ISD Programme 2010:

Artificial intelligence 2

Lecture 8, 2010-12-02

Particle swarm optimization (PSO)

Krister Wolff, PhD, senior researcher, Chalmers University of Technology
e-mail: krister.wolff@chalmers.se   url: www.am.chalmers.se/~wolff

Particle swarm optimization

- Contents: **MW**, pp. 117-137.

“Algorithms inspired by flocking behaviors of real birds, fish, etc.”

- A model of swarming: BOIDS
- PSO algorithms
- Applications of PSO

Krister Wolff, PhD, senior researcher, Chalmers University of Technology
e-mail: krister.wolff@chalmers.se url: www.am.chalmers.se/~wolff

Particle swarm optimization

- Simple local rules generate complex flocking behaviors

- [http://cmol.nbi.dk/models/boids/boids.html]

Particle swarm optimization

- **Swarming behavior** in animals have shown beneficial in:

  (1) Reproduction

  (2) Food gathering

  (3) Avoiding predator attacks: the "needle in a haystack"-problem.

  The ***search efficiency*** provided by *swarming* is what underlies particle swarm optimization (PSO) algorithms.

A model of swarming: BOIDS

- Craig Reynolds: **BOIDS**

  A numerical model introduced for simulation of the swarming of *bird-like objects* (=**BOIDS**)

  BOIDS = Bird-like objects

  No leader, only *local interactions* occur.

  Only a few simple, local rules for the interactions

  *Results in a **coherent swarm**!*

Krister Wolff, PhD, senior researcher, Chalmers University of Technology
e-mail: krister.wolff@chalmers.se url: www.am.chalmers.se/~wolff

A model of swarming: BOIDS

- Consider a swarm of *N* **BOIDS**:

$$\mathbf{S} = \{ p_i , i = 1, \ldots, N \}$$

where p_i is the i:th boid.

- *Visual range* defined for each boid:

$$\mathbf{V}_i = \{ p_j : \| \mathbf{x}_j - \mathbf{x}_i \| < r , \ j \neq i \}$$

"visibility sphere"

r = global constant

A model of swarming: BOIDS

- ***Positions*** and ***velocities*** update rule:

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \mathbf{a}_i \Delta t, \quad i = 1, \ldots, N$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i \Delta t, \quad i = 1, \ldots, N$$

$\boldsymbol{x}_i$ = position of boid i.

$\boldsymbol{v}_i$ = velocity of boid i.

$\boldsymbol{a}_i$ = acceleration of boid i.

Δt = timestep

Krister Wolff, PhD, senior researcher, Chalmers University of Technology
e-mail: krister.wolff@chalmers.se url: www.am.chalmers.se/~wolff

A model of swarming: BOIDS

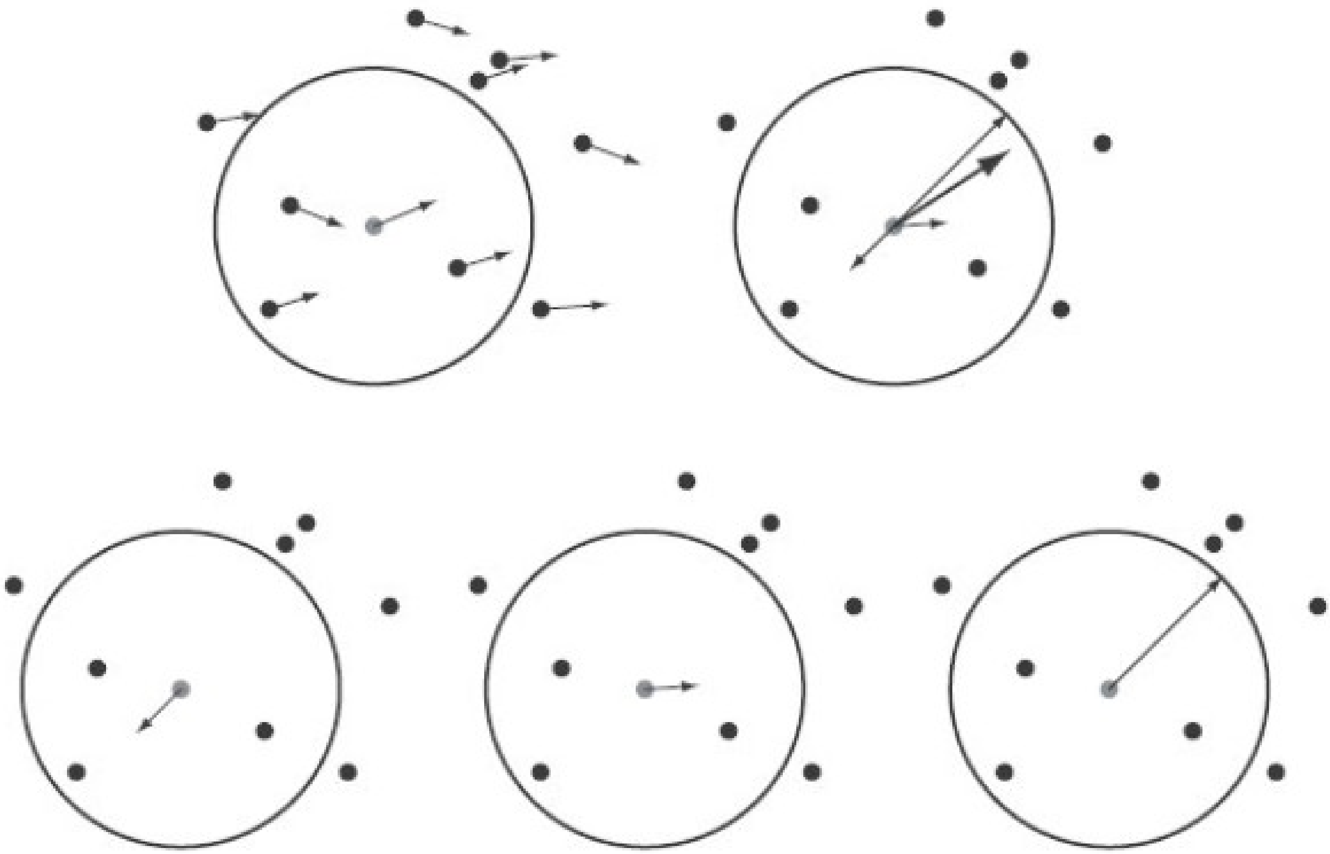
- The **movements** of each BOID is influenced by three *steers*:

(1) *Cohesion*: Stay near the center of the swarm.

(2) *Alignment*: Align velocity with the velocities of nearby swarm mates.

(3) *Separation*: Avoid collisions with nearby boids.

A model of swarming: BOIDS



- Steering vectors.

Krister Wolff, PhD, senior researcher, Chalmers University of Technology
e-mail: krister.wolff@chalmers.se url: www.am.chalmers.se/~wolff

A model of swarming: BOIDS

- **Cohesion**: *Center of density* of the boids within the visibility sphere of boid *i*:

$$\rho_i = \frac{1}{k_i} \sum_{p_j \in V_i} \mathbf{x}_j, \qquad k_i = \text{number of boids in } \mathbf{V}_i$$

- The *steering vector* representing **cohesion:**

$$\mathbf{c}_i = \frac{1}{T^2}(\rho_i - \mathbf{x}_i), \qquad T = \text{time constant}$$

If no boids are within $\mathbf{V}_i$ (=> $k_i = 0$) then set $\mathbf{C}_i = 0$.

Krister Wolff, PhD, senior researcher, Chalmers University of Technology
e-mail: krister.wolff@chalmers.se url: www.am.chalmers.se/~wolff

A model of swarming: BOIDS

- **Alignment**:

Steering vector:

$$\mathbf{l}_i = \frac{1}{T\,k_i} \sum_{p_j \in V_i} \mathbf{v}_j,$$

If no boids are within $\mathbf{V}_i$ (=> $k_i = 0$) then set $\mathbf{l}_i = 0$.

A model of swarming: BOIDS

- **Separation**:

  Steering vector:

$$\mathbf{s}_i = \frac{1}{T^2} \sum_{p_j \in V_i} \left(\mathbf{x}_i - \mathbf{x}_j \right)$$

If no boids are within $\mathbf{V}_i$ (=> $k_i = 0$) then set $\mathbf{s}_i = 0$.

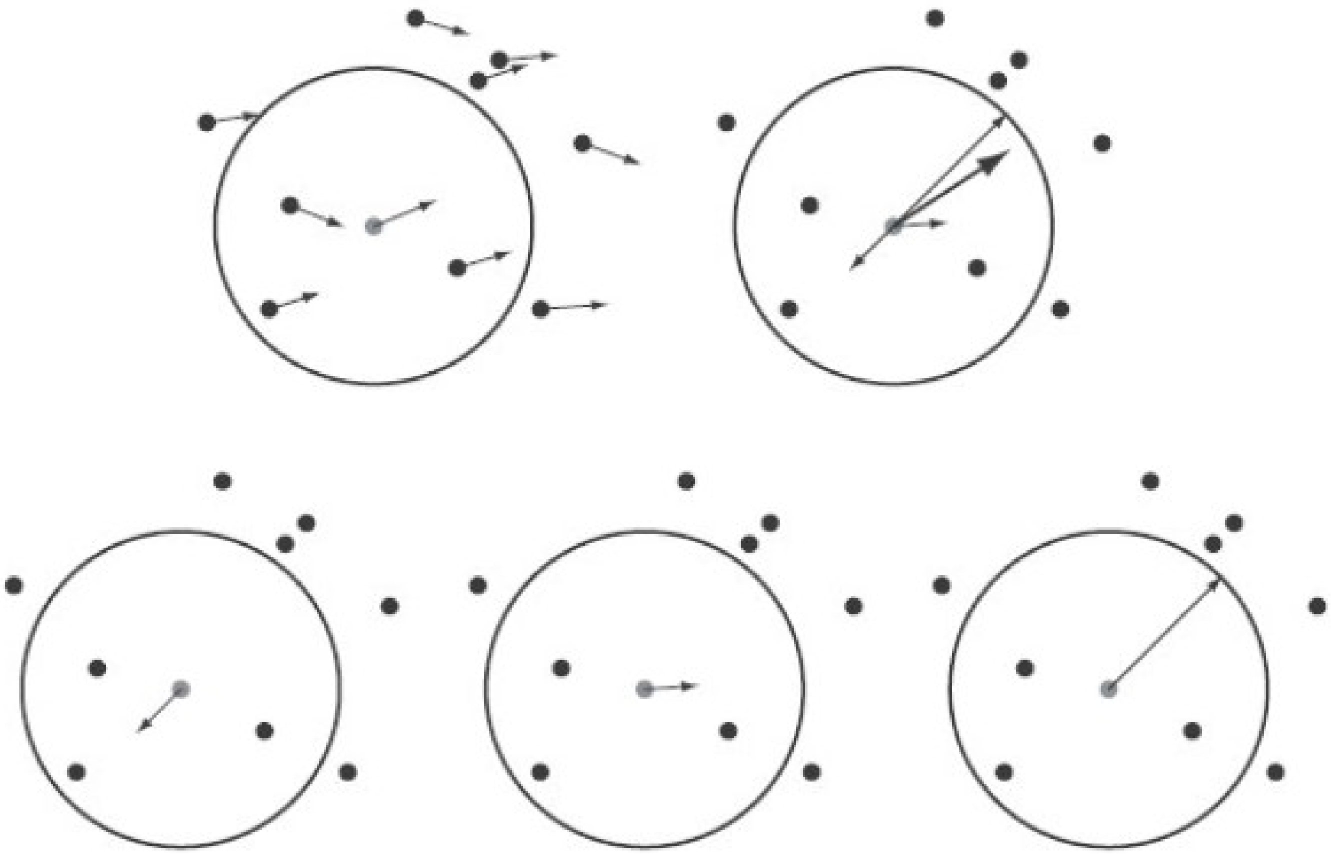
A model of swarming: BOIDS

- The **acceleration** of boid *i* is obtained by combining the three *steering vectors*:

$$\mathbf{a}_i = C_c \mathbf{c}_i + C_l \mathbf{l}_i + C_s \mathbf{s}_i$$

- C_c, C_l, and C_s are constants, in the range [0,1], defining the *relative impact* of $\mathbf{c}_i$, $\mathbf{l}_i$, and $\mathbf{s}_i$.

A model of swarming: BOIDS



- The steering vectors.

Krister Wolff, PhD, senior researcher, Chalmers University of Technology
e-mail: krister.wolff@chalmers.se url: www.am.chalmers.se/~wolff

A model of swarming: BOIDS

- **Initialization**:

  If the *initial speed* of the boids is *too large*, the swarm will *break apart* almost immediately!

  => (1) initial speed $\boldsymbol{v}_i$ should be set ~0, for all i.

  (1) limit speed to v_{max}

  (2) limit acceleration to a_{max}

  (3) Each boid should be placed within the visibility sphere of *at least* one other boid.

A model of swarming: BOIDS

- The BOIDS model leads to very realistic swarm behavior.
- Have been used (with small modifications) in movies (e.g. in Jurassic Park, to simulate herding dinosaurs).
- Go to:

  (1) http://www.red3d.com/cwr/boids/

  (2) http://www.youtube.com/watch?v=rN8DzlgMt3M

  (3) http://www.cs.ioc.ee/~ando/boids.php

PSO algorithms

- Based on the properties of ***swarms***.

  => *search efficiency*.

  Particle = Candidate solution.

  - Associated with a ***position*** and a ***velocity*** in the search space.

  - A *change* in velocity depends on the performance of the particle itself *and* that of other particles.

A basic PSO --->

Krister Wolff, PhD, senior researcher, Chalmers University of Technology
e-mail: krister.wolff@chalmers.se url: www.am.chalmers.se/~wolff

A basic PSO algorithm

- (1) **Initialization** of the position $\boldsymbol{x}_i$ and the velocity $\boldsymbol{v}_i$ of each particle p_i, i=1,...,N.

  N is in [20, 40], commonly.

  $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$ are initialized randomly:

$$x_{ij} = x_{\min} + r\left(x_{\max} - x_{\min}\right), \begin{cases} i = 1, \ldots, N \\ j = 1, \ldots, n \end{cases}$$

  r = random number in [0,1] with uniform distribution

  N = size of the swarm.

  n = number of variables of the problem.

A basic PSO algorithm

$$v_{ij} = \frac{\alpha}{\Delta t}\left(\frac{-x_{\max} - x_{\min}}{2} + r\left(x_{\max} - x_{\min}\right)\right), \begin{cases} i = 1, \ldots, N \\ j = 1, \ldots, n \end{cases}$$

r = random number in [0,1] (uniform distribution)

α = constant in [0,1].

Δt = timestep (=1, normally).

$x_{\min} = -x_{\max}$ is a common special case.

$$\Rightarrow v_{ij} = \alpha \frac{x_{\min} + r\left(x_{\max} - x_{\min}\right)}{\Delta t}, \begin{cases} i = 1, \ldots, N \\ j = 1, \ldots, n \end{cases}$$

A basic PSO algorithm

- (2) **Evaluate** performance of each particle:

  => compute objective function $f(\boldsymbol{x}_i)$, $i = 1, \ldots, N$

- (3) **Update** the *best position of each particle*, and the *global best position* (minimization).

$$\Rightarrow \forall\, p_i,\ i = 1, \ldots, N \text{ do:}$$

(3.1) $$\text{if } f(\boldsymbol{x}_i) < f(\boldsymbol{x}_i^{\text{pb}}) : \boldsymbol{x}_i^{\text{pb}} \leftarrow \boldsymbol{x}_i$$

(3.2) $$\text{if } f(\boldsymbol{x}_i) < f(\boldsymbol{x}_i^{\text{sb}}) : \boldsymbol{x}_i^{\text{sb}} \leftarrow \boldsymbol{x}_i$$

$\boldsymbol{x}_i^{\text{pb}}$ = best position so far of particle *i*.

$\boldsymbol{x}_i^{\text{sb}}$ = best position so far of *any* particle in the swarm.

A basic PSO algorithm

- (4) **Update particle velocities and positions.**

  (4.1) Velocities:

$$v_{ij} \leftarrow v_{ij} + \underbrace{c_1 q \frac{x_{ij}^{\mathrm{pb}} - x_{ij}}{\Delta t}}_{\textit{Cognitive component}} + \underbrace{c_2 r \frac{x_{ij}^{\mathrm{sb}} - x_{ij}}{\Delta t}}_{\textit{Social component}}, \quad \begin{cases} i = 1, \ldots, N \\ j = 1, \ldots, n \end{cases}$$

  q and *r* are random numbers in [0,1] (uniform).

  c_1 and c_2 are *weights* for the ***social*** and ***cognitive*** parts, respectively (= 2, normally).

Krister Wolff, PhD, senior researcher, Chalmers University of Technology
e-mail: krister.wolff@chalmers.se url: www.am.chalmers.se/~wolff

A basic PSO algorithm

- **Note:**

  (a) The contribution of the ***cognitive*** component determines to what extent the particle uses *its own* (previous) performance as guide towards better results.

  (b) The contribution of the ***social*** component determines to what extent the particle uses (previous) performance of the *other swarm members* as guide towards better results.

A basic PSO algorithm

- (4.2) In order to maintain *coherence*, restrict the velocities:

$$|v_{ij}| < v_{\max}$$

(4.3) Update position of particle p_i:

$$x_{ij} \leftarrow x_{ij} + v_{ij} \Delta t, \begin{cases} i = 1, \ldots, N \\ j = 1, \ldots, n \end{cases}$$

One iteration is now completed.

- (5) **Return** to (2) unless the termination criterion has been reached.

Algorithm 5.1 basic PSO

1. Initialize positions and velocities of the particles p_i:

   1.1 $x_{ij} = x_{\min} + r(x_{\max} - x_{\min}),\ i = 1, \ldots, N,\ j = 1, \ldots, n$

   1.2 $v_{ij} = \frac{\alpha}{\Delta t}\left(-\frac{x_{\max} - x_{\min}}{2} + r(x_{\max} - x_{\min})\right),\ i = 1, \ldots, N,\ j = 1, \ldots, n$

2. Evaluate each particle in the swarm, i.e. compute $f(\mathbf{x}_i)$, $i = 1, \ldots, N$.
3. Update the best position of each particle, and the global best position. Thus, for all particles p_i, $i = 1, \ldots, N$:

   3.1 if $f(\mathbf{x}_i) < f(\mathbf{x}_i^{\text{pb}})$ then $\mathbf{x}_i^{\text{pb}} \leftarrow \mathbf{x}_i$.

   3.2 if $f(\mathbf{x}_i) < f(\mathbf{x}^{\text{sb}})$ then $\mathbf{x}^{\text{sb}} \leftarrow \mathbf{x}_i$.

4. Update particle velocities and positions:

   4.1 $v_{ij} \leftarrow v_{ij} + c_1 q\left(\frac{x_{ij}^{\text{pb}} - x_{ij}}{\Delta t}\right) + c_2 r\left(\frac{x_j^{\text{sb}} - x_{ij}}{\Delta t}\right),\ i = 1, \ldots, N,\ j = 1, \ldots, n$

   4.2 Restrict velocities, such that $|v_{ij}| < v_{\max}$.

   4.3 $x_{ij} \leftarrow x_{ij} + v_{ij}\Delta t,\ i = 1, \ldots, N,\ j = 1, \ldots, n$.

5. Return to step 2, unless the termination criterion has been reached.

Properties of PSO

- Note that there exists *many variants* on the theme of PSO, just as in the case of EAs and ACOs. Some variants will now be described:

Best-in-current-swarm versus ***best-ever***.

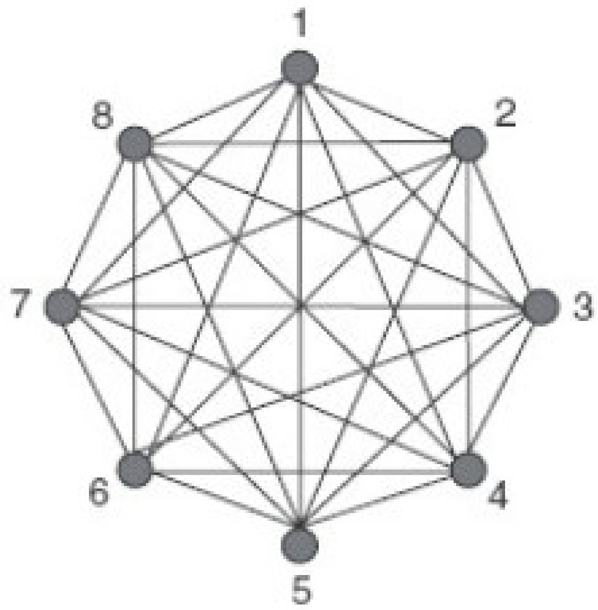
$\boldsymbol{x}^{\mathrm{sb,e}}$ = "***best-ever***"

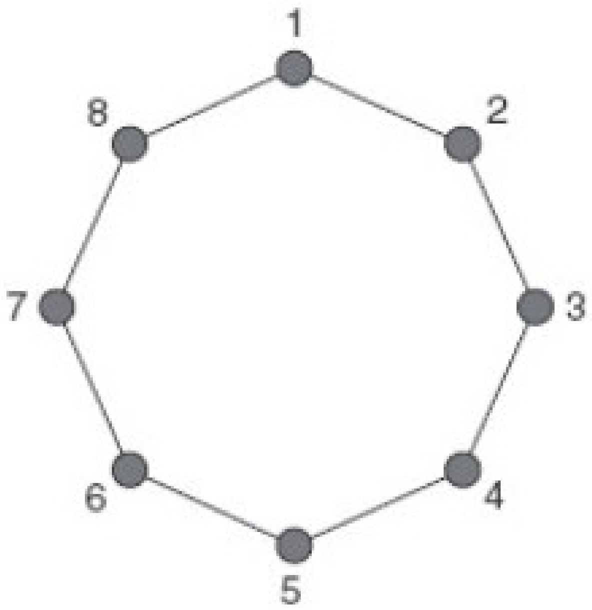
$\boldsymbol{x}^{\mathrm{sb,c}}$ = "***best-current***"

Neighborhood topologies

Consider the best particle in the *neighborhood* of particle p_i: $\boldsymbol{x}_i^{\mathrm{sb,n}}$

Properties of PSO





Fully connected. Restricted connectivity.

Other connectivities can be considered as well.

Properties of PSO

Neighborhood topologies.

Note:

(a) The topologies are constructed in *another,* abstract space than the visibility spheres.

(b) Neighborhood structures remain *constant* throughout the optimization, whereas the position $\boldsymbol{x}_i$ in search space *do not*!

The concept of *neighborhoods* where introduced for the purpose of *prevention of premature convergence* – but it *may slow down* the speed of convergence!

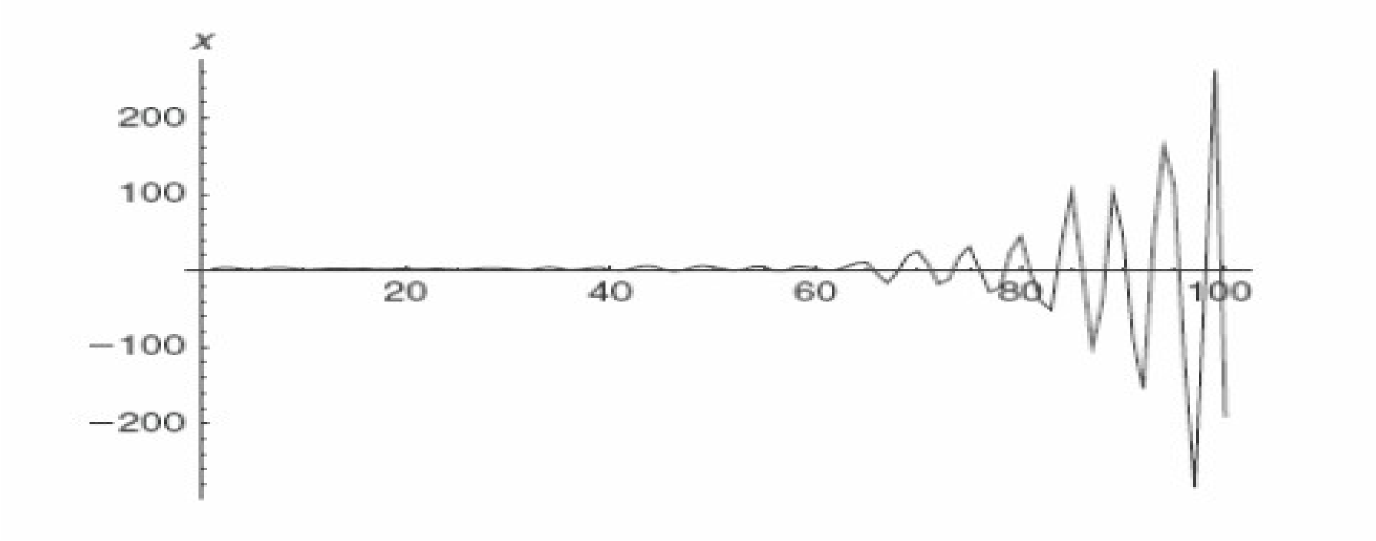
Krister Wolff, PhD, senior researcher, Chalmers University of Technology
e-mail: krister.wolff@chalmers.se url: www.am.chalmers.se/~wolff

Properties of PSO

- ***Maintaining coherence***.

  After <u>*removing*</u> the stochastic components q and r, the particle trajectories will remain bounded *only if* $c_1+c_2<4$ (Appendix B.4.1).

  <u>*Under the influence*</u> of q and r, the particle trajectories will diverge eventually, even if $c_1+c_2<4$!



Properties of PSO

- ***Maintaining coherence***.

  Thus, in order to "control" particle trajectories we introduce a *limit on particle velocities.* Restrict velocity of particle p_i as:

$$|v_{ij}| < v_{\max} = \frac{(x_{\max} - x_{\min})}{\Delta t}, \; j = 1, \ldots, n$$

if $v_{ij} > v_{max}$ set $v_{ij} = v_{max}$

if $v_{ij} < v_{max}$ set $v_{ij} = -v_{max}$

Properties of PSO

- ***Maintaining coherence***.

  Alternatively, use *constriction coefficients*:

$$v_{ij} \leftarrow X\left(v_{ij} + c_1 q \frac{x_{ij}^{\mathrm{pb}} - x_{ij}}{\Delta t} + c_2 r \frac{x_{ij}^{\mathrm{sb}} - x_{ij}}{\Delta t}\right), \begin{cases} i = 1, \ldots, N \\ j = 1, \ldots, n \end{cases}$$

  Then, trajectories do not diverge if:

$$X = \frac{2\kappa}{\left|2 - \xi - \sqrt{\xi^2 - 4}\right|}, \qquad \xi \equiv c_1 + c_2 > 4, \ \kappa \in \left]0, 1\right]$$

Properties of PSO

- ***Inertia weight***.

  Determines the *relative influence* of previous velocities on the current velocity:

$$v_{ij} \leftarrow w v_{ij} + c_1 q \frac{x_{ij}^{\mathrm{pb}} - x_{ij}}{\Delta t} + c_2 r \frac{x_{ij}^{\mathrm{sb}} - x_{ij}}{\Delta t}, \begin{cases} i = 1, \ldots, N \\ j = 1, \ldots, n \end{cases}$$

  w = inertia weight.

  If $w > 1$ particle favors ***exploration***.

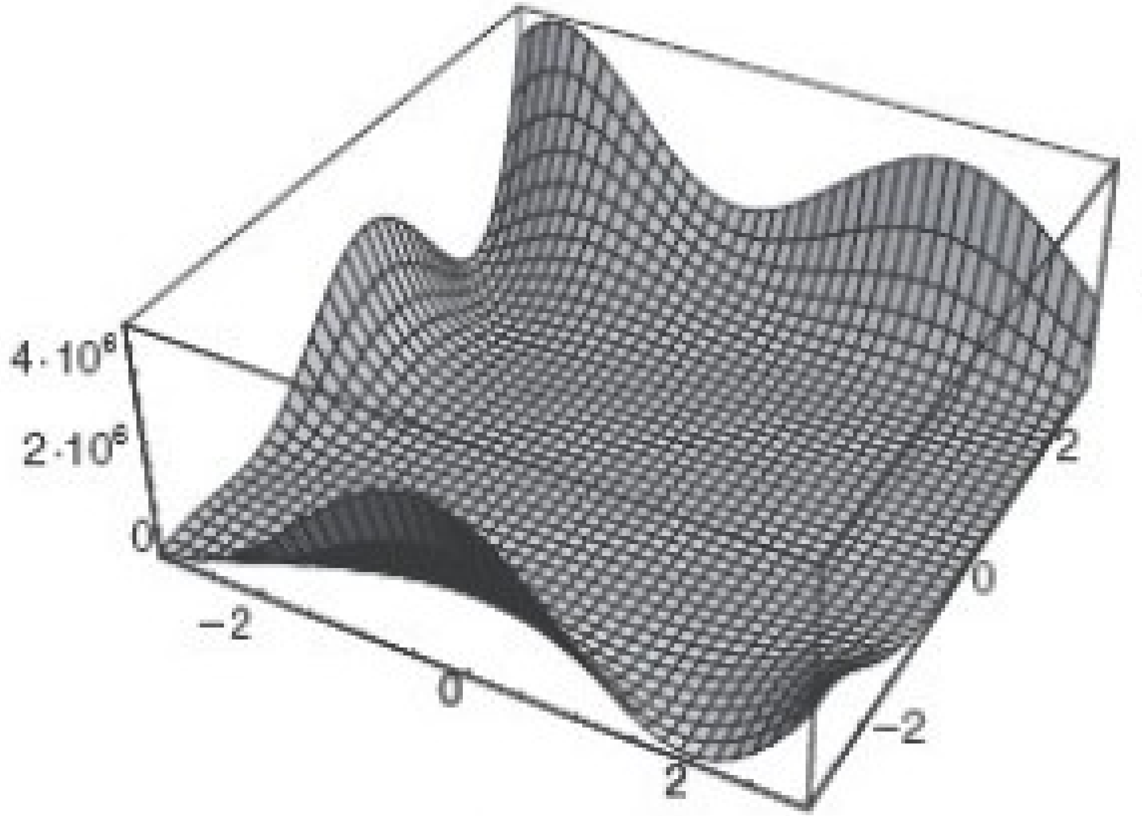
  If $w < 1$ particle favors ***exploitation***.

  Exploration is more important in the *early stages*.

  Start with w=1.4, reduce in each iteration until ~0.3

Properties of PSO

- Examples 5.2 and 5.3. The Goldstein-Price function:



Properties of PSO

- Results obtained using
  (a) Basic PSO, and
  (b) Standard PSO (including inertia weight varying from 1.4 to 0.4):

Parameters		$\Psi_1(x_1,x_2)$		$\Psi_4(x_1,x_2,x_3,x_4)$	
c_1	c_2	Avg.	S.D.	Avg.	S.D.
2	2	3.045	0.009	21.42	2.632
1	2	3.062	0.013	26.69	3.095
2	1	3.055	0.011	24.40	2.942
1	1	3.071	0.015	25.39	3.042

Parameters		$\Psi_1(x_1,x_2)$		$\Psi_4(x_1,x_2,x_3,x_4)$	
c_1	c_2	Avg.	S.D.	Avg.	S.D.
2	2	3.000	0.000	1.036	0.304
1	2	3.000	0.000	0.494	0.233
2	1	3.000	0.000	0.231	0.109
1	1	3.000	0.000	1.045	0.307

Properties of PSO

- ***Craziness operator***.

  With some (small) probability p_{cr} set:

$$v_{ij} = -v_{\max} + 2 r v_{\max}, \begin{cases} i = 1, \ldots, N \\ j = 1, \ldots, n \end{cases}$$

  r = (uniform) random number in [0, 1]

  Equivalent to *mutations*, in connection with EAs.

  Biologically motivated, inspired by the behavior of flocks of birds.

Discrete versions of PSO

- Generally, it is assumed that the variables x_j take ***real*** values.

  With small modifications PSO can be used with ***integer programming*** problems (i.e. where the variables take *integer* values only).

- (1) ***Variable truncation PSO*** is very straightforward:

  The internal workings of the PSO algorithm are *identical* to the standard (continuous) PSO, but each component of the position vector is ***truncated to the nearest integer value***.

  Truncation occurs both at *initialization*, as well as when *updating the new positions*.

Discrete versions of PSO

(2) **Binary PSO**.

Used when *binary representation* is needed, much as in the standard GA case, or *yes/no decision making*.

Same as standard PSO, but:

(a) Particle position is ***restricted*** to the set {0, 1}.

(b) The velocity v_{ij} is interpreted as a *probability* for setting the particle position to either 0 or 1, by means of an ***activation function***:

$$\sigma(v_{ij}) = \frac{1}{1 + e^{-v_{ij}}}$$

Discrete versions of PSO

(2) **Binary PSO**. *$\sigma(v_{ij})$* is interpreted as the *probability* of setting x_{ij} equal to **1**.

Thus, x_{ij} is set to **0** with probability 1-*$\sigma(v_{ij})$* .

Specifically,

$$x_{ij} = \begin{cases} 0 \text{ if } r > \sigma(v_{ij}) \\ 1 \text{ otherwise} \end{cases}$$

r = random number in [0,1].

Truncation of velocities is needed in order to avoid *too high* probability of always setting x_{ij} to 1:

$$|v_{ij}| < v_{\max} \approx 4$$

Applications of PSO

- Optimization of ANNs frequent application of PSO.

  PSO algorithms avoid the problems normally associated with GAs in connection with ANNs:
  (1) Using GA, crossover operator is *not* very likely to produce useful results from two different networks => avoid the *destructive effects* of crossover.

  (2) In PSO *there are no* completely random mutations (velocity vector can be interpreted as "almost a gradient").
- PSOs have shown good performance in recent studies (compared with backprop.).