# Particle Swarm Optimization: Part 1

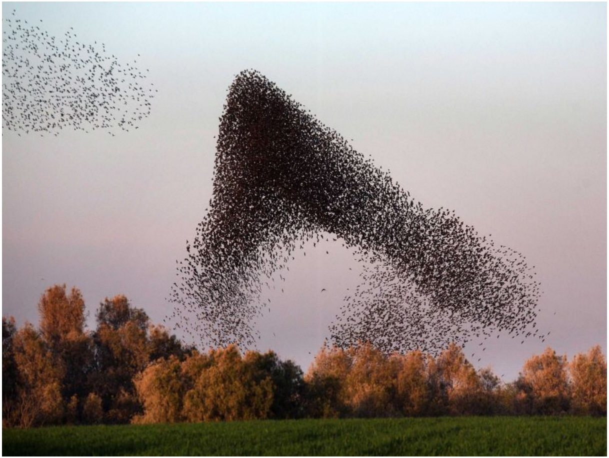
# What is Particle Swarm Optimization?

Particle swarm optimization (PSO) is a stochastic, population-based, search algorithm originally developed by Kennedy and Eberhart in 1995, the inspiration being the social behavior of birds within a flock.

- The initial intent of the particle swarm algorithm was to graphically simulate the movement of birds in an attempt understand what underlying mechanism allow the birds to behave in such a coordinated fashion. In particular their ability to suddenly change direction and then regroup back into an optimal formation.
- However PSO ended up being an effective approach to solving optimization algorithms

# What is Particle Swarm Optimization?

# What is Particle Swarm Optimization?



# What is Particle Swarm Optimization?

PSO operates in a similar fashion to many of the classical optimization approaches, in that we have a current position $\mathbf{x}(t)$ in the search space, from which we generate a new position by adding a movement vector to it $\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{v}(t+1)$.

The fundamental differences are that

- We work with a set of positions $S$, called the swarm.
- $\mathbf{v}(t+1)$ is constructed using
  - Information from the rest of the swarm.
  - Stochastic weighting is used on the information from the swarm.

# The PSO Formula

How do we construct $\mathbf{v}_i(t)$?

- First note the subscript is present because we are working with a swarm.
- The original formula is
  - $\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2 \otimes (\mathbf{g}(t) - \mathbf{x}_i(t))$
- First obverse that $\mathbf{v}_i(t)$ is a recurrence relation, so we would need an initial value for velocity, $\mathbf{v}_i(0)$ (more on this later).

# The PSO Formula

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2 \otimes (\mathbf{g}(t) - \mathbf{x}_i(t))$$

- $\mathbf{p}_i(t)$ indicates the position where particle $i$ obtained the best fitness (objective function evaluation)
- $\mathbf{g}(t)$ indicates the position where the best fitness has been obtained by the swarm as a whole. Specifically, this would be the "best" personal best position found by the whole swarm.

# The PSO Formula

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2 \otimes (\mathbf{g}(t) - \mathbf{x}_i(t))$$

- $\mathbf{p}_i(t) - \mathbf{x}_i(t)$ moves the new position in the direction where the particle $i$ had found success previously. i.e. guided by cognitive information.
- $\mathbf{g}(t) - \mathbf{x}_i(t)$ moves the new position in the direction where the swarm as a whole had found success previously. i.e. guided by social information.
- As a result $c_1$ and $c_2$ are called the cognitive and social coefficients or weights respectively.
- Optimal values for $c_1$ and $c_2$ are problem dependent

# The PSO Formula

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2 \otimes (\mathbf{g}(t) - \mathbf{x}_i(t))$$

- $\mathbf{r}_1$ and $\mathbf{r}_2$ are the stochastic parts of PSO, where $\mathbf{r}_1, \mathbf{r}_2 \sim U(0,1)^d$
- $c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t))$ is called the cognitive component of the PSO update equation.
- $c_2\mathbf{r}_2 \otimes (\mathbf{g}(t) - \mathbf{x}_i(t))$ is called the social component of the PSO update equation.
- In essence PSO is guided by a stochastically weighted sum of its old velocity, cognitive influence, and social influence.
- $\otimes$ indicates componentwise multiplication. Some authors like to instead use, $\mathbf{R}_1$, $\mathbf{R}_2$, as diagonal matrices where each diagonal component is sampled from $U(0,1)$.

# The PSO Formula

Before we delve deeper into the PSO there is a fundamental issue with the original update equations when used in practice?

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2 \otimes (\mathbf{g}(t) - \mathbf{x}_i(t)) \tag{1}$$
$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{2}$$

Can you intuitively guess?

# The PSO Formula

Before we delve deeper into the PSO there is a fundamental issue with the original update equations when used in practice?

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2 \otimes (\mathbf{g}(t) - \mathbf{x}_i(t)) \quad (1)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (2)$$

Can you intuitively guess?

- The velocity often explodes.

# Velocity Clamping

The first way people tried to handle the velocity explosion that equations (1), (2) causes is to utilize velocity clamping. Specifically

$$v_{i,j}(t+1) = \begin{cases} v_{i,j}(t+1) & \text{if } |v_{ij}(t+1)| < V_{max,j} \\ \text{sgn}(v_{i,j}(t+1))V_{max,j} & \text{if } |v_{ij}(t+1)| \geq V_{max,j} \end{cases}$$

While clamping limits the velocity it does not confine the positions, (remember this when we discuss convergence)

# Velocity Clamping

Problems?

# Velocity Clamping

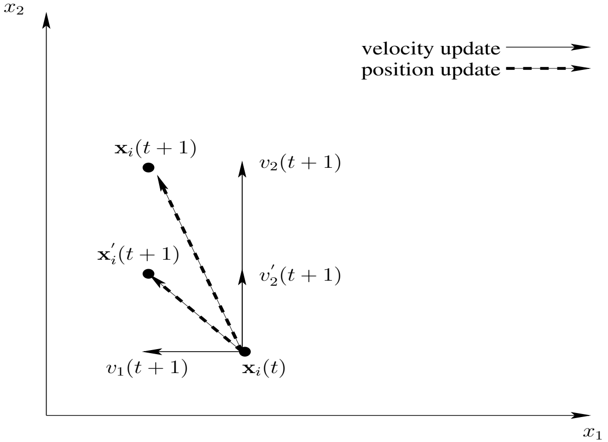
Problems?

- Picking the clamping thresholds. Should they be the same in each dimension?

# Velocity Clamping

Problems?

- Picking the clamping thresholds. Should they be the same in each dimension?
- It can change the search direction.



# Velocity Clamping

Time varying clamping approaches exist:

- Dynamically changing $V_{max}$ when $g$ does not improve over $\tau$ iterations

$$V_{max,j}(t+1) = \begin{cases} \beta V_{max,j}(t) & \text{if } f(\hat{\mathbf{y}}(t)) \geq f(\hat{\mathbf{y}}(t-t^{'})),\ \forall\ t^{'} = 1, \ldots, \tau \\ V_{max,j}(t) & \text{otherwise} \end{cases}$$

  $\beta$ decreases from 1.0 to 0.01, using some form of decrease schedule.

- Exponentially decaying $V_{max}$

$$V_{max,j}(t+1) = (1 - (t/n_t)^{\alpha})V_{max,j}(t)$$

  where $\alpha$ is a positive constant.

# Alternative to Clamping.

Velocity clamping handles a symptom rather then fix the underlying issue. There are two version of the PSO update equation which are currently in use that resolve (under some conditions) the underlying issue of uncontrolled velocity explosion.

- Proposed by Shi and Eberhart.

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2 \otimes (\mathbf{g}(t) - \mathbf{x}_i(t)) \qquad (3)$$

  where $w$ is called the inertia weight.

- Proposed by Clerc

$$\mathbf{v}_i(t+1) = \chi\left[\mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) + c_2\mathbf{r}_2 \otimes (\mathbf{g}(t) - \mathbf{x}_i(t))\right] \qquad (4)$$

  where $\chi$ is called the constriction coefficient. Clerc also provided criteria for selecting $\chi$ to ensuring "convergent" behavior.

*The models are equivalent

# Impact of Social and Cognitive Coefficient Choices

- $c_1 = c_2 = 0$?
- $c_1 > 0, c_2 = 0$:
  - particles are independent hill-climbers
  - local search by each particle
  - cognitive-only PSO model
- $c_1 = 0, c_2 > 0$:
  - swarm is "one" stochastic hill-climber
  - social-only PSO model
- $c_1 = c_2 > 0$:
  - particles are attracted towards the average of $\mathbf{p}_i$ and $\hat{\mathbf{g}}_i$
- $c_2 > c_1$: Social Bias
  - could be more beneficial for unimodal problems
- $c_2 < c_1$: Cognitive Bias
  - could be more beneficial for multimodal problems

# Impact of Social and Cognitive Coefficient Choices

- lower $c_1$ and $c_2$:
  - smooth particle trajectories
- higher $c_1$ and $c_2$:
  - more acceleration, abrupt movements

  Three useful theoretical studies on predicted particle trajectories based on parameter selection are
  - M. Clerc and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation, 6(1):58–73, 2002
  - J.L. Fernández-Martínez and E. García-Gonzalo. Stochastic Stability Analysis of the Linear Continuous and Discrete PSO Models. IEEE Transactions on Evolutionary Computation, 15(3):405–423, 2011
  - M.R. Bonyadi and Z. Michalewicz. Impacts of coefficients on movement patterns in the particle swarm optimization algorithm. IEEE Transactions on Evolutionary Computation, 21 , 378–390. (2017)

# Particle Convergence/Stability

There where a number of studies that proved that particle "convergence" could be obtained if specific coefficients where used. The most recent criteria for convergence are the following (if $c_1 = c_2$)

$$c_1 + c_2 < \frac{24(1 - w^2)}{7 - 5w}, \text{ and } |w| \leq 1 \tag{5}$$

"convergence" is in quotes, as the criteria of equation (5) are necessary and sufficient for

- order-1 and order-2 stability.
- and not deterministic convergence.

It has been shown that particle stability (achieved by satisfying the above criteria) has a substantial impact on performance.

# Social Network Structures

Up until now we have assume that all particle are connected, in the sense that the position **g** is derived from the swarm as a whole, implying all particles can "communicate" with each other.

- This type of PSO is refereed to as the **GBest** PSO.

There is exists a number of ways that PSO particles can be connected such that each particle $i$ has a neighborhood $\mathcal{N}_i$ from which it must derive its social information from.

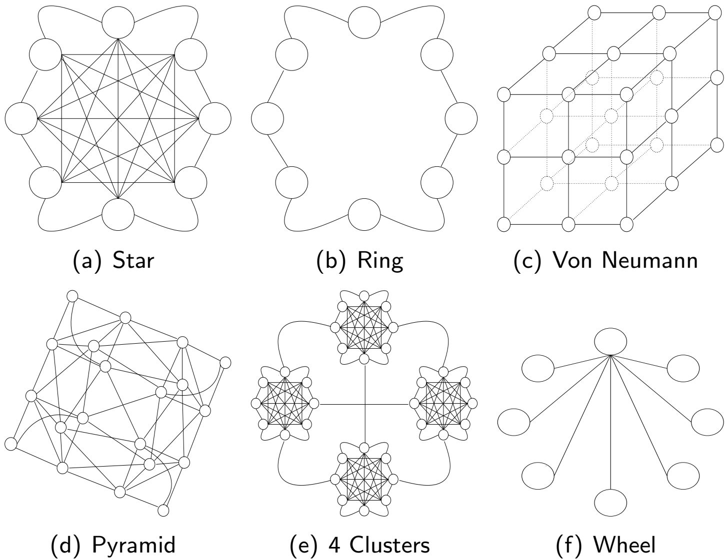
- In which case the social informer, is denoted $n_i$ as apposed to $g$, and is refereed to as the neighborhood best (or local best).
- This type of PSO is refereed to as the **LBest** PSO

# Social Network Structures

Some of the most common fixed social topologies are

- Star (used by **GBest** PSO)
- Ring
- Wheel
- N-Cluster
- Pyramid
- Von-Neumann

# Social Network Structures



# Social Network Structures

It is also possible to have social network structure that change during the course of a run.

- For example restructure the network based on physical proximity in the search space. For example a neighborhood could be seen as the $n$ closest particles.
- Use information theoretic ideas, such as breaking connections if no useful information was ever transported over the connection.

A subtle point which is worth noting, is the connection between particles need not be bi-directional,

- particle $i$ might be in $\mathcal{N}_j$ even if $j$ is not in $\mathcal{N}_i$.

# Algorithm Summary for PSO Minimization

1: Create and initialize a $d$-dimensional swarm, $\Omega(0)$, of $N$ particles uniformly within a predefined hypercube.
2: Let $f$ be the objective function.
3: Let $\boldsymbol{p}_i$ represent the personal best position of particle $i$, initialized to $\boldsymbol{x}_i(0)$.
4: Let $\boldsymbol{n}_i$ represent the neighborhood best position of particle $i$, initialized to $\boldsymbol{x}_i(0)$.
5: Initialize $\boldsymbol{v}_i(0)$ to $\mathbf{0}$. (may differ but there is good reason)

# Algorithm Summary for PSO Minimization

**repeat**
  **for all** particles $i = 1, \cdots, N$ **do**
    **if** $f(\boldsymbol{x}_i) < f(\boldsymbol{y}_i)$ **then**
      $\boldsymbol{p}_i = \boldsymbol{x}_i$
    **end if**
    **for all** particles $\hat{i}$ with particle $i$ in their NBD **do**
      **if** $f(\boldsymbol{p}_i) < f(\boldsymbol{n}_{\hat{i}})$ **then**
        $\boldsymbol{n}_{\hat{i}} = \boldsymbol{p}_i$
        **if** $f(\boldsymbol{n}_{\hat{i}}) < f(\boldsymbol{g})$ **then**
          $\boldsymbol{g} = \boldsymbol{n}_{\hat{i}}$
        **end if**
      **end if**
    **end for**
  **end for**
  **for all** particles $i = 1, \cdots, N$ **do**
    update velocity of particle $i$ using the chosen update equation
    update position of particle $i$ using the chosen update equation
  **end for**
**until** stopping condition is met

# Stopping conditions

When to end a run, is an important question for any optimization algorithm. Common options are

- End once a maximum number of iteration has been reached, or once a maximum number of function evaluation has been reached.
  - This can be seen as a computational budget, generally the evaluation of the objective function is the most expensive operation.
- Terminate once an acceptable solution is found, generally when you have a solution within an $\epsilon$ of the optimal target value.
  - This is more appropriate for comparative runs, when you already know the optimum.

# Stopping conditions

- Terminate when no improvement is observed over a number of iterations.
- When a “good enough” solution is found. It might be known that you want a solution that is at least $\alpha$ good and if you find one better or equal you can stop optimizing (through many people will always want the best solution you can find)
  - Consider a situation where a objective function might take a day to evaluate for a single position.

# Stopping conditions

A stopping condition that is a bit more specific to PSO is to stop once the normalized swarm radius is close to zero.

$$R_{norm} = \frac{R_{max}}{diameter\ \ of\ \ initial\ \ swarm} \approx 0$$

where

$$R_{max} = \max_{1 \leq m \leq ||\Omega||} ||\mathbf{x}_m - \mathbf{g}|| \tag{6}$$

with $\Omega$ the set of particles in the swarm.

# Synchronous versus Asynchronous Updates

- The PSO algorithm we presented in our algorithm summary, performs synchronous updates of the personal best and global (or local) best positions. Synchronous updates are done separately from the particle position updates.
- Though an alterntive asynchronous approach is possible.
  - The idea is that the personal best position is updated directly after a position is updated.
  - This implies the neighborhood (or even global) best position can change after each particles position is updated

# Synchronous versus Asynchronous Updates

- Asynchronous updates have the advantage that immediate feedback is given about the best regions of the search space, while feedback with synchronous updates is only given once per iteration.
- The immediate, potential disadvantage of the asynchronous updates is that it can lead to an over exploitation of new information.

# Most Basic PSO Variants

- **Cognition-only** model: The model excludes the social component from the original velocity equation. As mentioned earlier, same as setting $c_2 = 0$.

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_1\mathbf{r}_1 \otimes (\mathbf{p}_i(t) - \mathbf{x}_i(t)) \tag{7}$$

- **Social-only** model: The model excludes the cognitive component from the original velocity equation. As mentioned earlier, same as setting $c_1 = 0$.

$$\mathbf{v}_i(t+1) = w\mathbf{v}_i(t) + c_2\mathbf{r}_2 \otimes (\mathbf{g}(t) - \mathbf{x}_i(t)) \tag{8}$$

- **Selfless** model: the same as the social model, but the particle itself is not allowed to become the neighborhood best.

# Basic PSO Parameters

PSO has a number of problem dependent parameters, some are more sensitive than others.

- Swarm size
- Neighborhood size and Topology
- Number of iterations, or stopping condition threshold
- Acceleration coefficients, $c_1$, $c_2$, and $w$ or $\chi$ if constriction is being used.
- Velocity clamping limits (if being used).