UCD CASL

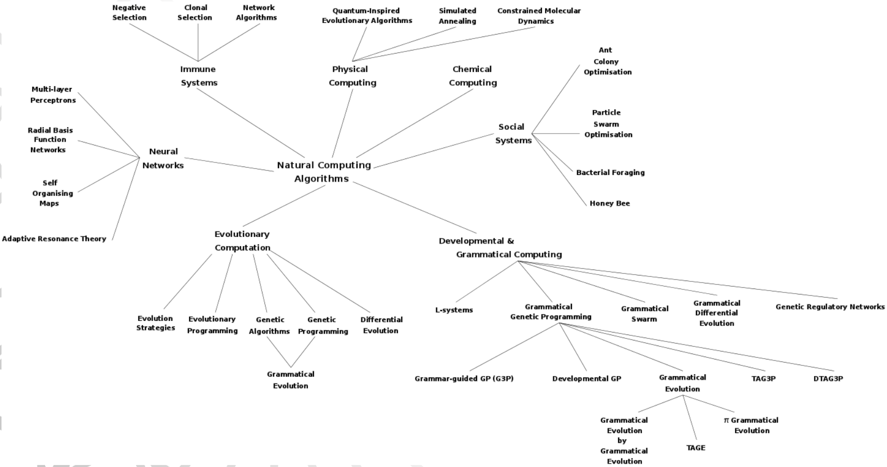
Complex & Adaptive Systems Laboratory

Dr. Michael O'Neill
Dr. Miguel Nicolau

Particle Swarm Optimisation

COMP30290 Natural Computing
COMP41190 Natural Computing and Applications

Natural Computing Algorithms

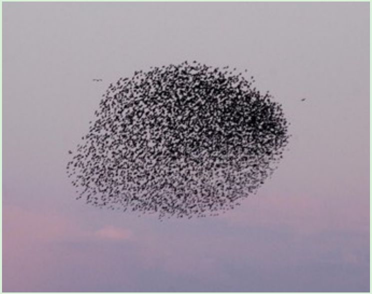


- Natural Computing Algorithms
  - Immune Systems
    - Negative Selection
    - Clonal Selection
    - Network Algorithms
  - Physical Computing
    - Quantum-Inspired Evolutionary Algorithms
    - Simulated Annealing
    - Constrained Molecular Dynamics
  - Chemical Computing
  - Social Systems
    - Ant Colony Optimisation
    - Particle Swarm Optimisation
    - Bacterial Foraging
    - Honey Bee
  - Neural Networks
    - Multi-layer Perceptrons
    - Radial Basis Function Networks
    - Self Organising Maps
    - Adaptive Resonance Theory
  - Evolutionary Computation
    - Evolution Strategies
    - Evolutionary Programming
    - Genetic Algorithms
    - Genetic Programming
      - Grammatical Evolution
    - Differential Evolution
  - Developmental & Grammatical Computing
    - L-systems
    - Grammatical Genetic Programming
      - Grammar-guided GP (G3P)
      - Developmental GP
      - Grammatical Evolution
        - Grammatical Evolution by Grammatical Evolution
        - TAGE
        - π Grammatical Evolution
      - TAG3P
      - DTAG3P
    - Grammatical Swarm
    - Grammatical Differential Evolution
    - Genetic Regulatory Networks

Social Algorithms

Inspiration

- School of fish / Flock of birds behaviour;
- No leader: **Local Interactions**;
  - match velocities of neighbours;
  - avoid collisions with neighbours;
  - avoid getting too far from neighbours.
- Simple rules simulate flocking:
  - Movie animations, graphics.

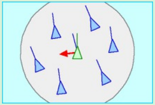


Boids

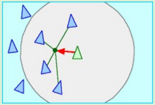
Flocking behaviour

- Craig Reynolds (1986);
- Artificial Life simulator of flocks, herds, schools;
- Based on three rules:

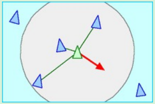
Alignment: steer towards the average heading of local flockmates.



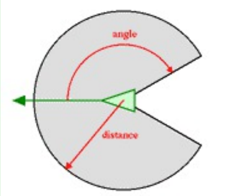
Cohesion: steer to move toward the average position of local flockmates.

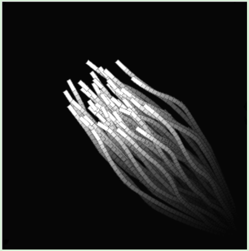


Separation: steer to avoid crowding local flockmates.



Boids neighbourhood





Boids

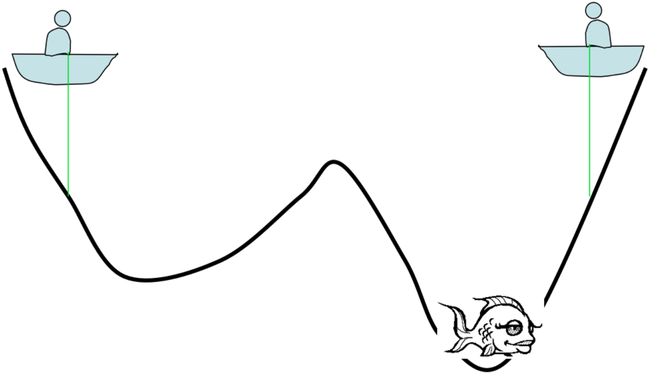
[Boids Demo]

PSO

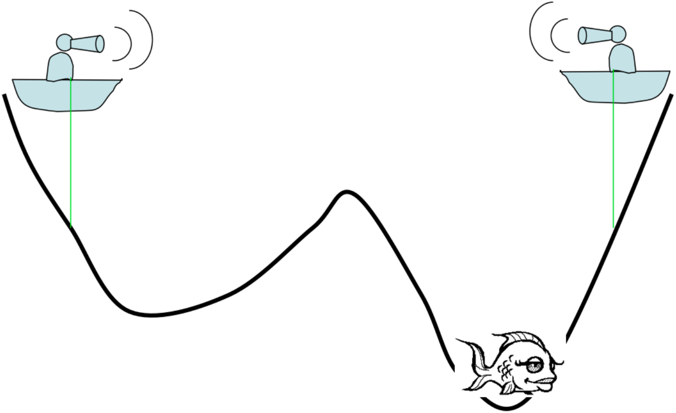
Particle Swarm

- Kennedy & Eberhart (1995);
- Particle Swarm Optimisation / Particle Swarm / Particle Swarm Algorithm;
- Intelligence through Social Interaction;
- Optimisation!

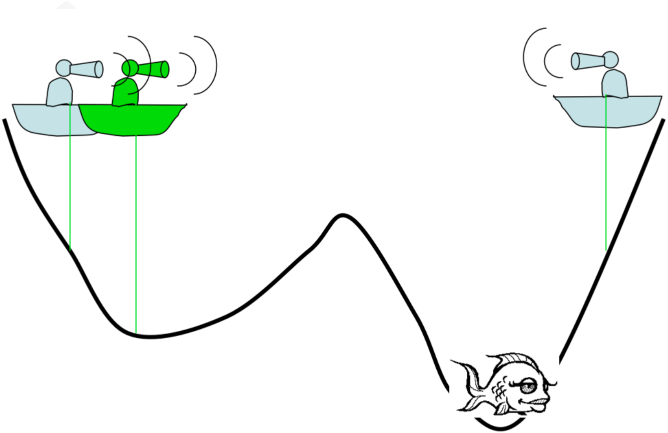
PSO - Communication & Cooperation



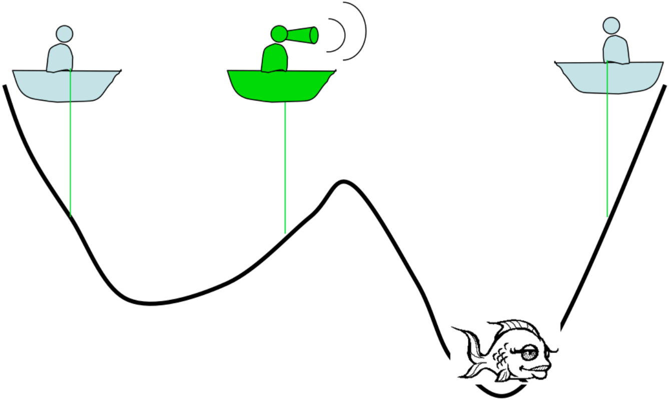
PSO - Communication & Cooperation



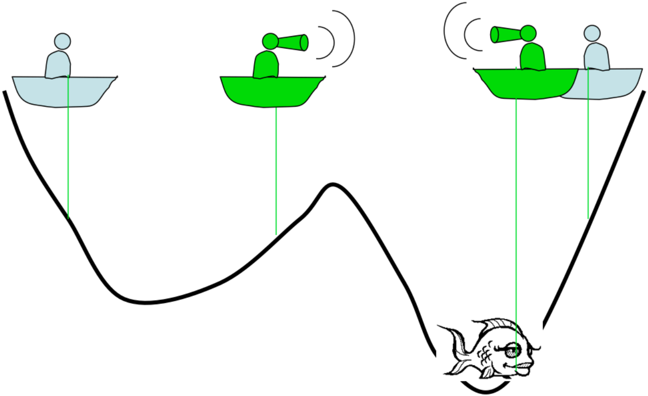
PSO - Communication & Cooperation



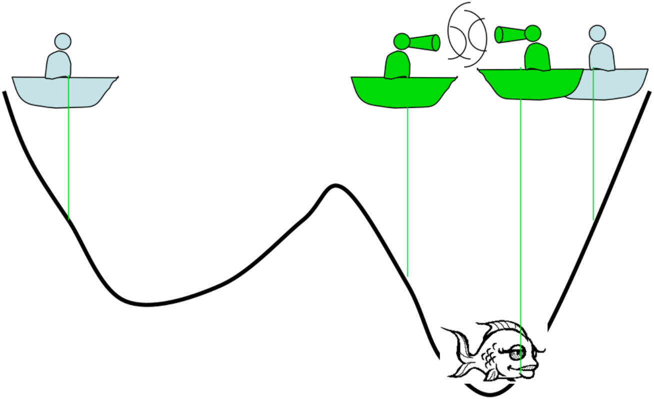
PSO - Communication & Cooperation



PSO - Communication & Cooperation



PSO - Communication & Cooperation

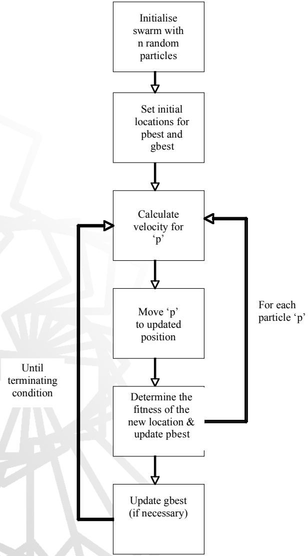


PSO

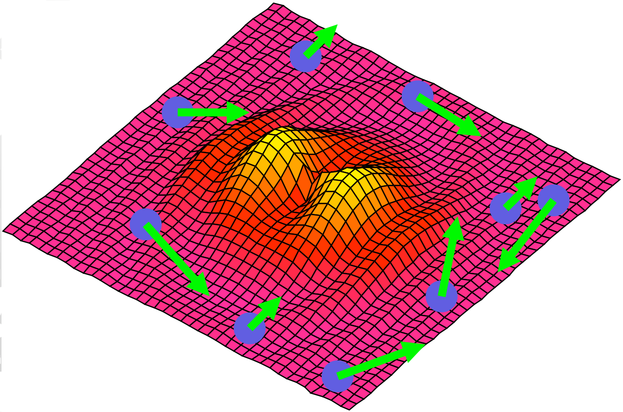
Representation

- Real-valued encoding;
- Fixed-length;
- Three vectors:
    - position, velocity, pbest (local best **ever**).
- 4^{th} global vector: gbest (global best **ever**).
- Two fitness values:
    - current;
    - pbest.

PSO - Algorithm


Initialise swarm with n random particles
Set initial locations for pbest and gbest
Calculate velocity for 'p'
Move 'p' to updated position
Determine the fitness of the new location & update pbest
Update gbest (if necessary)
For each particle 'p'
Until terminating condition

PSO - Initialisation



PSO - Update Rules

Velocity

Update velocity with pbest and gbest

$$\vec{v}_i^{t+1} = w \times \vec{v}_i^{t} + \varphi_1 \times \vec{r}_1^{t} \times (\vec{p}_i^{t} - \vec{x}_i^{t}) + \varphi_2 \times \vec{r}_2^{t} \times (\vec{g}^{t} - \vec{x}_i^{t})$$

cognitive term + social term

- w = inertia weight;
- φ_1, φ_2 = acceleration coefficients (learning factors);
- $\vec{r}_1^{t}, \vec{r}_2^{t}$ = random vectors (uniform in interval [0..1));

Position

Update positions with velocity vectors

$$\vec{x}_i^{t+1} = \vec{x}_i^{t} + \vec{v}_i^{t+1}$$

PSO - pbest and gbest

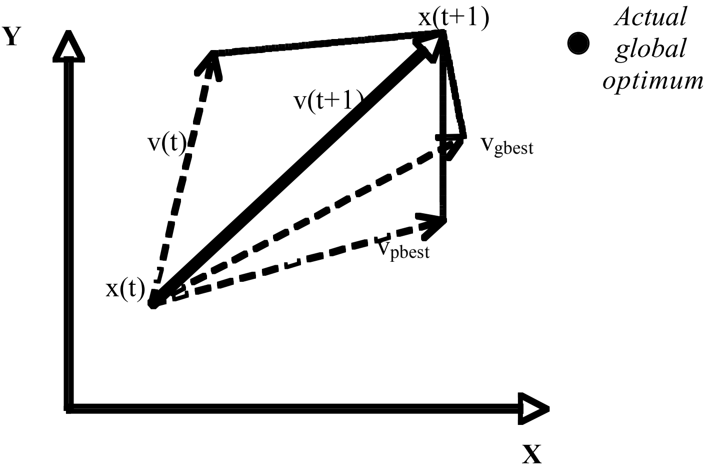
pbest

$$\vec{p}_i^{t+1} = \begin{cases} \vec{p}_i^t, & \text{if } f(\vec{x}_i^{t+1}) \leq f(\vec{p}_i^t) \\ \vec{x}_i^{t+1}, & \text{if } f(\vec{x}_i^{t+1}) > f(\vec{p}_i^t) \end{cases}$$

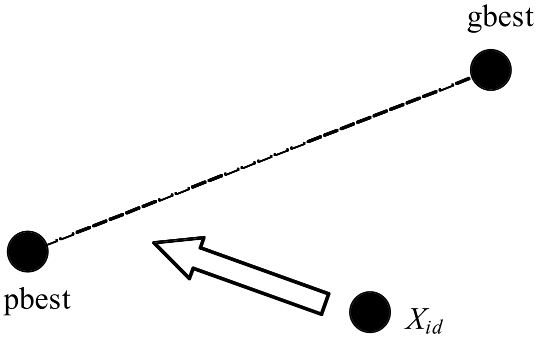
gbest

$$\vec{g}^{t+1} = \begin{cases} \vec{q} & \text{if } \vec{q} = max(f(\vec{p}_0^{t+1}) \ldots f(\vec{p}_n^{t+1})) > f(\vec{g}^t) \\ \vec{g}^t & \text{otherwise} \end{cases}$$

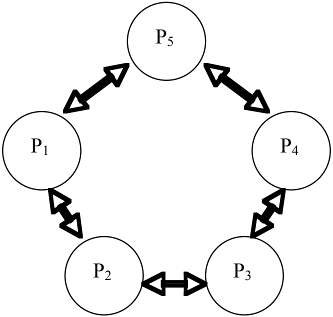
PSO - Essence

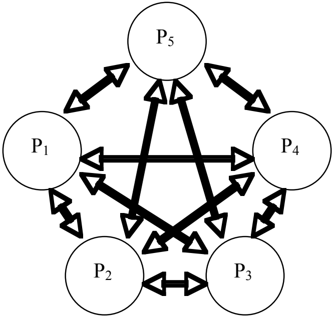


PSO - Oscillate



PSO - Local Neighbourhood





PSO - Parametrisation

Parameters

- Population size;
- Vector length;
- w, φ_1, φ_2 (can all be set to 1!);
- Range of values (open issue).

PSO

Features

- Population of particles;
- Fixed-length;
- Iteration ≡ Generation;
- No Selection;
- All particles updated;
- No mutation;
- No crossover.

PSO

[Mason Demo]

Project Ideas

PSO

- Explore Alternative Neighbourhoods;
- Variable-length PSO;
- Examine effect of bounding dimension values;
- Examine importance of φ_1 and φ_2;
- PSO+Xover;
- PSO+Selection;
- Interesting Applications of vanilla-PSO;
- Visualise an N-dimensional Swarm in flight;
- Novel velocity update strategies (evolve them!);
- Scalability of PSO as N increases.

Project Proposal

Checklist

- Run proposal idea past Mike and/or Miguel;
- Use template on website (max 2 pages);
- Submit printout to the School of CSI office;
- Deadline 3pm next Thursday 10th October;