



Particle Swarm Optimization in Dynamic Environments

Andries Engelbrecht

Computational Intelligence Research Group (CIRG)
Department of Computer Science
University of Pretoria
<http://cirg.cs.up.ac.za>

First Australian Computational Intelligence Symposium Series



- 1 Problem Statement
- 2 Basic Particle Swarm Optimization
- 3 Dynamic Environments
- 4 PSO in Dynamic Environments
- 5 Applications of Dynamic PSO
- 6 Conclusions



- The first particle swarm optimization (PSO) algorithms were developed to solve
 - ▶ unconstrained,
 - ▶ single-objective,
 - ▶ static, and
 - ▶ continuous-valued

optimization problems

- Basic PSO and many of its variants can not be effectively applied to find and track solutions in general dynamic environments
- PSO has to be adapted to increase exploration ability, and to maintain good levels of swarm diversity



The focus of this presentation is to show PSO can be

- adapted to find and track solutions in unconstrained, single-objective, dynamic environments
- used to train neural networks that experience concept drift
- adapted to track more than one solution in dynamic environments
- used to track Pareto fronts in dynamic multi-objective optimization
- applied to cluster temporal data



- PSO was developed by Kennedy and Eberhart in 1995
- 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
 - ▶ individuals follow a very simple behavior: emulate the success of neighboring individuals
 - ▶ emergent behavior: discovery of optimal regions in high dimensional search spaces



- 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

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

and \mathbf{v}_i is the velocity (step size)

- The search process:
 - ▶ Velocity updates per dimension

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

where

- ★ c_1, c_2 are positive acceleration coefficients
- ★ $r_{1j}(t), r_{2j}(t) \sim U(0, 1)$
- ★ y_i is the personal best position
- ★ \hat{y}_i is the neighborhood best position
- ★ $v_{ij}(0) = 0$ (usually, but can be random)



Create and initialize an n_x -dimensional swarm, S ;

repeat

for *each particle* $i = 1, \dots, S.n_s$ **do**

if $f(S.x_i) < f(S.y_i)$ **then**

$S.y_i = S.x_i$;

end

if $f(S.y_i) < f(S.\hat{y}_i)$ **then**

$S.\hat{y}_i = S.y_i$;

end

end

for *each particle* $i = 1, \dots, S.n_s$ **do**

 update the velocity and then the position;

end

until *stopping condition is true*;

- It has been proven that particles converge to a stable point

$$\frac{c_1 \mathbf{y}_i + c_2 \hat{\mathbf{y}}_i}{c_1 + c_2}$$

- Problem:
 - ▶ this point is not necessarily a minimum
 - ▶ may prematurely converge to a stable state
- 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,

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

- The above illustrates why the original PSO algorithms can not be applied to dynamic environments

Basic PSO: Convergent Trajectories

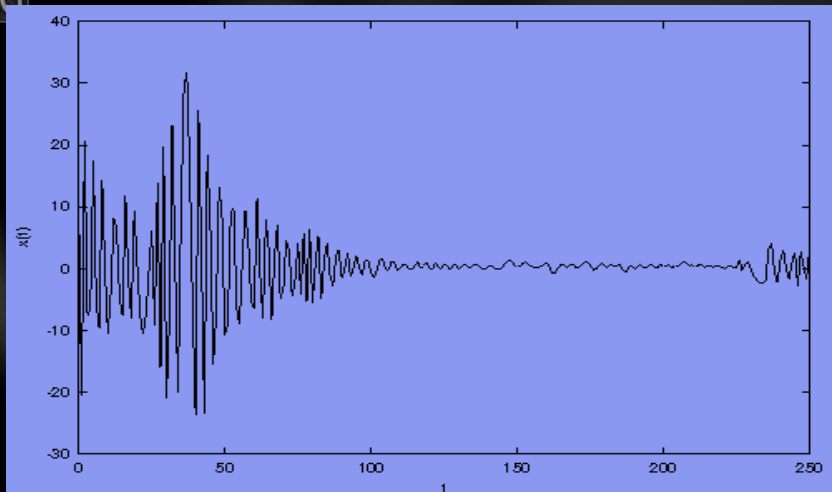


Figure: Stochastic Particle Trajectory for $w = 1.0$ and $c_1 = c_2 = 2.0$



Dynamic Environments: Formal Definition

Dynamic optimization problem:

$$\begin{aligned} & \text{minimize} && f(\mathbf{x}, \varpi(t)), \quad \mathbf{x} = (x_1, \dots, x_{n_x}), \varpi(t) = (\varpi_1(t), \dots, \varpi_{n_\varpi}) \\ & \text{subject to} && g_m(\mathbf{x}) \leq 0, \quad m = 1, \dots, n_g \\ & && h_m(\mathbf{x}) = 0, \quad m = n_g + 1, \dots, n_g + n_h \\ & && x_j \in \text{dom}(x_j) \end{aligned}$$

where $\varpi(t)$ is a vector of time-dependent objective function control parameters. The objective is to find

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

where $\mathbf{x}^*(t)$ is the optimum found at time step t .

Using the classification scheme of Eberhart and Shi:

- **Type I environments**, where the location of the optimum in problem space is subject to change. The change in the optimum, $\mathbf{x}^*(t)$ is quantified by the severity parameter, ζ , which measures the jump in location of the optimum.
- **Type II environments**, where the location of the optimum remains the same, but the value, $f(\mathbf{x}^*(t))$, of the optimum changes.
- **Type III environments**, where both the location of the optimum and its value changes.

Difficulty also determined by frequency of change and severity of change

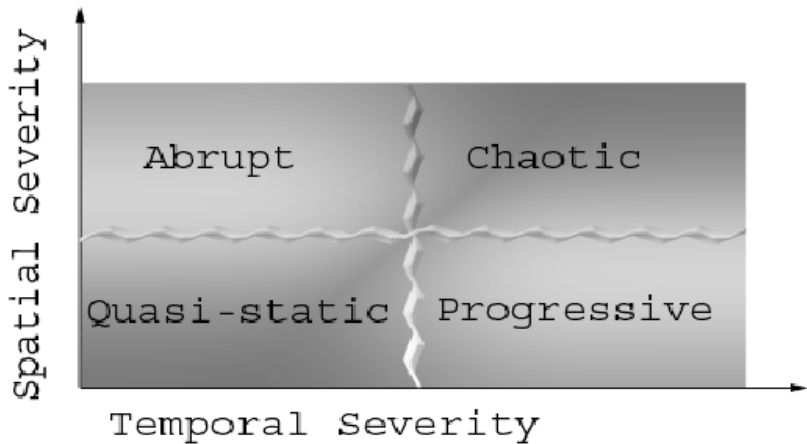


Figure: Classification Based on Temporal and Spatial Severity

$$f(\mathbf{x}, \varpi) = |f_1(\mathbf{x}, \varpi) + f_2(\mathbf{x}, \varpi) + f_3(\mathbf{x}, \varpi)|$$

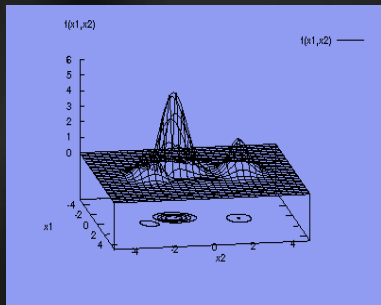
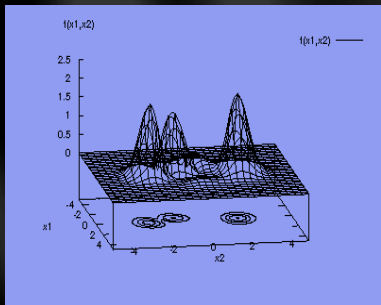
with

$$f_1(\mathbf{x}, \varpi) = \varpi_1(1 - x_1)^2 e^{-(x_1^2 - (x_2 - 1)^2)}$$

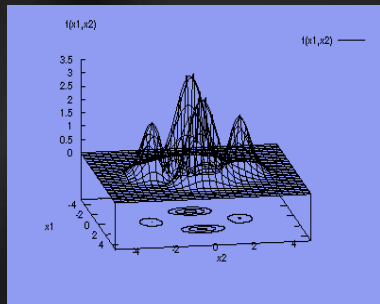
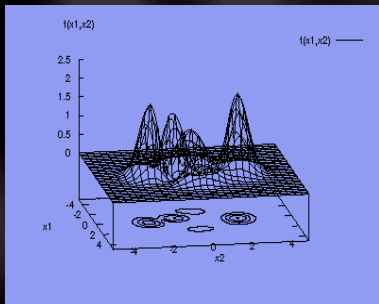
$$f_2(\mathbf{x}, \varpi) = -0.1 \left(\frac{x_1}{5} - \varpi_2 x_1^3 - x_2^5 \right)^2 e^{-(x_1^2 - x_2^2)}$$

$$f_3(\mathbf{x}, \varpi) = 0.5 e^{-(x_1 + 1)^2 - x_2^2}$$

Dynamic Environments: Illustration



Dynamic Environments: Illustration





PSO in Dynamic Environments: Issues

- Can PSO be used to track optima in dynamically changing landscapes?
- 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 attracted to old best positions
 - ▶ Small inertia weight values limit exploration
 - ▶ Velocity clamping limits exploration



PSO in Dynamic Environments: Some Solutions

- Environment change detection:
 - ▶ Optimization needs to react when a change is detected in order to increase diversity
 - ▶ Use sentry particles
- How to respond to environment changes?
 - ▶ Change the inertia update
 - ★ $w \sim N(0.72, \sigma)$, no velocity clamping
 - ★ If decreasing inertia is used, reset w to larger value
 - ▶ Reset velocity clamping to a larger value
 - ▶ Reset c_1 to larger value and c_2 to smaller value
 - ▶ Reinitialize particle positions
 - ★ Reinitialize the entire swarm
 - ★ Reinitialize parts of the swarm
 - ★ Total reinitialization versus keeping previous personal best positions
 - ▶ Do a local search around the previous optimum



PSO in Dynamic Environments: Charged Swarms

- Change the velocity update to maintain swarm diversity
- Have particles that attract one another, and others that repel one another
- Velocity update changes to

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

where \mathbf{a}_i is the particle acceleration

PSO in Dynamic Environments: Charged Swarms (cont)

- Acceleration determines the magnitude of inter-particle repulsion

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

- The repulsion force between particles i and l is

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

$$d_{il} = \|\mathbf{x}_i(t) - \mathbf{x}_l(t)\|$$

Q_i is the particle's charged magnitude

R_c is the core radius

R_p is the perception limit of each particle



PSO in Dynamic Environments: Quantum PSO

- Vaguely based on the model on an atom:
 - ▶ The orbiting electrons of an atom are replaced by a quantum cloud
 - ▶ Position of each electron is determined by a probability distribution and not its previous position or trajectory dynamics
- Quantum swarm:
 - ▶ A percentage of particles are treated as the quantum cloud
 - ▶ At each iteration, the cloud is randomized in a spherical area of radius r_{cloud} , centered around the *gbest* particle
 - ▶ Particles not part of the cloud behave according to the standard PSO
 - ▶ Randomization of the cloud at each iteration prevents convergence of the swarm, and maintains diversity, facilitating exploration
 - ▶ Non-quantum particles refine solutions, facilitating exploitation



Applications of Dynamic PSO: Content

- Neural networks with concept drift
- Track multiple solutions
- Dynamic multi-objective optimization
- Cluster temporal data



Applications of Dynamic PSO: Concept Drift in NN

- Concept drift in neural networks:
 - ▶ **concept drift** means that the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways
 - ▶ Considering classification problems, changes in the hidden context may cause class boundaries to shift over time
 - ▶ Concept drift can be
 - ★ gradual – decision boundaries shift over time
 - ★ abrupt – boundaries may be replaced by new ones
 - ▶ Learning algorithms should have the ability to track such changing decision boundaries



Applications of Dynamic PSO: Concept drift NN (cont)

- The algorithms used as NN training methods:
 - ▶ Standard back propagation with sigmoid activation functions
 - ▶ Charged PSO
 - ★ 50% of the particles charged, and the rest neutral
 - ★ $R_c = 1, R_p = 100$ and $Q = 0.3$
 - ▶ Quantum PSO,
 - ★ 50% of the swarm forming the quantum cloud
 - ★ $r_{cloud} = 2$
 - ▶ Swarm sizes of 30 particles
 - ▶ $w = 0.729844$ and $c_1 = c_2 = 1.496180, V_{max} = 2$



Applications of Dynamic PSO: Concept drift NN (cont)

Dynamic sphere problem:

- Hypersphere in \mathbb{R}^3 is used to classify patterns into two mutually exclusive classes

$$\sum_{i=1}^3 (x_i + c_i) = R^2$$

R is the radius and \mathbf{c} the center of the sphere

- Sphere was generated by randomizing (20 times)
 - ▶ $\mathbf{c} \in [0, 3]^3$
 - ▶ $R \in [0, 1]$
- For each sphere, 1000 random points, $\mathbf{x}_i \in [0, 1]^3$ were generated, giving 20000 patterns in total
- Class 1 are those patterns inside the sphere, and class 0 are patterns outside the sphere



Applications of Dynamic PSO: Concept drift NN (cont)

Sliding thresholds problem:

- Cartesian space was subdivided into three classes, using

$$\text{class}(x) = \begin{cases} A & \text{if } x \leq t_1 \\ B & \text{if } x \geq t_2 \\ C & \text{otherwise} \end{cases}$$

where $t_1 < t_2$

- The thresholds were changed 20 times by selecting $t_1 < t_2$ randomly from $[0, 1]$
- For each change in the threshold 1000 2-dimensional points were selected, with each $\mathbf{x}_i \in [0, 1]^2$
- Classification is based only of the first element of each vector



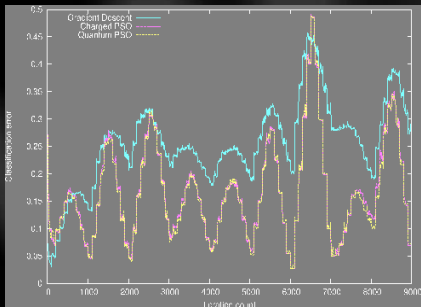
Applications of Dynamic PSO: Concept drift NN (cont)

Different dynamic environments tested:

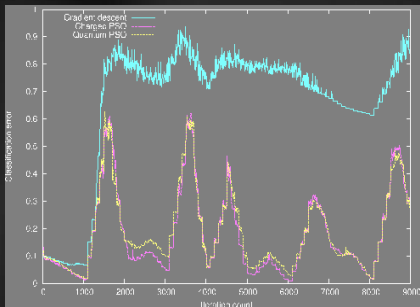
- **Scenario I:** A window of 1000 patterns was shifted by 100 patterns every 100 iterations of the training algorithm
- **Scenario II:** The step size is increased to 500 to simulate more drastic changes
- **Scenario III:** Step size is equal to the window size, meaning that all patterns in the window are replaced when a shift occurs.



Applications of Dynamic PSO: Concept drift NN (cont)



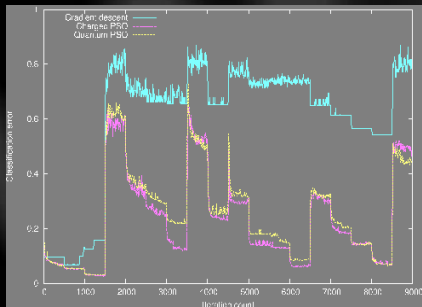
(e) Dynamic Sphere, Scenario I



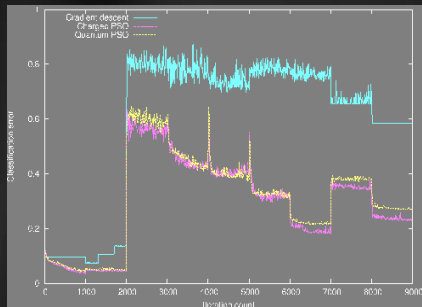
(f) Sliding Thresholds, Scenario I



Applications of Dynamic PSO: Concept drift NN (cont)



(g) Sliding Thresholds, Scenario II



(h) Sliding Thresholds, Scenario III



Applications of Dynamic PSO: Track Multiple Solutions

- Use the dynamic vector-based PSO, which is an extension of the vector-based PSO
- The vector-based PSO is a PSO niching algorithm with the ability to find multiple optima, including global and local optima

Algorithm: Enhanced Parallel Vector-Based PSO

Initialize the swarm;

Find niches and assign all particles to a niche;

Optimize and merge niches;



Applications of Dynamic PSO: Track Multiple Solutions

Step 1: Initialize Swarm

Create n_s particles;

Set *niche-id* of each particle to 0;

Initialize the granularity, g ;

for *each particle* **do**

 Calculate pbest position, $\mathbf{y}_i(t)$;

 Calculate $\mathbf{v}_{pi}(t) = \mathbf{y}_i(t) - \mathbf{x}_i(t)$;

end

Applications of Dynamic PSO: Track Multiple Solutions

Step 2: Find Niches

repeat

Set $\hat{\mathbf{y}}(t)$ to $\mathbf{y}_i(t)$ over particles with niche-id = 0;

for *each particle in the swarm* **do**

Calculate $\mathbf{v}_{gi} = \hat{\mathbf{y}}(t) - \mathbf{x}_i(t)$;

Calculate dot product, $\delta_i(t) = \mathbf{v}_{pi}(t) \bullet \mathbf{v}_{gi}(t)$;

Set radius $\rho_i(t)$ as distance between $\hat{\mathbf{y}}(t)$ and $\mathbf{x}_i(t)$;

end

Set niche radius to distance between $\hat{\mathbf{y}}(t)$ and closest particle with $\delta_i(t) < 0$;

for *each particle with $\rho_i <$ niche radius and $\delta_i > 0$* **do**

Set niche-id to the next niche number;

end

if *number of particles in niche $<$ 3* **then**

Create extra particles in niche to have at least 3 particles;

end

until *all particles have niche-id \neq 0*;

Applications of Dynamic PSO: Track Multiple Solutions

Step 3: Optimize and Merge Niches

for m times **do**

for k times **do**

for each particle **do**

 Create temporary particle $\mathbf{x}_{temp}(t)$;

 Calculate \mathbf{v}_{ptemp} , \mathbf{v}_{gtemp} and δ_{temp} ;

if $\delta_{temp} < 0$ **then**

 Retain original particle position and corresponding values;

end

else

 Set $\mathbf{x}_i(t) = \mathbf{x}_{temp}(t)$;

 Update $\mathbf{y}_i(t)$, $\hat{\mathbf{y}}(t)$, $\mathbf{v}_{pi}(t)$, $\mathbf{v}_{gi}(t)$, δ_i and ρ_i ;

end

end

end

 Merge Niches;

end



Applications of Dynamic PSO: Track Multiple Solutions

Algorithm: Dynamic Vector-Based PSO

Initialize the swarm;

Apply the vector-based PSO to find niches (optima);

while *function is still changing* **do**

if *function has changed* **then**

 Perform stage 1 to track existing optima;

 Perform stage 2 to explore to find new optima;

end

 Continue with the vector-based PSO;

end



Applications of Dynamic PSO: Track Multiple Solutions

- Stage 1: track existing optima
 - ▶ Discard all particle information, but keep neighborhood best position of each niche
 - ▶ For each niche, randomly create particles with a radius of the neighborhood best position
 - ▶ Update neighborhood best positions
- Stage 2: Locate new optima
 - ▶ Keep information about the best solution found by each niche
 - ▶ Discard all other particle information (except the new particles generated during stage 1)
 - ▶ Reinitialize particles within the entire search space



Applications of Dynamic PSO: Track Multiple Solutions

- Problems are generated using De Jong & Morrison's test function

$$f(x_1, x_2) = \max_{i=1, \dots, n} \left(H_i - R_i \sqrt{(x_1 - x_{1i})^2 + (x_2 - x_{2i})^2} \right)$$

where

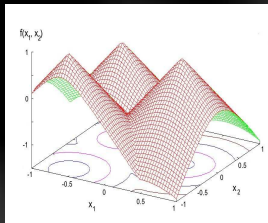
H_i is the height of each cone

R_i is the slope (x_{1i}, x_{2i}) is the position of the optimum

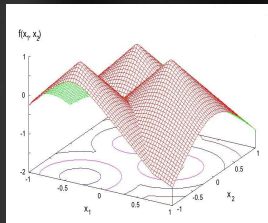
Applications of Dynamic PSO: Track Multiple Solutions



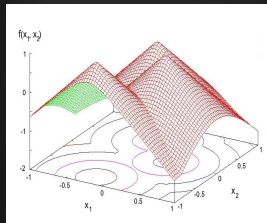
Scenario 1: 3 peaks which move around in the search space, 6 changes



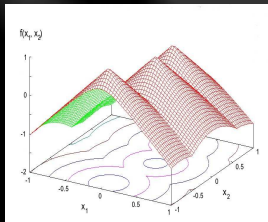
(i) Step 1



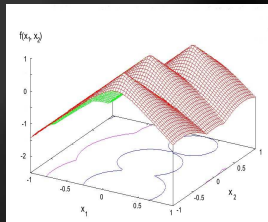
(j) Step 2



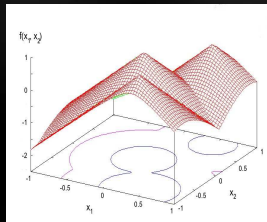
(k) Step 3



(l) Step 4



(m) Step 5



(n) Step 6



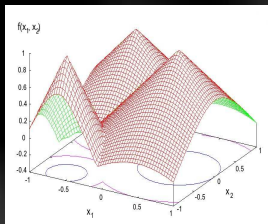
Applications of Dynamic PSO: Track Multiple Solutions

Step	Average		Average distance		Success rate
	evaluations	optima	from x_1	from x_2	
1	25168.4	3	0	1.77E-13	100%
2	6934.4	3	2.79E-19	3.65E-18	100%
3	6921.1	3	1.39E-19	4.32E-19	100%
4	6933.1	3	2.91E-17	1.76E-17	100%
5	6921.6	3	1.72E-19	1.22E-18	100%
6	6917.2	3	1.91E-17	1.91E-17	100%

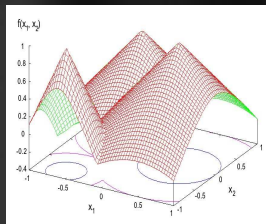
Applications of Dynamic PSO: Track Multiple Solutions



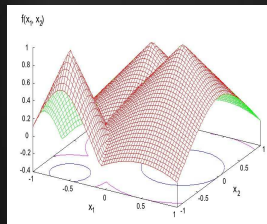
Scenario 2: 3 optima, 8 changes, 1 peak disappears to reappear later



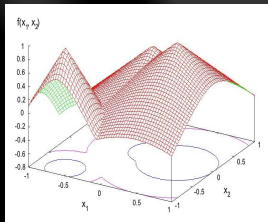
(o) Step 1



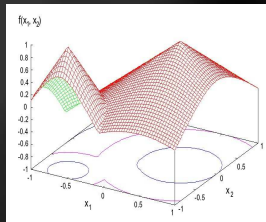
(p) Step 2



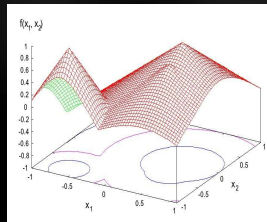
(q) Step 3



(r) Step 4



(s) Step 5



(t) Step 6



Applications of Dynamic PSO: Track Multiple Solutions

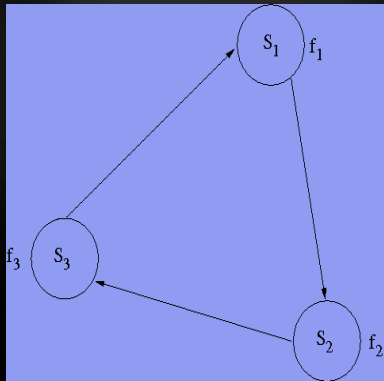
Step	Average		Average distance		Success rate
	evaluations	optima	from x_1	from x_2	
1	24474.0	3	4.07E-15	7.25E-16	100%
2	31665.5	3	5.47E-09	1.14E-09	100%
3	33905.7	3	1.24E-05	8.34E-07	93.33%
4	33378.9	3	3.92E-06	2.30E-07	80%
5	28050.3	2	3.41E-12	1.45E-12	100%
6	28324.2	3	3.21E-13	7.25E-07	84.33%

Dynamic multi-objective optimization problem:


$$\begin{aligned}
 &\text{minimize} && \mathbf{f}(\mathbf{x}, \varpi(t)), && \mathbf{x} = (x_1, \dots, x_{n_x}), \varpi(t) = (\varpi_1(t), \dots, \varpi_{n_\varpi}) \\
 &\text{subject to} && g_m(\mathbf{x}) \leq 0, && m = 1, \dots, n_g \\
 &&& h_m(\mathbf{x}) = 0, && m = n_g + 1, \dots, n_g + n_h \\
 &&& \mathbf{x} \in [\mathbf{x}_{min}, \mathbf{x}_{max}]^{n_x}
 \end{aligned}$$

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

- Use the vector evaluated PSO (VEPSO), but with a dynamic archive
- Each swarm optimizes only one objective function
- Information exchange is circular: one swarm uses global best position from neighboring swarm to update velocities
- Use sentry particles to detect environment change
- Update Pareto front after change to remove all dominated solutions



Applications of Dynamic PSO: Dynamic MOO



```
for number of iterations do
  if change occurred in environment then
    Re-evaluate solutions in Pareto front;
    Remove dominated solutions from Pareto front;
  end
  Perform one iteration of VEPSO;
  if new solutions are non-dominated then
    if space in archive then
      Add new solutions to Pareto front;
    end
    else
      Remove solutions from archive;
      Add new solutions to archive;
    end
  end
  Select new sentry particles;
end
```



Applications of Dynamic PSO: Dynamic MOO

The dynamic MOO, where the frequency of change, τ_t is 5: Minimize the function, FDA1:

$$\mathbf{f}(\mathbf{x}, t) = (f_1(x_I, t), g(\mathbf{x}_{II}, t)h(\mathbf{x}_{III}, f_i(x_I, t), g(\mathbf{x}_{II}, t), t))$$

where

$$f_1(x_I) = x_i$$

$$g(\mathbf{x}_{II}) = 1 + \sum_{x_i \in \mathbf{x}_{II}} (x_i - G(t))^2$$

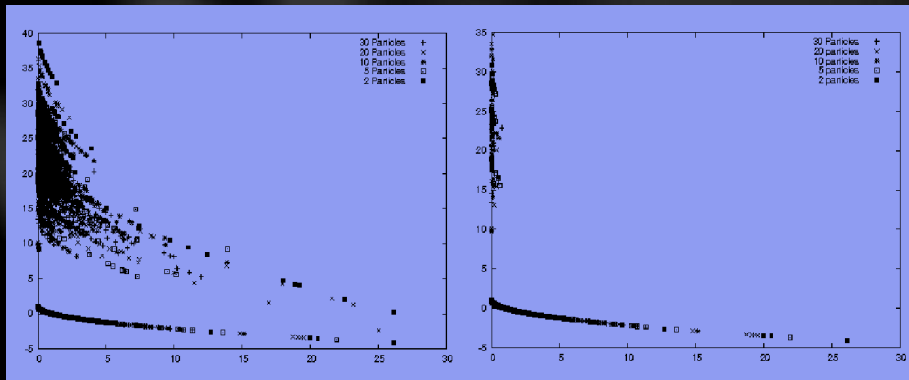
$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}}$$

$$G(t) = \sin(0.5\pi t), \quad t = \frac{1}{n_t} \left\lfloor \frac{\tau}{\tau_t} \right\rfloor$$

$$x_I \in [0, 1], \quad \mathbf{x}_{II} = (x_2, \dots, x_n) \in [-1, 1]$$



Applications of Dynamic PSO: Dynamic MOO



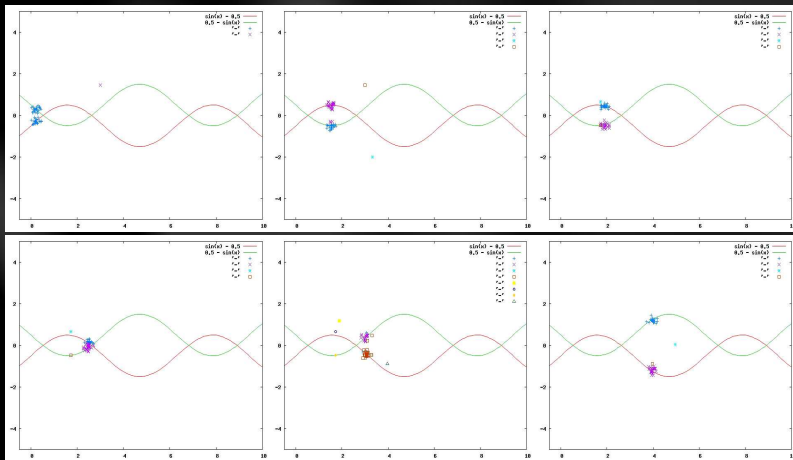


Applications of Dynamic PSO: Clustering Temporal Data

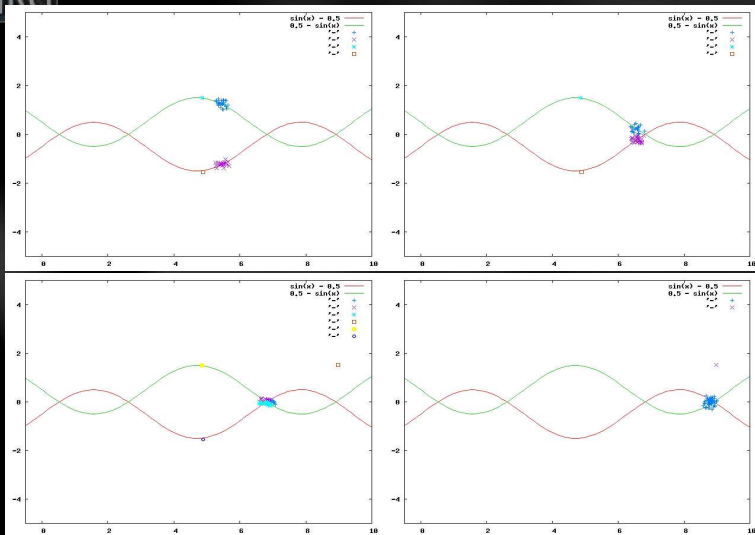
- Two clusters, with the center of each cluster moving on a sinus wave:
 - ▶ Cluster 1: $\sin(x) - 0.5$
 - ▶ Cluster 2: $0.5 - \sin(x)$
- At each time step, 1000 points are randomly created around the centroid
- The objective is to track these clusters over time
- Used the Charged PSO

Applications of Dynamic PSO: Clustering Temporal Data

CIRG

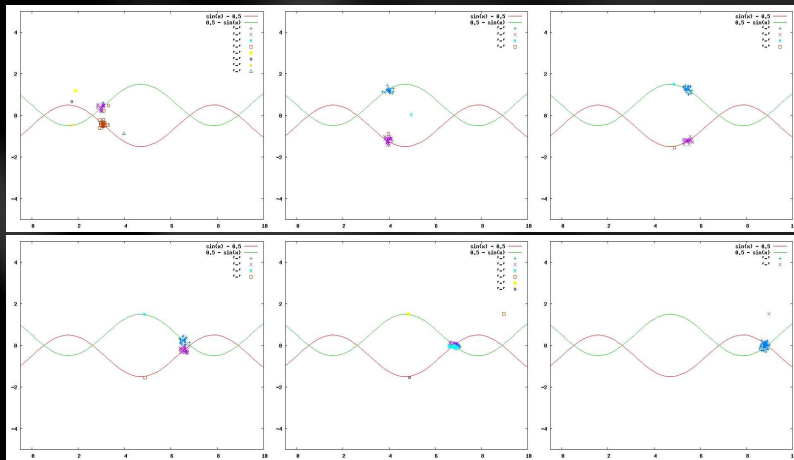


Applications of Dynamic PSO: Clustering Temporal Data



Applications of Dynamic PSO: Clustering Temporal Data

CIRG



- The basic PSO and most of its variations can not be used to find and track optima in dynamic environments
- PSO has to be adapted so that its exploration ability is improved and swarm diversity maintained
- Several succesfull PSO algoritms have been developed for dynamic environments
- This presentation illustrated that adaptations of PSO can be used to
 - ▶ train neural network classifiers under concept drift
 - ▶ locate and maintain multiple solutions in dynamically changing landscapes
 - ▶ solve dynamic multi-objective optimization problems
 - ▶ cluster temporal data