Particle Swarm Optimization: Pitfalls and Convergence Aspects

Andries Engelbrecht

Department of Computer Science, University of Pretoria, South Africa

Contents

Introduction

Overview of the basic PSO

Aspects of PSO

Particle Trajectories

Problem with PSO

Summary

Introduction

Particle swarm optimization (PSO):

- developed by Kennedy and Eberhart [5],
- first published in 1995, and
- with an exponential increase in the number of publications since then.

What is PSO?

- a simple, computationally efficient optimization method
- population-based, stochastic search
- based on a social-psychological model of social influence and social learning [6]
- individuals follow a very simple behavior: emulate the success of neighboring individuals
- emergent behavior: discovery of optimal regions of a high dimensional search space

What are the origins of PSO?

- in the work of Reynolds on "boids" [8],
- the work of Heppner and Grenander on using a "rooster" as attractor [4]
- simplified social model of determining nearest neighbors and velocity matching
- initial objective: to simulate the graceful, unpredictable choreography of collision-proof birds in a flock
- at each iteration, each individual determines its nearest neighbor and replaces its velocity with that of its neighbor
- resulted in synchronous movement of the flock
- random adjustments to velocities prevented individuals to settle too quickly on an unchanging direction
- adding roosters as attractors:
  - personal best
  - neighborhood best
  - $\rightarrow$ particle swarm optimization

Questions:

- Even with so much research done in PSO, and applications of PSO to solve complex real-world problems, do we really understand the behavior of PSO?
- How can we make sure that we have convergent trajectories?
- How can we prevent premature convergence?

Overview of Basic PSO

What are the main components?

- a swarm of particles
- each particle represents a candidate solution
- elements of a particle represent parameters to be optimized

The search process:

- Position updates

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

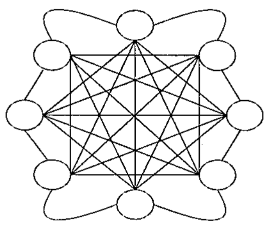
  where

$$\mathbf{x}_{ij}(0) \sim U(x_{min,j}, x_{max,j})$$

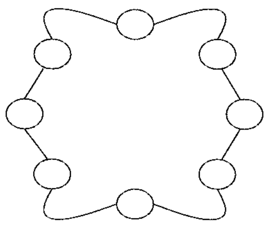
- Velocity
  - drives the optimization process
  - step size
  - reflects experiential knowledge and socially exchanged information

Social network structures

- social interaction based on neighborhoods
- envy
- first used network structures: star and ring topologies



(a) Star



(b) Ring

Figure 1: Social Network Structures

Global best (gbest) PSO

- uses the star social network
- velocity update per dimension:

$$\begin{aligned} v_{ij}(t+1) &= v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] \\ &\quad + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \end{aligned}$$

- $v_{ij}(0) = 0$ (usually)
- c_1, c_2 are positive acceleration coefficients
- $r_{1j}(t), r_{2j}(t) \sim U(0,1)$
- $\mathbf{y}(t)$ is the personal best position calculated as

$$\mathbf{y}_i(t+1) = \begin{cases} \mathbf{y}_i(t) & \text{if } f(\mathbf{x}_i(t+1)) \geq f(\mathbf{y}_i(t)) \\ \mathbf{x}_i(t+1) & \text{if } f(\mathbf{x}_i(t+1)) < f(\mathbf{y}_i(t)) \end{cases}$$

- $\hat{\mathbf{y}}(t)$ is the global best position calculated as

$$\begin{aligned} \hat{\mathbf{y}}(t) &\in \{\mathbf{y}_0(t), \ldots, \mathbf{y}_{n_s}(t)\} | f(\hat{\mathbf{y}}(t)) \\ &= \min\{f(\mathbf{y}_0(t)), \ldots, f(\mathbf{y}_{n_s}(t))\} \end{aligned}$$

  or

$$\hat{\mathbf{y}}(t) = \min\{f(\mathbf{x}_0(t)), \ldots, f(\mathbf{x}_{n_s}(t))\}$$

  where n_s is the number of particles in the swarm

Algorithm 1 *gbest* PSO

```
Create and initialize an $n_x$-dimensional swarm, $S$;
repeat
    for each particle $i = 1, \ldots, S.n_s$ do
        //set the personal best position
        if $f(S.\mathbf{x}_i) < f(S.\mathbf{y}_i)$ then
            $S.\mathbf{y}_i = S.\mathbf{x}_i$;
        end
        //set the global best position if $f(S.\mathbf{y}_i) < f(S.\hat{\mathbf{y}})$ then
            $S.\hat{\mathbf{y}} = S.\mathbf{y}_i$;
        end
    end
    for each particle $i = 1, \ldots, S.n_s$ do
        update the velocity;
        update the position;
    end
until stopping condition is true;
```

Local best (lbest) PSO

- uses the ring social network

$$\begin{aligned} v_{ij}(t+1) &= v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] \\ &\quad + c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)] \end{aligned}$$

- $\hat{\mathbf{y}}_i$ is the neighborhood best, defined as

$$\hat{\mathbf{y}}_i(t+1) \in \{\mathcal{N}_i | f(\hat{\mathbf{y}}_i(t+1)) = \min\{f(\mathbf{x})\}, \quad \forall \mathbf{x} \in \mathcal{N}_i\}$$

  with the neighborhood defined as

$$\begin{aligned} \mathcal{N}_i = \{&\mathbf{y}_{i-n_{\mathcal{N}_i}}(t), \mathbf{y}_{i-n_{\mathcal{N}_i}+1}(t), \ldots, \\ &\mathbf{y}_{i-1}(t), \mathbf{y}_i(t), \mathbf{y}_{i+1}(t), \ldots, \mathbf{y}_{i+n_{\mathcal{N}_i}}(t)\} \end{aligned}$$

  where $n_{\mathcal{N}_i}$ is the neighborhood size

- neighborhoods are based on particle indices, not spatial information
- neighborhoods overlap to facilitate information exchange

Algorithm 2 *lbest* PSO

```
Create and initialize an $n_x$-dimensional swarm, $S$;
repeat
    for each particle $i = 1, \ldots, S.n_s$ do
        //set the personal best position
        if $f(S.\mathbf{x}_i) < f(S.\mathbf{y}_i)$ then
            $S.\mathbf{y}_i = S.\mathbf{x}_i$;
        end
        //set the neighborhood best position
        if $f(S.\mathbf{y}_i) < f(S.\hat{\mathbf{y}}_i)$ then
            $S.\hat{\mathbf{y}} = S.\mathbf{y}_i$;
        end
    end
    for each particle $i = 1, \ldots, S.n_s$ do
        update the velocity;
        update the position;
    end
until stopping condition is true;
```

gbest PSO vs *lbest* PSO

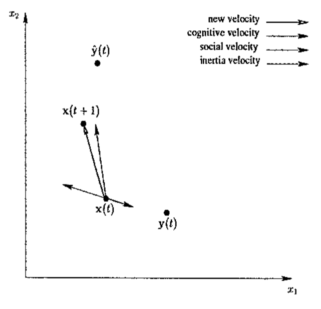
- speed of convergence
- susceptibility to local minima?

Aspects of Basic PSO

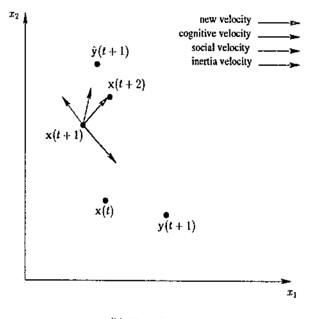
Velocity components:

- previous velocity, $\mathbf{v}_i(t)$
  - inertia component
  - memory of previous flight direction
  - prevents particle from drastically changing direction
- cognitive component, $c_1\mathbf{r}_1(\mathbf{y}_i - \mathbf{x}_i)$
  - quantifies performance relative to past performances
  - memory of previous best position
  - nostalgia
- social component, $c_2\mathbf{r}_2(\hat{\mathbf{y}}_i - \mathbf{x}_i)$
  - quantifies performance relative to neighbors
  - envy

Geometric illustration



(a) Time Step t



(b) Time Step $t + 1$

Figure 2: Geometrical Illustration of Velocity and Position Updates for a Single Two-Dimensional Particle

Exploration–exploitation tradeoff

- exploration – the ability to explore regions of the search space
- exploitation – the ability to concentrate the search around a promising area to refine a candidate solution
- c_1 vs c_2 and the influence on the exploration–exploitation tradeoff

Velocity clamping:

- the problem: velocity quickly explodes to large values
- solution:

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{if } v'_{ij}(t+1) < V_{max,j} \\ V_{max,j} & \text{if } v'_{ij}(t+1) \geq V_{max,j} \end{cases}$$

- controlling the global exploration of particles
- problem-dependent
- does not confine the positions, only the step sizes

- problem to be aware of

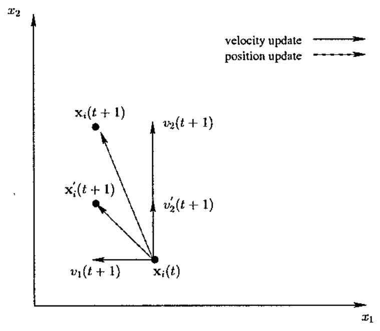


Figure 3: Effects of Velocity Clamping

- dynamically changing V_{max} when *gbest* does not improve over τ iterations [9]

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

- exponentially decaying V_{max} [3]

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

Inertia weight [10]

- to control exploration and exploitation
- controls the momentum
- velocity update changes to

$$\begin{aligned} v_{ij}(t+1) &= wv_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] \\ &\quad + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \end{aligned}$$

- for $w \geq 1$
  - velocities increase over time
  - swarm diverges
  - particles fail to change direction towards more promising regions
- for $0 < w < 1$
  - particles decelerate
  - convergence also dependent on values of c_1 and c_2
- exploration–exploitation
  - large values – favor exploration
  - small values – promote exploitation
- problem-dependent

- dynamically changing inertia weights
  - $w \sim N(0.72, \sigma)$
  - linear decreasing [11]

$$w(t) = (w(0) - w(n_t))\frac{(n_t - t)}{n_t} + w(n_t)$$

  - non-linear decreasing [15]

$$w(t+1) = \alpha w(t')$$

  with $w(t) = 1.4$
  - based on relative improvement [1]

$$w_i(t+1) = w(0) + (w(n_t) - w(0))\frac{e^{m_i(t)} - 1}{e^{m_i(t)} + 1}$$

  where the relative improvement, m_i, is estimated as

$$m_i(t) = \frac{f(\hat{\mathbf{y}}_i(t)) - f(\mathbf{x}_i(t))}{f(\hat{\mathbf{y}}_i(t)) + f(\mathbf{x}_i(t))}$$

Constriction Coefficient [2]

- to ensure convergence to a stable point without the need for velocity clamping

$$\begin{aligned} v_{ij}(t+1) &= \chi[v_{ij}(t) + \phi_1(y_{ij}(t) - x_{ij}(t)) \\ &\quad + \phi_2(\hat{y}_j(t) - x_{ij}(t))] \end{aligned}$$

where

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi(\phi - 4)}|}$$

with

$$\begin{aligned} \phi &= \phi_1 + \phi_2 \\ \phi_1 &= c_1 r_1 \\ \phi_2 &= c_2 r_2 \end{aligned}$$

- if $\phi \geq 4$ and $\kappa \in [0,1]$, then the swarm is guaranteed to converge
- $\chi \in [0,1]$
- κ controls exploration–exploitation
  $\kappa \approx 0$: fast convergence, local exploitation
  $\kappa \approx 1$: slow convergence, high degree of exploration

- effectively equivalent to inertia weight for specific χ:
  $w = \chi, \phi_1 = \chi c_1 r_1$ and $\phi_2 = \chi c_2 r_2$

Synchronous vs asynchronous updates

- synchronous:
  - personal best and neighborhood bests updated separately from position and velocity vectors
  - slower feedback
  - better for *gbest*
- asynchronous:
  - new best positions updated after each particle position update
  - immediate feedback about best regions of the search space
  - better for *lbest*

Acceleration coefficients (trust parameters):

- $c_1 = c_2 = 0$?
- $c_1 > 0, c_2 = 0$:
  - particles are independent hill-climbers
  - local search by each particle

- $c_1 = 0, c_2 > 0$:
  - swarm is one stochastic hill-climber
- $c_1 = c_2 > 0$:
  - particles are attracted towards the average of $\mathbf{y}_i$ and $\hat{\mathbf{y}}_i$
- $c_2 > c_1$:
  - more beneficial for unimodal problems
- $c_1 < c_2$:
  - more beneficial for multimodal problems
- low c_1 and c_2:
  - smooth particle trajectories
- high c_1 and c_2:
  - more acceleration, abrupt movements
- adaptive acceleration coefficients [7]

$$c_1(t) = (c_{1,min} - c_{1,max})\frac{t}{n_t} + c_{1,max}$$

$$c_2(t) = (c_{2,max} - c_{2,min})\frac{t}{n_t} + c_{2,min}$$

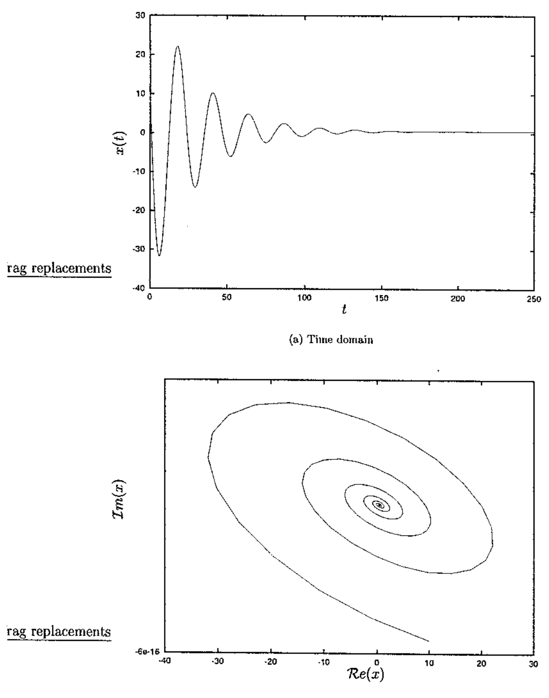
Particle Trajectories

Simplified particle trajectories [13]

- no stochastic component
- single, one-dimensional particle
- using w
- personal best and global best are fixed:
  $y = 1.0, \hat{y} = 0.0$

Example trajectories:

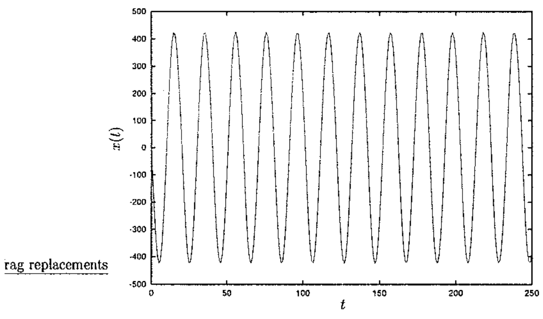
- Convergence to an equilibrium (figure 4)
- Cyclic behavior (figure 5)
- Divergent behavior (figure 6)



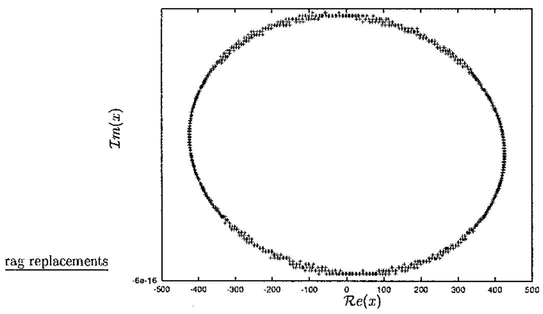
(a) Time domain

(b) Phase space

Figure 4: Convergent Trajectory for Simplified System, with $w = 0.5$ and $\phi_1 = \phi_2 = 1.4$

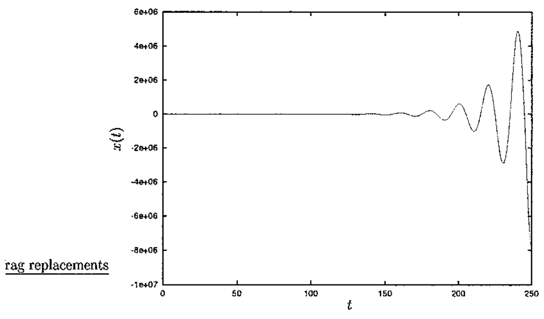


(a) Time domain

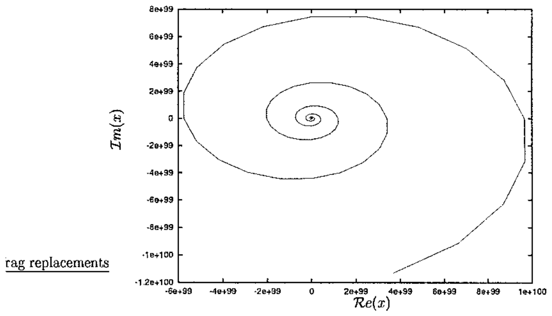


(b) Phase space

Figure 5: Cyclic Trajectory for Simplified System, with $w = 1.0$ and $\phi_1 = \phi_2 = 1.999$



(a) Time domain



(b) Phase space

Figure 6: Divergent Trajectory for Simplified System, with $w = 0.7$ and $\phi_1 = \phi_2 = 1.9$

Convergence conditions:

- What do we mean by the term convergence?
- Convergence map for values of w and $\phi = \phi_1 + \phi_2$, where $\phi_1 = c_1 r_1, \phi_2 = c_2 r_2$

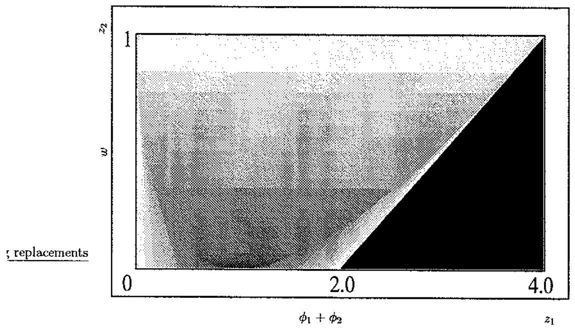


Figure 7: Convergence Map for Values of w and $\phi = \phi_1 + \phi_2$

- conditions on values of w, c_1 and c_2:

$$1 > w > \frac{1}{2}(\phi_1 + \phi_2) - 1 \geq 0$$

Stochastic trajectories:

- $w = 1.0, c_1 = c_2 = 2.0$
  - violates the convergence condition
  - for $w = 1.0$, $c_1 + c_2 < 4.0$ to validate the condition

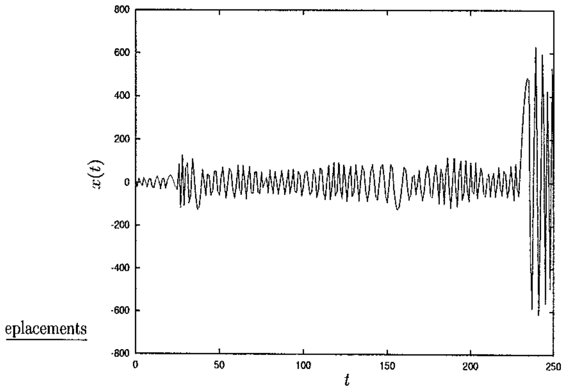


Figure 8: Stochastic Particle Trajectory for $w = 1.0$ and $c_1 = c_2 = 2.0$

- $w = 0.9, c_1 = c_2 = 2.0$
  - violates the convergence condition
  - for $w = 0.9$, $c_1 + c_2 < 3.8$ to validate the condition

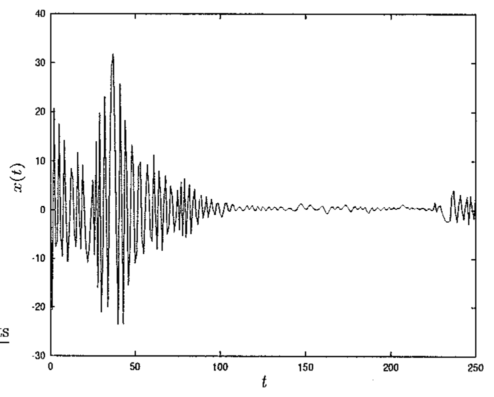


Figure 9: Stochastic Particle Trajectory for $w = 0.9$ and $c_1 = c_2 = 2.0$

- What is happening here?
  - since $0 < \phi_1 + \phi_2 < 4$,
  - and $r_1, r_3 \sim U(0, 1)$.
  - $\text{prob}(c_1 + c_2 < 3.8) = \frac{3.8}{4} = 0.95$

- good convergent parameter choices:
  - $w = 0.7, c_1 = 1.4 = c_2 = 1.4$
  - validates the convergence condition

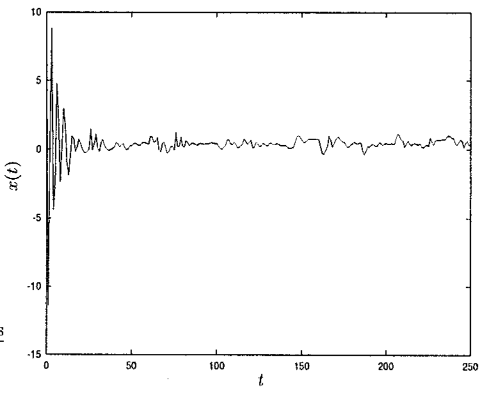


Figure 10: Stochastic Particle Trajectory for $w = 0.7$ and $c_1 = c_2 = 1.4$

- under stochastic ϕ_1 and ϕ_2, convergent behavior results when [13]

$$\phi_{ratio} = \frac{\phi_{crit}}{c_1 + c_2}$$

is close to 1.0, where

$$\phi_{crit} = \sup \phi \,|\, 0.5\,\phi - 1 < w, \quad \phi \in (0, c_1 + c_2]$$

Problem with Basic PSO

It has been proven that particles converge to a stable point [13, 2, 12]

$$\frac{\phi_1 y + \phi_2 \hat{y}}{\phi_1 + \phi_2}$$

Problem:

- this point is not necessarily a minimum
- may prematurely converge to a stable state
- formal proofs in [13]

Potential dangerous property:

- when $\mathbf{x}_i = \mathbf{y}_i = \hat{\mathbf{y}}_i$
- then the velocity update depends only on $w\mathbf{v}_i$
- if this condition persists for a number of iterations,

$$w\mathbf{v}_i \rightarrow 0$$

Solution:

- prevent the condition from occurring
- guaranteed convergence PSO (GCPSO) [13, 14]
- change the position update of the global best (or neighborhood best) to

  $$x_{\tau j}(t+1) = \hat{y}_j(t) + w v_{\tau j}(t) + \rho(t)(1 - 2r_2(t))$$

  where τ is the index of the global (neighborhood) best particle
- updates velocity of the global (neighborhood) best using

  $$\begin{aligned} v_{\tau j}(t+1) &= -x_{\tau j}(t) + \hat{y}_j(t) + w v_{\tau j}(t) \\ &\quad + \rho(t)(1 - 2r_{2j}(t)) \end{aligned}$$

  where $\rho(t)$ is a scaling factor
- the term $\rho(t)(1 - 2r_{2j}(t))$ forces the PSO to perform a random search in an area around $\hat{\mathbf{y}}(t)$

- $\rho(t)$ controls the diameter of this search area:

$$\rho(t{+}1) = \begin{cases} 2\rho(t) & \text{if } \#successes(t) > \epsilon_s \\ 0.5\rho(t) & \text{if } \#failures(t) > \epsilon_c \\ \rho(t) & \text{otherwise} \end{cases}$$

  where $\#successes$ and $\#failures$ respectively denote the number of consecutive successes and failures, with a failure defined as $f(\hat{\mathbf{y}}(t)) \leq f(\hat{\mathbf{y}}(t+1))$
- GCPSO is proven to be a local minimizer [14]

Summary

Despite its simplicity, PSO has been very successful

However, care has to be taken in the selection of parameter values to ensure convergent trajectories

The original PSO as a flaw which may cause it to prematurely converge to an equilibrium which does not represent an optimum

References

[1] M. Clerc. Think Locally, Act Locally: The Way of Life of Cheap-PSO, an Adaptive PSO. Technical report, http://clerc.maurice.free.fr/pso/, 2001.

[2] 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.

[3] H-Y. Fan. A Modification to Particle Swarm Optimization Algorithm. *Engineering Computations*, 19(7-8):970–989, 2002.

[4] F. Heppner and U. Grenander. A Stochastic Nonlinear Model for Coordinated Bird Flocks. In S. Krasner, editor, *The Ubiquity of Chaos*. AAAS Publications, 1990.

[5] J. Kennedy and R.C. Eberhart. Particle Swarm Optimization. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, pages 1942–1948. IEEE Press, 1995.

[6] J. Kennedy and R. Mendes. Neighborhood Topologies in Fully-Informed and Best-of-Neighborhood Particle Swarms. In *Proceedings of the IEEE International Workshop on Soft Computing in Industrial Applications*, pages 45–50, June 2003.

[7] A.C. Ratnaweera, S.K. Halgamuge, and H.C. Watson. Particle Swarm Optimiser with Time Varying Acceleration Coefficients. In *Proceedings of the International Conference on Soft Computing and Intelligent Systems*, pages 240–255, 2002.

[8] C.W. Reynolds. Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, 21(4):25–34, 1987.

[9] J.F. Schutte and A.A. Groenwold. Sizing Design of Truss Structures using Particle Swarms. *Structural and Multidisciplinary Optimization*, 25(4):261–269, 2003.

[10] Y. Shi and R.C. Eberhart. A Modified Particle Swarm Optimizer. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 69–73, May 1998.

[11] P.N. Suganthan. Particle Swarm Optimiser with Neighborhood Operator. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1958–1962. IEEE Press, 1999.

[12] I.C. Trelea. The Particle Swarm Optimization Algorithm: Convergence Analysis and Parameter Selection. *Information Processing Letters*, 85(6):317–325, 2003.

[13] F. van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[14] F. van den Bergh and A.P. Engelbrecht. A New Locally Convergent Particle Swarm Optimizer. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 96–101, 2002.

[15] G. Venter and J. Sobieszczanski-Sobieski. Multidisciplinary Optimization of a Transport Aircraft Wing using Particle Swarm Optimization. In *Ninth AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002.