Particle Swarm Optimization

AP Engelbrecht

Computational Intelligence Research Group (CIRG)
Department of Computer Science
University of Pretoria
South Africa
engel@cs.up.ac.za
http://cirg.cs.up.ac.za

GECCO'15 Companion, July 11–15, 2015, Madrid, Spain.
ACM 978-1-4503-3488-4/15/07.
http://dx.doi.org/10.1145/2739482.2756564

Instructor

Andries Engelbrecht received the Masters and PhD degrees in Computer Science from the University of Stellenbosch, South Africa, in 1994 and 1999 respectively. He is a Professor in Computer Science at the University of Pretoria, and serves as Head of the department. He also holds the position of South African Research Chair in Artificial Intelligence. His research interests include swarm intelligence, evolutionary computation, artificial neural networks, artificial immune systems, and the application of these Computational Intelligence paradigms to data mining, games, bioinformatics, finance, and difficult optimization problems. He is author of two books, Computational Intelligence: An Introduction and Fundamentals of Computational Swarm Intelligence.

Presentation Outline

Introduction
The Origins

Particle swarm optimization (PSO):
- developed by Kennedy & Eberhart [11, 22],
- 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
- individuals follow very simple behaviors:
  - emulate the success of neighboring individuals,
  - but also bias towards own experience of success
- emergent behavior: discovery of optimal regions within a high dimensional search space

What are the origins of PSO?

- In the work of Reynolds on “boids” [36]
  - collision avoidance
  - velocity matching
  - flock centering
- The work of Heppner and Grenander on using a “rooster” as attractor of all birds in the flock [18]

- Original PSO is a 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
  - Randomly initializes positions of birds
  - At each iteration, each individual determines its nearest neighbor and replaces its velocity with that of its neighbor
- This resulted in synchronous movement of the flock, but flock settled too quickly on an unanimous, unchanging flying direction

- Random adjustments to velocities (referred to as craziness) prevented individuals to settle too quickly on an unchanging direction
- To further expand the model, roosters were added as attractors:
  - personal best
  - neighborhood best

  → particle swarm optimization

Overview of Basic PSO

Main Components

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:

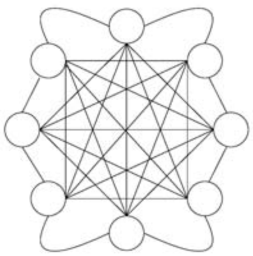
- Position updates

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1), \quad \mathbf{x}_{ij}(0) \sim U(x_{min,j}, x_{max,j})$$

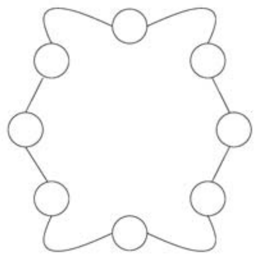
- Velocity (step size)
  - drives the optimization process
  - step size
  - reflects experiential knowledge and socially exchanged information

Overview of Basic PSO
Social Network Structures

Social network structures are used to determine best positions/attractors



: Star Topology

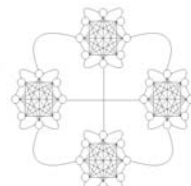


: Ring Topology

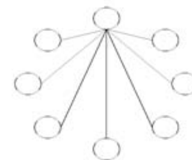
Overview of Basic PSO
Social Network Structures (cont)

(a) Von Neumann

(b) Pyramid



(c) 4 Clusters



(d) Wheel

Overview of Basic PSO
global best (gbest) PSO

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

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

- $v_{ij}(0) = 0$ (preferred)
- c_1, c_2 are positive acceleration coefficients
- $r_{1j}(t), r_{2j}(t) \sim U(0,1)$
- note that a random number is sampled for each dimension

Overview of Basic PSO
gbest PSO (cont)

- $\mathbf{y}_i(t)$ is the personal best position calculated as (assuming minimization)

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

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

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

or (removing memory of best positions)

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

where n_s is the number of particles in the swarm

Overview of Basic PSO
gbest PSO Algorithm

Create and initialize an n_x-dimensional swarm, S;
repeat
    for *each particle* $i = 1, \ldots, S.n_s$ **do**
        if $f(S.\mathbf{x}_i) < f(S.\mathbf{y}_i)$ **then**
            $S.\mathbf{y}_i = S.\mathbf{x}_i$;
        end
        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*;

Overview of Basic PSO
local best (lbest) PSO

- uses the ring social network

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

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

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

  with the neighborhood defined as

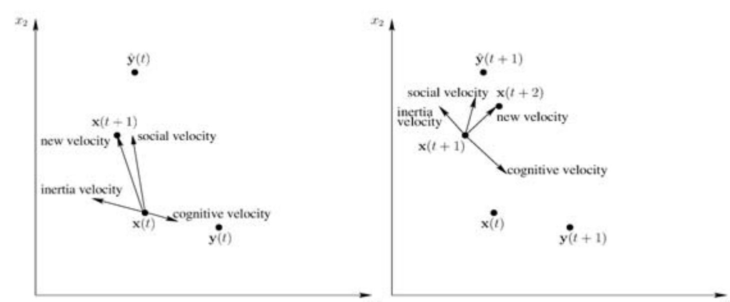
$$\mathcal{N}_i = \{\mathbf{y}_{i-n_{\mathcal{N}_i}}(t), \mathbf{y}_{i-n_{\mathcal{N}_i}+1}(t), \ldots, \mathbf{y}_{i-1}(t), \mathbf{y}_i(t), \mathbf{y}_{i+1}(t), \ldots, \mathbf{y}_{i+n_{\mathcal{N}_i}}(t)\}$$

  where $n_{\mathcal{N}_i}$ is the neighborhood size
- neighborhoods based on particle indices, not spatial information
- neighborhoods overlap to facilitate information exchange

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

Aspects of Basic PSO
Geometric Illustration



(a) Time Step t (b) Time Step $t+1$

Aspects of Basic PSO
Velocity Clamping

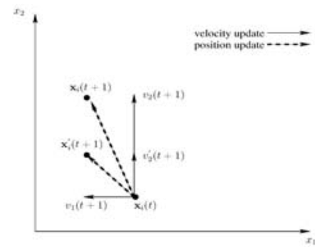
- the problem: velocity quickly explodes to large values
- solution:

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

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

Aspects of Basic PSO
Velocity Clamping (cont)

- Issues with velocity clamping:
  - dimensions with ranges smaller than V_{max} will never be clamped
  - changes search direction – normalized clamping



: Change in Search Direction Due to Velocity Clamping

Aspects of Basic PSO
Velocity Clamping (cont)

- problem-dependent
  - dynamically changing V_{max} when *gbest* does not improve over τ iterations [38]

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

  β decreases from 1.0 to 0.01
  - exponentially decaying V_{max} [16]

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

Aspects of Basic PSO
Inertia Weight

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

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

- 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, depending on c_1 and c_2
- exploration–exploitation
  - large values – favor exploration
  - small values – promote exploitation
- problem-dependent

Aspects of Basic PSO
Inertia Weight (cont)

Dynamically changing inertia weights

- $w \sim N(0.72, \sigma)$
- linear decreasing [39]

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

- non-linear decreasing [44]

$$w(t+1) = \alpha w(t'), \quad w(0) = 1.4$$

- based on relative improvement [6]

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

where the relative improvement, m_i, is estimated as

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

Aspects of Basic PSO
Constriction Coefficient

- 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

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

Aspects of Basic PSO
Constriction Coefficient (cont)

- 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, exploitation
  $\kappa \approx 1$: slow convergence, exploration
- effectively equivalent to inertia weight for specific χ:
  $w = \chi, \phi_1 = \chi c_1 r_1$ and $\phi_2 = \chi c_2 r_2$

Aspects of Basic PSO
Iteration Strategies

- Synchronous interation strategy
  - personal best and neighborhood bests updated separately from position and velocity vectors
  - slower feedback of new best positions
- Asynchronous iteration strategy
  - new best positions updated after each particle position update
  - immediate feedback of new best positions
  - lends itself well to parallel implementation

Aspects of Basic PSO
Iteration Strategies (cont)

Synchronous Iteration Strategy

Create and initialize the swarm;
repeat
    for *each particle* **do**
        Evaluate particle's fitness;
        Update particle's personal best position;
        Update particle's neighborhood best position;
    end
    for *each particle* **do**
        Update particle's velocity;
        Update particle's position;
    end
until *stopping condition is true*;

Asynchronous Iteration Strategy

Create and initialize the swarm;
repeat
    for *each particle* **do**
        Update the particle's velocity;
        Update the particle's position;
        Evaluate particle's fitness;
        Update the particle's personal best position;
        Update the particle's neighborhood best position;
    end
until *stopping condition is true*;

Aspects of Basic PSO
Acceleration Coefficients

- $c_1 = c_2 = 0$?
- $c_1 > 0, c_2 = 0$:
  - particles are independent hill-climbers
  - local search by each particle
  - cognitive-only PSO
- $c_1 = 0, c_2 > 0$:
  - swarm is one stochastic hill-climber
  - social-only PSO
- $c_1 = c_2 > 0$:
  - 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

Aspects of Basic PSO
Acceleration Coefficients (cont)

- low c_1 and c_2:
  - smooth particle trajectories
- high c_1 and c_2:
  - more acceleration, abrupt movements
- problem dependent
  - adaptive acceleration coefficients [35]

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

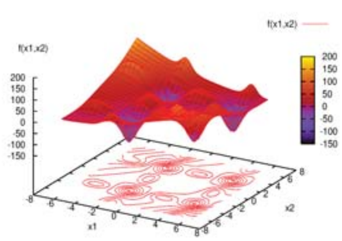
$$c_2(t) = (c_{2,max} - c_{2,min})\frac{t}{n_t} + c_{2,min}$$

Some Basic Applications of PSO
Function Optimization

Minimize the 2-D Bird function

$$f(\mathbf{x}) = \sin(x_1)e^{(1-cos(x_2))^2} + \cos(x_2)e^{(1-sin(x_1))^2} + (x_1 - x_2)^2$$

with $x_j \in [-2\pi, 2\pi]$



Some Basic Applications of PSO
Training A Feedforward Neural Network

- Objective is to find weight and bias values that minimizes an error function, e.g. sum-squared error
- Representation: particle represents weight vector and biases
- Fitness function: Sean-squared error, classification error
- Initialization:
  - Small initial weights to prevent velocity from growing too fast
  - Zero initial velocity, to start with as small as possible step sizes
  - Small V_{max} to prevent too fast growth in velocity

Some Basic Applications of PSO
Data Clustering

- Objective is to find centroids such that intra-cluster distances are minimized and inter-cluster distances maximized
- Representation of centroid vectors:

$$\mathbf{x}_i = (\mathbf{m}_{i1}, \ldots, \mathbf{m}_{ik}, \ldots, \mathbf{m}_{iK})$$

- Fitness function: Quantization error

$$J_{e,i} = \frac{\sum_{k=1}^{K}[\sum_{\forall \mathbf{z}_p \in C_{ki}} \mathcal{E}(\mathbf{z}_p, \mathbf{m}_{ki})]/n_{ki}}{K}$$

PSO Issues
About Convergence

Particles are guaranteed under certain conditions to converge to an equilibrium [8, 40, 9]:

- Particles will converge to

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

- This is not necessarily even a local minimum
- It has been proven that standard PSO is not a local minimizer [10]

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,

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

PSO Issues
Roaming Particles

- Empirical analysis [15] and theoretical proofs [17] showed that particles leave search boundaries very early during the optimization process
- Potential problems:
  - **Infeasible solutions**: Should better positions be found outside of boundaries, and no boundary constraint method employed, personal best and neighborhood best positions are pulled outside of search boundaries
  - **Wasted search effort**: Should better positions not exist outside of boundaries, particles are eventually pulled back into feasible space.
  - **Incorrect swarm diversity calculations**: As particles move outside of search boundaries, diversity increases

PSO Issues
Roaming Particles (cont)

Goal of this experiment: To illustrate

- particle roaming behavior, and
- infeasible solutions may be found

Experimental setup:

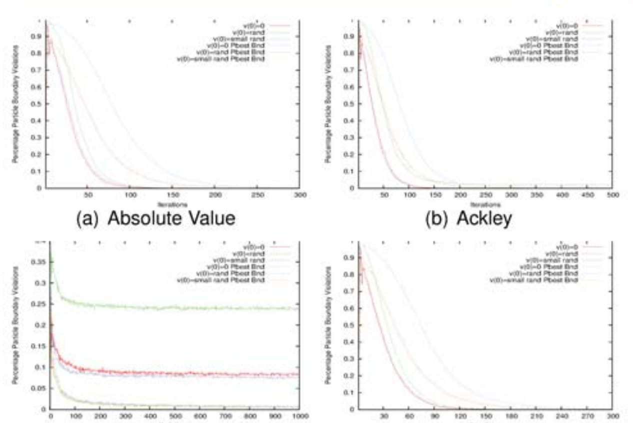
- A standard gbest PSO was used
- 30 particles
- $w = 0.729844$
- $c_1 = c_2 = 1.496180$
- Memory-based global best selection
- Synchronous position updates
- 50 independent runs for each initialization strategy

PSO Issues
Roaming Particles (cont)

: Functions Used for Empirical Analysis to Illustrate Roaming Behavior

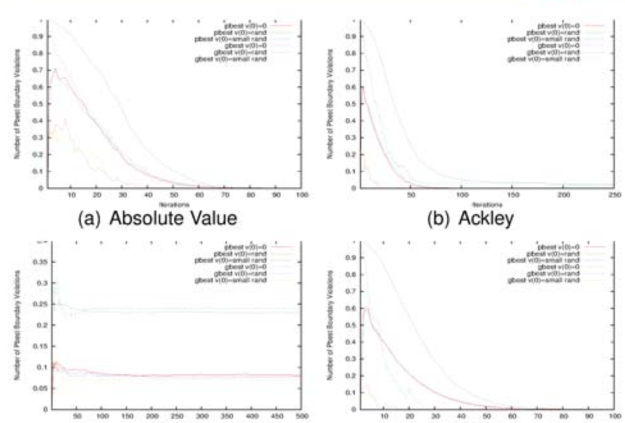
Function	Definition	Domain
AbsValue	$f(\mathbf{x}) = \sum_{j=1}^{n_x} \lvert x_j \rvert$	[-100,100]
Ackley	$f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n_x}\sum_{j=1}^{n_x} x_j^2}} - e^{\frac{1}{n_x}\sum_{j=1}^{n_x}\cos(2\pi x_j)} + 20 + e$	[-32.768,32.768]
Bukin 6	$f(\mathbf{x}) = 100\sqrt{\lvert x_2 - 0.01x_1^2 \rvert} + 0.01\lvert x_1 + 10 \rvert$	[-15,5],[-3,3]
Griewank	$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{j=1}^{n_x} x_j^2 - \prod_{j=1}^{n_x}\cos\left(\frac{x_j}{\sqrt{j}}\right)$	[-600,600]
Quadric	$f(\mathbf{x}) = \sum_{l=1}^{n_x}\left(\sum_{j=1}^{l} x_j\right)^2$	[-100,100]
Rastrigin	$f(\mathbf{x}) = 10n_x + \sum_{j=1}^{n_x}\left(x_j^2 - 10\cos(2\pi x_j)\right)$	[-5.12,5.12]
Rosenbrock	$f(\mathbf{x}) = \sum_{j=1}^{n_x-1}\left(100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2\right)$	[-2.048,2.048]

PSO Issues: Roaming
Percentage Particles that Violate Boundaries



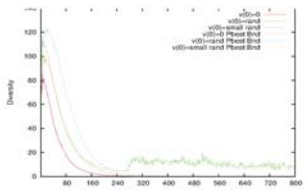
(a) Absolute Value (b) Ackley (c) Bukin 6 (d) Griewank

PSO Issues: Roaming
Percentage Best Position Boundary Violations

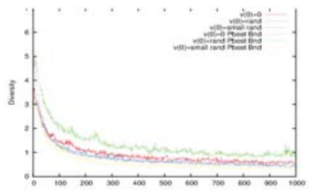


(a) Absolute Value (b) Ackley (c) Bukin 6 (d) Griewank

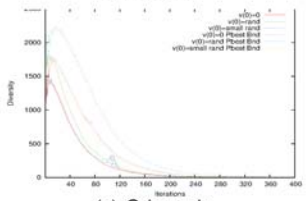
PSO Issues: Roaming
Diversity Profiles



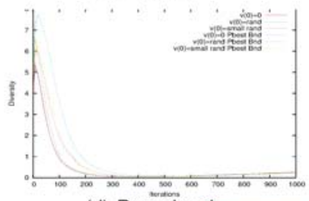
(a) Ackley



(b) Bukin 6



(c) Griewank



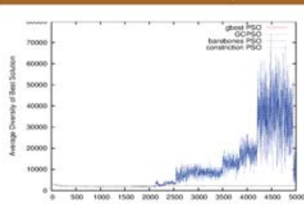
(d) Rosenbrock

PSO Issues: Roaming
Finding Infeasible Solutions

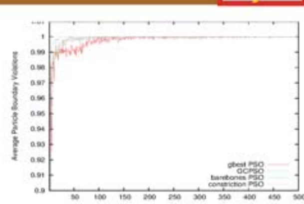
: Functions Used for Empirical Analysis to Illustrate Finding of Infeasible Solutions

Function	Domain	Function Definition
Ackley	[10,32.768]	$f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n_x} x_j^2}} - e^{\frac{1}{n}\sum_{j=1}^{n_x}\cos(2\pi x_j)} + 20 + e$
Ackley^{SR}	[-32.768,0]	$f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n_x} z_j^2}} - e^{\frac{1}{n}\sum_{j=1}^{n_x}\cos(2\pi z_j)} + 20 + e$
Eggholder	[-512,512]	$f(\mathbf{x}) = \sum_{j=1}^{n_x-1}\left(-(x_{j+1}+47)\sin(\sqrt{\lvert x_{j+1}+x_j/2+47\rvert}) + \sin(\sqrt{\lvert x_j-(x_{j+1}+47)\rvert})(-x_j)\right)$
Griewank^{SR}	[0,600]	$f(\mathbf{x}) = 1 + \frac{1}{4000}\sum_{j=1}^{n_x} z_j^2 - \prod_{j=1}^{n_x}\cos\left(\frac{z_j}{\sqrt{j}}\right)$
Norwegian^S	[-1.1,1.1]	$f(\mathbf{x}) = \prod_{j=1}^{n_x}\left(\cos(\pi z_j^3)\left(\frac{99+z_j}{100}\right)\right)$
Rosenbrock^S	[-30,30]	$f(\mathbf{x}) = \sum_{j=1}^{n_x-1}\left(100(z_{j+1}-z_j^2)^2 + (z_j-1)^2\right)$
Schwefel1.2^S	[0,100]	$f(\mathbf{x}) = \sum_{j=1}^{n_x}\left(\sum_{k=1}^{j} z_k\right)^2$
Schwefel2.26	[-50,50]	$f(\mathbf{x}) = -\sum_{j=1}^{n_x}\left(x_j\sin\left(\sqrt{\lvert x_j\rvert}\right)\right)$
Spherical^S	[0,100]	$f(\mathbf{x}) = \sum_{j=1}^{n_x} z_j^2$
Salomon	[-100,5]	$f_{14}(\mathbf{x}) = -\cos(2\pi\sum_{j=1}^{n_x} x_j^2) + 0.1\sqrt{\sum_{j=1}^{n_x} x_j^2} + 1$

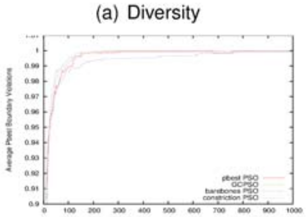
PSO Issues: Roaming
Finding Infeasible Solutions: Ackley



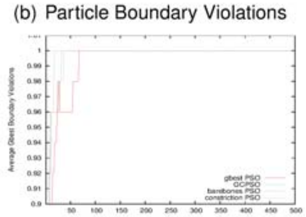
(a) Diversity



(b) Particle Boundary Violations

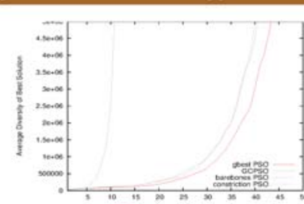


(c) Personal Best Boundary Viola-

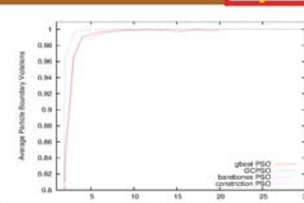


(d) Global Best Boundary Violations

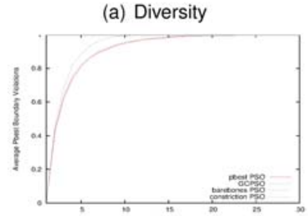
PSO Issues: Roaming
Finding Infeasible Solutions: Eggholder



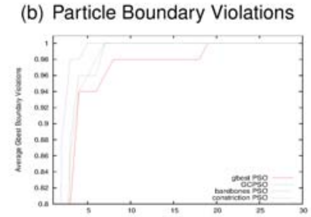
(a) Diversity



(b) Particle Boundary Violations



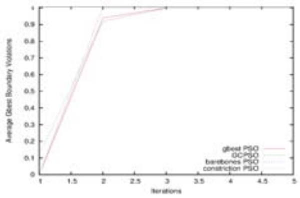
(c) Personal Best Boundary Viola-



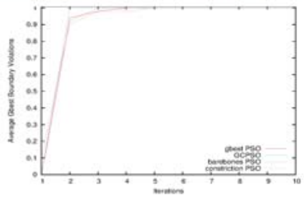
(d) Global Best Boundary Violations

PSO Issues: Roaming

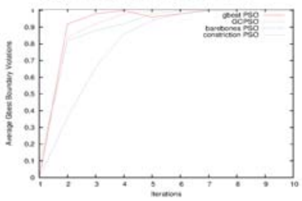
Finding Infeasible Solutions: gbest Boundary Violations



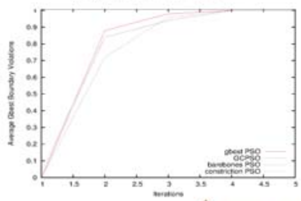
(a) Shifted Schwefel 1.2



(b) Schwefel 2.26



(c) Salomon



(d) Shifted Spherical

PSO Issues

Velocity Initialization

Velocties have been initialized using any of the following:

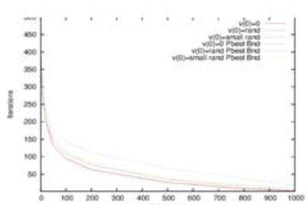
- $\mathbf{v}_i(0) = \mathbf{0}$
  - Critique: Limits exploration ability, therefore extent to which the search space is initially covered
  - Counter argument: Initial positions are uniformly distributed
  - Flocking analogy: Physical objects, in their initial state, do not have any momentum
- $\mathbf{v}_i(0) \sim U(-x_{min}, x_{max})^{n_x}$, where n_x is the problem dimension
  - Argument in favor: Initial random velocities help to improve exploration abilities of the swarm, therefore believed to obtain better solutions, faster
  - Argument against: large initial step sizes cause particles to leave search boundaries:

$$\mathbf{v}_i(0) \sim U(-x_{min}, x_{max})^{n_x} \longrightarrow \mathbf{x}_i(1) \sim U(-2x_{min}, 2x_{max})^{n_x}$$

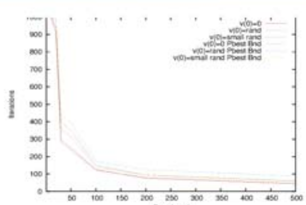
- Initialize to small random values

PSO Issues: Velocity Initialization

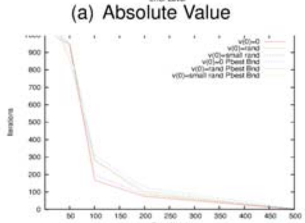
Fitness Reduction Profiles



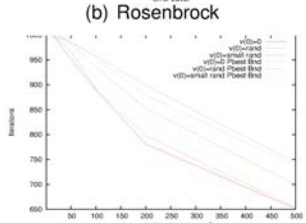
(a) Absolute Value



(b) Rosenbrock



(c) Rastrigin



(d) Quadric

PSO Issues: Velocity Initialization

Fitness After 1000 Iterations

Function	Zero Init No Pbest Bound	Random Init No Pbest Bound
Absolute Value	3.53E-001±2.87E+000	2.46E-001±1.47E+000
Ackley	2.49E+000±1.35E+000	2.68E+000±2.67E+000
Bukin 6	6.20E-002±4.50E-002	6.65E-002±5.56E-002
Griewank	3.72E-002±5.26E-002	3.91E-002±5.57E-002
Quadric	9.04E+001±8.70E+001	1.80E+002±3.15E+002
Rastrigin	6.66E+001±1.71E+001	7.37E+001±2.16E+001
Rosenbrock	2.65E+001±1.53E+001	2.73E+001±1.66E+001

PSO Issues: Velocity Initialization
Observations

The following general observations are made:

- Small random initialization and zero initialization have similar behaviors
- Random initialization
  - slower in improving fitness of best solution
  - resulted in larger diversity
  - had more roaming particles, roaming for longer
  - significantly more best positions left boundaries
  - took longer to reduce number of particle and best position violations
  - very slow in increasing number of converged dimensions

PSO Issues
gbest PSO versus lbestPSO

Current opinions about gbest PSO:

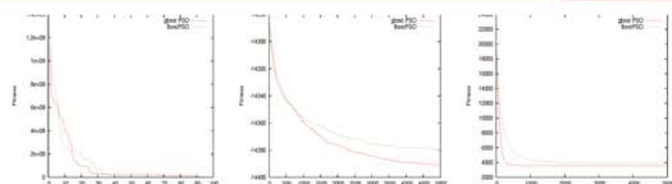
- Gbest PSO should not be used due to premature convergence to local optima as observed for a number of optimization problems [19, 24, 27, 32, 37]
- Gbest PSO converges fast due to the faster transfer of the best position throughout the swarm, and therefore the strong attraction to one best position [2, 13, 14, 19, 24, 25, 27, 28, 29]
- Gbest PSO is more susceptible to being trapped in local minima than lbest PSO [13, 14, 25].
- Gbest PSO is best suited to unimodal problems and should not be used for multimodal problems [2, 7, 21, 24, 32]
- Gbest PSO does not perform well for non-separable problems [25].

PSO Issues
gbest PSO versus lbestPSO (cont)

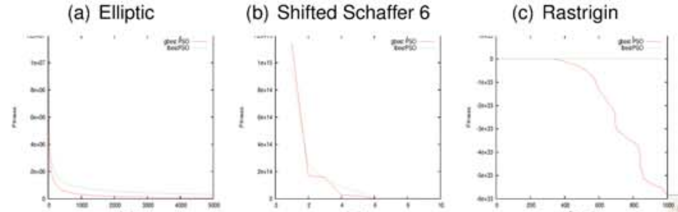
: Outcomes of Statistical Analysis Comparing Gbest with Lbest PSO

Function Class		Number of Functions	Accuracy			Success Rate			Efficiency			Diversity		
			>	=	<	>	=	<	>	=	<	>	=	<
UM	Seperable	7	5	0	2	6	0	1	2	0	5	5	0	2
	Non-separable	3	2	1	0	2	1	0	2	1	0	2	0	1
	Noisy	2	1	0	1	1	1	0	2	0	0	1	0	1
	Shifted	5	2	3	0	2	3	0	2	3	0	1	0	4
	Rotated	1	1	0	0	1	0	0	0	1	0	0	0	1
MM	Seperable	6	1	2	3	2	2	2	3	1	2	6	0	0
	Non-seperable	9	4	1	4	3	4	2	4	3	2	1	0	8
	Shifted	10	3	4	3	5	5	0	8	1	1	1	0	9
	Rotated	4	0	3	1	1	2	1	2	1	1	0	0	4
	Noisy	1	0	1	0	0	1	0	0	1	0	0	0	1
	Composition	11	1	2	8	0	4	7	1	5	5	0	0	11
Overall Total		59	20	17	22	23	23	13	26	17	16	11	0	48
Overall Unimodal		18	11	4	3	12	5	1	8	5	5	9	0	9
Overall Multimodal		41	13	19	14	11	18	12	18	12	11	2	0	39
Overall Seperable		17	7	4	6	9	5	3	12	1	4	5	0	12
Overall Non-seperable		42	13	13	16	14	18	10	11	16	9	6	0	36

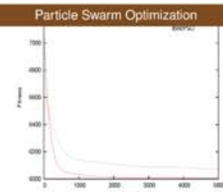
PSO Issues: gbest vs lbest
Fitness Profiles



(a) Elliptic (b) Shifted Schaffer 6 (c) Rastrigin



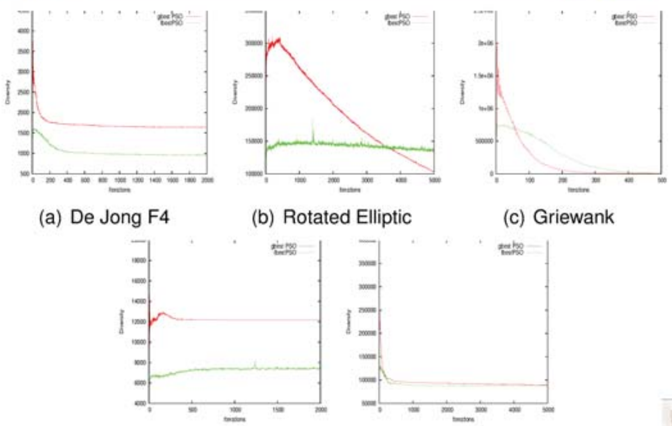
(d) Noisy Shifted Schwefel 1.2 (e) Schwefel 2.22 (f) Shubert



(g) Shifted Rotated Weierstrass

PSO Issues: gbest vs lbest

Diversity Profiles



(a) De Jong F4 (b) Rotated Elliptic (c) Griewank

(d) Rastrigin (e) Shifted Rosenbrock

PSO Issues: gbest vs lbest

Observations

- None of the star or ring topologies can be considered outright best for any of the main function classes
- Very similar performance over 60 functions with respect to solution accuracy
- gbest PSO performed slightly better than lbets PSO with respect to success rate and efficiency
- lbest PSO is slightly more consistent than gbest PSO
- Which topology is best, is function specific

PSO Issues

Iteration Strategies

- Should a synchronous iteration strategy (SIS) or and asynchronous iteration strategy (AIS) be used?
- General opinions:
  - AIS is generally faster and less costly than SIS [4, 26, 20, 33, 38]
  - AIS generally provides better results [26, 20, 33, 38]
  - AIS is better suited for lbest PSO, while SIS is better for gbest PSO [4]
- Recently, it was shown that SIS generally yields better results than AIS, specifically unimodal functions, and equal to AIS or better for multimodal functions [34]
- It was also recently stated that the choice of iteration strategy is very function dependent [45]

PSO Issues: Iteration Strategies

Accuracy Scores

: Ranks based on Final Fitness Values

Function Class		Number of Functions	gbest PSO			lbest PSO			GCPSO			BBPSO		
			>	=	<	>	=	<	>	=	<	>	=	<
UM	Sep	7	0	0	7	0	1	6	0	0	7	0	1	6
	Non-sep	3	1	1	1	0	2	1	0	2	1	0	3	0
	Noisy	2	0	0	2	1	1	0	1	0	1	1	0	1
	Shifted	5	0	5	0	0	4	1	0	5	0	0	5	0
	Rotated	1	0	0	1	0	0	1	0	0	1	0	1	0
MM	Sep	6	0	5	1	0	6	0	0	4	2	0	6	0
	Non-sep	9	0	7	2	0	9	0	1	7	1	0	9	0
	Shifted	10	2	6	2	0	10	0	1	7	2	1	8	1
	Rotated	4	0	1	3	0	4	0	1	0	3	1	1	2
	Noisy	1	1	0	0	0	1	0	1	0	0	1	0	0
	Composition	11	7	4	0	0	11	0	7	3	1	10	0	1
Overall Total		59	11	29	19	1	49	9	12	28	19	14	34	11
Overall UM		18	1	6	11	1	8	9	1	7	10	1	10	7
Overall MM		41	10	23	8	0	41	0	11	21	9	13	24	4
Overall Sep		17	1	7	9	1	10	6	0	7	10	0	10	7
Overall Non-sep		42	10	23	9	0	39	3	12	21	9	13	25	4

PSO Issues: Iteration Strategies
Observation

- Unimodal functions: AIS better accuracy for most functions
- Multimodal functions:
  - No significant difference for most of the functions
  - For the remainder of the functions, no clear winner
  - For lbest PSO not significant difference over all functions – insensitive to iteration strategy
- Separable functions: SIS not the preferred strategy for most of the functions
- Non-separable:
  - AIS bad for BBPSO
  - For lbest PSO AIS slightly better than SIS
  - For gbest PSO, GCPSO, SIS slightly better
  - However, for most functions no significant difference

Particle Trajectories
Theoretical Results

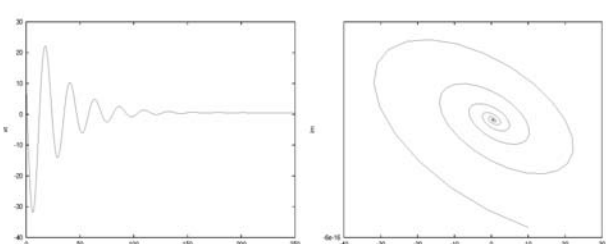
Simplified particle trajectories [41, 9]

- 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 9)
- Cyclic behavior (figure 10)
- Divergent behavior (figure 11)

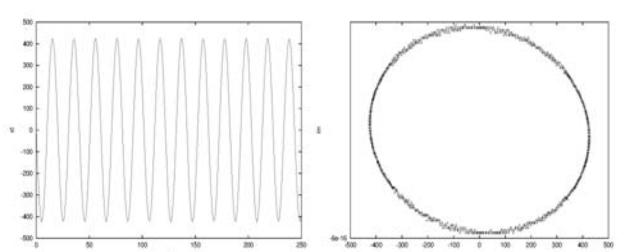
Particle Trajectories
Convergent Trajectory



(a) Time domain (b) Phase space

: $w = 0.5$ and $\phi_1 = \phi_2 = 1.4$

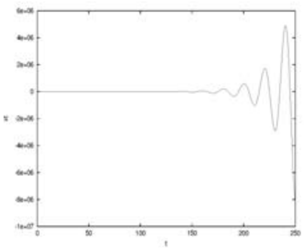
Particle Trajectories
Cyclic Trajectory



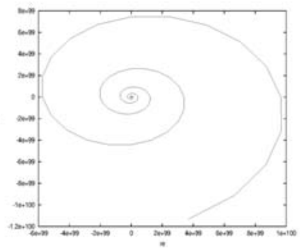
(a) Time domain (b) Phase space

: $w = 1.0$ and $\phi_1 = \phi_2 = 1.999$

Particle Trajectories
Divergent Trajectory



(a) Time domain

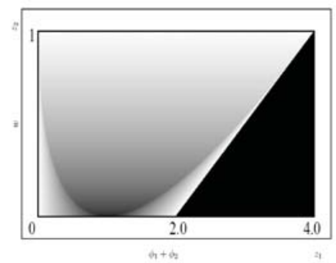


(b) Phase space

: $w = 0.7$ and $\phi_1 = \phi_2 = 1.9$

Particle Trajectories
Convergence Conditions

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



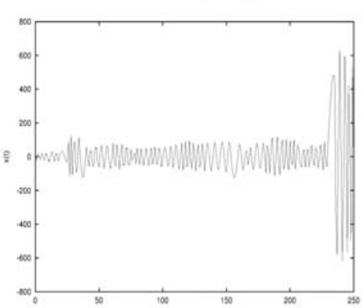
Convergence conditions on values of w, c_1 and c_2:

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

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

Particle Trajectories
Stochastic Trajectories

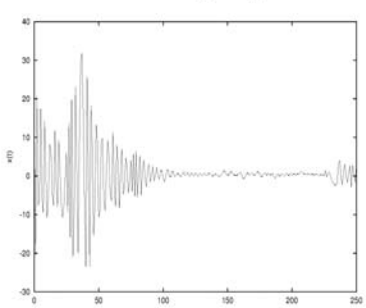
: $w = 1.0$ and $c_1 = c_2 = 2.0$



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

Particle Trajectories
Stochastic Trajectories (cont)

: $w = 0.9$ and $c_1 = c_2 = 2.0$



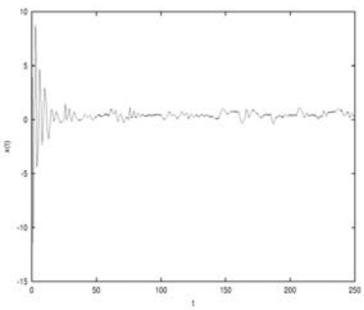
- violates the convergence condition
- for $w = 0.9$, $c_1 + c_2 < 3.8$ to validate the condition

What is happening here?

- since $0 < \phi_1 + \phi_2 < 4$,
- and $r_1, r_2 \sim U(0, 1)$,
- $\text{prob}(c_1 + c_2 < 3.8) = \frac{3.8}{4} = 0.95$

Particle Trajectories
Good Convergent Parameter Choices

: $w = 0.7$ and $c_1 = c_2 = 1.4$



- validates the convergence condition

PSO as Universal Optimizer

- Many different classes of optimization problems exist, for example,
  - Discrete-valued versus continuous-valued
  - Boundary constrained versus constrained
  - Single versus multi-objective
  - Static versus dynamic and noisy
  - Large scale
  - Unimodal versus multimodal
- Original PSO was developed to solve boundary constrained, single-objective, static, continuous-valued optimization problems
- Can PSO be used to solve optimization problems of these different problem classes, without changing the main principles of PSO?

PSO as Universal Optimizer (cont)
Exploration vs Exploitation

- For each problem type, what are the issues, and how can PSO be adapted to address these issues, while still maintaining the behavioral principles of PSO?
- An issue that relates to all of these problems: Exploration–exploitation tradeoff
  - **exploration**
    - ability to explore the search space
    - need to maintain swarm diversity
  - **exploitation**
    - ability to concentrate the search around a promising area to refine a candidate solution
    - need ways to ensure that all particles converge on the same point

Discrete-Valued Optimization Problems
Binary PSO

- What is the problem?
  - PSO originally developed for optimizing continuous-valued variables
  - Uses vector algebra on floating-point vectors
- How to adapt PSO for binary-valued variables?
  - Binary PSO (binPSO) of Kennedy and Eberhart [23]
  - 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?

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

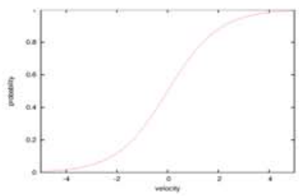
  - Then, position update changes to

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

Discrete-Valued Optimization Problems
Binary PSO (cont)

Issues:

- Interpretation of control parameters changes
  - w: small values facilitate longer exploration
  - V_{max}: smaller values promote exploration
- Initial velocities should be zero
- Velocities should never move to zero, but to $\pm\infty$
- Curse of dimensionality
- What happens if binary representations of consecutive numbers have a large Hamming distance?



: Sigmoid Function

Discrete-Valued Optimization Problems
Angle Modulated PSO

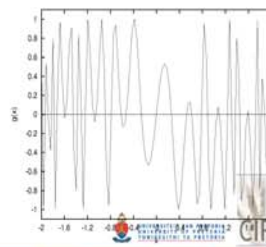
- Velocities and positions remain floating-point vectors
- Find a bitstring generating function to generate bitstring solution
- The generating function:

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

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

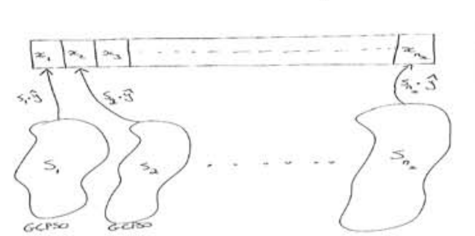


Discrete-Valued Optimization Problems
Angle Modulated PSO (cont)

Use a standard PSO to find the best values for these coefficients
Generate a swarm of 4-dimensional particles;
repeat
  Apply any PSO for one iteration;
  for *each particle* **do**
    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;
  end
until *a convergence criterion is satisfied*;

Large Scale Optimization Problems
Cooperative PSO

- Each particle is split into K separate parts of smaller dimension [41, 42, 43]
- Each part is then optimized using a separate sub-swarm
- If $K = n_x$, each dimension is optimized by a separate sub-swarm
- What are the issues?
  - Problem if there are strong dependencies among variables
  - How should the fitness of sub-swarm particles be evaluated?



Large Scale Optimization Problems
Cooperative PSO (cont)

$K_1 = n_x \bmod K$ and $K_2 = K - (n_x \bmod K)$;
Initialize K_1 $\lceil n_x/K \rceil$-dimensional and K_2 $\lfloor n_x/K \rfloor$-dimensional swarms;
repeat
  for *each sub-swarm* $S_k, k = 1, \ldots, K$ **do**
    for *each particle* $i = 1, \ldots, S_k.n_s$ **do**
      if $f(\mathbf{b}(k, S_k.\mathbf{x}_i)) < f(\mathbf{b}(k, S_k.\mathbf{y}_i))$ **then**
        $S_k.\mathbf{y}_i = S_k.\mathbf{x}_i$;
      end
      if $f(\mathbf{b}(k, S_k.\mathbf{y}_i)) < f(\mathbf{b}(k, S_k.\hat{\mathbf{y}}))$ **then**
        $S_k.\hat{\mathbf{y}} = S_k.\mathbf{y}_i$;
      end
    end
    Apply velocity and position updates;
  end
until *stopping condition is true*;

Multiple Solutions to Multimodal Problems

Niching capability of PSO:

- Can the *gbest* PSO find more than one solution?
  - Formal proofs showed that all particles converge to a weighted average of their personal best and global best positions
  - Therefore, only one solution can be found
  - What if we re-run the algorithm? No guarantee to find different solutions
- What about *lbest* PSO?
  - Neighborhoods may converge to different solutions
  - However, due to overlapping neighborhoods, particles are still attracted to one solution

Multiple Solutions to Multimodal Problems
Objective Function Stretching

Sequential niching, stretching the function to remove found minima [30, 31]

Create and initialize a n_x-dimensional swarm, S, and $\mathcal{X} = \emptyset$;
repeat
  Perform a single PSO iteration;
  if $f(S.\hat{\mathbf{y}}) \leq \epsilon$ **then**
    Isolate $S.\hat{\mathbf{y}}$;
    Perform a local search around $S.\hat{\mathbf{y}}$;
    if *a minimizer* $\mathbf{x}^*_{\mathcal{N}}$ *is found by the local search* **then**
      $\mathcal{X} \leftarrow \mathcal{X} \cup \{\mathbf{x}^*_{\mathcal{N}}\}$;
      Let $f(\mathbf{x}) \leftarrow H(\mathbf{x})$;
    end
  end
  Reinitialize the swarm S;
until *stopping condition is true*;
Return $\mathcal{X}$ as the set of multiple solutions;

Multiple Solutions to Multimodal Problems
NichePSO

Parallel niching PSO [3]

Create and initialize a n_x-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;

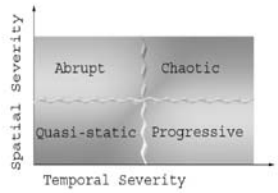
Dynamic Environments

- Objective: To find and track solutions in dynamically changing search spaces

$$\mathbf{x}^*(t) = \min_{\mathbf{x}} f(\mathbf{x}, \varpi(t))$$

where $\mathbf{x}^*(t)$ is the optimum found at time step t, and $\varpi(t)$ is a vector of time-dependent objective function control parameters

- Environment types:
  - Location of optima may change
  - Value of optima may change
  - Optima may disappear and new ones appear
  - Change frequencey
  - Change severity



: Environment Classes

Dynamic Environments
Consequences for PSO

- PSO can not be applied to dynamic environments without any changes to maintain swarm diversity
- Recall that particles converge to a weighted average of their personal best and global best positions
- At the point of convergence, $\mathbf{v}_i = 0$, and the contributions of the cognitive and social components are approximately zero
- New velocities are zero, therefore no change in position
- When the environment changes, personal best positions becomes stale, and will cause particles to be attrackted to old best positions
- Small inertia weight values limit exploration
- Velocity clamping limits exploration

Dynamic Environments
Consequences for PSO (cont)

- Environment change detection:
  - Optimization algorithm needs to react when a change is detected in order to increase diversity
  - Use sentry particles [5]
  - Gbest versus pbest versus arbitrary positions as sentries
- How to respond to environment changes?
  - Change the inertia update
    - $w \sim N(0.72, \sigma)$ [12], using no velocity clamping
    - If decreasing inertia is used, reset w to larger value

Dynamic Environments
Consequences for PSO (cont)

- Reinitialize particle positions [12]:
  - Reinitialize the entire swarm
  - Reinitialize parts of the swarm
  - Total reinitialization versus keeping previous personal best positions
- Limit memory
  - Reinitialize the personal best position to the particle's current position – only effective if swarm has not yet converged
  - Reset personal best positions only if significant change in fitness is observed
  - Recalculate global best after resetting personal best positions
- Do a local search around the previous optimum [46]

Dynamic Environments
Charged PSO

- Some particles attract one another, and others repell one another
- Velocity changes to

$$v_{ij}(t+1) = wv_{ij}(t)+c_1r_1(t)[y_{ij}(t)-x_{ij}(t)]+c_2r_2(t)[\hat{y}_j(t)-x_{ij}(t)]+a_{ij}(t)$$

where $\mathbf{a}_i$ is the particle acceleration, determining the magnitude of inter-particle repulsion [1]

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

- The repulsion force between particles i and l is

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

Dynamic Environments
Charged PSO (cont)

- $d_{il} = ||\mathbf{x}_i(t) - \mathbf{x}_l(t)||$
  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 repel from each other
- 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
- Acceleration is fixed at the core radius to prevent too severe repelling

Dynamic Environments
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
- Swarm contains
  - neutral particles following standard PSO updates
  - charged, or quantum particles, randomly placed within a multi-dimensional sphere

$$\mathbf{x}_i(t+1) = \begin{cases} \mathbf{x}_i(t) + \mathbf{v}_i(t+1) & \text{if } Q_i = 0 \\ \mathbf{B}_{\hat{\mathbf{y}}}(r_{cloud}) & \text{if } Q_i \neq 0 \end{cases}$$

Constrained Optimization Problems
Definition

Constrained optimization problem:

$$\begin{aligned} \text{minimize} \quad & f(\mathbf{x}), \quad \mathbf{x} = (x_1, \ldots, x_{n_x}) \\ \text{subject to} \quad & g_m(\mathbf{x}) \leq 0, \; m = 1, \ldots, n_g \\ & h_m(\mathbf{x}) = 0, \; m = n_g+1, \ldots, n_g+n_h \\ & x_j \in \text{dom}(x_j) \end{aligned}$$

where n_g and n_h are the number of inequality and equality constraints resp ectively

- Optimization problem is reformulated as

$$\text{minimize} \quad F(\mathbf{x}, t) = f(\mathbf{x}, t) + \lambda p(\mathbf{x}, t)$$

λ is the penalty coefficient
$p(\mathbf{x}, t)$ is the (possibly) time-dependent penalty function

- How to find the best penalty coefficients?
- And the penalty?

$$p(\mathbf{x}_i, t) = \sum_{m=1}^{n_g+n_h} \lambda_m(t) p_m(\mathbf{x}_i)$$

where

$$p_m(\mathbf{x}_i) = \begin{cases} \max\{0, g_m(\mathbf{x}_i)^\alpha\} & \text{if } m \in [1, \ldots, n_g] \\ |h_m(\mathbf{x}_i)|^\alpha & \text{if } m \in [n_g + 1, \ldots, n_g + n_h] \end{cases}$$

α is a positive constant, representing the power of the penalty

- Define the Lagrangian for the constrained problem
- The Lagrangian:

$$L(\mathbf{x}, \lambda_g, \lambda_h) = f(\mathbf{x}) + \sum_{m=1}^{n_g} \lambda_{gm} g_m(\mathbf{x}) + \sum_{m=n_g+1}^{n_g+n_h} \lambda_{hm} h_m(\mathbf{x})$$

$\lambda_g \in \mathbb{R}^{n_g}$ and $\lambda_h \in \mathbb{R}^{n_h}$ are the Lagrangian multipliers

- The new optimization problem (the primal problem):

$$\begin{array}{ll} \text{maximize}_{\lambda_g, \lambda_h} & L(\mathbf{x}, \lambda_g, \lambda_h) \\ \text{subject to} & \lambda_{gm} \geq 0, \; m = 1, \ldots, n_g + n_h \end{array}$$

- The vector $\mathbf{x}^*$ that solves the primal problem, as well as the Lagrange multiplier vectors, λ_g^* and λ_h^*, can be found by solving the min-max problem,

$$\min_{\mathbf{x}} \max_{\lambda_g, \lambda_h} L(\mathbf{x}, \lambda_g, \lambda_h)$$

- A coevolutionary PSO approach to solve the above min-max problem uses two swarms
- Swarm S_1 uses fitness function

$$f(\mathbf{x}) = \max_{\lambda_g, \lambda_h \in S_2} L(\mathbf{x}, \lambda_g, \lambda_h)$$

- Swarm S_2 uses fitness function

$$f(\lambda_g, \lambda_h) = \min_{\mathbf{x} \in S_1} L(\mathbf{x}, \lambda_g, \lambda_h)$$

Create and initialize two swarms, S_1 and S_2, where S_1 is n_x-dimensional and S_2 is $n_g + n_h$ dimensional;
repeat
  Run a PSO algorithm on swarm S_1 for $S_1.n_t$ iterations;
  Re-evaluate $S_2.\mathbf{y}_i(t), \forall i = 1, \ldots, S_2.n_s$;
  Run a PSO algorithm on swarm S_2 for $S_2.n_t$ iterations;
  Re-evaluate $S_1.\mathbf{y}_i(t), \forall i = 1, \ldots, S_1.n_s$;
until *stopping condition is true*;

Constrained Optimization Problems
Formulate as a MOP

- Reformulate constraints as additional sub-objective(s)
- Solve using a multi-objective PSO

Multi-Objective Optimization
Definition

Multi-objective problem:

$$\begin{array}{ll} \text{minimize} & \mathbf{f}(\mathbf{x}) \\ \text{subject to} & g_m(\mathbf{x}) \leq 0, \quad m = 1, \ldots, n_g \\ & h_m(\mathbf{x}) = 0, \quad m = n_g + 1, \ldots, n_g + n_h \\ & \mathbf{x} \in [\mathbf{x}_{min}, \mathbf{x}_{max}]^{n_x} \end{array}$$

where $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_{n_k}(\mathbf{x})) \in \mathcal{O} \subseteq \mathbb{R}^{n_k}$

$\mathcal{O}$ is referred to as the *objective space*
The search space, $\mathcal{S}$, is also referred to as the *decision space*

Multi-Objective Optimization
Issues

- Can PSO be used to simultaneously optimize more than one, possibly conflicting objective?
- How can PSO be used to find a set of solutions which optimally balances the trade-offs among these conflicting objectives?
- The task is to find a set of non-dominating solutions
- Formal definition of domination:
  A decision vector, $\mathbf{x}_1$ dominates a decision vector, $\mathbf{x}_2$ (denoted by $\mathbf{x}_1 \prec \mathbf{x}_2$), if and only if
  - $\mathbf{x}_1$ is not worse than $\mathbf{x}_2$ in all objectives, i.e. $f_k(\mathbf{x}_1) \leq f_k(\mathbf{x}_2), \forall k = 1, \ldots, n_k$, and
  - $\mathbf{x}_1$ is strictly better than $\mathbf{x}_2$ in at least one objective, i.e. $\exists k = 1, \ldots, n_k : f_k(\mathbf{x}_1) < f_k(\mathbf{x}_2)$

Multi-Objective Optimization
Aggregation-based Methods

- The objective is redefined as

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{n_k} \omega_k f_k(\mathbf{x}) \\ \text{subject to} & g_m(\mathbf{x}) \leq 0, \quad m = 1, \ldots, n_g \\ & h_m(\mathbf{x}) = 0, \quad m = n_g + 1, \ldots, n_g + n_h \\ & \mathbf{x} \in [\mathbf{x}_{min}, \mathbf{x}_{max}]^{n_x} \\ & \omega_k \geq 0, k = 1, \ldots, n_k \end{array}$$

  where $\sum_{k=1}^{n_k} \omega_k = 1$
- Problem with getting the best values for ω_k
- Has to be applied repeatedly to get more than one solution

A multi-swarm approach:

- Assume K sub-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 swarm $S_{(k+1) \mod K}$

- Assume two objectives

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

  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
  - Update personal best position only if the new particle position dominates the previous personal best position
- Alternative KTS: random

- Objective of archive is to keep track of all non-dominated solutions
- Non-dominated solutions added to archive after each iteration
- Fixed-sized archives versus unlimited sizes
- Local versus global guides

Let $t = 0$;
Initialize the swarm, $S(t)$, and archive, $A(t)$;
repeat
  Evaluate $(S(t))$;
  $A(t+1) \leftarrow$ Update$(S(t), A(t))$;
  $S(t+1) \leftarrow$ Generate$(S(t), A(t))$;
  $t = t+1$;
until *stopping condition is true*;

- Dynamically changing constraints, in
  - static and dynamic environments
  - single- and multi-objectives
- Tracking multiple optima in dynamic environments
- Dynamic multi-objective optimization problems

References I

[1] T.M. Blackwell and P.J. Bentley.
Dynamic Search with Charged Swarms.
In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 19–26, 2002.

[2] D. Bratton and J. Kennedy.
Defining a Standard for Particle Swarm Optimization.
In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 120–127, 2007.

[3] R. Brits, A.P. Engelbrecht, and F. van den Bergh.
Locating Multiple Optima using Particle Swarm Optimization.
Applied Mathematics and Computation, 189(2):1859–1883, 2007.

References II

[4] A. Carlisle and G. Dozier.
An Off-the-Shelf PSO.
In *Proceedings of the Workshop on Particle Swarm Optimization*, pages 1–6, 2001.

[5] A. Carlisle and G. Dozier.
Tracking Changing Extrema with Particle Swarm Optimizer.
Technical report, CSSE01-08, Auburn University, 2001.

[6] 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.

References III

[7] M. Clerc.
From Theory to Practice in Particle Swarm Optimization.
In *Handbook of Swarm Intelligence: Adaptation, Learning, and Optimization*, volume 8, pages 3–36. Springer, 2010.

[8] 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.

[9] F. Van den Bergh and A.P. Engelbrecht.
A Study of Particle Swarm Optimization Particle Trajectories.
Information Sciences, 176(8):937–971, 2006.

References IV

[10] F. Van den Bergh and A.P. Engelbrecht.
A Convergence Proof for the Particle Swarm Optimizer.
Fundamenta Informaticae, 105(4):341–374, 2010.

[11] R.C. Eberhart and J. Kennedy.
A New Optimizer using Particle Swarm Theory.
In *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, pages 39–43, 1995.

[12] R.C. Eberhart and Y. Shi.
Tracking and Optimizing Dynamic Systems with Particle Swarms.
In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 94–100, May 2001.

References V

[13] W. Elshamy, H.M. Emara, and A. Bahgat.
Clubs-based Particle Swarm Optimization.
In *Proceedings of the IEEE Swarm Intelligence Symposium*, 2007.

[14] A.P. Engelbrecht.
Fundamentals of Computational Swarm Intelligence.
Wiley, first edition, 2005.

[15] A.P. Engelbrecht.
Particle Swarm Optimization: Velocity Initialization.
In *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE Press, 2012.

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

CIRG

References VI

[17] S. Helwig and R. Wanka.
Theoretical Analysis of Initial Particle Swarm Behavior.
In *Proceedings of the Tenth International Conference on Parallel Problem Solving from Nature*, pages 889–898, 2008.

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

[19] T. Huang and A.S. Mohan.
Significance of Neighborhood Topologies for the Reconstruction of Microwave Images using Particle Swarm Optimization.
In *Proceedings of the Asia-Pacific Microwave Conference*, volume 1, 2005.

CIRG

References VII

[20] Luo J and Z. Zhang.
Research on the Parallel Simulation of Asynchronous Pattern of Particle Swarm Optimization.
Computer Simulation, 22(6), 2006.

[21] J. Kennedy.
Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance.
In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1931–1938, 1999.

[22] 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.

CIRG

References VIII

[23] J. Kennedy and R.C. Eberhart.
A Discrete Binary Version of the Particle Swarm Algorithm.
In *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, pages 4104–4109, 1997.

[24] J. Kennedy and R. Mendes.
Population Structure and Particle Performance.
In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1671–1676. IEEE Press, 2002.

[25] 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.

CIRG

References IX

[26] B-I. Koh, A. George, R. Haftka, and B.Fregly.
Parallel Asynchronous Particle Swarm Optimization.
International Journal for Numerical Methods and Engineering, 67:578–595, 2006.

[27] Y. Marinakis and M Marinaki.
Particle Swarm Optimization with Expanding Neighborhood Topology for the Permutation Flowshop Scheduling Problem.
Soft Computing, 2013.

[28] R. Mendes, J. Kennedy, and J. Neves.
Watch thy Neighbor or How the Swarm can Learn from its Environment.
In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 88–94, April 2003.

References X

[29] R. Mendes, J. Kennedy, and J. Neves.
The Fully Informed Particle Swarm: Simpler, Maybe Better.
IEEE Transactions on Evolutionary Computation, 8(3):204–210, 2004.

[30] K.E. Parsopoulos, V.P. Plagianakos, G.D. Magoulas, and M.N. Vrahatis.
Stretching Technique for Obtaining Global Minimizers through Particle Swarm Optimization.
In *Proceedings of the IEEE Workshop on Particle Swarm Optimization*, pages 22–29, 2001.

[31] K.E. Parsopoulos and M.N. Vrahatis.
Modification of the Particle Swarm Optimizer for Locating all the Global Minima.
In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 324–327, 2001.

References XI

[32] E.S. Peer, F. van den Bergh, and A.P. Engelbrecht.
Using Neighborhoods with the Guaranteed Convergence PSO.
In *Proceedings of the IEEE Swarm Intelligence Symposium*, pages 235–242. IEEE Press, 2003.

[33] J.R. Perez and J. Basterrechea.
Particle Swarm Optimization and Its Application to Antenna Farfield-Pattern Prediction from Planar Scanning.
Microwave and Optical Technology Letters, 44(5):398–403, 2005.

[34] J. Rada-Vilela, M. Zhang, and W. Seah.
A Performance Study on Synchronous and Asynchronous Updates in Particle Swarm Optimization.
In *Proceedings of the Genetic an Evolutionary Computation Conference*, pages 21–28, 2011.

References XII

[35] 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.

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

[37] M. Richards and D. Ventura.
Dynamic Sociemery in Particle Swarm Optimization.
In *Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing*, 2003.

References XIII

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

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

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

References XIV

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

[42] F. van den Bergh and A.P. Engelbrecht.
Cooperative Learning in Neural Networks using Particle Swarm Optimizers.
South African Computer Journal, 26:84–90, 2000.

[43] F. van den Bergh and A.P. Engelbrecht.
A Cooperative Approach to Particle Swarm Optimization.
IEEE Transactions on Evolutionary Computation, 8(3):225–239, 2004.

References XV

[44] 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.

[45] C.A. Voglis, K.E. Parsopoulos, and I.E. Lagaris.
Particle Swarm Optimization with Deliberate Loss of Information.
Soft Computing, 16(8):1373–1392, 1012.

[46] X. Zhang, L. Yu, Y. Zheng, Y. Shen, G. Zhou, L. Chen, L. Xi, T. Yuan, J. Zhang, and B. Yang.
Two-Stage Adaptive PMD Compensation in a 10 Gbit/s Optical Communication System using Particle Swarm Optimization Algorithm.
Optics Communications, 231(1-6):233–242, 2004.