Particle Swarm Optimization: Part 3

Sub-Swarm PSO

Sub-swarm PSO approaches divide the main swarm of particles into subgroups.

- Each subgroup performs a different task, or exhibits a different behavior.
  - The amount of difference is variable
- The behavior of a group or task performed by a group usually changes over time in response to the groups interaction with the environment

Sub-Swarm PSO

Consider the following idea, inspired by multi-phase PSO (MPPSO). A swarm can be divided into sub-swarm where

- **Full-Attraction** based update, particles are attracted to both the neighborhood and personal best positions
  - $c_1 > 0, c_2 > 0$
- **Social-Repulsion, Cognitive-Attraction** based update, particles are repelled from the neighborhood best position but attracted to the personal best position.
  - $c_1 > 0, c_2 < 0$
- **Cognitive-Repulsion, Social-Attraction** based update, particles are attracted to the neighborhood best position but replied from personal best position.
  - $c_1 < 0, c_2 > 0$
- **Social-Repulsion, Cognitive Attraction** based update, particles are repelled from neighborhood and personal best positions.
  - $c_1 < 0, c_2 < 0$

Sub-Swarm PSO

Particles subswarm allocation can be handled in a number of was.

1. Assigned a sub-swarm for the whole run.
2. Based on new information or the phase of the search, the sub-swarm could be intelligently reallocated.
3. Particles should be exchanged between sub-swarm or simply migrate.
4. There are countless variations.

Attractive and Repulsive PSO

For example the attractive and repulsive PSO (ARPSO) developed by Riget and Vesterstrøm. Particle would alternate the whole swarm between the **Full-Attraction** phase and **Social-Repulsion, Cognitive Attraction** phase based on the swarm diversity. Specifically

$$Diverity(\Omega(t)) = \frac{1}{\|\Omega\|} \sum_{i=1}^{\|\Omega\|} \|\mathbf{x}_i(t) - \bar{\mathbf{x}}_i(t)\| \tag{1}$$

$$\bar{\mathbf{x}}_i(t) = \frac{\sum_{i=1}^{\|\Omega\|} \mathbf{x}_i(t)}{\|\Omega\|} \tag{2}$$

Then if

- In **Full-Attraction** state and $Diverity(\Omega(t)) < \rho_{min}$ switch to **(Social-Repulsion, Cognitive Attraction)**.
- In **(Social-Repulsion, Cognitive Attraction)** state and $Diverity(\Omega(t)) > \rho_{max}$ switch to **Full-Attraction**.

Sub-Swarm PSO: Attractive and Repulsive

Can we use this concept in conjunction with sub-swarms?

Life-Cycle PSO

Krink and Lvberg used the life-cycle model to change the behavior of individuals. Using the life-cycle model, an individual can be in any of three phases

- a PSO particle,
- a GA individual,
- a stochastic hill-climber. (generally the last phase)

An individual switches from one phase to the next if its fitness is not improved over a number of consecutive iterations.

Cooperative Split PSO: High Dimensional Problems

The cooperative split PSO (CPSO-S_k), first introduced by Van den Bergh and Engelbrecht.

- Split a d-dimensional problem into k subproblems
- Each of the k sub-problems is optimized by it's own PSO's

The immediate question is

- How do we evaluate fitness?

Cooperative Split PSO

Since each sub-swarm cannot evaluate a particles fitness by itself, a **Context vector** is used.

- a context vector represent the d-dimensional solution

How do we construct the context vector?

- Construct by concatenating the global best positions from the K sub-swarms.
- Particles in sub-swarm S_k are then swapped into the corresponding positions of the context vector, and the original fitness function is used to evaluate the fitness of the context vector.

Cooperative Split PSO

The approach can works well for high dimensional problems, but when can this approach fail?

Cooperative Split PSO

The approach can works well for high dimensional problems, but when can this approach fail?

- When there is an interlinking between variables, or more formally
  - The components that swam s_k optimize may be non separable from the components that swarm s_j optimizes

Cooperative Split PSO

The approach can works well for high dimensional problems, but when can this approach fail?

- When there is an interlinking between variables, or more formally
  - The components that swam s_k optimize may be non separable from the components that swarm s_j optimizes
- Are there any special implementation type considerations?

PSO and Dynamic Optimization

Traditional PSO is not well suited for dynamic optimization. Why?

PSO and Dynamic Optimization

Traditional PSO is not well suited for dynamic optimization. Why?

- Diversity collapse often occurs
  - Makes it hard to detect change
  - During collapse, the swarm aspect of PSO is compromised.

PSO and Dynamic Optimization

Traditional PSO is not well suited for dynamic optimization. Why?

- Diversity collapse often occurs
  - Makes it hard to detect change
  - During collapse, the swarm aspect of PSO is compromised.
- This implies that the most significant alteration needed for PSO is the inclusion of a diversity ol mechanism.

Predator-Prey PSO

The concept of predator-prey is a common one from nature. Silva *et al* proposed its use in PSO to promote exploration. The idea

- Have one predator in the swarm that uses

$$\mathbf{v}_p(t+1) = \mathbf{r} \otimes (\hat{\mathbf{y}}(t) - \mathbf{x}_p) \quad (3)$$

  with $\mathbf{r} \sim U(0, V_{max,p})$
  - if $V_{max,p} > 1$ the predator can jump over the prey.

Predator-Prey PSO

- The prey (rest of swarm) are affected by fear and use

$$\begin{aligned}\mathbf{v}_i(t+1) =& w\mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2 \otimes (\mathbf{n}_i(t) - \mathbf{x}_i(t)) \\ &+ c_3\mathbf{r}_3\mathbf{D}(\|\mathbf{x}_i(t) - \mathbf{v}_p(t)\|)\end{aligned} \tag{4}$$

where

$$\mathbf{D}(\|\mathbf{x}_i(t) - \mathbf{v}_p(t)\|) = \mathbf{1}\alpha e^{-\beta\|\mathbf{x}_i(t) - \mathbf{v}_p(t)\|} \tag{5}$$

- Equation (4) is only used of $U(0,1) < P_f$, where P_f is the fear level. Otherwise the standard update is used.

Multi-Start PSOs

Sometimes PSO stops improving

- One of the most common reasons is swarm collapse

In order to prevent this when swarm collapse is detected is to introduce randomness or chaos

- reinitialize some particles
  - positions
  - velocities
  - memories
- Change particle phase
- to name a few

The detection of collapse if often the critical aspect.

Charged PSO

Blackwell and Bentley developed the charged PSO. The idea is to introduce two opposing forces within the dynamics of the PSO:

- an attraction to the center of mass of the swarm and inter-particle repulsion.
- The attraction force facilitates convergence to a single solution, while the repulsion force preserves diversity.

- Some particles repel one another with the same charge
- Velocity changes to

$$\begin{aligned}\mathbf{v}_i(t+1) =& w\mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2 \otimes (\mathbf{n}_i(t) - \mathbf{x}_i(t)) \\ &+ \hat{\mathbf{a}}_i \end{aligned} \tag{6}$$

where $\hat{\mathbf{a}}_i$ is the charge-based particle acceleration, determining the magnitude of inter-particle repulsion

Charged PSO

$$\hat{\mathbf{a}}_i(t) = \sum_{l=1, i \neq l}^{n_s} \mathbf{a}_{il}(t)$$

- The repulsion force between particles i and l is

$$\mathbf{a}_{il}(t) = \begin{cases} \left(\frac{Q_i Q_l}{\|\mathbf{x}_i(t) - \mathbf{x}_l(t)\|^3}\right)(\mathbf{x}_i(t) - \mathbf{x}_l(t)) & \text{if } R_c \leq \|\mathbf{x}_i(t) - \mathbf{x}_l(t)\| \leq R_p \\ \frac{Q_i Q_l}{R_c^2} \frac{(\mathbf{x}_i(t) - \mathbf{x}_l(t))}{\|\mathbf{x}_i(t) - \mathbf{x}_l(t)\|} & \text{if } \|\mathbf{x}_i(t) - \mathbf{x}_l(t)\| < R_c \\ 0 & \text{if } \|\mathbf{x}_i(t) - \mathbf{x}_l(t)\| > R_p \end{cases}$$

  where Q_i is the charged magnitude of particle i, R_c is referred to as the core radius, and R_p is the perception limit of each particle.

- Acceleration is fixed at the core radius to prevent too severe repelling

Charged PSO (cont)

- Q_i is the particle's charged magnitude
  R_c is the core radius
  R_p is the perception limit of each particle
- If $Q_i = 0$, particles are neutral and there is no repelling
- If $Q_i \neq 0$, particles are charged, and attract or repel each other
  - Proposed algorithm actually only allows $Q_i > 0$, so repulsion
- Inter-particle repulsion occurs only when the separation between two particles is within the range $[R_c, R_p]$
- The smaller the separation, the larger the repulsion between the corresponding particles

Quantum PSO

- Based on quantum model of an atom, where orbiting electrons are replaced by a quantum cloud which is a probability distribution governing the position of the electron
- Developed as a simplified and less expensive version of the charged PSO.
  - What was expensive?
- Swarm contains
  - neutral particles following standard PSO updates
  - charged, or quantum particles, randomly placed within a multi-dimensional sphere

$$\mathbf{x}_i(t+1) = \begin{cases} \mathbf{x}_i(t) + \mathbf{v}_i(t+1) & \text{if } Q_i = 0 \\ \mathcal{H}(n_i, \phi) & \text{if } Q_i \neq 0 \end{cases}$$

    where ϕ is the cloud radius.
  - Originally proposed in the context of gbest $\implies \mathbf{n}_i = \mathbf{g}$

PSO and Binary Optimization

PSO was originally developed to solve real valued optimization problems. However it can be altered to work for discrete binary values problems. We will discuss two options.

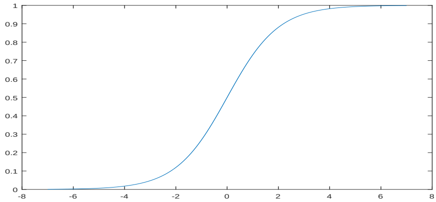
- Binary PSO (binPSO)
- Angular-modulated PSO

Binary PSO

Binary PSO (binPSO) was proposed by Kennedy and Eberhart

- Velocity remains a floating-point vector, but meaning changes
- Velocity is no longer a step size, but is used to determine a probability of selecting bit 0 or bit 1
- Position is a bit vector, i.e. $x_{ij} \in \{0, 1\}$
- How to interpret velocity as a probability?

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



Binary PSO

- Then, position update changes to

$$x_{ij}(t+1) = \begin{cases} 1 & \text{if } U(0,1) < p_{ij}(t+1) \\ 0 & \text{otherwise} \end{cases}$$

- A velocity of 0.7 implies a 70% or being set to 1

Angle Modulated PSO

A novel way of handling binary problems is to transform the problem into a simple real valued problem.

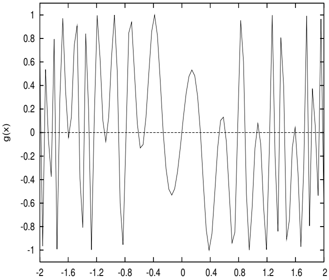
- Velocities and positions remain floating-point vectors
- Find a bitstring generating function to generate bitstring solution
- The generating function:

$$g(x) = \sin(2\pi(x - a) \times b \times \cos(2\pi(x - a) \times c)) + d$$

sampled at evenly spaced positions, x

The coefficients determine the shape of the generating function:

- a: horizontal shift of generating function
- b: maximum frequency of the sin function
- c: frequency of the cos function
- d: vertical shift of generating function



Use a standard PSO to find the best values for these coefficients

- Generate a swarm of 4-dimensional particles
- Repeat until stopping condition met.
  - Apply any PSO for one iteration
  - For each particle:
    - Substitute values for coefficients a, b, c and d into generating function
    - Produce n_x bit-values to form a bit-vector solution
    - Calculate the fitness of the bit-vector solution in the original bit-valued space

PSO and Multi-Objective Optimization

As mentioned earlier multi-objective optimization is the task of simultaneously optimizing more than one objective.

- The simplest approach is to use weighted aggregation $\hat{f}(\mathbf{x}) = \sum_{f=0}^{F} \lambda_f f_f(\mathbf{x})$
  - The common issue is still present of how to select λ_fs
  - The other issue is that the optimally of each objective could be competing against each other. Which means we are unlikely to be considering good trade off solution.
- If we wish to rather use the dominance/pareto front approach we cannot use an off-the-shelve PSO.

Vector-Evaluated PSO

Another approach is to allocate a swarm for the optimization of each objective individually, and then transfer position information from well performing particles. Such an approach is used by VEPSO (proposed by Parsopoulos and Vrahatis). Specifically,

- Assume K objectives
- K sub-swarms are used, where each optimizes one of the objectives
- Need a knowledge transfer strategy (KTS) to transfer information about best positions between sub-swarms
- Exchanged information are via selection of global guides, replacing the global best positions in the velocity updates
- Standard KTS: the ring KTS
    - Sub-swarms are arranged in a ring topology
    - Global guide of swarm S_k is from swarm $S_{(k+1) \mod K}$

Vector-Evaluated PSO

- Assume two objectives

$$\begin{aligned} S_1.v_{ij}(t+1) &= wS_1.v_{ij}(t) + c_1 r_{1j}(t)(S_1.y_{ij}(t) - S_1.x_{ij}(t)) \\ &+ c_2 r_{2j}(t)(S_2.\hat{g}_i(t) - S_1.x_{ij}(t)) \\ S_2.v_{ij}(t+1) &= wS_2.v_{ij}(t) + c_1 \hat{r}_{1j}(t)(S_2.y_{ij}(t) - S_2.x_{ij}(t)) \\ &+ c_2 \hat{r}_{2j}(t)(S_1.\hat{g}_j(t) - S.x_{2j}(t)) \end{aligned}$$

  where sub-swarm S_1 evaluates individuals on the basis of objective $f_1(\mathbf{x})$, and sub-swarm S_2 uses objective $f_2(\mathbf{x})$
- Local guide selection:
  - Local guide replaces the personal best
  - Personal best is determined by the fitness on the objective the particles swarm is optimizing.
    - A variant is: Update personal best position only if the new particle position dominates the previous personal best position.
- Alternative KTS: Random (the swarm from which you obtain your random guide is different on subsequent iterations)

Vector-Evaluated PSO

Another effective KTS approach is to utilize an **archive**, based on pareto-dominance.

- The archive should contain a set of non-dominated solutions
- the Global guide is selected from the archive.
- If a new solution is found that is not dominated by any solution in the archive it is added.
- If this results in a solution becoming dominated it is removed.

The archive could grow rapidly so there is normally a cap in size.

- If the cap is reached and a non-dominated solution is found, what do we do?
  - We evict a solution from the archive The most common way is to remove the position from the densest area of the pareto optimal front.
    - other metrics can be used.

Multi-Objective Performance (MOOP) Inverted Generational Distance

How do we measure performance of a MOOP? In comparative studies on known benchmark functions IGD is the most common measurement

- Inverted generational distance (IGD) was proposed by Sierra and Coello Coello
- You need two sets
  - POF^* contains elements of the POF generated by your algorithm.
  - POF' contains elements of the POF generated by you subsampling the **true** POF.

$$IGD = \frac{\sqrt{\sum_{i=1}^{\|POF'\|} d_i^2}}{\|POF'\|} \tag{7}$$

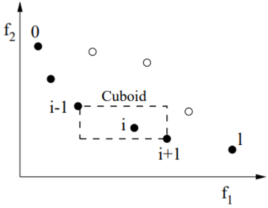
where d_i is the Euclidean distance in the objective space between solution i of POF' and the nearest member of POF^*.

When the true POF is not known you can use Hypervolume/ S-metric

Multi-Guide Particle Swarm Optimizer

The MGPSO algorithm was proposed by Scheepers *et al* and is inspired by the vector evaluated particle swarm optimizer (VEPSO)

- The authors proposed the introduction of a third attractor, in addition to the usual social and cognitive attractors.
  - The aim of the new attractor, $\hat{\boldsymbol{a}}$ is to pull particles towards the Pareto-optimal front (POF).
  - The third attractor is selected from the archive of non-dominated solutions.
  - The archive attractor is selected from the tournament pool as the one with the largest crowding distance (as used by NSGAII) to promote convergence to a diverse pareto-front.



Multi-Guide Particle Swarm Optimizer

The velocity and position update equation of MGPSO are defined as follows:

$$\begin{aligned}
\boldsymbol{v}_i(t+1) &= w\boldsymbol{v}_i(t) + c_1\boldsymbol{r}_1 \otimes (\boldsymbol{y}_i(t) - \boldsymbol{x}_i(t)) \\
&\quad + \lambda_i c_2 \boldsymbol{r}_2 \otimes (\hat{\boldsymbol{y}}_i(t) - \boldsymbol{x}_i(t)) \\
&\quad + (1-\lambda_i) c_3 \boldsymbol{r}_3 \otimes (\hat{\boldsymbol{a}}_i(t) - \boldsymbol{x}_i(t)) &(8)\\
\boldsymbol{x}_i(t+1) &= \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1), &(9)
\end{aligned}$$

where $\boldsymbol{r}_1$, $\boldsymbol{r}_2$, $\boldsymbol{r}_3 \sim U(0,1)^d$, and d is the dimension of the problem PSO is attempting to solve. λ_i is the exploitation trade-off coefficient for particle i, is initialized as a random constant sampled uniformly from $(0,1)$ (λ_i does not vary over iterations).

From which we can obtain the following stability criteria

Sufficient and Necessary condition both for Order-1 and order-2 stability MGPSO

$$0 < c < \frac{12(1-w^2)}{(\lambda^2-\lambda+7)(w+1)-12w}, \quad |w| < 1 \tag{10}$$

where $c = c_1 = c_2 = c_3$

Deriving the Stable Regions Cont.

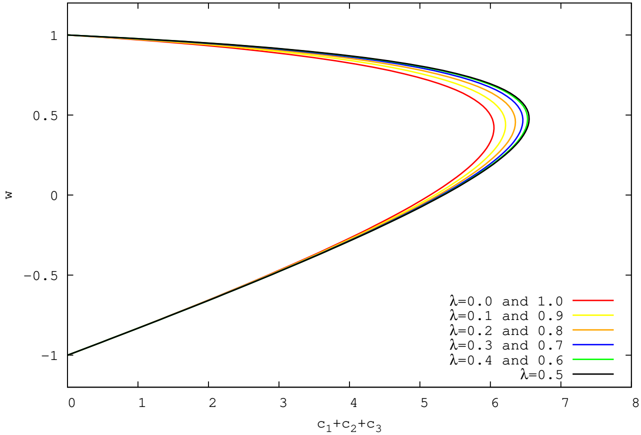


Figure: MGPSO regions of stability

Niching PSO

The basic PSO can be used for niching. However it has inherent properties which make it a poor choice.

- A gbest swarm generally converge around a single point
- Even with a Lbest structure you don't always promote a spread of solutions
  - Better than Gbest but not good enough. Only a small number of niches are found.
- PSO doesn't maintain niches. For example if a PSO finds a more promising area of a search space the particle are likely to almost completely abandons a current niche.

Niching PSO

Instead of using a basic PSO we rather adapt PSO to have true niching ability. We will focus on the NichePSO:

- The basic operating principle of NichePSO is the **self-organization** of particles into independent sub-swarms.
- Each sub-swarm locates and maintains a niche.
- Information exchange is only within the sub-swarm

Niching PSO

NichePSO overview:

- The NichePSO starts with one swarm, referred to as the main swarm, containing all particles.
- As soon as a particle converges on a potential solution, a sub-swarm is created by grouping together particles that are in close proximity to the potential solution.
- Information exchange is only within the sub-swarm
- These particles are then removed from the main swarm, and continue within their sub-swarm to refine (and to maintain) the solution/niche
- Over time, the main swarm shrinks as sub-swarms are spawned from it.

Niching PSO

Create and initialize a d-dimensional *main* swarm, S;
repeat
    Train main swarm, S, for one iteration using *cognition-only* model;
    Update the fitness of each main swarm particle, $S.\mathbf{x}_i$;
    for *each sub-swarm* S_k **do**
        Train sub-swarm particles, $S_k.\mathbf{x}_i$, using a full model PSO;
        Update each particle's fitness;
        Update the swarm radius $S_k.R$;
    endFor
    If possible, merge sub-swarms;
    Allow sub-swarms to absorb any particles from the main swarm that moved into the sub-swarm;
    If possible, create new sub-swarms;
until *stopping condition is true*;
Return $S_k.\hat{\mathbf{y}}$ for each sub-swarm S_k as a solution;

Niching PSO

Identification of the Niche

- if the normalized standard deviation of a particles *i* fitness drops below a ϵ a sub-swarm is created from that particle.
- who joins the sub-swarm?
  - The particle with the smallest euclidean distance is added.

Niching PSO

Absorption of particles into a sub-swarm

- Why do we want to absorb?
  - prevent two sub-swarm from focusing on the same niche.
  - improve the sub-swarm diversity.
- Particles absorption is simple.
  - if $\|\mathbf{x}_i - S_k.\mathbf{g}\| < S_k.R$ then $\mathbf{x}_i$ is removed from S and added to S_k
  - $S_k.R$ is the max distance between the sub-swarm ks gbest and all particles in the sub-swarm.

Niching PSO

Merging Sub-swarms

- It is possible that more than one sub-swarm form to represent the same optimum.
- At which point having two sub-swarms does not make sense.
- Swarms are considered similar if the hyperspace defined by their particle positions and radii intersect.

$$\|S_{k_1}.\mathbf{g} - S_{k_2}.\mathbf{g}\| < S_{k_1}.R + S_{k_2}.R \tag{11}$$

Any of a number of stopping conditions can be used to terminate the search for multiple solutions. It is important that the stopping conditions ensure that each sub-swarm has converged onto a unique solution.