Natural Computing

Lecture 13: Particle swarm optimisation

School of Informatics

Michael Herrmann
mherrman@inf.ed.ac.uk
phone: 0131 6 517177
Informatics Forum 1.42

INFR09038
5/11/2010
Swarm intelligence

• Collective intelligence: A super-organism emerges from the interaction of individuals

• The super-organism has abilities that are not present in the individuals (‘is more intelligent’)

• “The whole is more than the sum of its parts”

• Mechanisms: Cooperation and competition self-organisation, … and communication

• Examples: Social animals (incl. ants), smart mobs, immune system, neural networks, internet, swarm robotics

Swarm intelligence: Application areas

- Biological and social modeling
- Movie effects
- Dynamic optimization
  - routing optimization
  - structure optimization
  - data mining, data clustering
- Organic computing
- Swarm robotics
Swarms in robotics and biology

• **Robotics/AI**
  – Main interest in pattern synthesis
    • Self-organization
    • Self-reproduction
    • Self-healing
    • Self-configuration
  – Construction

• **Biology/Sociology**
  – Main interest in pattern analysis
    • Recognizing best pattern
    • Optimizing path
    • Minimal conditions
    • not “what”, but “why”
  – Modeling

*Dumb parts, properly connected into a swarm, yield smart results.*

Kevin Kelly
Complex behaviour from simple rules

Rule 1: *Separation*
Avoid Collision with neighboring agents

Rule 2: *Alignment*
Match the velocity of neighboring agents

Rule 3: *Cohesion*
Stay near neighboring agents
Towards a computational principle

- **Evaluate** your present position
- **Compare** it to your previous best and neighborhood best
- **Imitate** self and others

Hypothesis: There are two major sources of cognition, namely, own experience and communication from others.

Particle Swarm Optimization (PSO)

- Methods for finding an optimal solution to an objective function
- Direct search, i.e. gradient free
- Simple and quasi-identical units
- Asynchronous; decentralized control
- ‘Intermediate’ number of units: $\sim 10^{1-10^{23}}$
- **Redundancy** leads to reliability and adaptation
- PSO is one of the computational algorithms in the field of swarm intelligence (another one is ACO)

PSO algorithm: Initialization

- Fitness function
  \[ f : \mathbb{R}^m \rightarrow \mathbb{R} \]
- Number of particles
  \[ n = 20, \ldots, 200 \]
- Particle positions
  \[ x_i \in \mathbb{R}^m, \quad i = 1, \ldots, n \]
- Particle velocities
  \[ v_i \in \mathbb{R}^m, \quad i = 1, \ldots, n \]
- Current best of each particle
  ("simple nostalgia")
  \[ \hat{x}_i \]
- Global best
  ("group norm")
  \[ \hat{g} \]
- Initialize constants
  \[ \omega, \alpha_{1/2} \]
The canonical PSO algorithm

For each particle

For all members of the swarm, i.e. $1 \leq i \leq n$

- create random vectors $r_1, r_2$ with components drawn from $U[0,1]$

- update velocities
  $$v_i \leftarrow \omega v_i + \alpha_1 r_1 \circ (\hat{x}_i - x_i) + \alpha_2 r_2 \circ (\hat{g} - x_i)$$

- update positions
  $$x_i \leftarrow x_i + v_i$$

- update local bests
  $$\hat{x}_i \leftarrow x_i \quad \text{if} \quad f(x_i) < f(\hat{x}_i)$$

- update global best
  $$\hat{g} \leftarrow x_i \quad \text{if} \quad f(x_i) < f(\hat{g})$$
Comparison of GA and PSO

• Generally similar:
  1. Random generation of an initial population
  2. Calculation of a fitness value for each individual.
  3. Reproduction of the population based on fitness values.
  4. If requirements are met, then stop. Otherwise go back to 2.

• Modification of individuals
  • In GA: by genetic operators
  • In PSO: Particles update themselves with the internal velocity. They also have memory.

• Sharing of information
  • Mutual in GA. Whole population moves as a group towards optimal area.
  • One-way in PSO: Source of information is only gBest (or lBest). All particles tend to converge to the best solution quickly.

• Representation
  • GA: discrete
  • PS: continuous

www.swarmintelligence.org/tutorials.php
PSO as MBS

- As in GA the “model” is actually a population (which can be represented by a probabilistic model)
- Generate new samples from the individual particles of the previous iteration by random modifications
- Use memory of global, neighborhood or personal best for learning
Initialization

# Initialize the particle positions and their velocities
X = lower_limit + (upper_limit - lower_limit) * 
    rand(n_particles, m_dimensions)
assert X.shape == (n_particles, m_dimensions)
V = zeros(X.shape)

# Initialize the global and local fitness to the worst possible
fitness_gbest = inf
fitness_lbest = fitness_gbest * ones(n_particles)
w=0.1  # omega range 0.01 ... 0.7
a1=a2=2  # alpha range 0 ... 4, both equal
n=25  # range 20 ... 200
max velocity  # no larger than: range of x per step
    or 10-20% of this range

Main loop (next page)
for k = 1 .. T_iterations:    # loop until convergence
    fitness_X = evaluate_fitness(X)  # evaluate fitness of each particle
    for I = 1 .. n_particles:        # update local bests
        if fitness_X[I] < fitness_lbest[I]:
            fitness_lbest[I] = fitness_X[I]
            for J = 1 .. m_dimensions:
                X_lbest[I][J] = X[I][J]; end J; end I;
        end I;
    end I;
    min_fitness_index = argmin(fitness_X)    # update global best
    min_fitness = fitness_X[min_fitness_index]
    if min_fitness < fitness_gbest:
        fitness_gbest = min_fitness;
        X_gbest = X[min_fitness_index, :]
    end I;
end k;

for I = 1 .. n_particles:        # update velocities and positions
    for J = 0 .. m_dimensions:
        R1 = uniform_random_number()
        R2 = uniform_random_number()
        V[I][J] = (w*V[I][J] +
                   a1*R1*(X_lbest[I][J] - X[I][J]) + a2*R2*(X_gbest[J] - X[I][J]))
        X[I][J] = X[I][J] + V[I][J]
    end I, end J, end k;
Illustrative example

1. Create a ‘population’ of agents (called particles) uniformly distributed over $\mathcal{X}$.
2. Evaluate each particle’s position according to the objective function.
How does it work?

- Exploratory behaviour: Search a broad region of space
- Exploitative behaviour: Locally oriented search to approach a (possibly local) optimum

Parameters to be chosen to properly balance between exploration and exploitation, i.e. to avoid premature convergence to a local optimum yet still ensure a good rate of convergence to the optimum.

Convergence

- Exploration: Swarm collapses (or rather diverges, oscillates, or is critical)
- Exploitation: Global best approaches global optimum (or rather, for a collapse of the swarm, a local optimum)

Mathematical attempts (typically oversimplified): Convergence to global optimum for a 1-particle swarm after infinite time (F. v. d. Bergh, 2001)

see PSO at en.wikipedia.org
Repulsive PSO algorithm

For each particle $1 \leq i \leq n$

- create random vectors $r_1, r_2, r_3$ with components drawn from $U[0,1]$

$$v_i \leftarrow \omega \ v_i + \alpha_1 r_1 \circ (\hat{x}_i - x_i) + \alpha_2 r_2 \circ (\hat{y} - x_i) + \alpha_3 \omega r_3 \circ z$$

- update velocities
  - $\hat{y}$ best of random neighbors, $\alpha_2 < 0$
  - $z$ random velocity

- update positions etc.

- Properties: sometimes slower, more robust and efficient

$\circ$ componentwise multiplication
Constriction factor in canonical PSO

- Introduced by Clerc (1999)
- Simplest form:

\[ v_i \leftarrow K \left[ \omega \ v_i + \alpha_1 \ r_1 \circ (\hat{x}_i - x_i) + \alpha_2 \ r_2 \circ (\hat{g} - x_i) \right] \]

\[ K = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}, \text{ where } \phi = \alpha_1 + \alpha_2 > 4 \]

\[ e.g. \ \phi = 4.1 \Rightarrow K = 0.729, \ i.e. \ \text{prefactors } \alpha \approx 1.5 \]

- May replace interia \( \omega \)
- Meant to improve convergence by an enforced decay (more about this later)
Topology: Restricted competition/coordination

- Topology determines with whom to compare and thus how solutions spread through the population.
- Traditional ones: gbest, lbest.
- Global version is faster but might converge to local optimum for some problems.
- Local version is a somewhat slower but not easy to be trapped into local optimum.
- Combination: Use global version to get rough estimate. Then use local version to refine the search.
- For some topologies analogous to islands in GA.
Innovative topologies

- Specified by:
  Mean degree, clustering, heterogeneity etc.
Comparison of GA and PSO

• Generally similar:
  1. Random generation of an initial population
  2. Calculate of a fitness value for each individual.
  3. Reproduction of the population based on fitness values.
  4. If requirements are met, then stop. Otherwise go back to 2.

• Modification of individuals
  • In GA: by genetic operators
  • In PSO: Particles update themselves with the internal velocity. They also have memory.

• Sharing of information
  • Mutual In GA. Whole population moves as a group towards optimal area.
  • One-way in PSO: Source of information is only gBest (or lBest). All particles tend to converge to the best solution quickly.

• Representation
  • GA: discrete
  • PS: continuous

www.swarmintelligence.org/tutorials.php
Literature on swarms

- Eric Bonabeau, Marco Dorigo, Guy Theraulaz: Swarm Intelligence: From Natural to Artificial Systems (Santa Fe Institute Studies on the Sciences of Complexity) OUP USA (1999)
- Eberhart Y. Shi (2001) PSO: Developments, applications ressources. IEEE.
- www.engr.iupui.edu/~eberhart/web/PSObook.html
- Tutorials: www.particlesswarm.info/
- Bibliography: icdweb.cc.purdue.edu/~hux/PSO.shtml