathbf{v}_i = \mathbf{v}_i + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + c_2\mathbf{r}_2 \otimes (\mathbf{p}_g - \mathbf{x}_i)$$

Neighbourhood

Social component

$$\mathbf{v}_i = \mathbf{v}_i + \boxed{\varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i)} + \boxed{\varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i)}$$

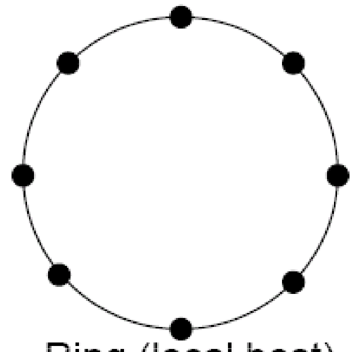
- Position $\mathbf{p}_g$ in the "social" part is the best position found by particles in the neighborhood of the ith particle.
- How is this neighbourhood defined?

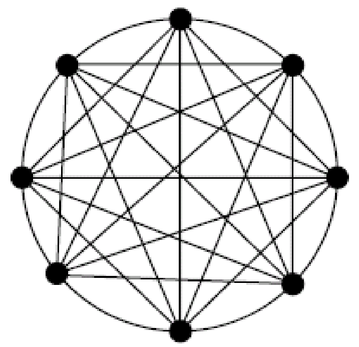
Geographical Neighbourhood

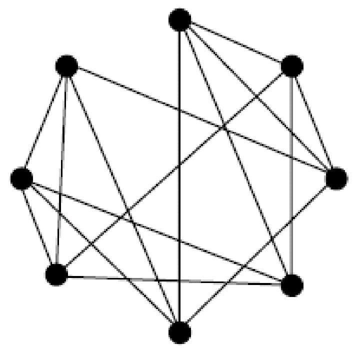
- Based on Euclidean proximity in the search space
- Close to the real-world paradigm but computationally expensive

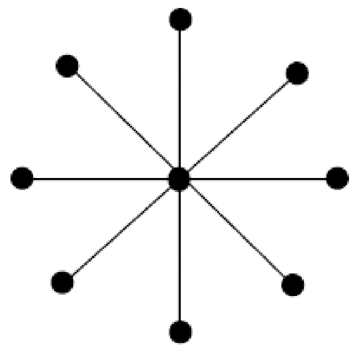
Communication Topologies

- A PSO implementation that chooses $\mathbf{p}_g$ from within a restricted local neighborhood is referred to as **lbest PSO**
- Choosing $\mathbf{p}_g$ without any restriction (hence from the entire swarm) results in a **gbest PSO**.


Ring (local best)


Global best


Random graph


Star

Inertia Weight

- In order to better control search Shi and Eberhart proposed

$$\mathbf{v}_i = \omega\, \mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i)$$

Inertia Weight

- If ω, φ_1 and φ_2 are set correctly, this update rule allows for convergence without the use of $\mathbf{v}_{\mathbf{max}}$
  - $\omega \geq 1$: velocities increase over time, swarm diverges
  - $0 < \omega < 1$: particles decelerate, convergence depends on ϕ_1 and ϕ_2
  - $\omega < 0$: velocities decrease over time and the swarm convergences
- Studies show that setting $\omega = 0.7298$ and $\varphi_1 = \varphi_2 = 1.49618$ provides good convergence behavior
- If we interpret $\varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i)$ as external force f_i then the change in the particle's velocity

$$\Delta \mathbf{v}_i = f_i - \boxed{(1 - \omega)}\, \mathbf{v}_i$$

Friction coefficient

Constriction PSO

- Clerc and Kennedy (2000) suggested a general PSO, where a constriction coefficient is applied to the velocity formula.

$$\mathbf{v}_i = \chi\,[\mathbf{v}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i)]$$

where $\chi = \dfrac{2k}{|2-\varphi-\sqrt{\varphi^2-4\varphi}\,|}$ and $\varphi = \varphi_1 + \varphi_2$ with $\varphi_1 = c_1 r_1$, $\varphi_2 = c_2 r_2$

- If $\varphi \geq$ **4** and **k** is in [0,1], then the swarm is guaranteed to converge. **k** controls the balance between exploration and exploitation.
- Typically, **k** is set to 1, and c_1=c_2=2.05; and the constriction coefficient χ is 0.7298
- Also limit V_{max} to X_{max}, the dynamic range of each variable on each dimension. This gives the **Canonical PSO**

Fully Informed Particle Swarm

- In the standard version of PSO, the effective sources of influence are in fact only two: self and best neighbor.
- Information from the remaining neighbors is unused.
- In Mendes’ fully informed particle swarm (FIPS), the particle is affected by all its neighbors

PSO Variants

- Tribes – An adaptive PSO version where swarm size is determined by strategies for generating new particles as well as removing poorly performing ones.
- ARPSO – The Attractive-Repuslive PSO uses a diversity measure
- Dissipative PSO – increases randomness;
- PSO with self-organized criticality – aims to improve diversity;
- Self-organizing Hierachicl PSO;
- FDR-PSO – The Fitness-distance ratio PSO encourages interactions between particles that are both fit and close to each other;
- PSO with mutation
- DEPSO – aims to combine DE with PSO;
- CLPSO – incorporate learning from more previous best particles.

PSO Demo

- PSO Demonstration - Christian Borgelt
- http://www.borgelt.net/psopt.html

Binary PSO

- Each solution in the population is a binary string.
  - Each binary string is of dimension **n** which is evaluated to give parameter values.
- Each binary string represents a particle
- Strings are updated bit-by-bit

BPSO

- In regular (real valued) PSO, everything is in terms of a velocity.
- In BPSO, how does one define a velocity for a single bit?
  - Generally the velocity is defined in terms of a probability of the bit changing

BPSO

- In BPSO, bit-by-bit updates are done probabilistically
  - For a chosen bit (d) in a chosen string (i) it is changed to a 1 with a probability (P) that is a function of
    - its predisposition to be a 1,
    - the best value of itself to date, and
    - the best value of its neighbors.
  - 1-P is the probability of changing to a 0
  - Once P is determined, we generate a random number R, and if R < P, then the bit becomes a 1; otherwise it becomes a 0

BPSO

- The formula for an individual bit's update is:

$$P(x_{id}(t)=1)=f(x_{id}(t),v_{id}(t-1),p_{id},p_{gd})$$

- The function *P* is a probability, and thus once this value is computed for a given particle bit, we must generate a uniform random number to see whether it should be a 1 or a 0

BPSO

- $P(x_{id}(t) = 1)$ is the probability that an invididual i will choose 1 for the bit at the d^{th} site in the bitstring
- $\mathrm{x}_{\mathrm{id}}(\mathrm{t})$ is the current state of string i at bit d
- $v_{id}(t-1)$ is a measure of the string's current probability to choose a 1
- p_{id} is the best state found so far (to date) for bit d of individual i, i.e. a 1 or a 0
- p_{gd} is 1 or 0 depending on what the value of bit d is in the best neighbor to date

BPSO

- What is f() ?
- There are several measures or expressions used for f, one that is commonly used is the sigmoid function

$$\mathrm{f}(\mathrm{v}_{\mathrm{id}}(t)) = \frac{1}{1+e^{-v_{id}(t)}}$$

$$v_{id}(t) = v_{id}(t-1) + (\varphi_1)(p_{id} - x_{id}(t-1)) + (\varphi_2)(p_{gd} - x_{id}(t-1))$$

BPSO Example

- As an example, let's say that we are dealing with a population of 5 bit binary particles and a population of 4 particles

  10101

  01011

  11100

  01101

- We are updating particle 2 (01011), bit 3 (0)

BPSO Example

- We assume that the current velocity of this bit to be a 1 is 0.25.
- Assume that the best value of this particle (to date) was 00100
- And the best value of the whole population (to date) was 01111

BPSO Example

- Thus we have:

$$v_{23}(t-1) = 0.25 \qquad \mathrm{x}_{23}(t-1) = 0$$

$$\mathrm{p}_{23} = 1 \qquad \mathrm{p}_{g3} = 1$$

$$\varphi_1 = 2.5 \qquad \varphi_2 = 1.7$$

$$v_{id}(t) = v_{id}(t-1) + (\varphi_1)(p_{id} - x_{id}(t-1)) + (\varphi_2)(p_{gd} - x_{id}(t-1))$$

$$v_{23}(t) = 0.25 + (2.5)(1-0) + (1.7)(1-0) = 4.45$$

$$f(v_{23}(t)) = \frac{1}{1+e^{-4.45}} = 0.988$$

BPSO Example

- Now, with the value for f, we generate a random number, and if it is < f then bit x becomes a 1 otherwise, it becomes a 0.

BPSO

- Initializations
  - Initial population (particle) values – just randomly generate binary strings
  - Initial velocities can be generated as

$$v_{id}(0) = V_{\min} + (V_{\max} - V_{\min}) * rand()$$

where rand() is a random number chosen from a uniform distribution $[0,1)$

The End