Advances in 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

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

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. He holds an A-rating from the South African National Research Foundation.

Presentation Outline I

Introduction

Particle swarm optimization (PSO):

- developed by Kennedy & Eberhart,
- 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

Introduction (cont)

Many variations of PSO have been developed, mainly to improve

- accuracy
- speed to convergence
- balance between exploration and exploitation

Focus of the above mainly on the class of problems for which PSO was developed, i.e.

- continuous-valued,
- single-objective,
- static, and
- boundary constrained

Some theoretical analyses of particle trajectories and convergence have been done to better understand PSO

Introduction (cont)

PSO versions were also developed to solve the following types of optimization problems:

- Discrete-valued (binary, integer, sets)
- Constrained (inequality and/or equality constraints, static and dynamic)
- Dynamic & noisy
- Multi-objective (static and dynamic, and many-objectives)
- Finding multiple optima (static and dynamic)
- Large-scale optimization problems

Other PSO developments include

- Self-adaptive PSO
- Heterogeneous PSO
- Inclusion in hyper-heuristics

Objectives of Tutorial

The main objective of this tutorial is to discuss recent advances in PSO, specifically

- recent studies to better understand PSO
  - with a focus on the PSO control parameters, and
  - particle search behavior
- recent PSO algorithms, with a focus on
  - Self-adaptive PSO
  - Heterogeneous PSO
  - Dynamic multi-objective optimization PSO
- recent developments not covered
  - Many-objective PSO
  - Set-based PSO
  - Dynamically changing constraints

What are the main components?

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

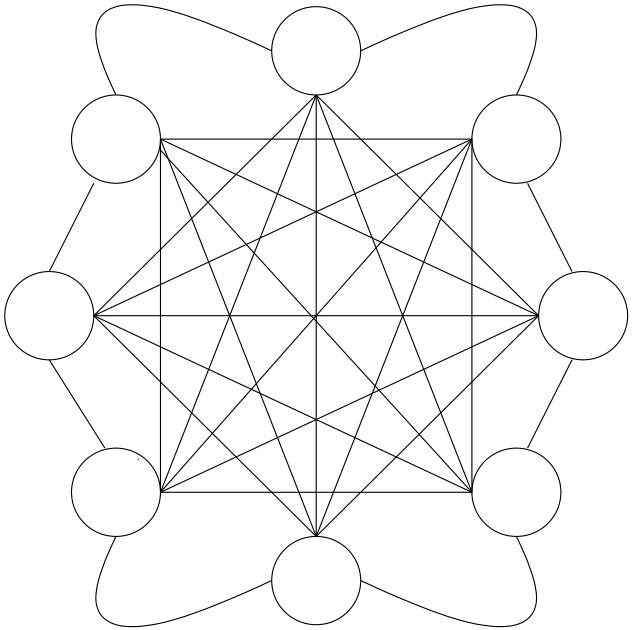
The search process:

- Position updates

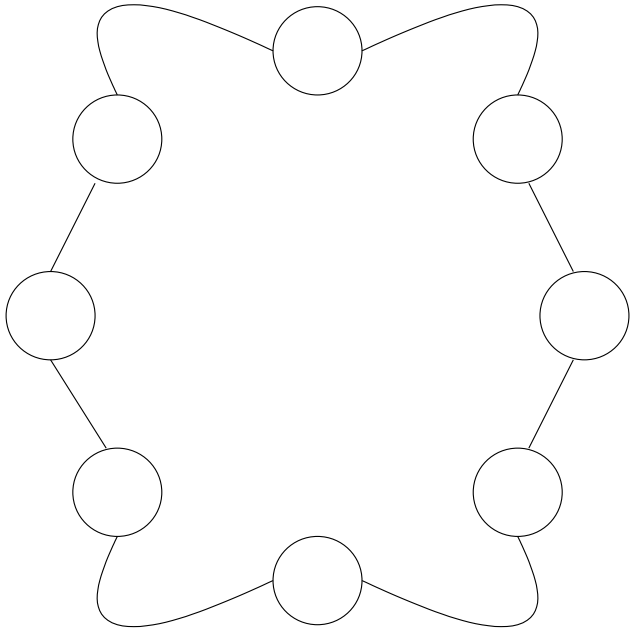
$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1), \quad \mathbf{x}_{ij}(0) \sim U(x_{min,j}, x_{max,j})$$

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

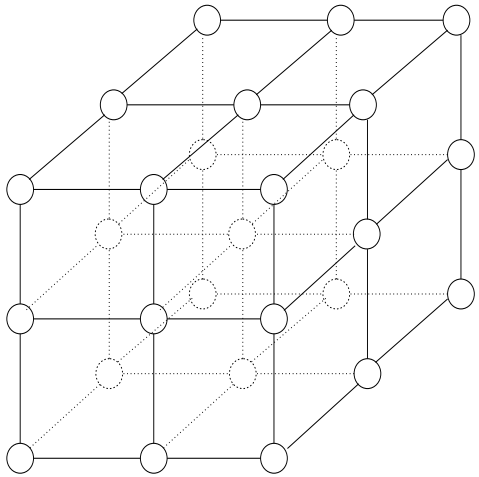
Social network structures are used to determine best positions/attractors



: Star Topology



: Ring Topology



: Von Neumann Topology

- 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

- $\mathbf{y}_i(t)$ 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

- 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

- 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

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;
```

PSO Control Parameters

What is the issue

Performance, specifically convergence behavior, of PSO is highly dependent on the values of the control parameters:

- the inertia weight, w
- the acceleration coefficients, $c_1 + c_2$
- the random values, r_1 and r_2

Consequences for bad values include:

- divergent search trajectories
- cyclic trajectories
- too fast, premature convergence
- too long search, slow convergence
- undesirable exploration versus exploitation trade-off

PSO Control Parameters

r_1 and r_2

PSO is a stochastic search algorithm, with the stochasticity due to r_1 and r_2

They should be vectors of random values, i.e.

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

and not scalars, that is not

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

PSO Control Parameters

r_1 and r_2 (cont)

Note, the random values should be sampled

- per iteration
- per individual
- per dimension

What is the consequence if r_1 and r_2 are scalars?

Can only reach points in the search space that are linear combinations of the original particle positions

Formal proof in U Paquet, AP Engelbrecht, *Particle Swarms for Equality-Constrained Optimization*, Fundamenta Informaticae, vol 76, pp 1–24, 2006

PSO Control Parameters

Effect of w

Controls the tendency of particles to keep searching in the same direction

- 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
- very problem-dependent

PSO Control Parameters

Effect of c_1 and c_2

Respectively scales the influence of the two attractors, the personal best and the neighborhood best positions

Consequences of different values:

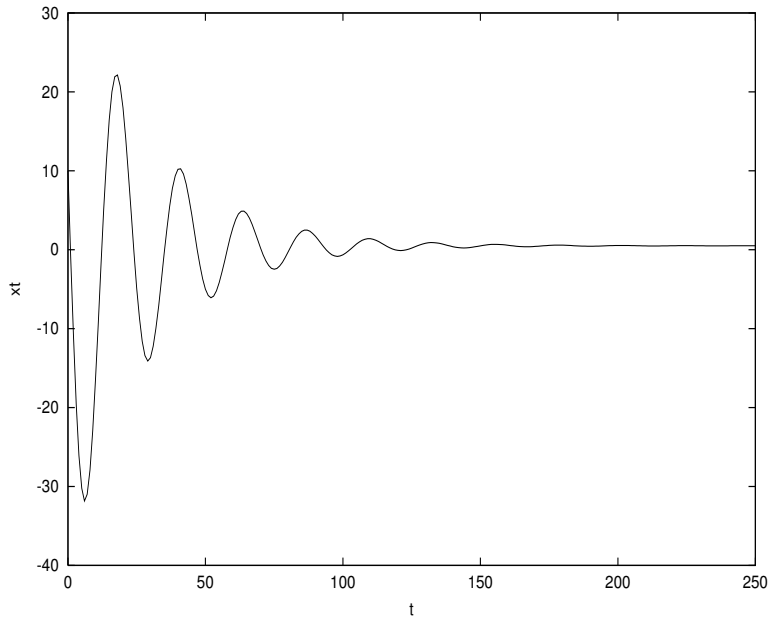
- $c_1 = c_2 = 0$?
- $c_1 > 0, c_2 = 0$:
  - particles are independent hill-climbers
  - local search by each particle
  - cognitive-only PSO
- $c_1 = 0, c_2 > 0$:
  - swarm is one stochastic hill-climber
  - social-only PSO
- $c_1 = c_2 > 0$:
  - particles are attracted towards the average of $\mathbf{y}_i$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

Simplified particle trajectories:

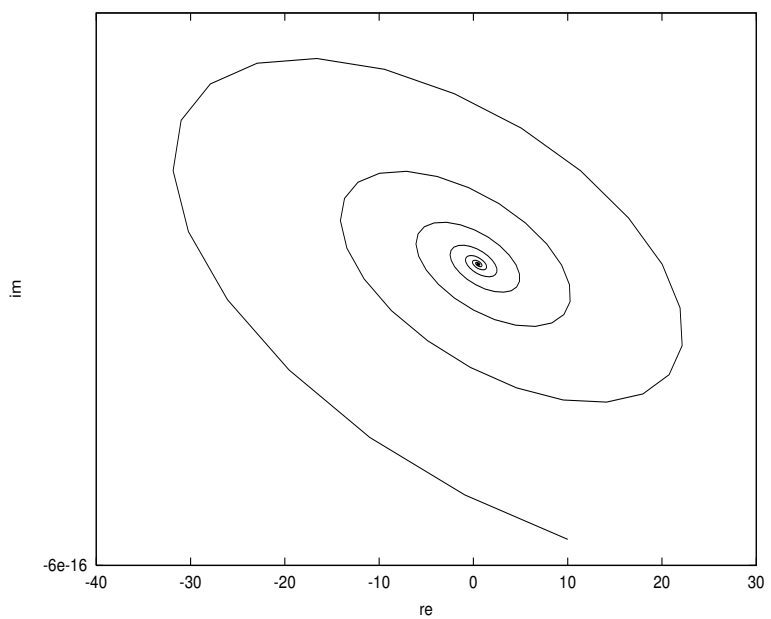
- no stochastic component
- single, one-dimensional particle
- using w
- personal best and global best are fixed: $y = 1.0, \hat{y} = 0.0$

Example trajectories:

- Convergence to an equilibrium
- Cyclic behavior
- Divergent behavior



(a) Time domain

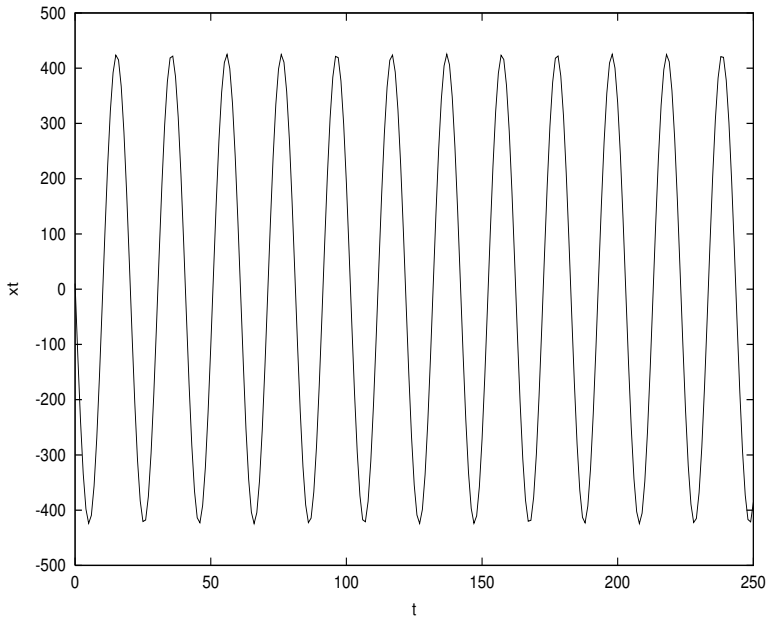


(b) Phase space

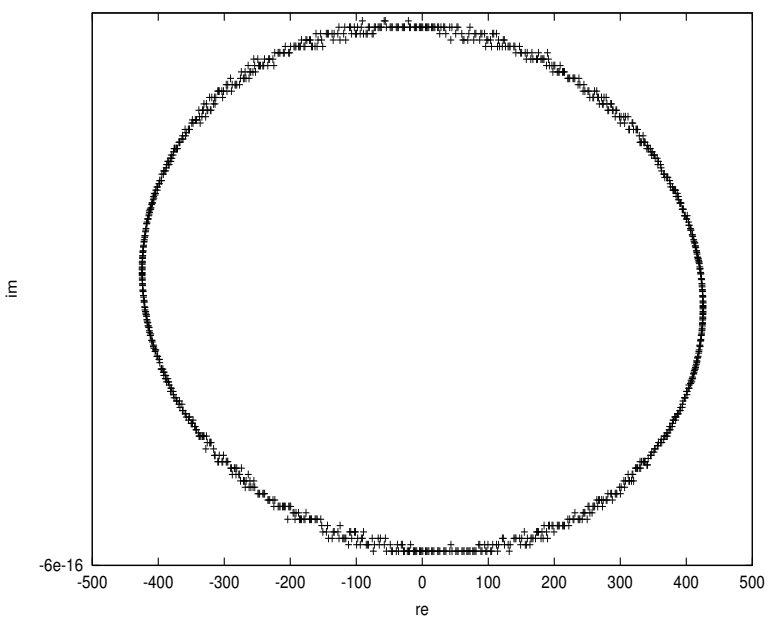
: $w = 0.5$ and $\phi_1 = \phi_2 = 1.4$

PSO Control Parameters

Example Particle Trajectories: Cyclic Trajectories

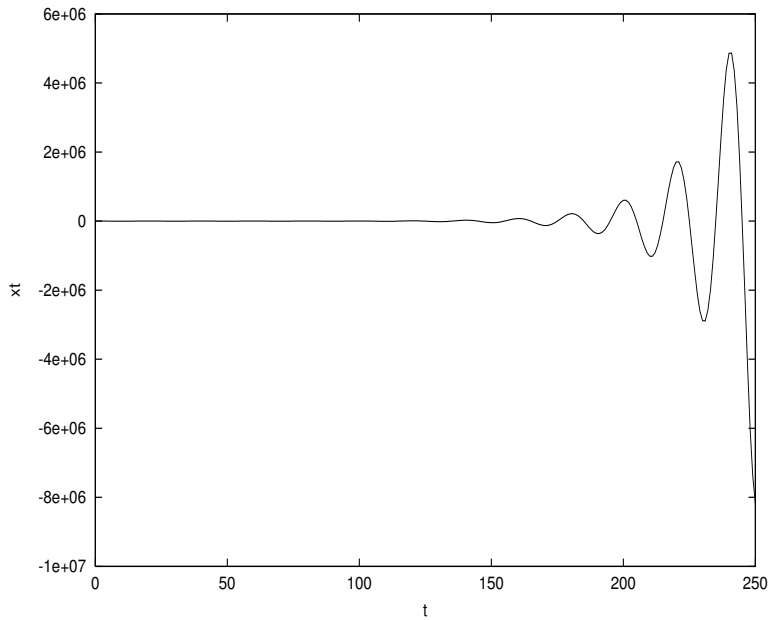


(a) Time domain

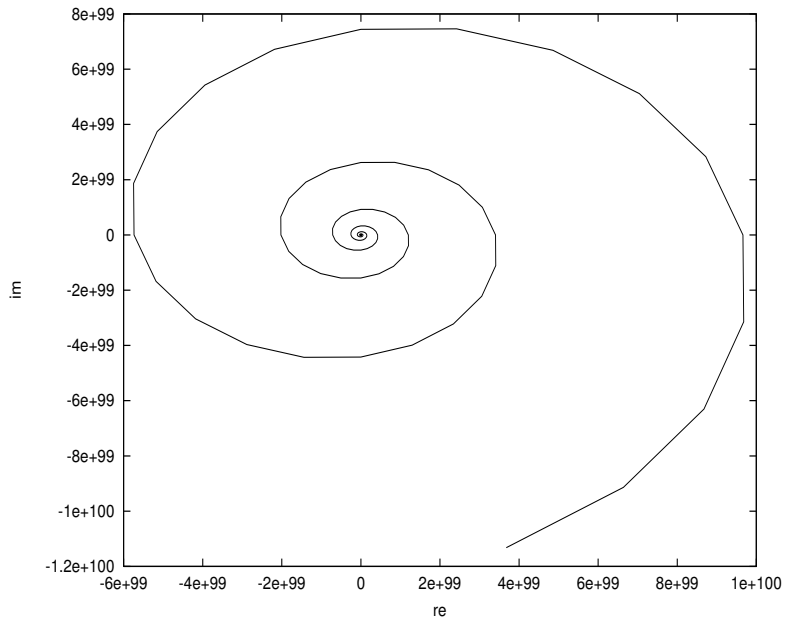


(b) Phase space

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



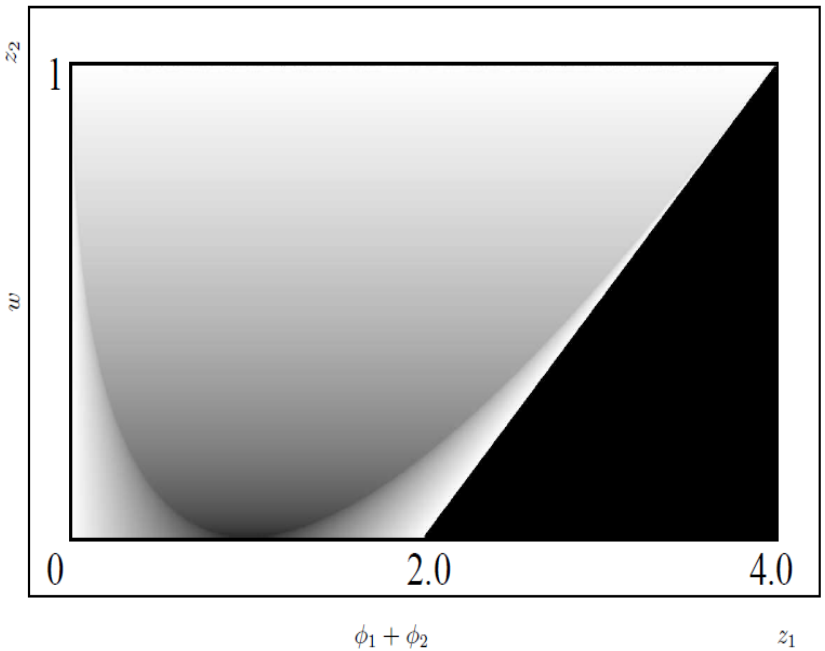
(a) Time domain



(b) Phase space

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

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

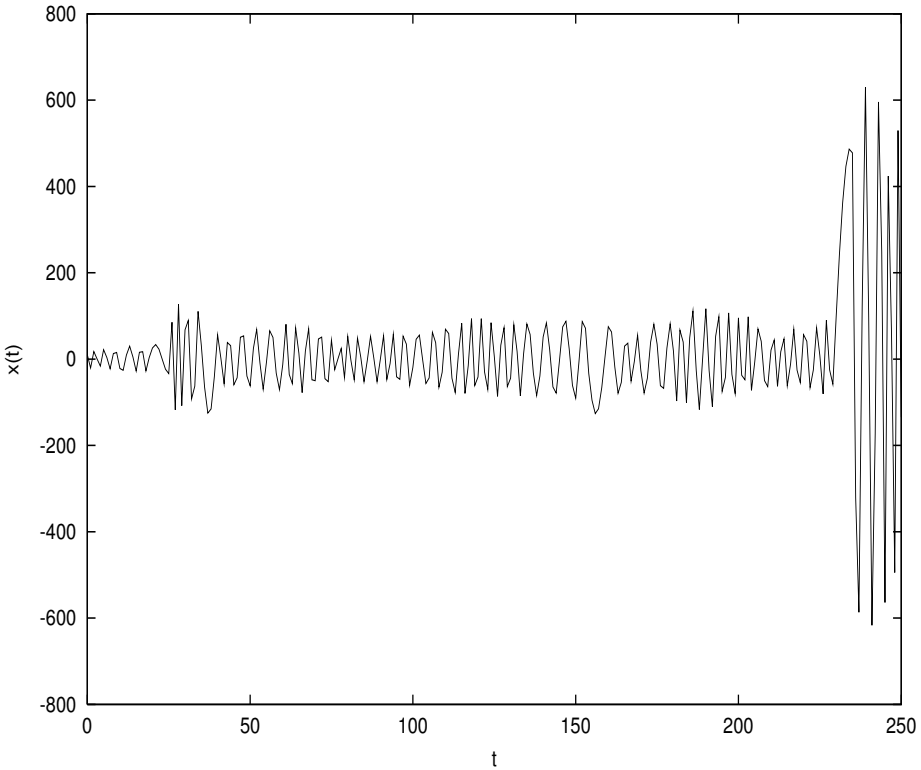


Convergence conditions on values of w, c_1 and c_2:

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

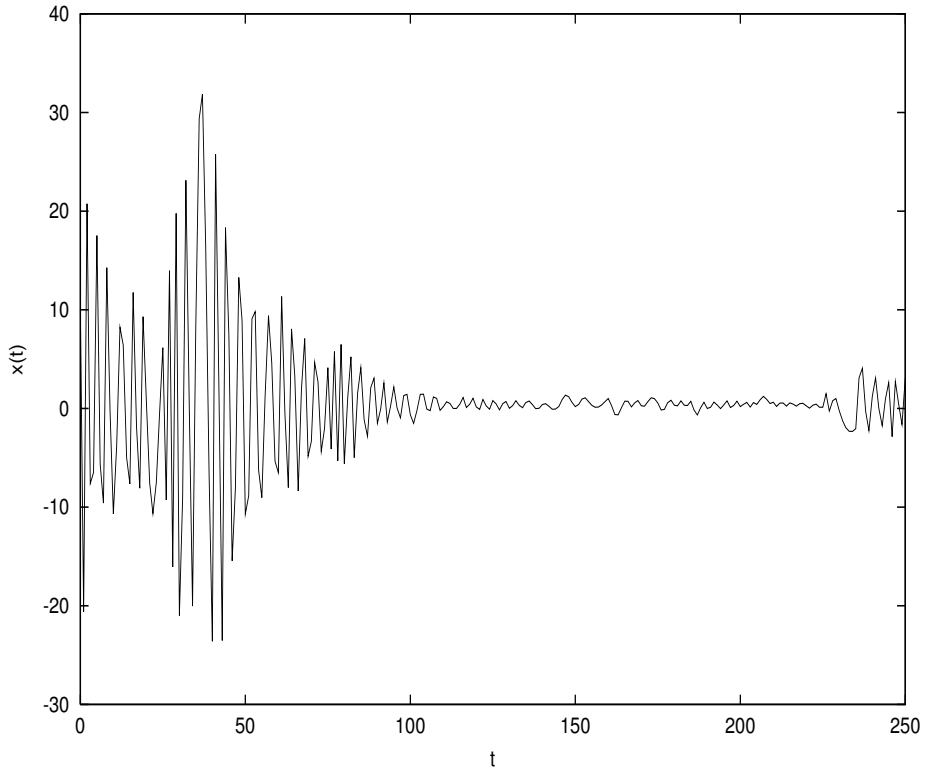
: Convergence Map for Values of w and $\phi = \phi_1 + \phi_2$

: $w = 1.0$ 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

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

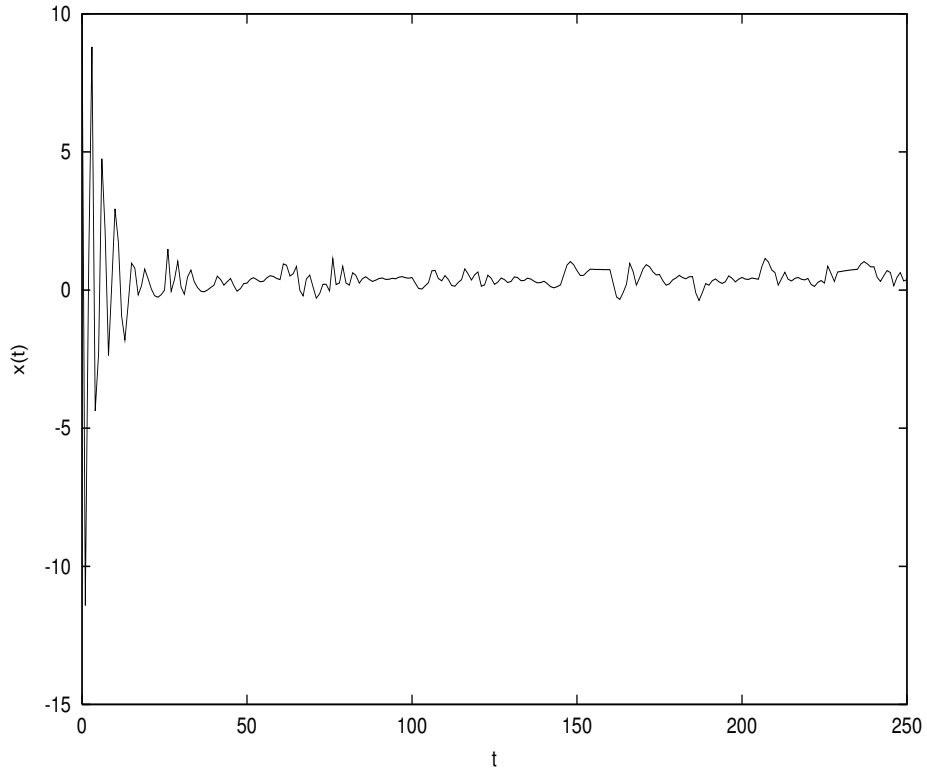


- violates the convergence condition
- for $w = 0.9$, $c_1 + c_2 < 3.8$ to validate the condition

What is happening here?

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

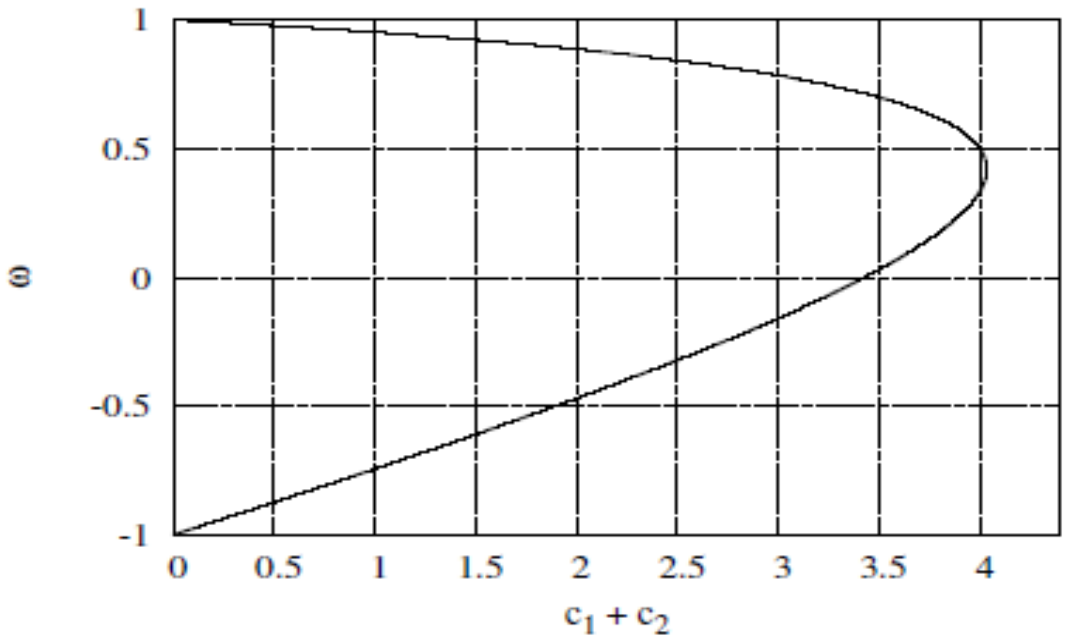
: $w = 0.7$ and $c_1 = c_2 = 1.4$



- validates the convergence condition

A more recent and accurate convergence condition:

$$c_1 + c_2 < \frac{24(1 - w^2)}{7 - 5w} \text{ for } w \in [-1, 1]$$



Empirically shown to be the best matching convergence condition

PSO Issues: Roaming Particles

- Empirical analysis and theoretical proofs 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

Empirical Analysis

Goal of this experiment: To illustrate

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

Experimental setup:

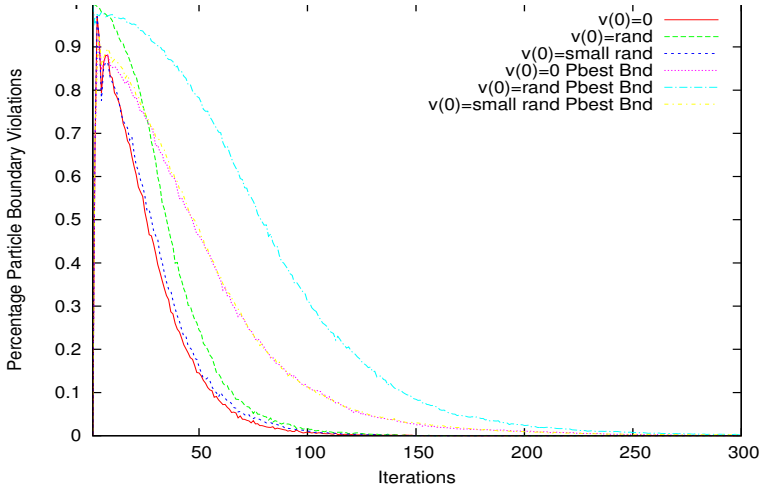
- A standard gbest PSO was used
- 30 particles
- $w = 0.729844$
- $c_1 = c_2 = 1.496180$
- Memory-based global best selection
- Synchronous position updates
- 50 independent runs for each initialization strategy

Functions Used for Empirical Analysis to Illustrate Roaming Behavior

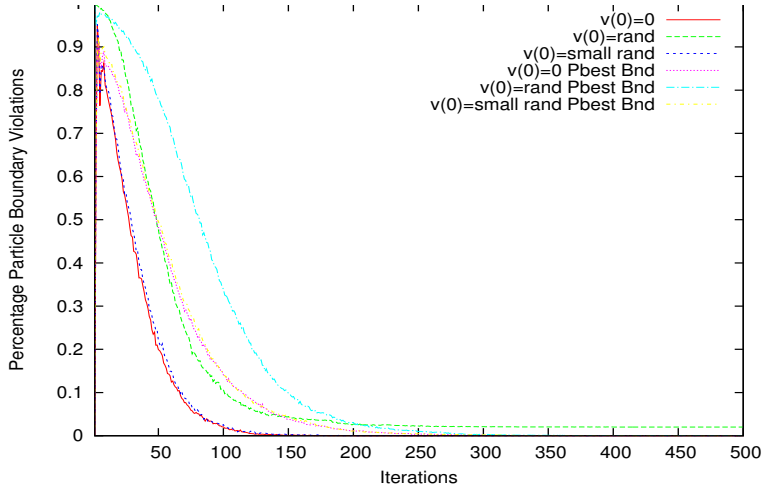
Function	Definition	Domain
AbsValue	$f(\mathbf{x}) = \sum_{j=1}^{n_x} \lvert x_i \rvert$	[-100,100]
Ackley	$f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n_x}\sum_{j=1}^{n_x} x_j^2}} - e^{\frac{1}{n_x}\sum_{j=1}^{n_x}\cos(2\pi x_j)} + 20 + e$	[-32.768,32.768]
Bukin 6	$f(\mathbf{x}) = 100\sqrt{\lvert x_2 - 0.01x_1^2 \rvert} + 0.01\lvert x_1 + 10 \rvert$	[-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 Particles

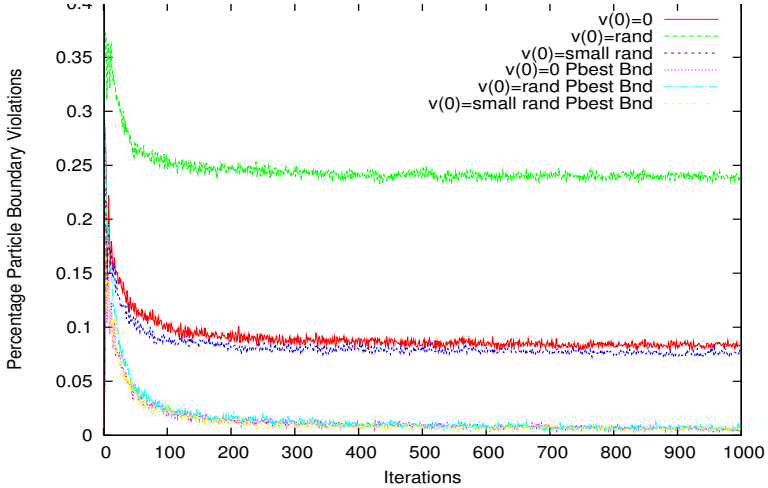
Percentage Particles that Violate Boundaries



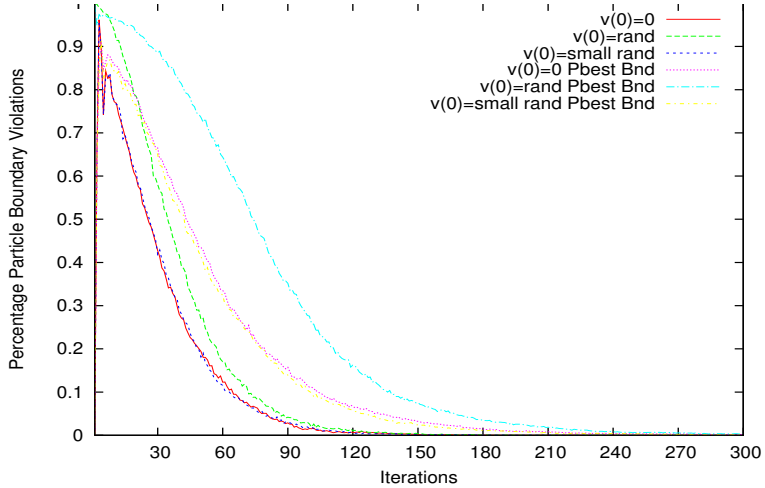
(a) Absolute Value



(b) Ackley



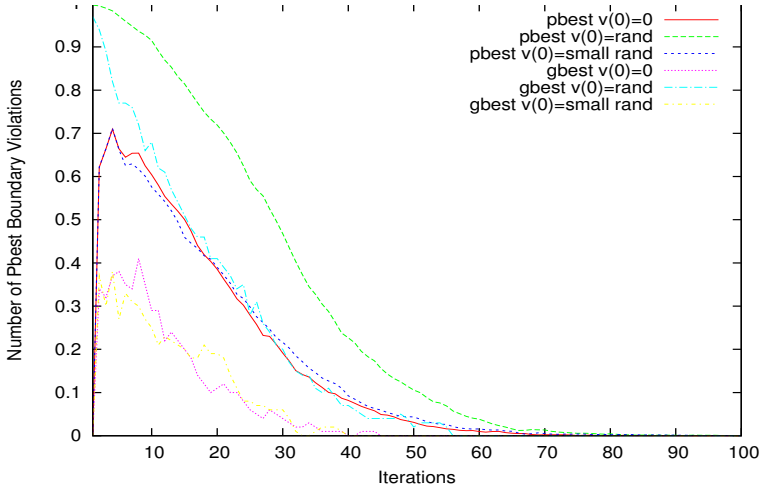
(c) Bukin 6



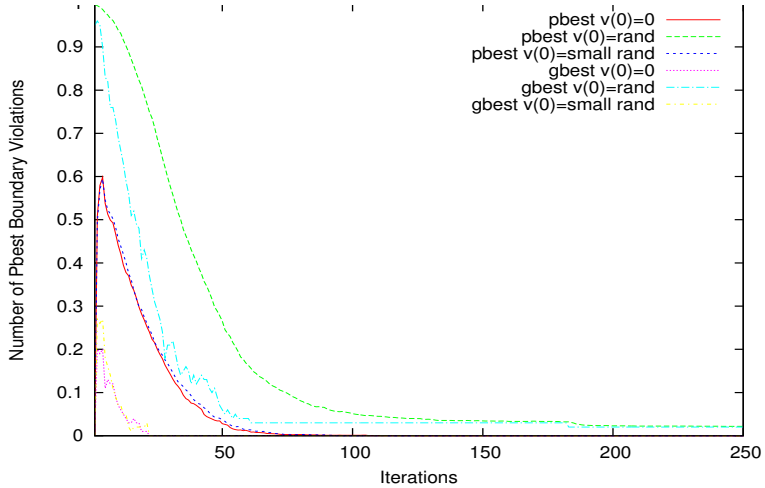
(d) Griewank

PSO Issues: Roaming Particles

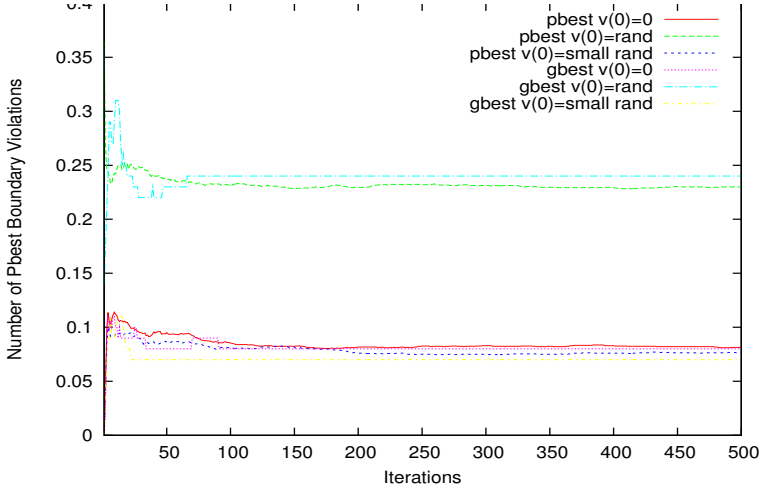
Percentage Best Position Boundary Violations



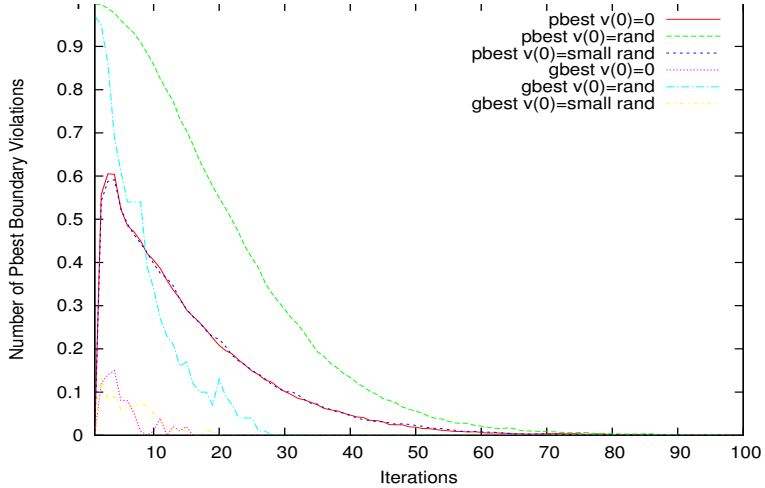
(a) Absolute Value



(b) Ackley



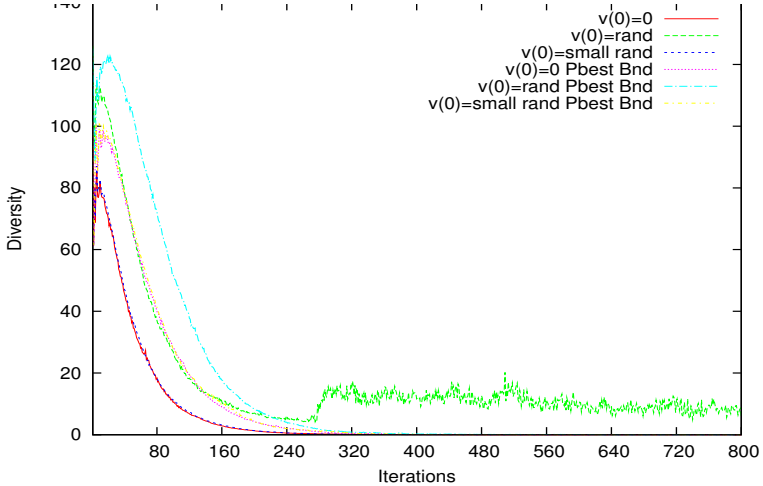
(c) Bukin 6



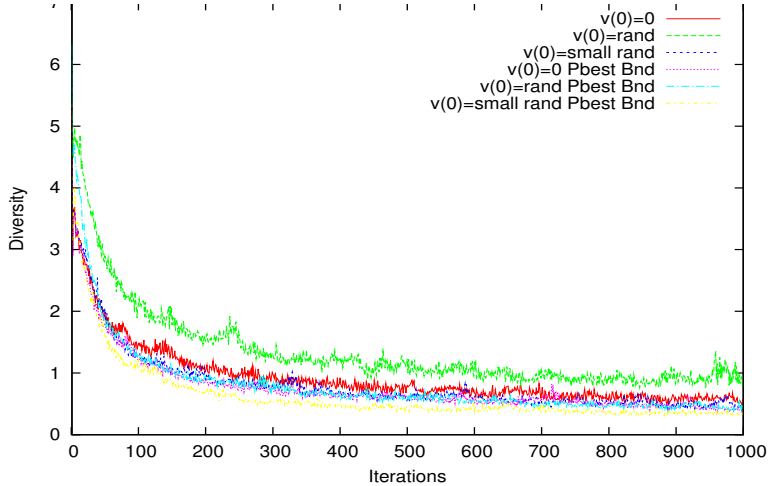
(d) Griewank

PSO Issues: Roaming Particles

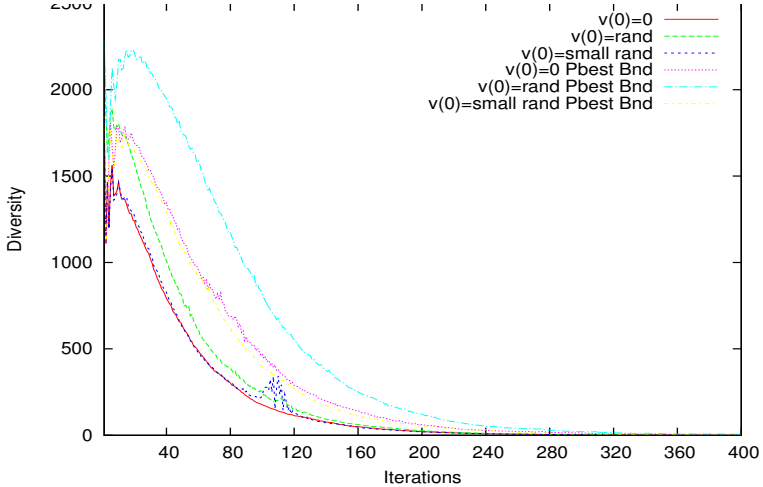
Diversity Profiles



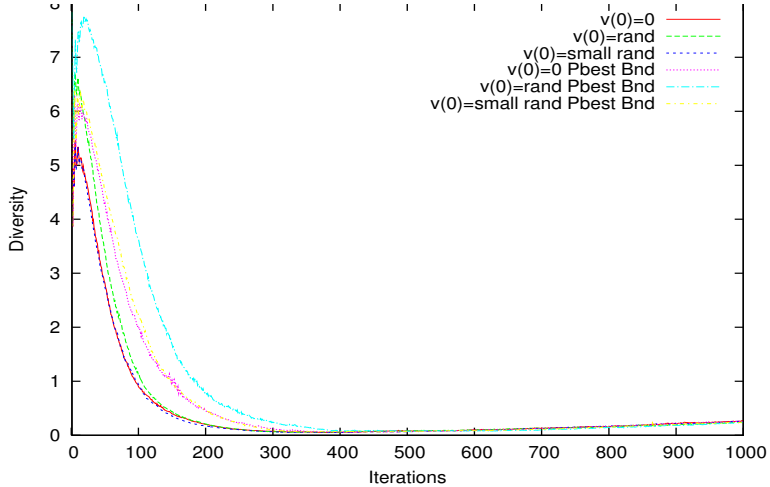
(a) Ackley



(b) Bukin 6



(c) Griewank



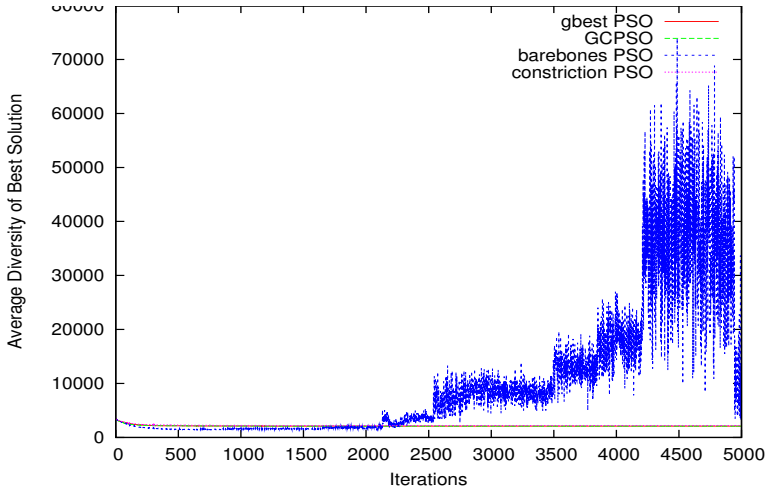
(d) Rosenbrock

Functions Used for Empirical Analysis to Illustrate Finding of Infeasible Solutions

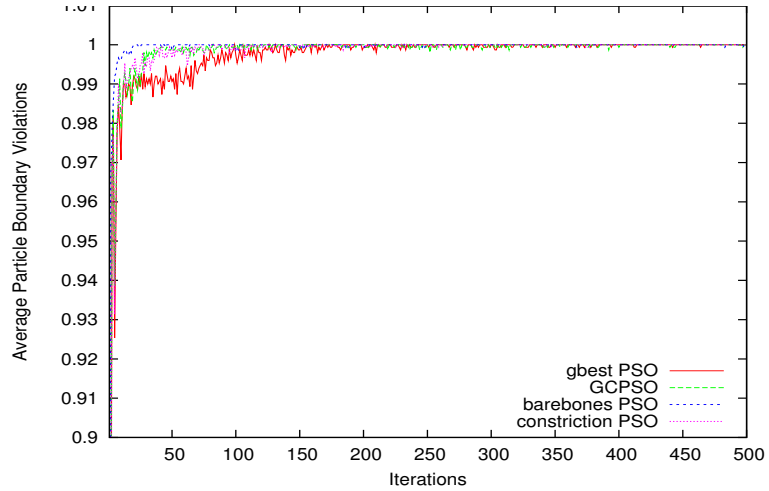
Function	Domain	Function Definition
Ackley	[10,32.768]	$f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n_x} x_j^2}} - e^{\frac{1}{n}\sum_{j=1}^{n_x}\cos(2\pi x_j)} + 20 + e$
Ackley^{SR}	[-32.768,0]	$f(\mathbf{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{j=1}^{n_x} z_j^2}} - e^{\frac{1}{n}\sum_{j=1}^{n_x}\cos(2\pi z_j)} + 20 + e$
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]	$f(\mathbf{x}) = \sum_{j=1}^{n_x} z_i^2$
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 Particles

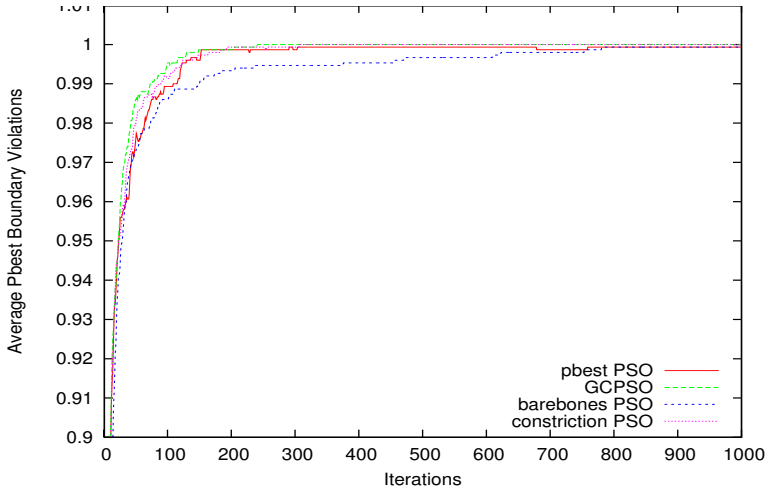
Finding Infeasible Solutions: Ackley



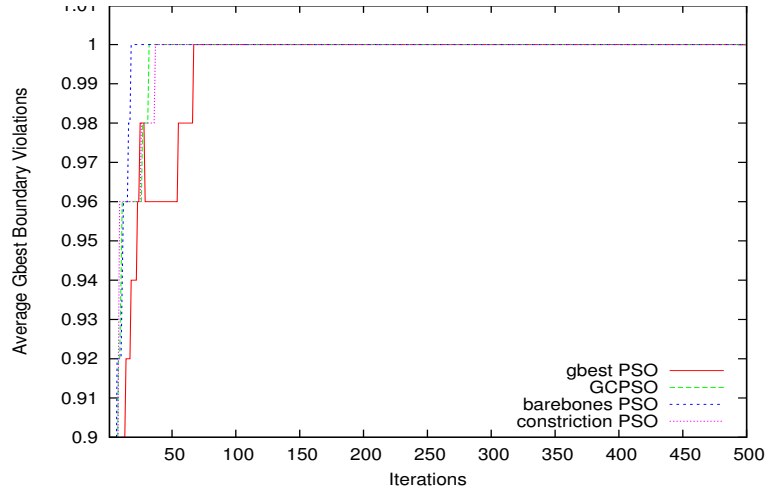
(a) Diversity



(b) Particle Boundary Violations



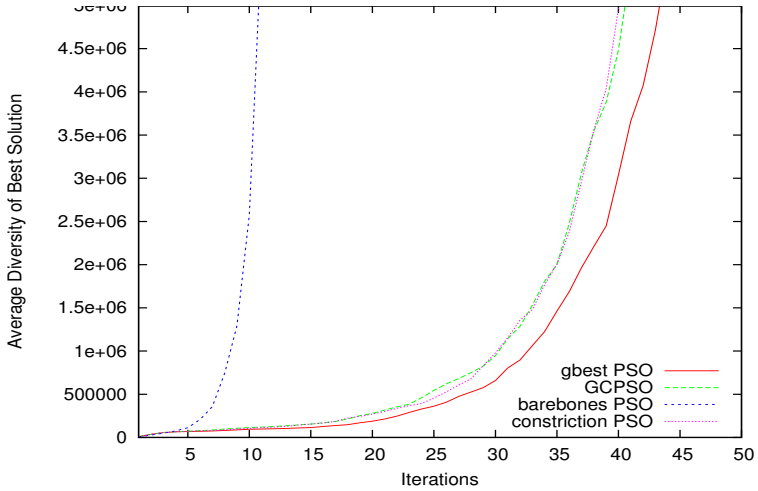
(c) Personal Best Boundary Violations



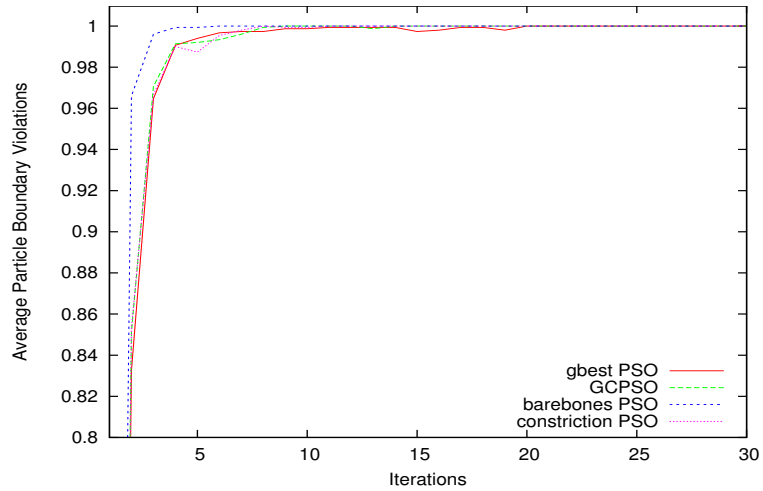
(d) Global Best Boundary Violations

PSO Issues: Roaming Particles

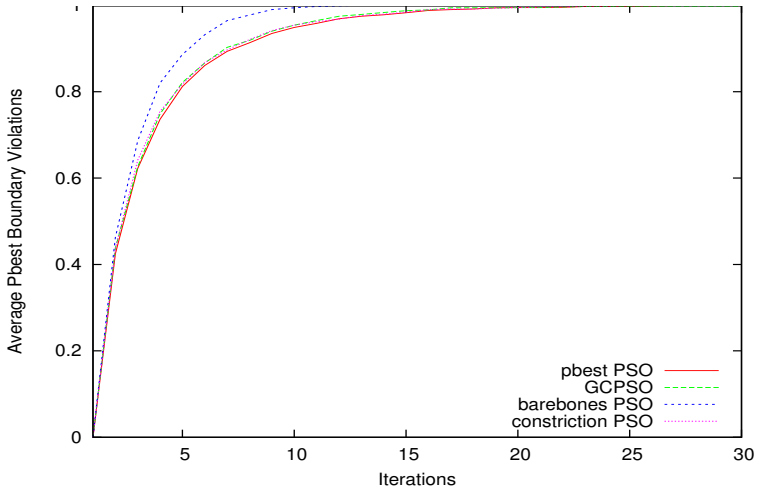
Finding Infeasible Solutions: Eggholder



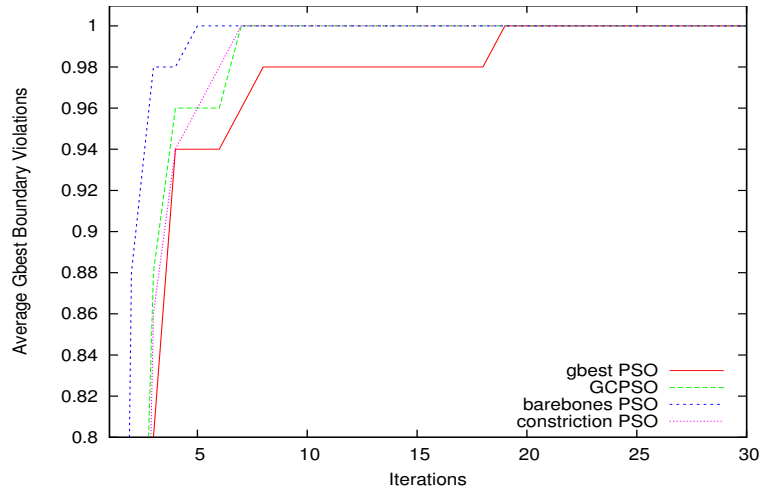
(a) Diversity



(b) Particle Boundary Violations



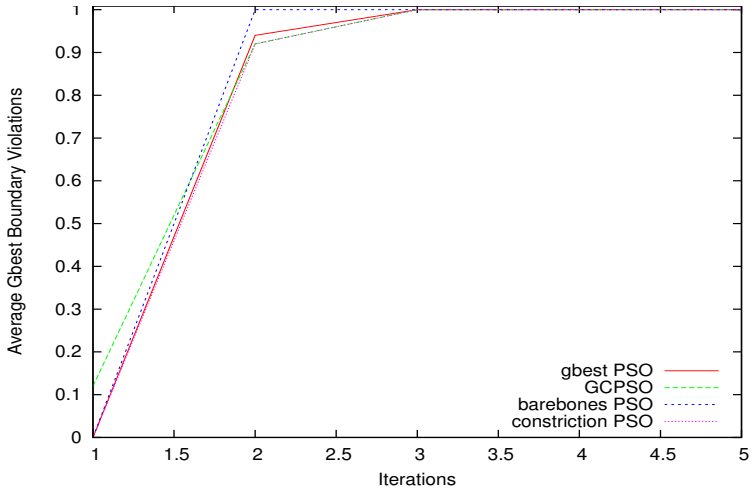
(c) Personal Best Boundary Violations



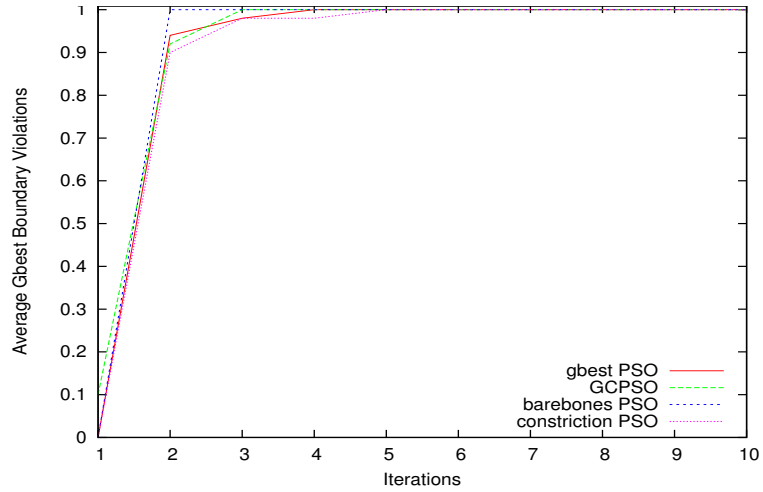
(d) Global Best Boundary Violations

PSO Issues: Roaming Particles

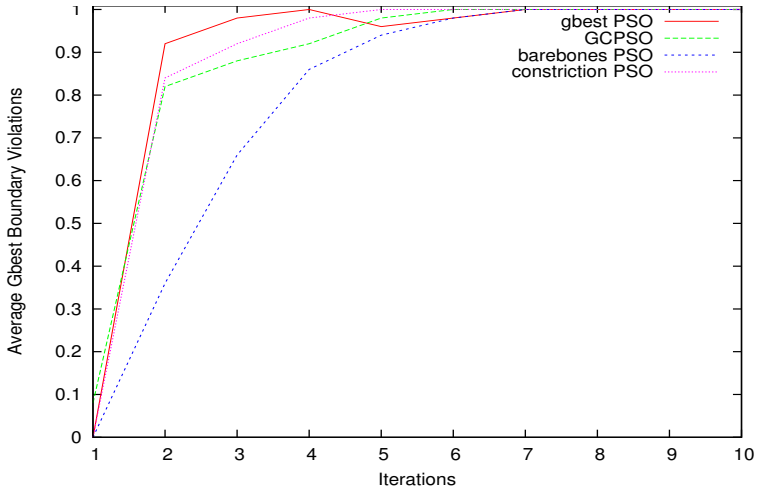
Finding Infeasible Solutions: gbest Boundary Violations



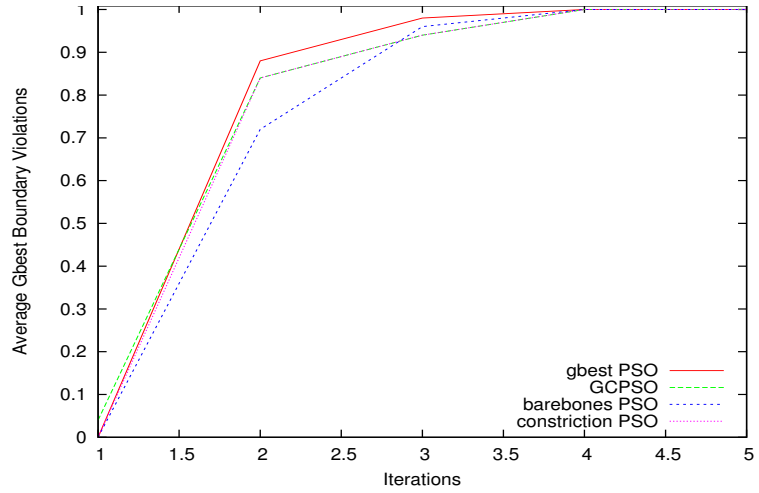
(a) Shifted Schwefel 1.2



(b) Schwefel 2.26



(c) Salomon



(d) Shifted Spherical

PSO Issues: Roaming Particles

Solving the Problem

The roaming problem can be addressed by

- using boundary constraint mechanisms
- update personal best positions only when solution quality improves AND the particle is feasible (within bounds)

However, note that some problems do not have boundary constraints, such as neural network training...

- PSO performance is very sensitive to control parameter values
- Approaches to find the best values for control parameters:
  - Just use the values published in literature?
    - Do you want something that works,
    - or something that works best?
  - Fine-tuned static values
  - Dynamically changing values
  - Self-adaptive control parameters

- Factorial design
- F-Race
- Control parameter dependencies
- Problem dependency
- Computationally expensive

- Time-Varying Inertia Weight (PSO-TVIW)

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

where w_s and w_f are the initial and final inertia weight values, n_t is the maximum number of iterations

- Time-Varying Acceleration Coefficients (PSO-TVAC)

$$c_1(t) = c_{1s} + (c_{1f} - c_{1s})\frac{t}{n_t}$$

$$c_2(t) = c_{2s} + (c_{2f} - c_{2s})\frac{t}{n_t}$$

PSO with Simulated Annealing:

- Inertia weight is adapted as

$$w_i(t) = w_a F(\eta_i(t)) + w_b$$

where w_a and w_b are user-specified positive constants, and

$$F(\eta_i(t)) = \begin{cases} 2 & \text{if } \eta_i(t) < 0.0001 \\ 1 & \text{if } 0.0001 \leq \eta_i(t) < 0.01 \\ 0.3 & \text{if } 0.01 \leq \eta_i(t) < 0.1 \\ -0.8 & \text{if } 0.1 \leq \eta_i(t) < 0.9 \\ -5.5 & \text{if } 0.9 \leq \eta_i(t) \leq 1.0 \end{cases}$$

and the relative particle performance is

$$\eta_i(t) = \frac{f(\hat{\mathbf{y}}_i(t-1))}{f(\mathbf{x}_i(t-1))}$$

$\eta_i(t) \approx 0$ denotes that particle is much worse than the nbest
$\eta_i(t) = 1$ denotes particle is as good as nbest

- Social acceleration adapted as

$$c_{2i}(t) = c_{2a}G(\eta_i(t)) + c_{2b}$$

and

$$G(\eta_i(t)) = \begin{cases} 2.5 & \text{if } \eta_i(t) < 0.0001 \\ 1.2 & \text{if } 0.0001 \leq \eta_i(t) < 0.01 \\ 0.5 & \text{if } 0.01 \leq \eta_i(t) < 0.1 \\ 0.2 & \text{if } 0.1 \leq \eta_i(t) < 0.9 \\ 0.1 & \text{if } 0.9 \leq \eta_i(t) \leq 1.0 \end{cases}$$

For η_i low, c_2 increases

Particle swarm optimization with individual coefficients adjustment:

- Inertia weight:

$$w_i(t) = w_a F(\xi_i(t)) + w_b$$

with

$$F(\xi_i(t)) = 2\left(1 - cos\left(\frac{\pi \xi_i(t)}{2}\right)\right)$$

- Social acceleration

$$c_{2i}(t) = c_{2a} G(\xi_i(t)) + c_{2b}$$

$$G(\xi_i(t)) = 2.5\left(1 - cos\left(\frac{\pi \xi_i(t)}{2}\right)\right)$$

and

$$\xi_i(t) = \begin{cases} 0 & \text{if } f(\mathbf{x}_i(t-1)) = 0 \\ \frac{f(\mathbf{x}_i(t-1)) - f(\hat{\mathbf{y}}_i(t-1)}{f(\mathbf{x}_i(t-1))} & \text{otherwise} \end{cases}$$

Improved Particle Swarm Optimization adapts inertia weight as:

$$w(t) = e^{-\lambda(t)}$$

with

$$\lambda(t) = \frac{\alpha(t)}{\alpha(t-1)}$$

and

$$\alpha(t) = \frac{1}{n_s} \sum_{i=1}^{n_s} |f(\mathbf{x}_i(t)) - f(\hat{\mathbf{y}}^*(t))|$$

where $\hat{\mathbf{y}}^*(t)$ is the iteration-best

Adaptive particle swarm optimization based on velocity information:

- Inertia weight updated using

$$w(t+1) = \begin{cases} \max\{w(t) - \Delta w, w_{min}\} & \text{if } \overline{v(t)} \geq v_{ideal}(t+1) \\ \min\{w(t) + \Delta w, w_{max}\} & \text{otherwise} \end{cases}$$

where Δw is a step size, and the ideal velocity is

$$v_{ideal}(t) = v_s \left(\frac{1 + \cos(\pi \frac{t}{T_{0.95}})}{2} \right)$$

where $v_s = \frac{x_{max} - x_{min}}{2}$ is the initial ideal velocity, $T_{0.95}$ is the point where 95% of the search is complete, and

$$\overline{v(t)} = \frac{1}{n_x n_s} \sum_{i=1}^{n_s} \sum_{j=1}^{n_x} |v_{ij}(t)|$$

Adaptive inertia weight particle swarm optimization:

- Inertia weight update:

$$w(t) = (w_{max} - w_{min})P_s(t) + w_{min}$$

with

$$P_s(t) = \frac{\sum_{i=1}^{n_s} S_i(t)}{n_s}$$

and

$$S_i(t) = \begin{cases} 1 & \text{if } f(\mathbf{y}_i(t)) < f(\mathbf{y}_i(t-1)) \\ 0 & \text{otherwise} \end{cases}$$

Increases w when particle successes are high

The Self-Adaptive PSO, adapts inertia weight as

$$w_i(t) = 0.15 + \frac{1}{1 + e^{\overline{f(\mathbf{y}(t))} - f(\mathbf{y}_i(t))}}$$

where $\overline{f(\mathbf{y}(t))}$ is the average pbest fitness values of the swarm

Issues with current self-adaptive approaches:

- Most, at some point in time, violate convergence conditions
- Converge prematurely, with little exploration of control parameter space
- Introduce more control parameters

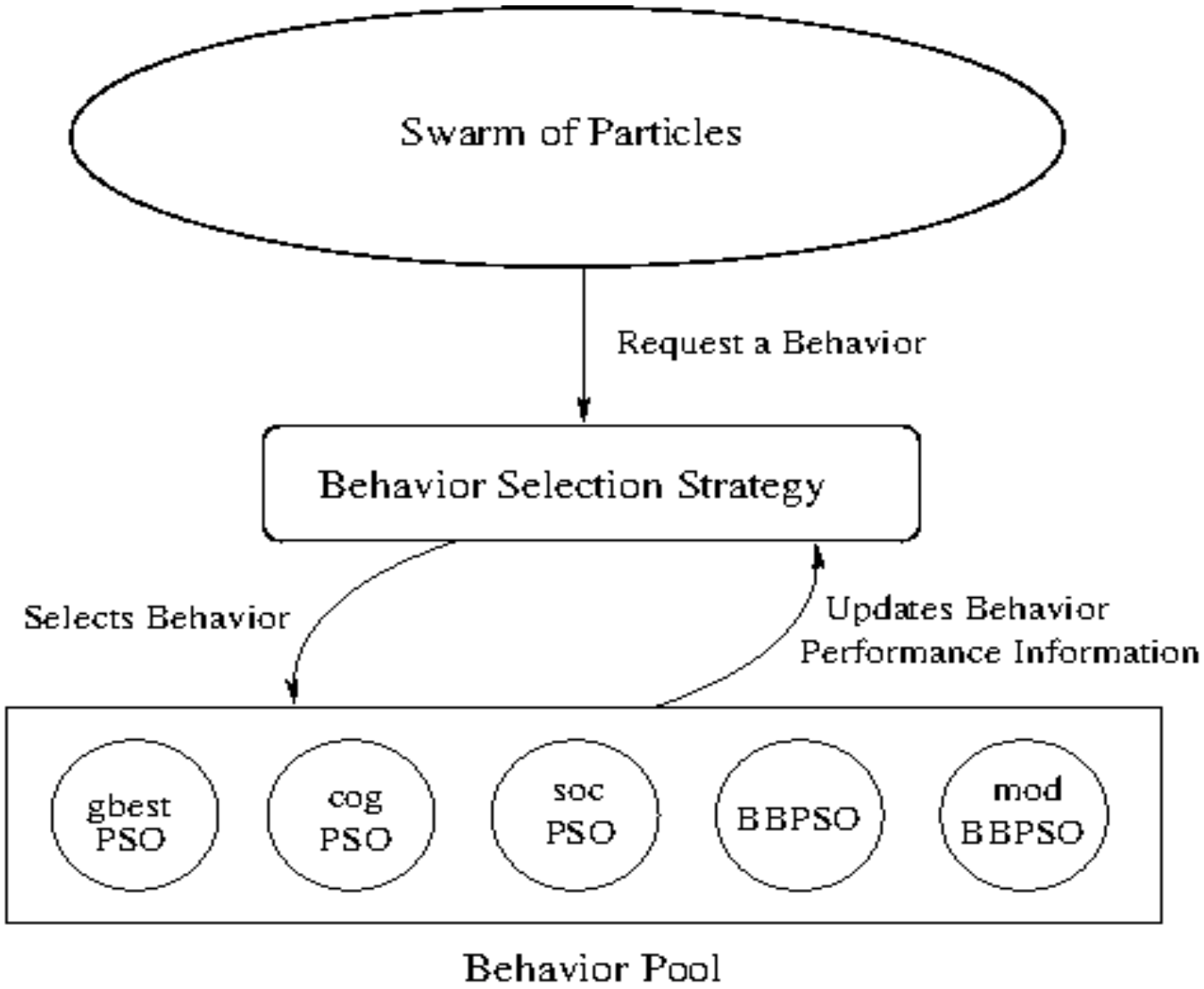
Particles in standard particle swarm optimization (PSO), and most of its modifications, follow the same behavior:

- particles implement the same velocity and position update rules
- particles therefore exhibit the same search behaviours
- the same exploration and explotation abilities are achieved

Heterogeneous swarms contain particles that follow different behaviors:

- particles follow different velocity and position update rules
- some particles may explore longer than others, while some may exploit earlier than others
- a better balance of exploration and exploitation can be achieved provided a pool of different behaviors is used

Akin to hyper-heuristics, ensemble methods, multi-methods


Swarm of Particles
Request a Behavior
Behavior Selection Strategy
Selects Behavior
Updates Behavior
Performance Information
gbest PSO
cog PSO
soc PSO
BBPSO
mod BBPSO
Behavior Pool

A behavior is defined as both

- the velocity update, and
- the position update

of a particle

Requirements for behaviors in the behavior pool:

- must exhibit different search behaviors
- different exploration-exploitation phases
- that is, different exploration-exploitation finger prints

Exploration-exploitation finger print determined through diversity profile

Diversity calculated as

$$\mathcal{D} = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\sum_{j=1}^{n_x} (x_{ij} - \overline{x}_j)^2}$$

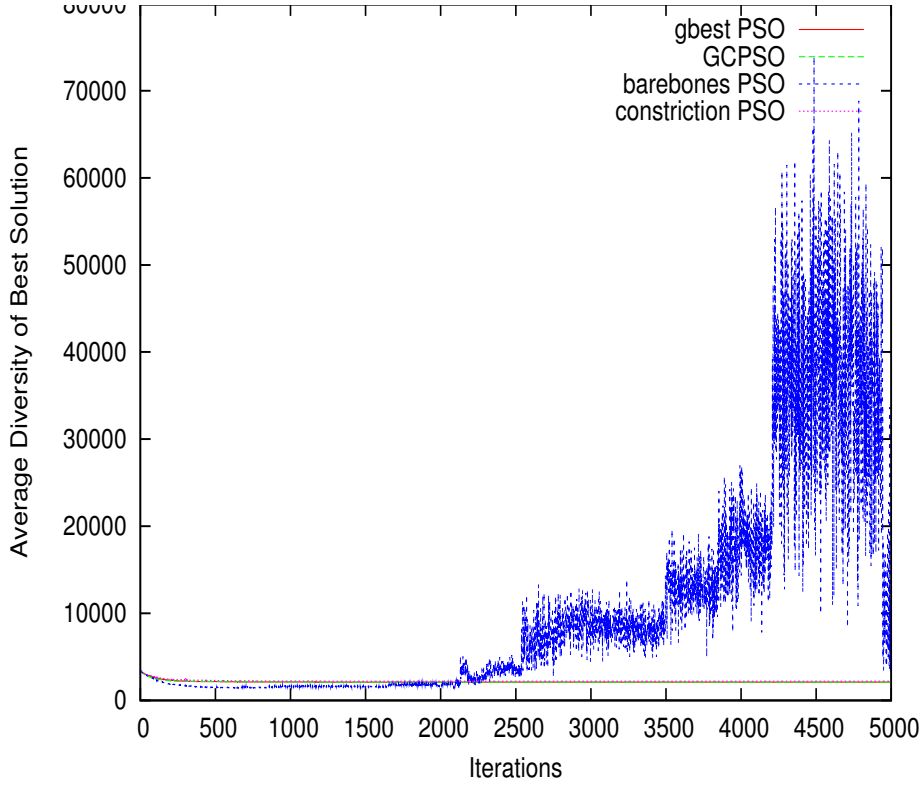
where the swarm center is

$$x_j = \frac{\sum_{i=1}^{n_s} x_{ij}}{n_s}$$

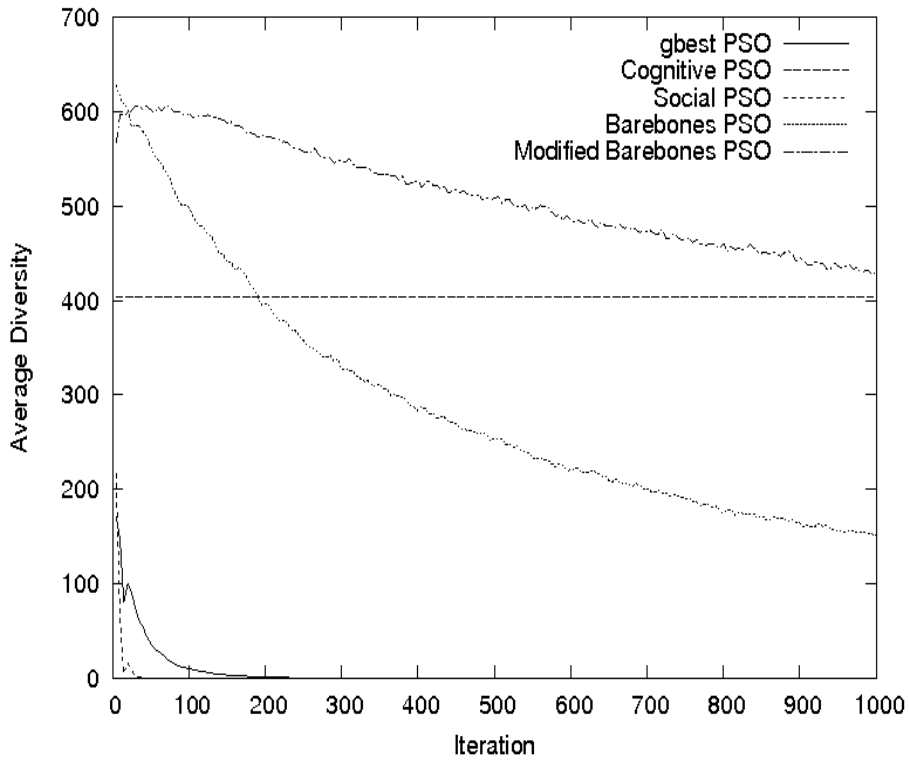
and n_s is the number of particles

Heterogeneous Particle Swarm Optimization

Exploration-Exploitation Finger Prints (cont)



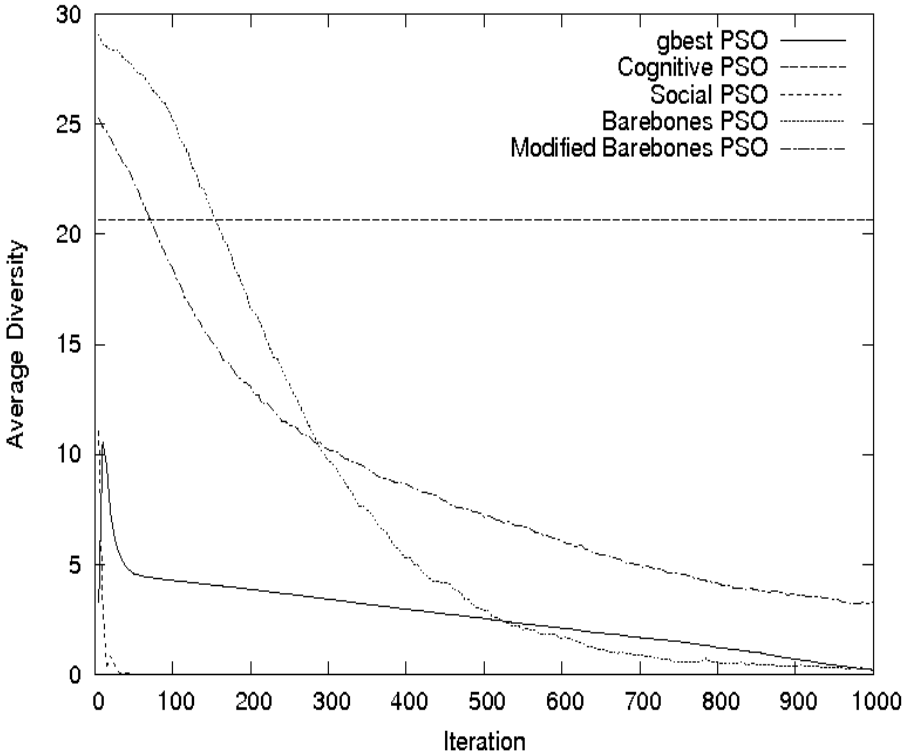
(a) Ackley



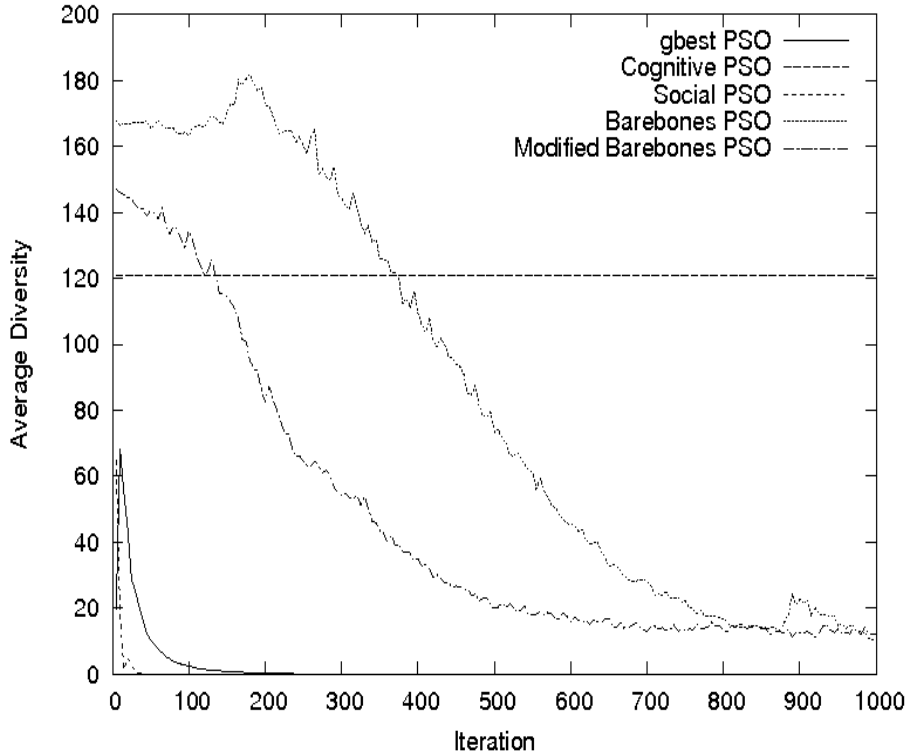
(b) Quadric

Heterogeneous Particle Swarm Optimization

Exploration-Exploitation Finger Prints (cont)



(c) Rastrigin



(d) Rosenbrock

A number of PSO algorithms do already exist which allow particles to follow different search behaviors:

- Division of labor PSO: Particles are allowed to switch to local search near the end of the search process
- Life-cycle PSO: Particles follow a life-cycle, changing from a PSO particle, to GA individual, to a hill-climber
- Predator-prey PSO: Predator particles are attracted to only the gbest position, prey particles follow the standard velocity update rule
- Guaranteed convergence PSO (GCPSO): Global best particle follows a different, exploitaitve search around best position, while other particles follow normal velocity updates

- NichePSO: Main swarm of particles folllow cognitive PSO, while sub-swarms follow GCPSO behavior
- Charged PSO: Charged particles adds a repelling force to the velocity update
- Heterogeneous cooperative PSO: Sub-swarms use different meta-heuristics

Two different initial HPSO models:

- **static** HPSO (SHPSO)
  - behaviors randomly assigned from behavior pool during initialization
  - behaviors do not change
- **dynamic** HPSO (DHPSO)
  - behaviors are randomly assigned
  - may change during search
  - when particle stagnates, it randomly selects a new behavior
  - a particle stagnates if its personal best position does not change over a number of iterations

What is the problem?

- Current HPSO models do not make use of any information about the search process to guide selection towards the most promising behaviors

What is the solution?

- An approach to self-adapt behaviors, i.e. to select the best behaviors probabilistically based on information about the search process

Related self-adaptive approaches to HPSO algorithms:

- Difference proportional probability PSO (DPP-PSO)
  - particle behaviors change to the nbest particle behavior
  - based on probability proportional to how much better the nbest particle is
  - includes static particles for each behavior, i.e. behaviors do not change
  - only two behaviors, i.e. FIPS and original PSO
- Adaptive learning PSO-II (ALPSO-II)
  - Behaviors are selected probabilistically based on improvements that they affect to the quality of the corresponding particles
  - Overly complex and computationally expensive

Heterogeneous Particle Swarm Optimization

Self-Adaptive HPSO: General Structure

```
1:  initialize swarm
2:  while stopping conditions not met do
3:      for each particle do
4:          if change schedule is triggered then
5:              select new behavior for the particle
6:          end if
7:      end for
8:      for each particle do
9:          update particle's velocity and position based on selected behavior
10:     end for
11:     for each particle do
12:         update pbest and gbest
13:     end for
14:     for each behavior do
15:         update behavior score/desirability
16:     end for
17: end while
```

Expanded behavior pool: That of dHPSO, plus

- Quantum PSO (QPSO)
- Time-varying inertia weight PSO (TVIW-PSO)
- Time-varying acceleration coefficients (TVAC-PSO)
- Fully informed particle swarm (FIPS)

Two self-adaptive strategies inspired by foraging behavior of ants

- Ants are able to find the shortest path between their nest and a food source
- Paths are followed probabilistically based on pheromone concentrations on the paths

Definitions:

- B is the total number of behaviors
- b is a behavior index
- p_b is the pheronome concentration for behavior b
- $prob_b(t)$ is the probability of selecting behavior b at time step t

$$prob_b(t) = \frac{p_b(t)}{\sum_{i=1}^{B} p_i(t)}$$

Each particle selects a new behavior, using Roulette wheel selection, when a behavior change is triggered

- Constant strategy (pHPSO-const):
  - Rewards behaviors if they improve or maintain a particle's fitness regardless the magnitude of improvement

$$p_b(t) = p_b(t) + \sum_{i=1}^{S_b} \begin{cases} 1.0 & \text{if } f(x_i(t)) < f(x_i(t-1)) \\ 0.5 & \text{if } f(x_i(t)) = f(x_i(t-1)) \\ 0.0 & \text{if } f(x_i(t)) > f(x_i(t-1)) \end{cases}$$

  - S_b is the number of particles using behavior b

- Linear strategy (pHPSO-lin):
  - Behaviors are rewarded proportional to the improvement in particle fitness

$$p_b(t) = p_b(t) + \sum_{i=1}^{s_b} \left(f\left(x_i(t-1)\right) - f\left(x_i(t)\right) \right)$$

  - A lower bound of 0.01 is set for each p_b to prevent zero or negative pheromone concentrations
  - Each behavior therefore always has a non-zero probability of being selected

To maintain diversity in the behavior pool, pheromone evaporates:

$$p_b(t+1) = \frac{\left(\sum_{i=1, i\neq b}^{B} p_i\right)}{\sum_{i=1}^{B} p_i} \times p_b$$

Amount of evaporation is proportional to the behavior's pheromone concentration as a ratio to the total pheromone concentration

A more desirable behavior has stronger evaporation to prevent domination

The pHPSO strategies are

- computationally less expensive than others,
- behaviors are self-adapted based on success of the corresponding behaviors, and
- better exploration of behavior space is achieved through pheromone evaporation
- introduces no new control parameters

Frequency-based HPSO is based on the premise that behaviors are more desirable if they frequently perform well:

- Each behavior has a success counter
- Success counter keeps track of the number of times that the behavior improved the fitness of a particle
- Only the successes of the previous k iterations are considered, so that behaviors that performed well initially, and bad later, do not continue to dominate in the selection process
- Behaviors change when a particle's pbest position stagnates
- Next behavior chosen using tournament selection
- Two new control parameters: k and tournament size

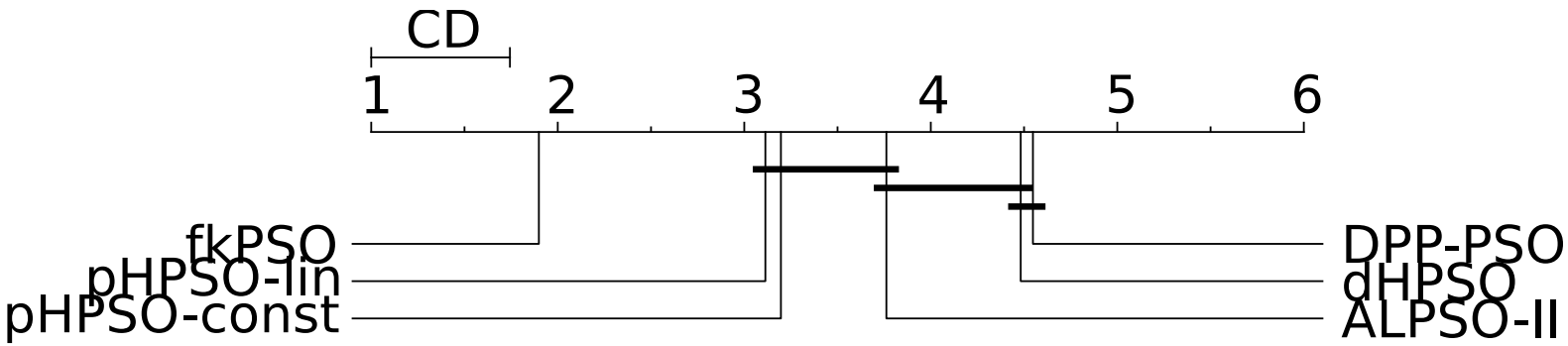
Functions used in 10, 30, 50 dimensions

- CEC 2013 benchmark functions
- 5 unimodal
- 15 basic multimodal
- 8 composition
- domain of $[-100, 100]$

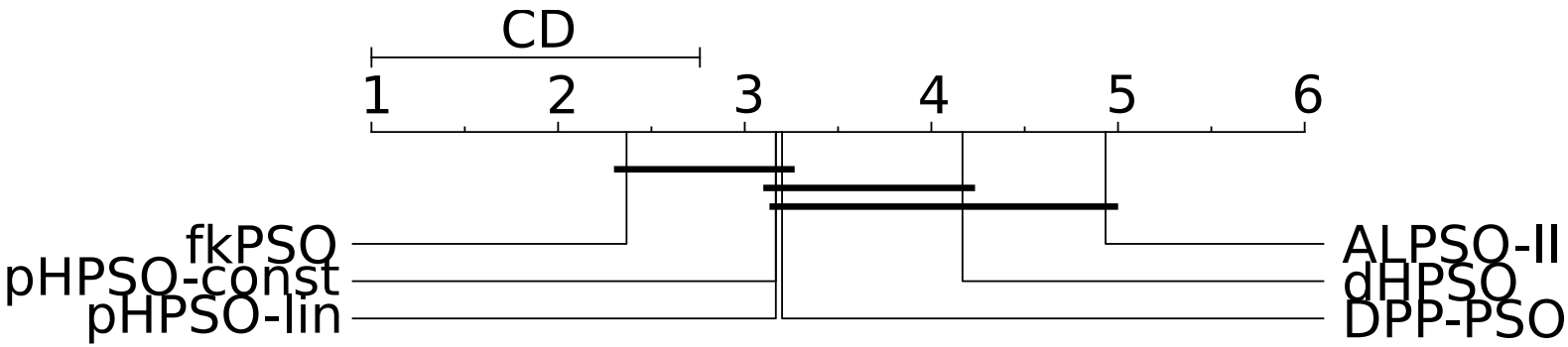
Control Parameters

- All parameters optimized using iterated F-Race
- Swarm size of 50
- Code implemented in CIlib

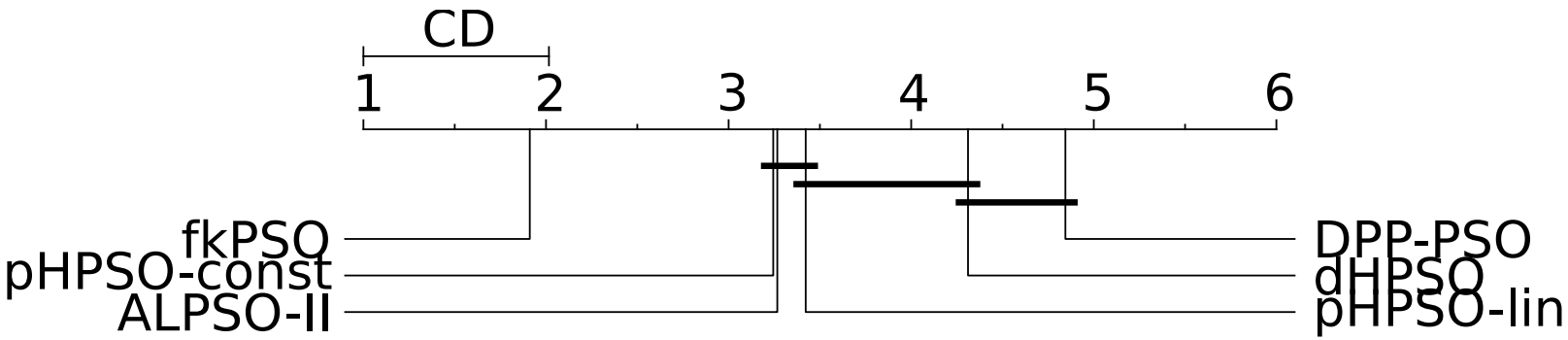
For all functions using Bonferroni-Dunn post-hoc test



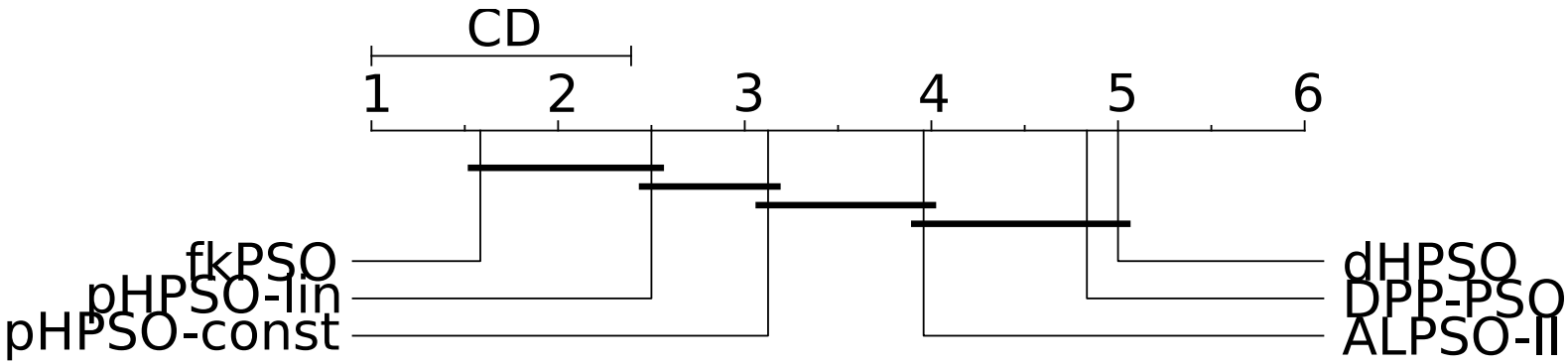
For unimodal functions using Bonferroni-Dunn post-hoc test



For multimodal functions using Bonferroni-Dunn post-hoc test

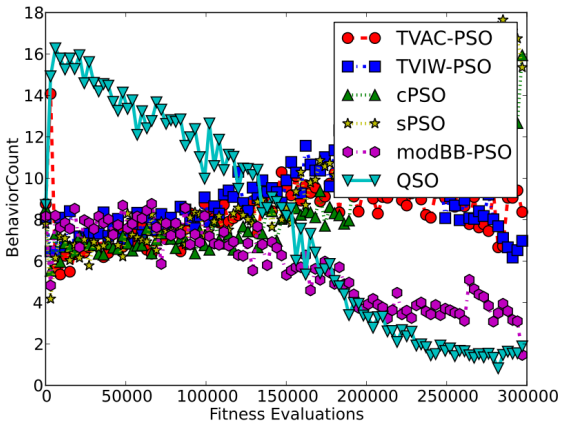


For composition functions using Bonferroni-Dunn post-hoc test

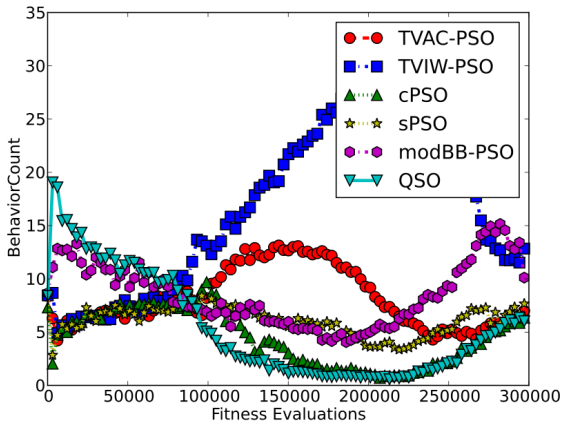


Self-Adaptive Heterogeneous PSO

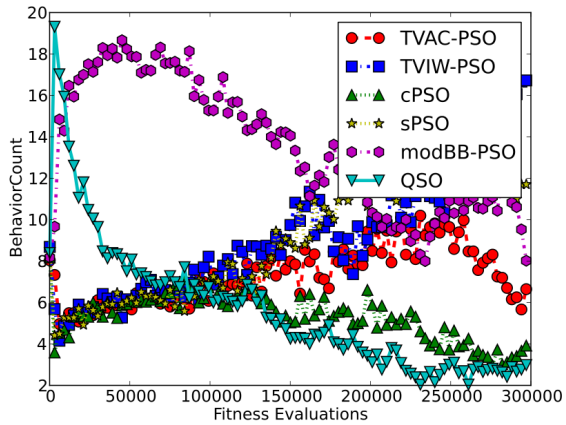
Behavior Profile Plots



(a) f_4

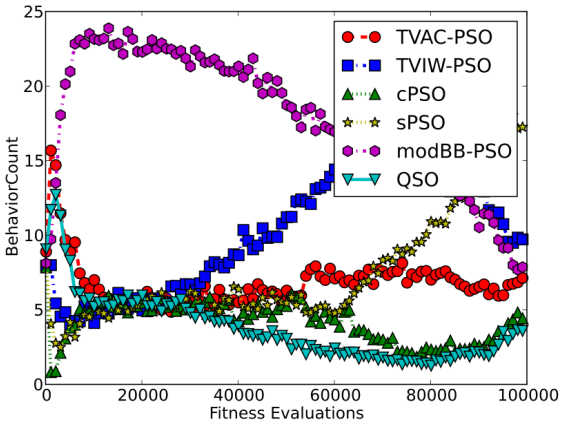


(b) f_{12}

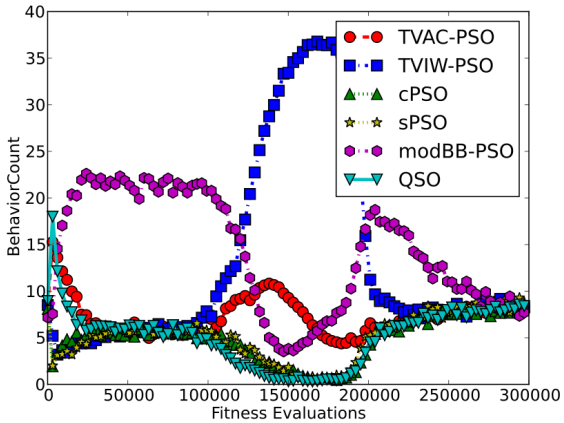


(c) f_{27}

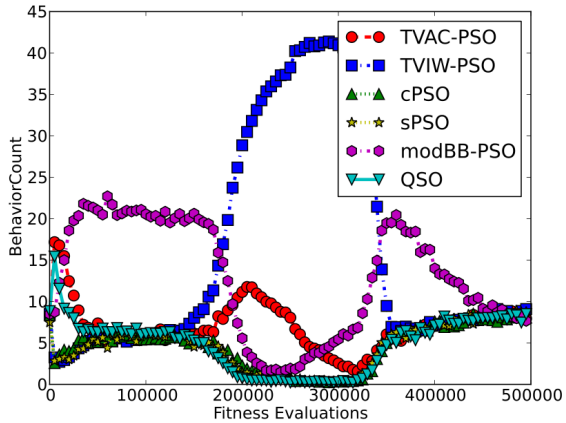
: Behavior profile plots for functions f_4, f_{12} and f_{27} in 30 dimensions



(a) 10D



(b) 30D

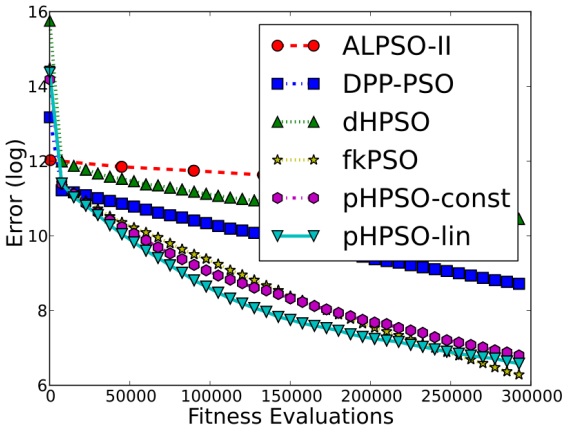


(c) 50D

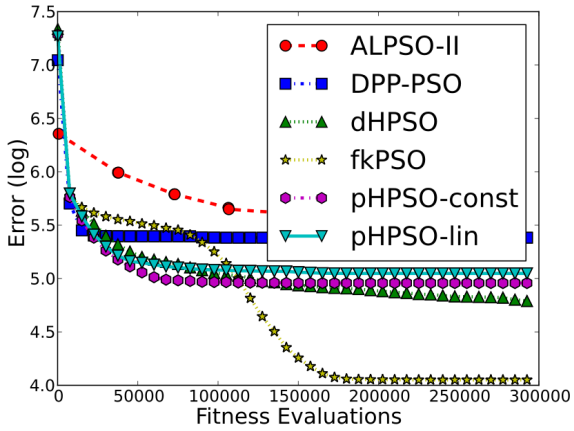
: Behavior profile plots for functions f_{10} in 10, 30 and 50 dimensions

Self-Adaptive Heterogeneous PSO

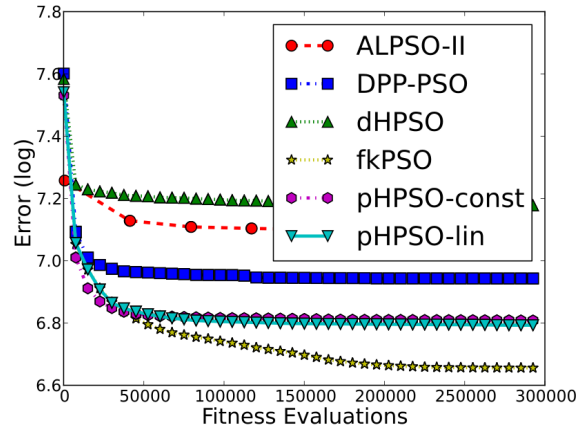
Convergence Plots



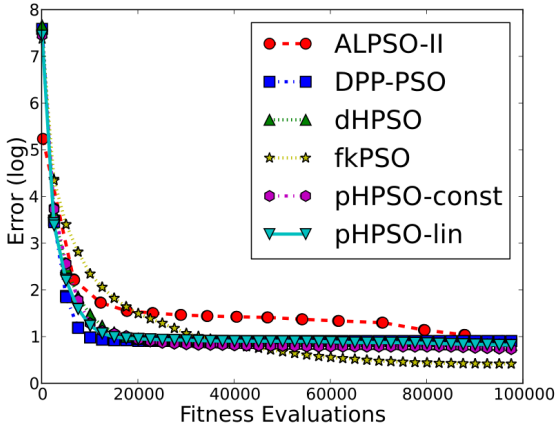
(a) f_4 - 30D

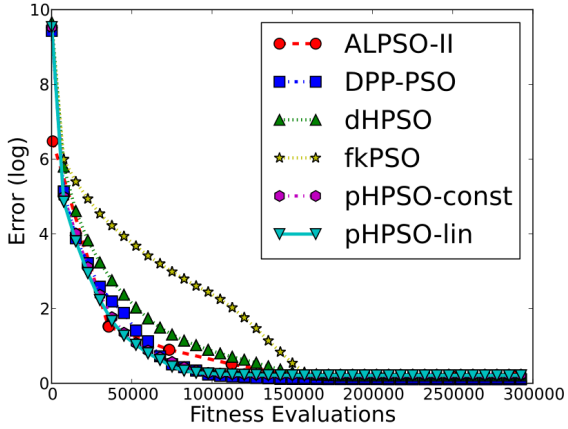


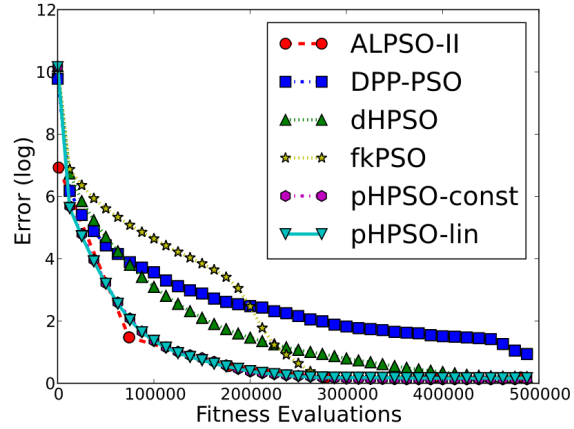
(b) f_{12} - 30D



(c) f_{27}- 30D







What is the issue?

- Particles should be allowed to change their behavior during the search process
- Change should occur when the behavior no longer contribute to improving solution quality

Current approaches:

- Select at every iteration
- Select when pbest position stagnates over a number of iterations

The following behavior selection schedules are proposed:

- Periodic
  - Behaviors change every m iterations
  - Small m prevents bad behaviors from being used too long, but may provide insufficient time to determine the desirability of behaviors
  - Larger m may waste time on bad behaviors, but sufficient time to "learn" good behaviors
- Random
  - Select irregular intervals, based on some probability
- Fitness stagnation
  - Select when fitness of the particle's position does not improve for m iterations

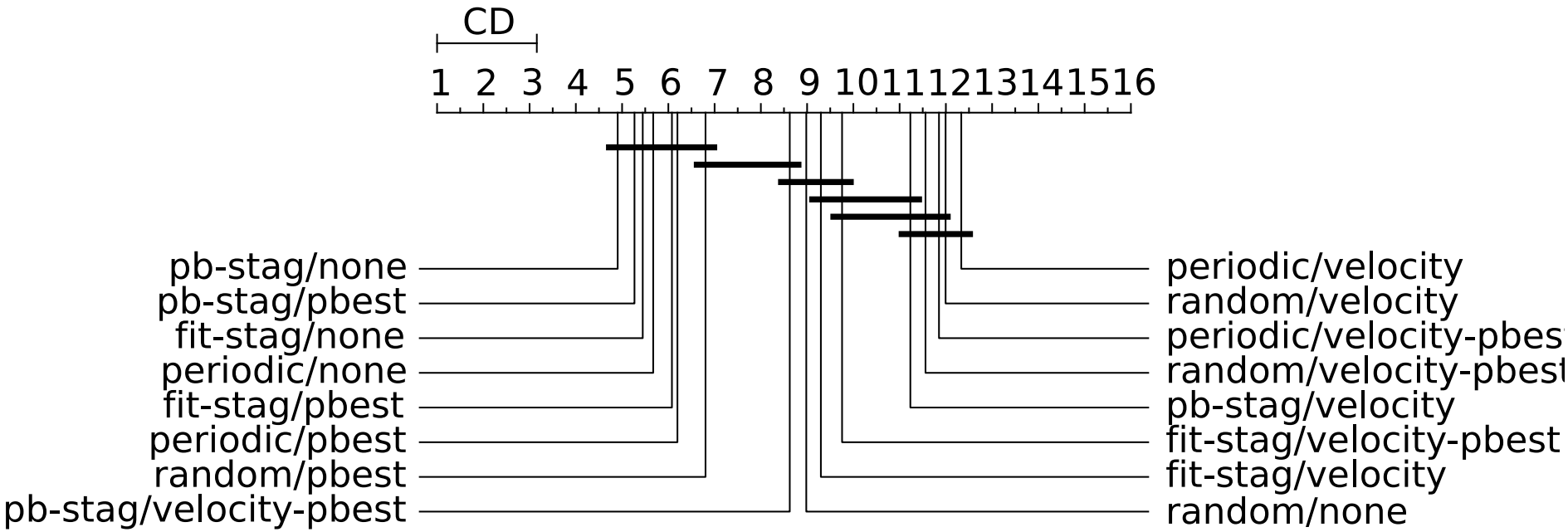
The following particle state resetting strategies are considered:

- No resetting of velocity and personal best position
- Reset velocity upon behavior change to random value
- Personal best reset, which sets particle to its pbest position after behavior change
- Reset both velocity and personal best

Self-Adaptive Heterogeneous PSO

Resetting Strategies (cont)

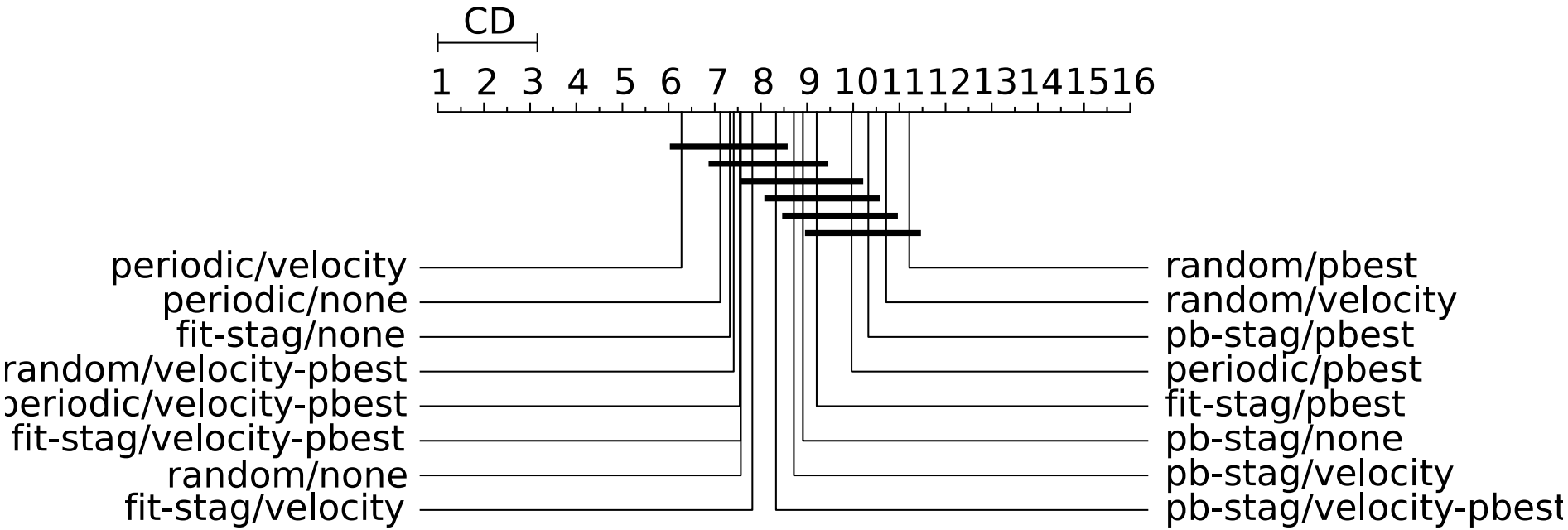
For the CEC 2013 function, using the same experimental procedure as previous, 16 different behavior changing schedules were evaluated for self-adaptive HPSO algorithms



: Critical difference diagram for dHPSO

Self-Adaptive Heterogeneous PSO

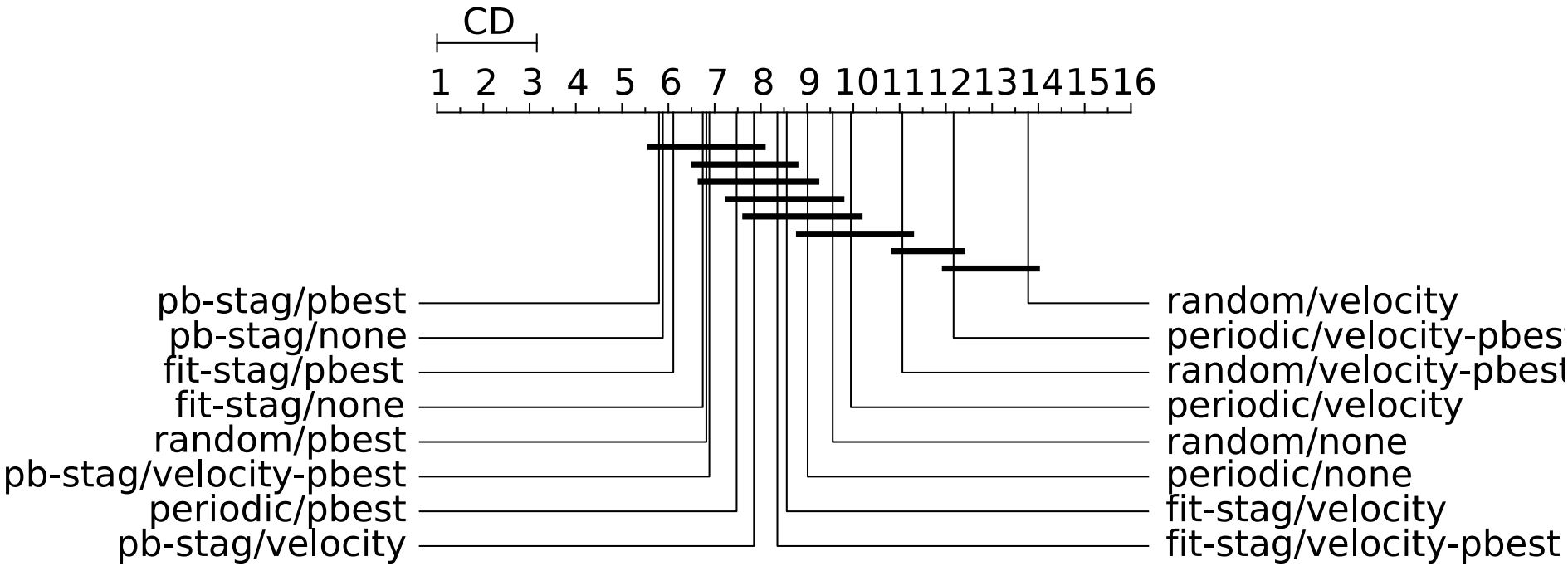
Resetting Strategies (cont)



: Critical difference diagram for f_k-PSO

Self-Adaptive Heterogeneous PSO

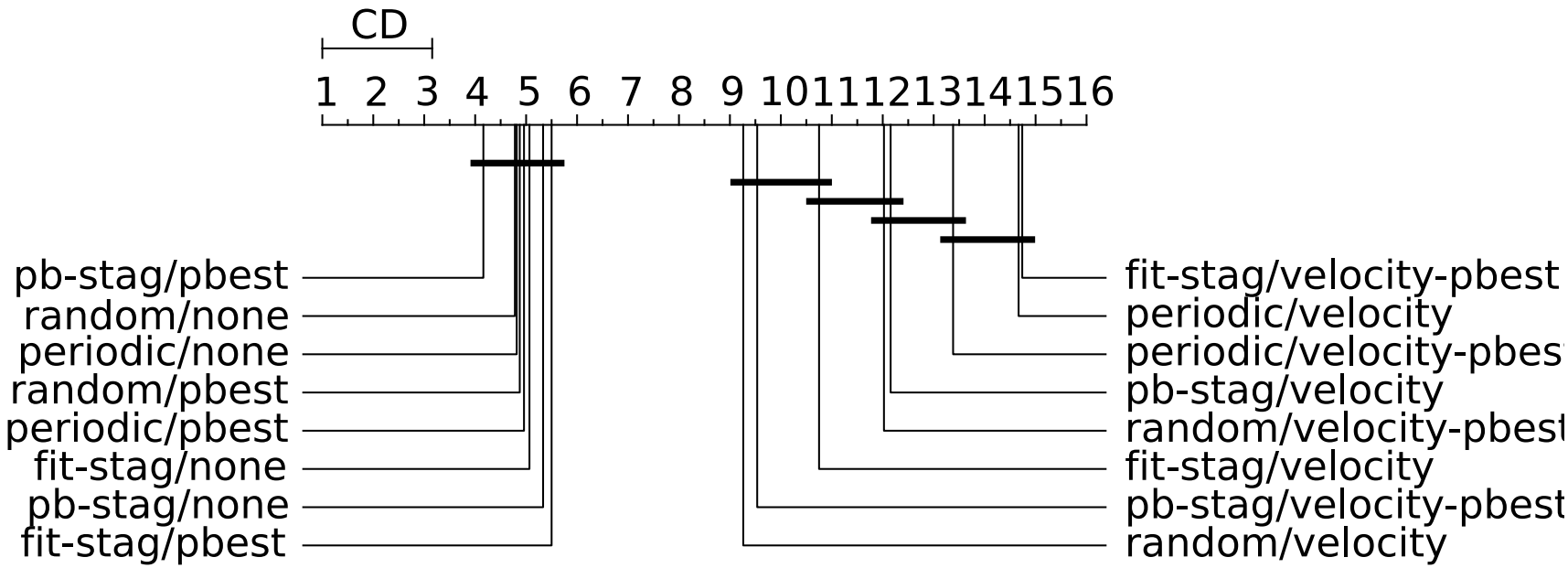
Resetting Strategies (cont)



: Critical difference diagram for pHPSO-const

Self-Adaptive Heterogeneous PSO

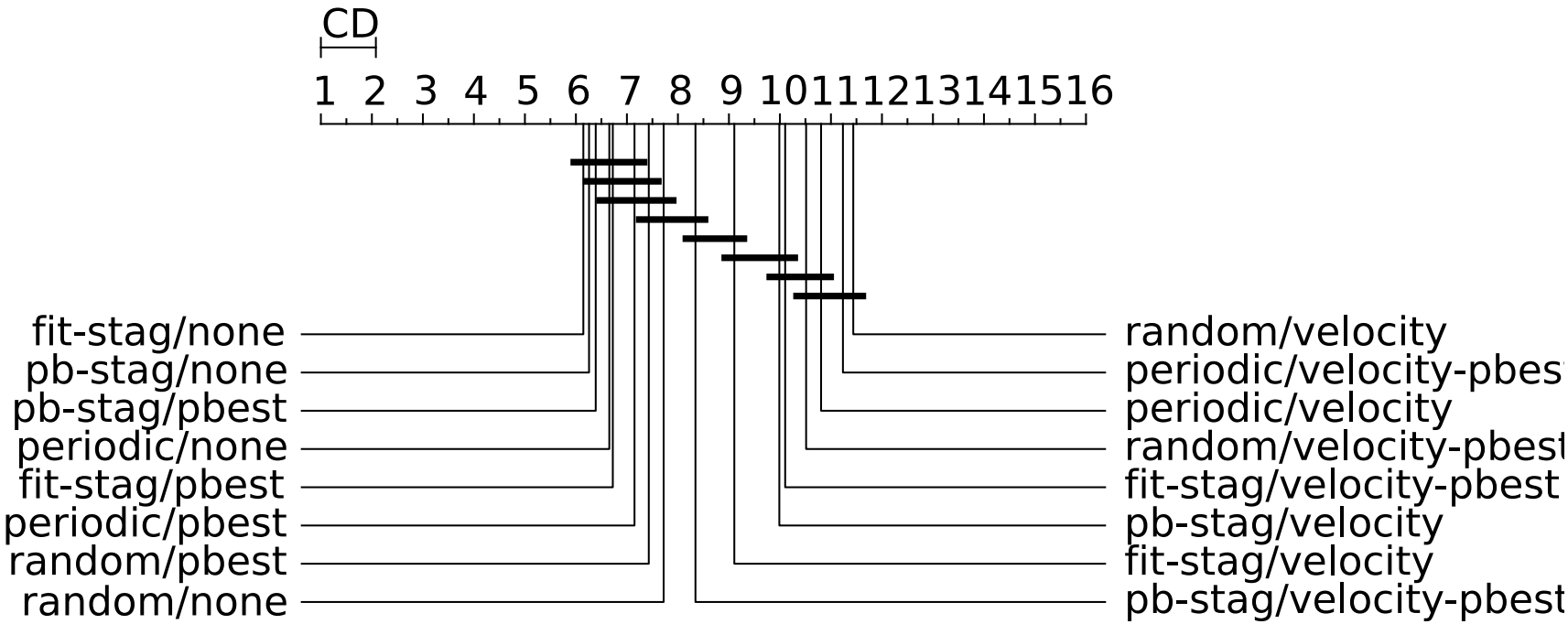
Resetting Strategies (cont)



: Critical difference diagram for pHPSO-lin

Self-Adaptive Heterogeneous PSO

Resetting Strategies (cont)



: Critical difference diagram for all the algorithms

Summary

- While the original PSO and most of its variants force all particles to follow the same behavior, different PSO variants show different search behaviors
- Heterogeneous PSO (HPSO) algorithms allow different particles to follow different search behaviors
- Proposed the following HPSO strategies:
  - Static HPSO: Behaviors randomly assigned upon initialization and do no change
  - Dynamic HPSO: Behaviors randomly selected when pbest position stagnates
  - Self-Adaptive HPSO:
    - Pheromone-based strategies, where probability of being selected is proportional to success
    - Frequency-based strategy, where behaviors are selected based on frequency of improving particle fitness

Summary

- sHPSO and dHPSO improve performance significantly in comparison with the individual behaviors
- sHPSO and dHPSO are highly scalable compared to individual behaviors
- pHPSO and f_k-HPSO perform better than other HPSO algorithms, with f_k-HPSO performing best
- Self-adaptive HPSO strategies show clearly how different behaviors are preferred at different points during the search process
- Proposed self-adaptive HPSO strategies are computationally less complex than other HPSO strategies
- Behavior changing schedules have been shown to have an effect on performance

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

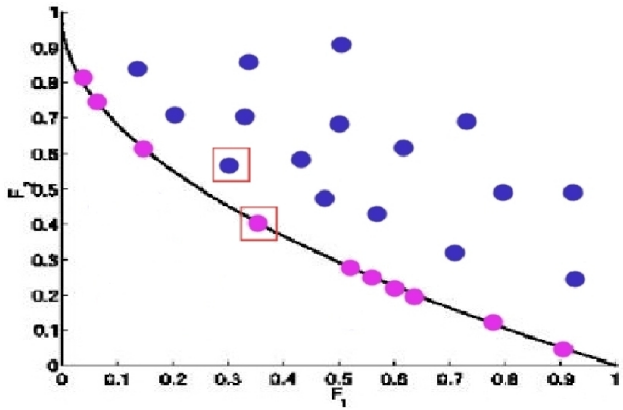
where

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

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

Goals when solving a MOOP: finding

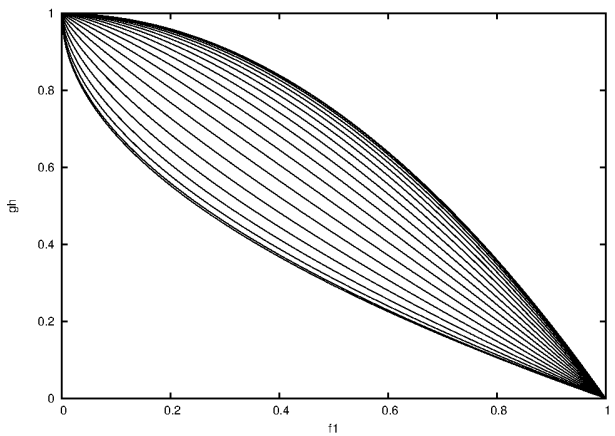
- the set of optimal trade-off solutions (POF)
- a diverse set of solutions



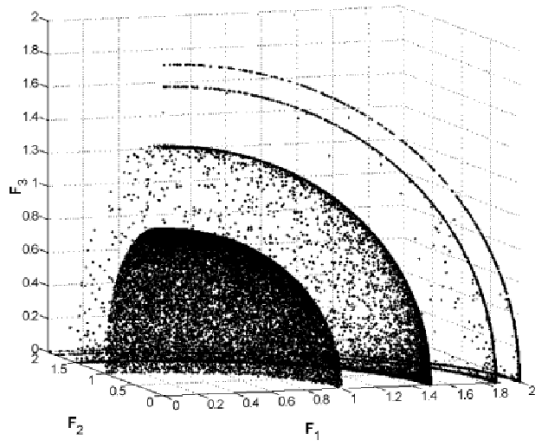
: Example of non-dominated solutions

Goals when solving a DMOOP:

- in addition to goals of solving a MOOP $\Rightarrow$ want to track the POF over time



(a) POF of dMOP2



(b) POF of FDA5

: Example of DMOOPs' POFs

The following gaps are identified in DMOO literature:

- Most research in MOO was done on static multi-objective optimization problems (SMOOPs) and dynamic single-objective optimization problems (DSOOPs)
- Even though PSO successfully solved both SMOOPs and DSOOPs
  - Mostly evolutionary algorithms (EAs) were developed to solve DMOOPs
  - Less than a handful of PSOs were developed for DMOO
- For DMOO, there is a lack of standard
  - benchmark functions, and
  - performance measures

$\Rightarrow$ Difficult to evaluate and compare dynamic multi-objective optimization algorithms (DMOAs)

An ideal MOO (static or dynamic) set of benchmark functions should

- Test for difficulties to converge towards the Pareto-optimal front (POF)
  - Multimodality
    - There are many POFs
    - The algorithm may become stuck on a local POF
  - Deception (∄)
    - There are at least to POFs
    - The algorithm is "tricked" into converging on the local POF
  - Isolated optimum (∄)
    - Fitness landscape have flat regions
    - In such regions, small perturbations in decisionvariables do not change objective function values
    - There is very little useful information to guide the search towards a POF

- Different shapes of POFs
  - Convexity or non-convexity in the POF
  - Discontinuous POF, i.e. disconnected sub-regions that are continuous ($\nexists$)
  - Non-uniform distribution of solutions in the POF
- Have various types or shapes of Pareto-optimal set (POS) ($\nexists$)
- Have decision variables with dependencies or linkages

An ideal DMOOP benchmark function suite should include problems with the following characteristics:

- Solutions in the POF that over time may become dominated
- Static POF shape, but its location in decision space changes
- Distribution of solutions changes over time
- The shape of the POFs should change over time:
  - from convex to non-convex or vice versa
  - from continuous to disconnected or vice versa
- Have decision variables with different rates of change over time
- Include cases where the POF depends on the values of previous POSs or POFs
- Enable changing the number of decision variables over time
- Enable changing the number of objective functions over time

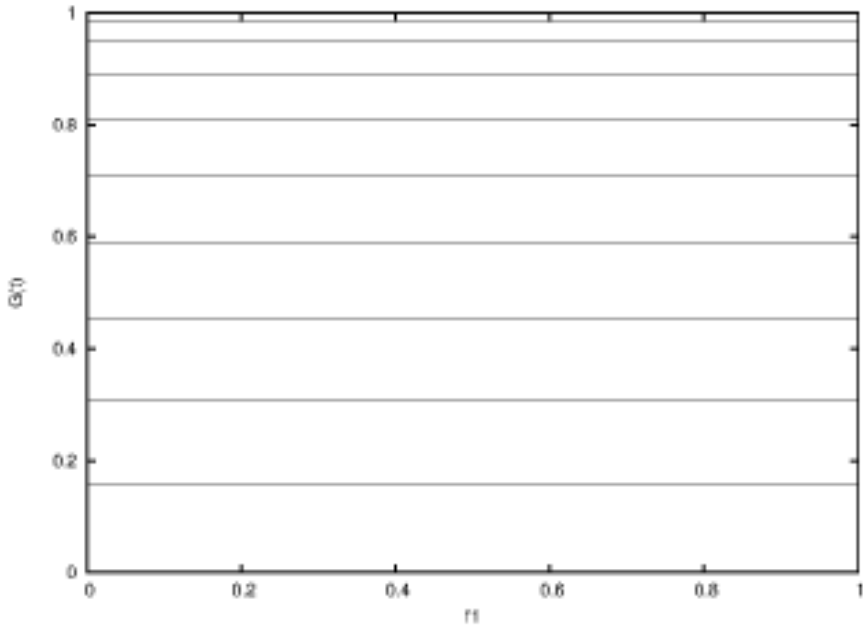
Four categories of dynamic environments for DMOOPs:

	POS	
POF	No Change	Change
No Change	Type IV	Type I
Change	Type III	Type II

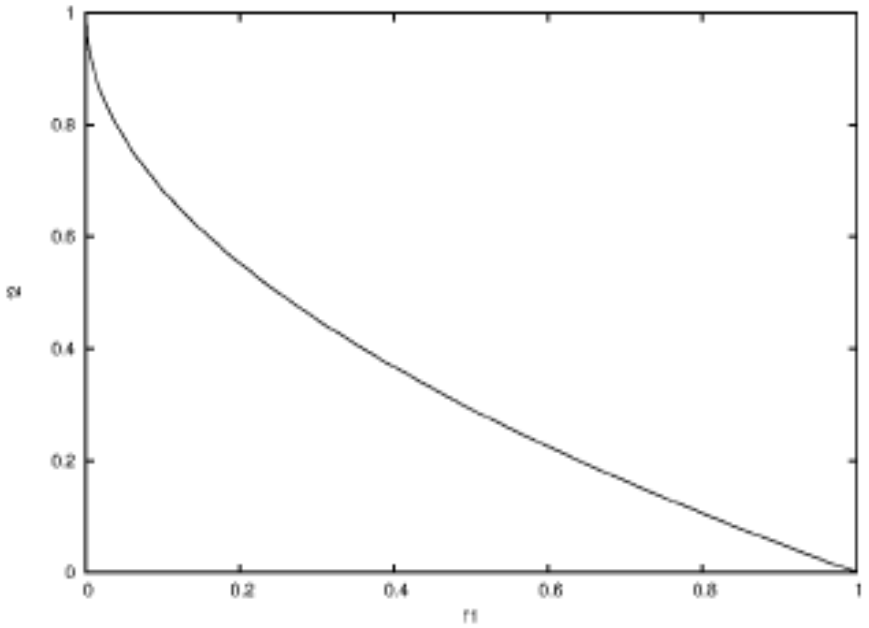
Other considerations:

- Frequency of change
- Severity of change

Has a convex POF



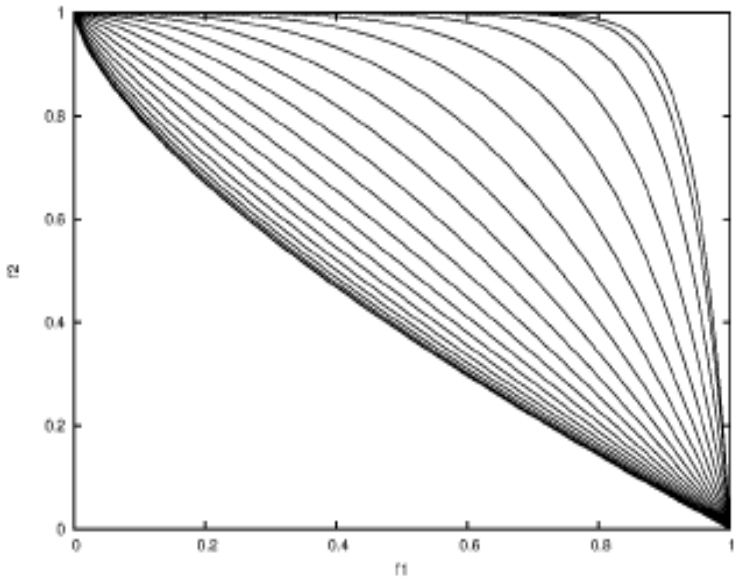
(a) POS



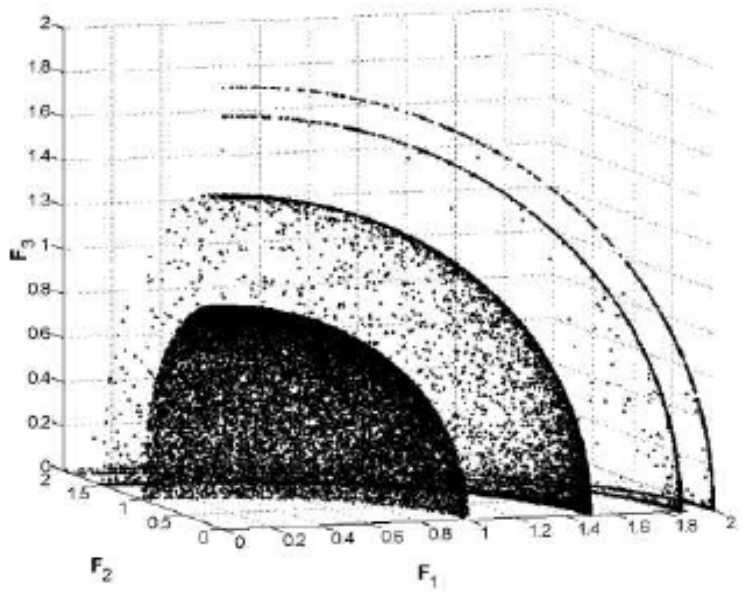
(b) POF

POF changes from convex to concave

Non-convex POF
Spread of solutions change over time



(a) FDA2



(b) FDA5

Discontinuous POF

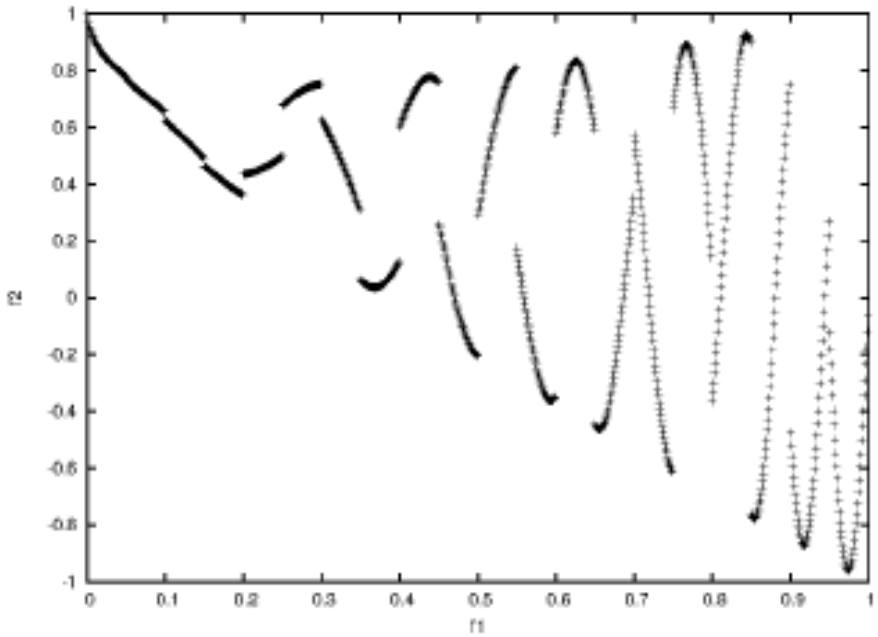
$$POF = 1 - \sqrt{f_1} - f_1 \sin(10\pi t f_1)$$

Discontinuous POF

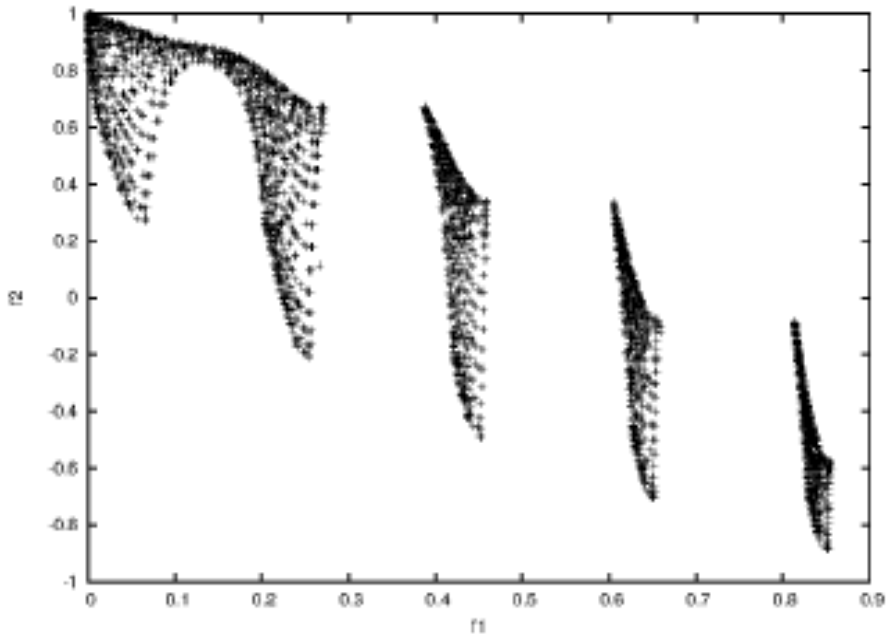
$$POF = 1 - \sqrt{f_1}^{H(t)} - f_1^{H(t)} \sin(10\pi t f_1)$$

$$H(t) = 0.75 \sin(0.5\pi t) + 1.25$$

$$t = \frac{1}{n_t} \lfloor \frac{\tau}{\tau_t} \rfloor$$



(a) POF of HE1



(b) POF of HE2

M Helbig, AP Engelbrecht, *Benchmarks for Dynamic Multi-Objective Optimisation Algorithms*, ACM Computing Surveys, 46(3), Article number 37, 2014

Comprehensive reviews and studies of performance measures exist for SMOO and DSOO

In the field of DMOO:

- no comprehensive overview of performance measures existed
- no standard set of performance measures existed

$\Rightarrow$ Difficult to compare DMOO algorithms

Therefore:

- a comprehensive overview of measures was done
- issues with currently used measures were highlighted

Accuracy Measures:

- Variational Distance (VD)
  - Distance between solutions in POS^* and POS'
  - POS' is a reference set from the true POS
  - Can be applied to the POF too
  - calculated just before a change in the environment, as an average over all environments
- Success Ratio (SR)
  - ratio of found solutions that are in the true POF
  - averaged over all changes

Diversity Measures:

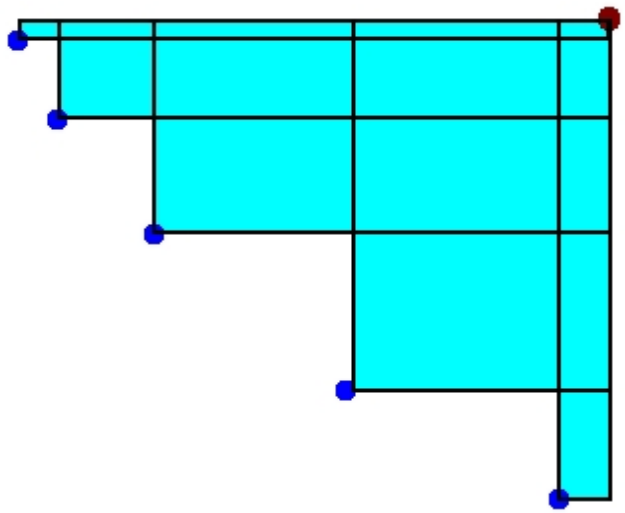
- Maximum Spread (MS')
  - length of the diagonal of the hyperbox that is created by the extreme function values of the non-dominated set
  - measures how well the POF^* covers POF'
- Path length (PL)
  - consider path lengths (length between two solutions)/path integrals
  - take the shape of the POF into account
  - difficult to calculate for many objectives and discontinuous POFs
- Set Coverage Metric (η)
  - a measure of the coverage of the true POF
- Coverage Scope (CS)
  - measures Pareto front extent
  - average coverage of the non-dominated set

Robustness:

- Stability
  - measures difference in accuracy between two time steps
  - low values indicate more stable algorithm
- Reactivity
  - how long it takes for an algorithm to recover after a change in the environment

Combined Measures:

- Hypervolume (HV) or Lebesque integral
  - reference vector is worst value in each objective
  - large values indicate better approximated front



- Hypervolume difference (HVD)

$$HVD = HV(POF') - HV(POF^*)$$

Dynamic Multi-Objective Optimization

Performance Measures (cont)

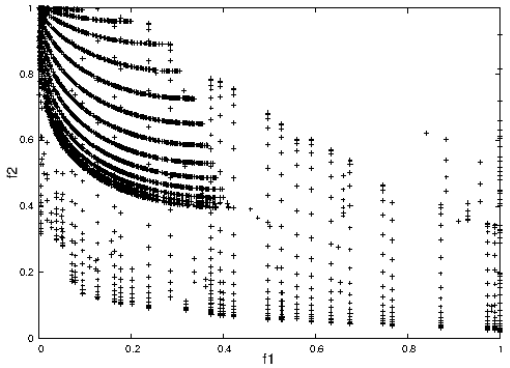
Issues arise with current DMOO performance measures when:

1. Algorithms lose track of the changing POF
2. The found POF contains outlier solutions
3. Boundary constraint violations are not managed
4. Calculated in the decision space

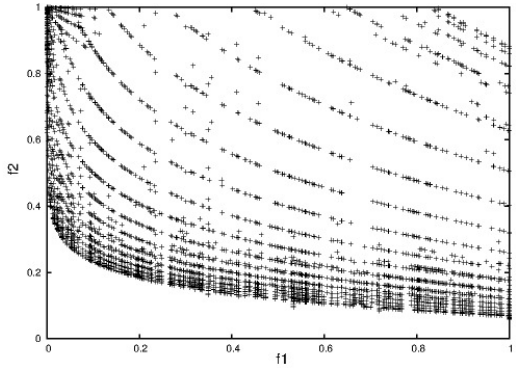
Dynamic Multi-Objective Optimization

Performance Measures (cont)

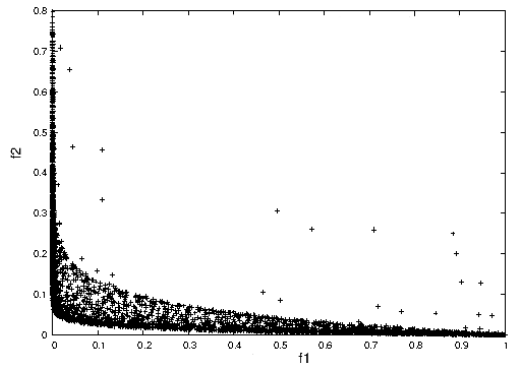
- DMOA loses track of changing POF, i.e. failed to track the moving POF
- POF changes over time in such a way that the HV decreases
  $\Rightarrow$ DMOAs that lose track of POF obtain the highest HV
- Issue of losing track of changing POF is unique to DMOO
- First observed where five algorithms solved the FDA2 DMOOP:
  - DVEPSO-A: uses clamping to manage boundary constraints
  - DVEPSO-B: uses dimension-based reinitialization to manage boundary constraints
  - DNSGAII-A: %individuals randomly selected and replaced with new randomly created individuals
  - DNSGAII-B: %individuals replaced with mutated individuals, randomly selected
  - dCOEA: competitive coevlutionary EA



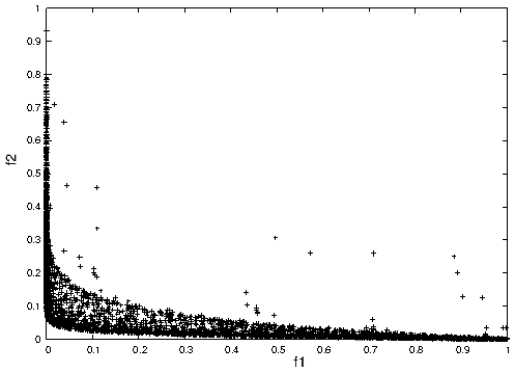
(c) DVEPSO-A



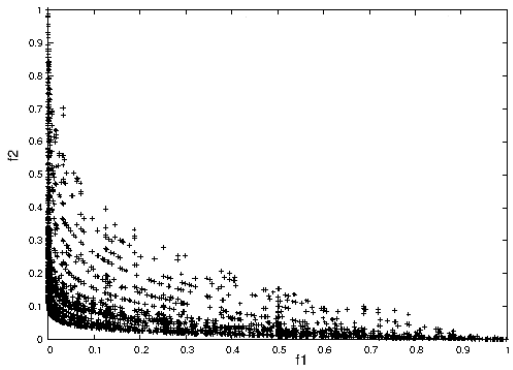
(d) DVEPSO-B



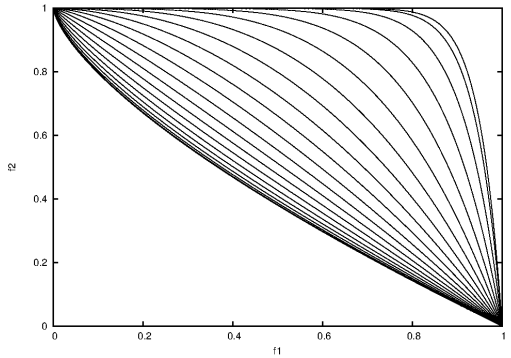
(e) DNSGA-II-A



(f) DNSGA-II-B



(g) dCOEA



(h) *POF* of FDA2

: *POF* and *POF** found by various algorithms for FDA2 with $n_t = 10$, $\tau_t = 10$ and 1000 iterations

Performance measure values for $\tau_t = 10$ (bold indicates best values)

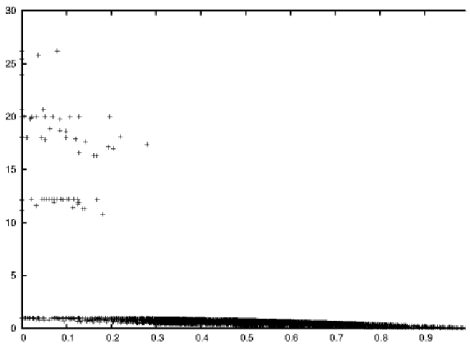
: Performance Measure Values for FDA2

τ_t	Algorithm	NS	S	HVR	Acc	Stab	VD	MS	R
10	DVEPSO-A	**73.4**	0.00118	0.99533	0.97848	0.00049	0.45824	**0.90878**	4
10	DVEPSO-B	63	0.00391	0.99905	0.98157	0.00029	**0.43234**	0.88916	3
10	DNSGAII-A	39.4	0.00044	1.0044	0.98681	$9.565\text{x}10^{-06}$	0.71581	0.77096	2
10	DNSGAII-B	39.6	**0.00042**	**1.00441**	**0.98683**	**$9.206\text{x}10^{-06}$**	0.71681	0.77866	**1**
10	dCOEA	38.4	0.00051	1.00209	0.98454	0.00122	0.70453	0.61923	5

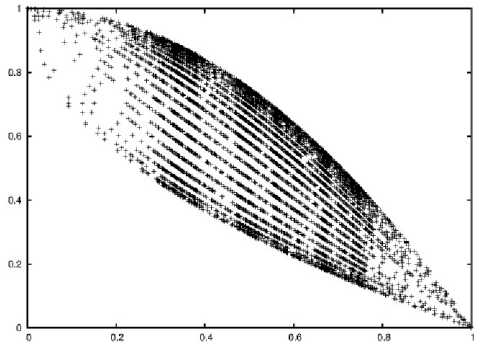
When the environment changes frequently

- Algorithm finds non-dominated solutions further away from the true *POF*
- In time available, algorithm does not find any solutions that dominate outliers

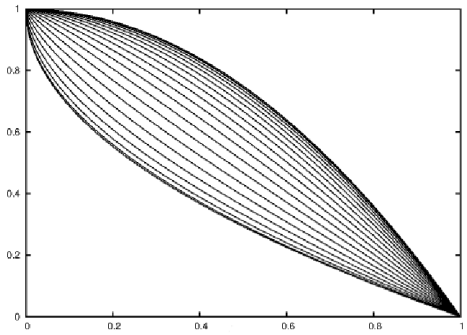
  $\Rightarrow$ *POF** for a specific time step may contain outliers
- Issue of outliers is applicable to both SMOO and DMOO



(a) With outliers



(b) Zoomed into POF region of (a)



(c) *POF* of dMOP2

: Example of a *POF** that contains outlier solutions.

Outliers will skew results obtained using:

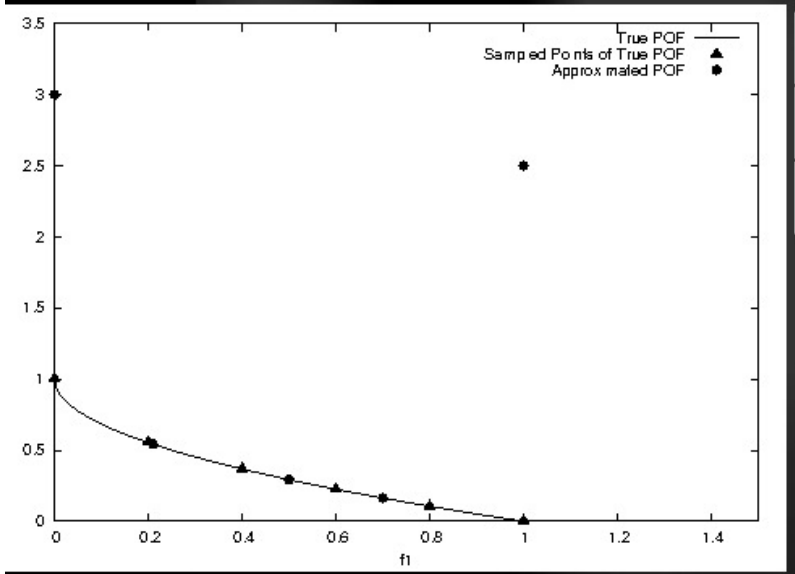
- distance-based performance measures, such as GD, VD, PL, CS (large distances)
- performance measures that measure the spread of the solutions, such as *MS* (large diagonal), and
- the HV performance measures, such as HV, HVR (outliers become reference vector)

Dynamic Multi-Objective Optimization

Performance Measures (cont)

For FDA1:

- GD & VD values much larger with outliers present (smaller values are better)
- Spread is incorrectly inflated
- For HV, outliers influence the selection of reference vectors, resulting in larger HV values



Outliers	GD	VD	MS
Yes	2.05565	4.596574	0.91833
No	0.00942	0.016311	0.4342

Outliers	HV	HVR	acc_{alt}
Yes	2.49898	0.84461	0.45974
No	0.69798	0.91994	0.06074

Dynamic Multi-Objective Optimization

Performance Measures (cont)

- Solutions tend to move outside the boundary constraints
- Most unconstrained DMOOPs have boundary constraints
- An algorithm does not manage boundary constraint violations
  $\Rightarrow$ infeasible solutions may be added to POF^*
- Infeasible solutions may dominate feasible solutions in POF^*
  $\Rightarrow$ feasible solutions removed from POF^*
- Infeasible solutions may cause misleading results

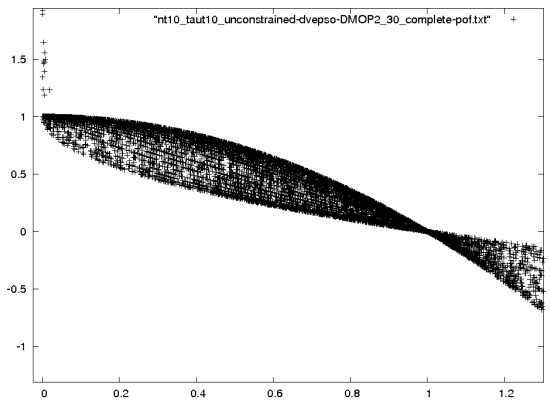
: HVR values for dMOP2

Algorithm	**HVR**
$DVEPSO_u$	1.00181
$DVEPSO_c$	0.99978

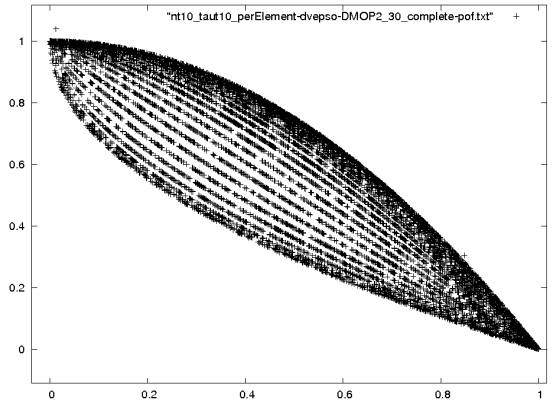
Dynamic Multi-Objective Optimization

Performance Measures (cont)

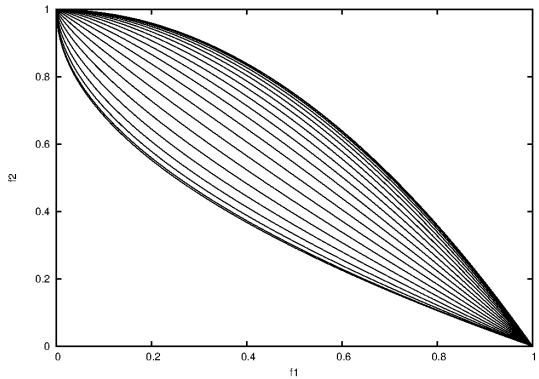
- When comparing various algorithms with one another
  $\Rightarrow$ important to check that all algorithms manage boundary contraints
- Issue of boundary constraint violations is applicable to both SMOO and DMOO



(a) Contains Infeasible Solutions



(b) No Infeasible Solutions



(c) True POF

: Example of a *POF** that contains infeasible solutions due to boundary constraint violations

Dynamic Multi-Objective Optimization

Performance Measures (cont)

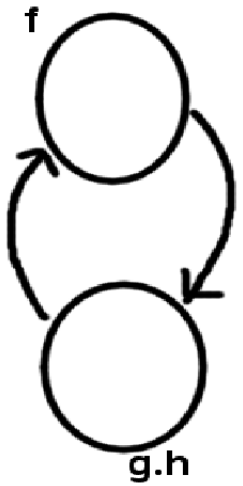
- Accuracy measures (*VD* or *GD*) can be calculated in decision or objective space
- In objective space, *VD* = distance between POF^* and POF'
- One goal of solving a DMOOP is to track the changing POF
  $\Rightarrow$ Accuracy should be measured in objective space
- *VD* in decision space measures the distance between POS^* and *POS*
- May be useful to determine how close POS^* is from *POS*
- It may occur that even though algorithm's POS^* is very close to *POS*, POF^* is quite far from *POF*
- Small change in POS may result in big change in POF, so calculation wrt decision space will be misleading

M Helbig, AP Engelbrecht, *Performance Measures for Dynamic Multi-objective Optimisation Algorithms*, Information Sciences, 250:61–81, 2013

Dynamic Multi-Objective Optimization

Vector-Evaluated Particle Swarm Optimization (VEPSO)

- introduced by Parsopoulos and Vrahatis
- based on the Vector Evaluated Genetic Algorithm
- each swarm solves only one objective
- swarms share knowledge with each other
- shared knowledge is used to update particles' velocity



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

Dynamic Multi-Objective Optimization

VEPSO (cont)

The VEPSO has been extended to include:

- an archive of non-dominated solutions
- boundary contraint management
- various ways to share knowledge among sub-swarms
- updating pbest and gbest using Pareto-dominance

Dynamic Multi-Objective Optimization

VEPSO (cont)

The archive on non-dominated solutions

- has a fixed size
- a new solution that is non-dominated wrt all solutions in the archive is added to the archive if there is space
- a new non-dominated solution that is dominated by any solution in the archive is rejected, and not added to the archive
- if a new solution dominates any solution in the archive, the dominated solution is removed
- if the archive is full, a solution from a dense area of the POF* is removed

Dynamic Multi-Objective Optimization

VEPSO (cont)

If a particle violates a boundary constraint in decision space, one of the following strategies can be followed:

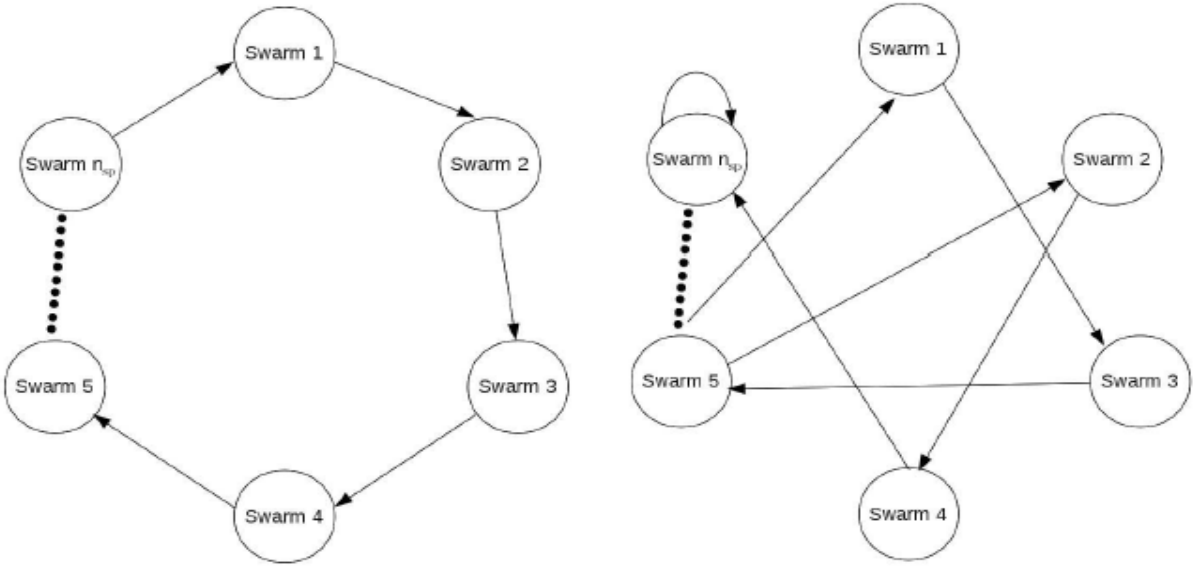
- do nothing.....
- clamp violating dimensions at the boundary, or close to the boundary
- deflection (bouncing) – invert the search direction
- dimension-based reinitialization to position within boundaries
- reinitialize entire particle
  - keep current velocity, or
  - reset velocity

Dynamic Multi-Objective Optimization

VEPSO (cont)

Knowledge transfer strategies (KTS):

- ring KTS
- random KTS
- parent-centric based crossover on non-dominated solutions KTS



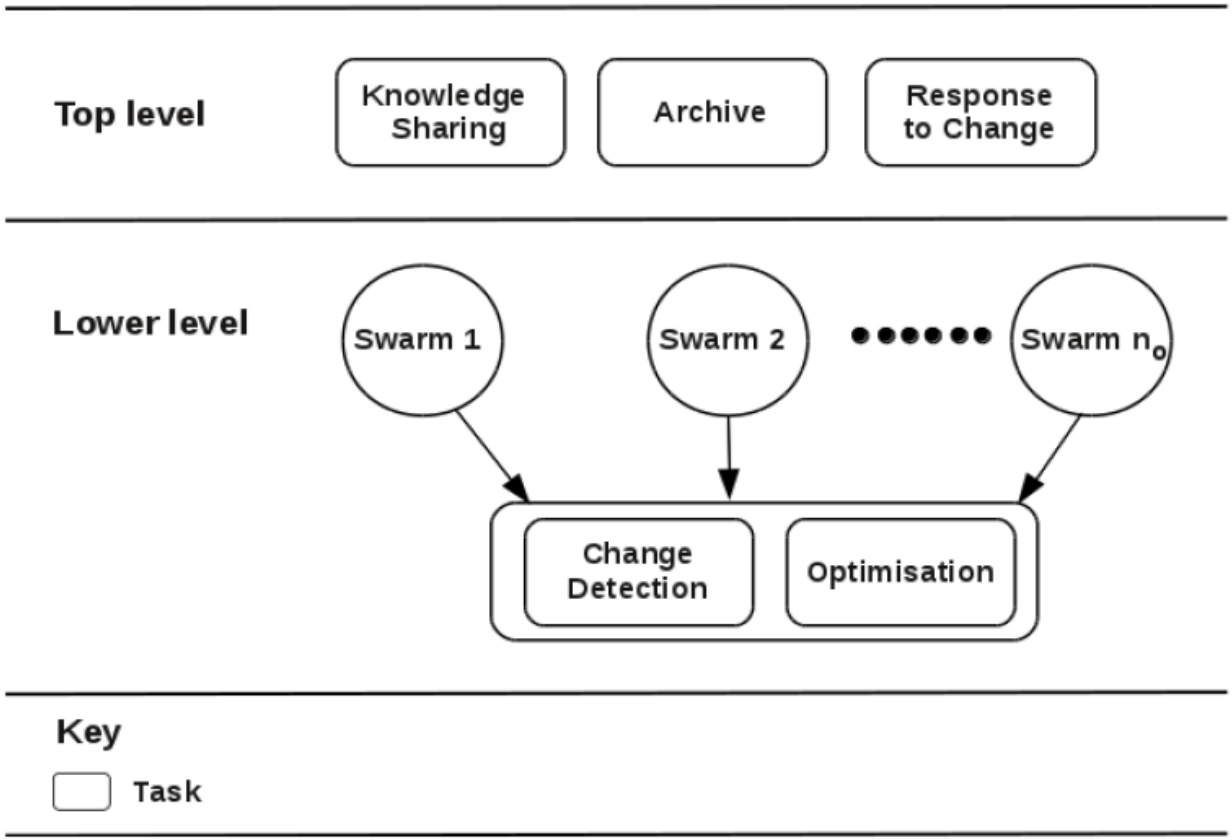
(a) Ring Topology (b) Random Topology

Dynamic Multi-Objective Optimization

Dynamic VEPSO (DVEPSO)

Adapted the extended VEPSO for dynamic environments to include:

- change detection using sentry particles
- change responses



Dynamic Multi-Objective Optimization

DVEPSO (cont)

```
1.  for number of iterations do
2.      check whether a change has occurred
3.      if change has occurred
4.          respond to change
5.          remove dominated solutions from archive
6.      perform PSO iteration
7.      if new solutions are non-dominated
8.          if space in archive
9.              add new solutions to archive
10.         else
11.             remove solutions from archive
12.             add new solutions to archive
13.     select sentry particles
```

The following change detection strategy is used:

- Sentry particles are used
- Objective function value of sentry particles evaluated at beginning and end of iteration
- If there is a difference, then a change has occured
- If any one or more of the sub-swarms detect a change, then a response is triggered

When a change is detected, a response is activated for the affected sub-swarm

- Re-evaluation
  - all particles and best positions re-evaluated
  - remove stale information
- Re-initialization
  - percentage of particles in sub-swarm(s) re-initialized
  - introduces diversity
- Reset pbest (local guide) to current particle position
- Determine a new gbest (global guide) position

Dynamic Multi-Objective Optimization

DVEPSO (cont)

- Archive update:
  - remove all solutions, or
  - remove non-dominated solutions with large changes in objective function values, or
  - re-evaluate solutions
    - remove solutions that have become dominated, or
    - apply hillclimbing to adapt dominated solution to become non-dominated

Dynamic Multi-Objective Optimization

DVEPSO (cont)

- Guide updates:
  - local guides vs global guides
  - update by not using dominance
  - use dominance relation

Experimental setup:

- 30 runs of 1000 iterations each
- 15 benchmark functions:
  - change frequency, τ_t: 10 (1000 changes), 25 or 50
  - change severity, n_t: 1, 10, 20
- Three performance measures:
  - #non-dominated solutions
  - accuracy: $|HV(POF'(t)) - HV(POF^*(t))|$
  - stability: $\max\{acc(t-1) - acc(t)\}$

Analysis of data:

- pairwise Mann-Whitney U tests
- statistical difference $\Rightarrow$ winner awarded a win, loser a loss
- ranked based on $diff = \#wins - \#losses$ with regards to:
  - each performance measure (measured over all DMOOPs and n_t-τ_t)
  - each environment (n_t-τ_t) (measured over all DMOOPs) and performance measures)
  - each DMOOP type with regards to:
    - each performance measure
    - each environment
  - overall performance (measured over all DMOOPs, n_t-τ_t and performance measures)
- best configuration of DVEPSO selected

- DVEPSO compared against four DMOAs:
  1. DNSGA-II-A: NSGA-II replaces % of random individuals with new individuals
  2. DNSGA-II-B: NSGA-II replaces % of random individuals with mutated individuals
  3. dCOEA: dynamic competitive-cooperative coevolutionary algorithm
  4. DMOPSO: MOPSO adapter for DMOO
- Parameters for each algorithm optimised for same set of DMOOPs

Overall results with regards to performance measures:

PM	Results	DMOO Algorithm				
		DNSGA-II-A	**DNSGA-II-B**	**dCOEA**	**DMOPSO**	**DVEPSO**
acc	Wins	94	109	129	118	119
acc	Losses	120	104	164	86	95
acc	Diff	-26	5	-35	32	24
acc	Rank	4	3	5	**1**	2
stab	Wins	67	94	39	117	96
stab	Losses	89	66	200	39	50
stab	Diff	-22	28	-161	78	46
stab	Rank	4	3	5	**1**	2
NS	Wins	185	187	116	53	111
NS	Losses	83	78	202	195	129
NS	Diff	102	109	-86	-142	-18
NS	Rank	2	**1**	4	5	3

Overall results with regards to performance measures for Type I DMOOPs:

- *acc*:
  - Best:DVEPSO, dCOEA; Second: DNSGA-II-B; Worst: DNSGA-II-A
  - More wins than losses: DVEPSO and dCOEA
- *stab*:
  - Best: DNSGA-II-B, DMOPSO; Second: dCOEA; Worst: DVEPSO
  - More wins than losses: DNSGA-II-B and DMOPSO
- *NS*:
  - Best: DNSGA-II-B; Second: DNSGA-II-A; Worst: DVEPSO
  - More wins than losses: DNSGA-II-B

Overall results for Type I DMOOPs:

- Best: DNSGA-II-B; Second: dCOEA; Worst: DNSGA-II-A
- More wins than losses: DNSGA-II-A

- DVEPSO:
  - struggled to converge to POF of dMOP3 (density of non-dominated solutions changes over time)
  - only algorithm converging to POF of DIMP2 (each decision variable have a different rate of change)

Results with regards to performance measures for Type II DMOOPs:

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	*acc*	Wins	55	55	36	56	63
all	all	*acc*	Losses	43	41	106	41	34
all	all	*acc*	Diff	12	14	-70	15	29
all	all	*acc*	Rank	4	3	5	2	**1**
all	all	*stab*	Wins	36	45	18	53	59
all	all	*stab*	Losses	43	30	104	20	14
all	all	*stab*	Diff	-7	15	-86	33	45
all	all	*stab*	Rank	4	3	5	2	**1**
all	all	*NS*	Wins	47	49	118	108	47
all	all	*NS*	Losses	95	91	62	34	87
all	all	*NS*	Diff	-48	-42	56	74	-40
all	all	*NS*	Rank	5	4	2	**1**	3

- Overall results for Type II DMOOPs:

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	all	Wins	138	149	172	217	169
all	all	all	Losses	181	162	272	95	135
all	all	all	Diff	-43	-13	-100	122	34
all	all	all	Rank	4	3	5	**1**	2

Results with regards to performance measures for Type III DMOOPs:

n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	*acc*	Wins	32	43	70	53	36
all	all	*acc*	Losses	60	50	43	32	49
all	all	*acc*	Diff	-28	-7	27	21	-13
all	all	*acc*	Rank	5	3	**1**	2	4
all	all	*stab*	Wins	25	36	15	57	42
all	all	*stab*	Losses	28	22	82	7	22
all	all	*stab*	Diff	-3	14	-67	50	20
all	all	*stab*	Rank	4	3	5	**1**	2
all	all	*NS*	Wins	17	12	81	72	57
all	all	*NS*	Losses	73	78	48	17	36
all	all	*NS*	Diff	-56	-66	33	55	21
all	all	*NS*	Rank	4	5	2	**1**	3

- Overall results for Type III DMOOPs:

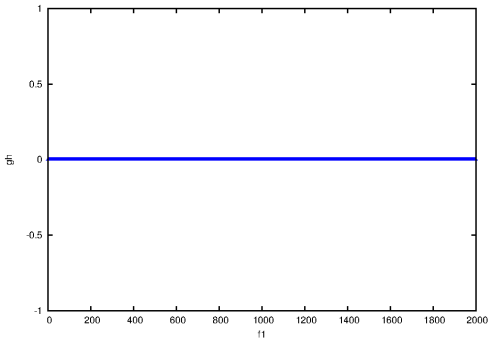
n_t	τ_t	PM	Results	DMOO Algorithm				
				DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
all	all	all	Wins	74	91	166	182	135
all	all	all	Losses	161	150	173	56	107
all	all	all	Diff	-87	-59	-7	126	28
all	all	all	Rank	5	4	3	**1**	2

- DVEPSO struggled to converge to discontinuous POFs

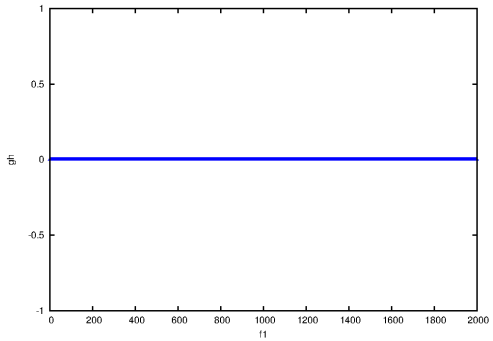
- Overall results:

Results	DMOO Algorithm				
	DNSGA-II-A	DNSGA-II-B	dCOEA	DMOPSO	DVEPSO
Wins	266	303	435	455	384
Losses	429	386	499	233	290
Diff	-163	-83	-64	222	94
Rank	5	4	3	**1**	2

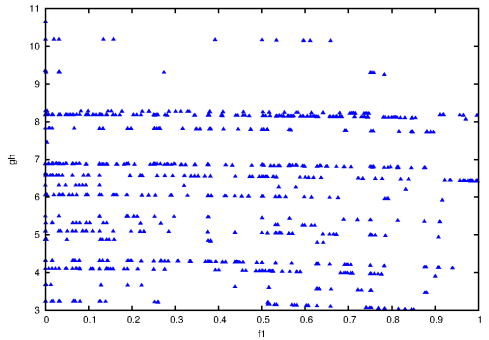
- DVEPSO second best
- However, performance is very problem-dependant



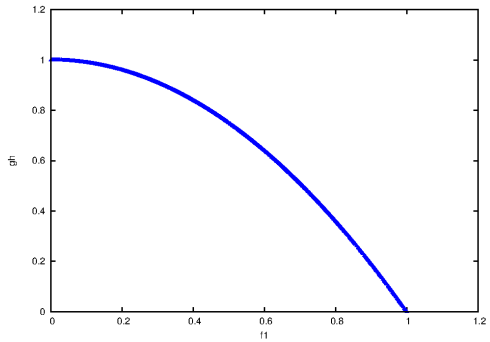
(a) DNSGA-II-A



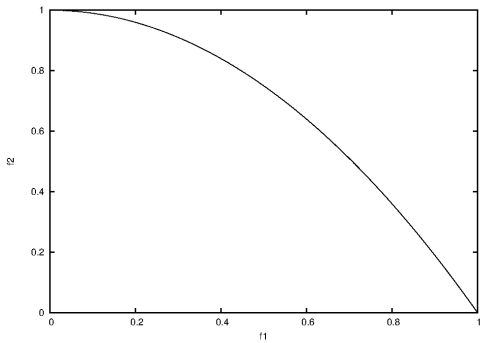
(b) DNSGA-II-B



(c) dCOEA

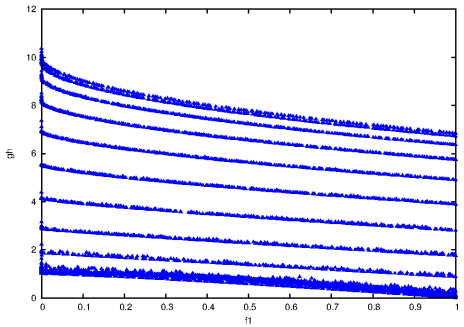


(d) DVEPSO

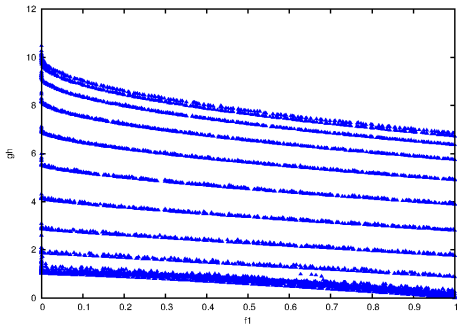


(e) *POF*

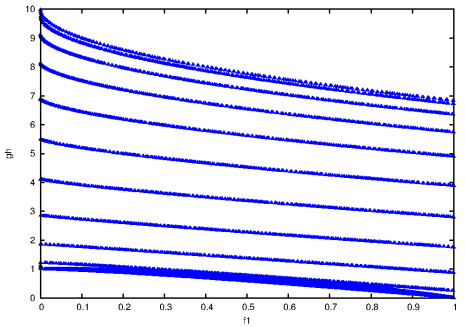
: *POF** for DIMP2 for $n_t = 10$ and $\tau_t = 10$. $NS(DMOPSO) = 0$.



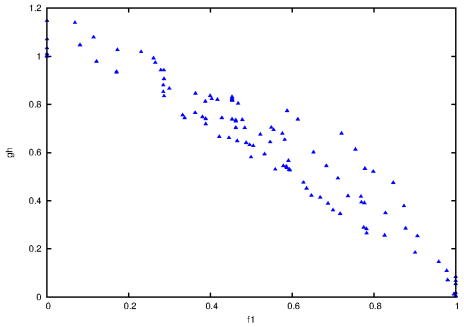
(a) DNSGA-II-A



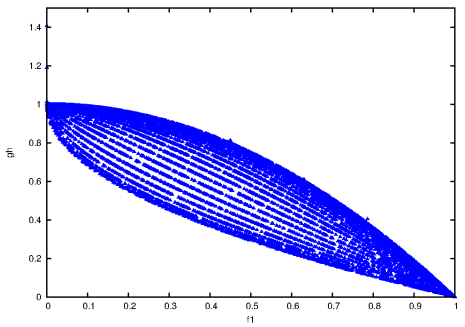
(b) DNSGA-II-B



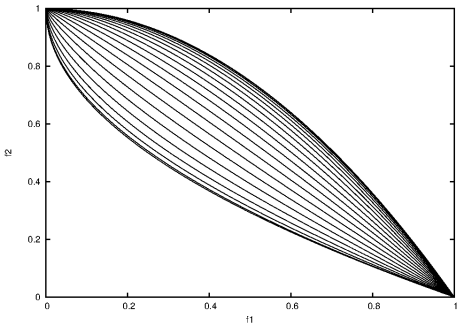
(c) dCOEA



(d) DMOPSO

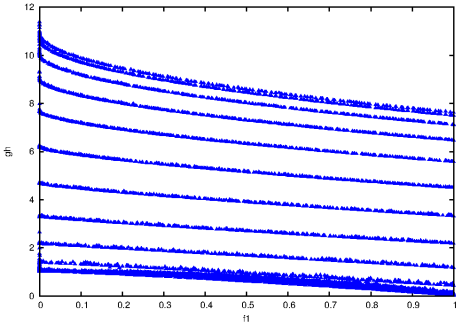


(e) DVEPSO

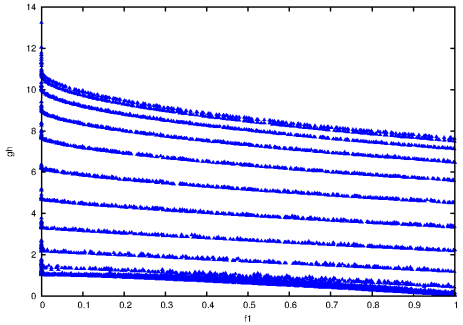


(f) *POF*

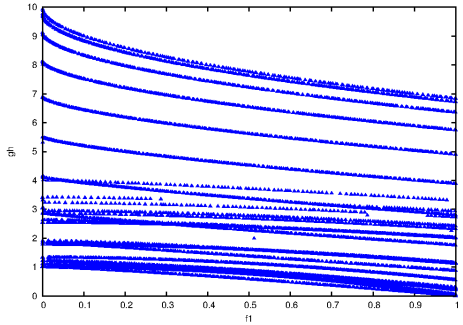
: POF^* for dMOP2 for $n_t = 10$ and $\tau_t = 10$. $NS(DMOPSO) = 0$



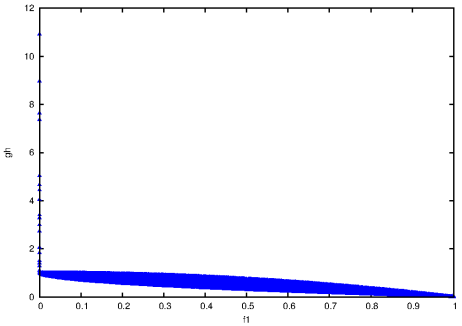
(a) DNSGA-II-A



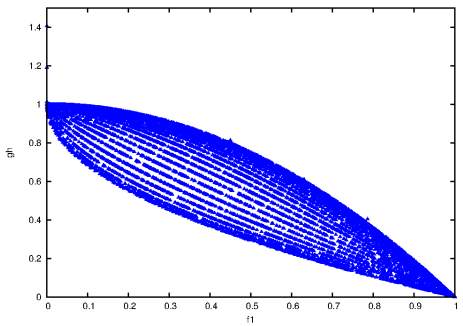
(b) DNSGA-II-B



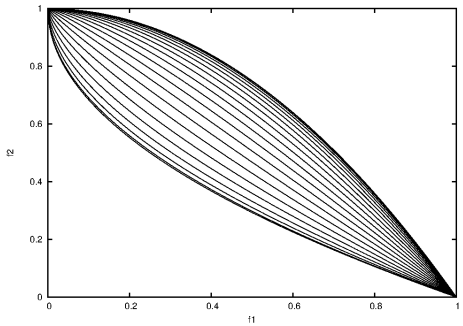
(c) dCOEA



(d) DVEPSO

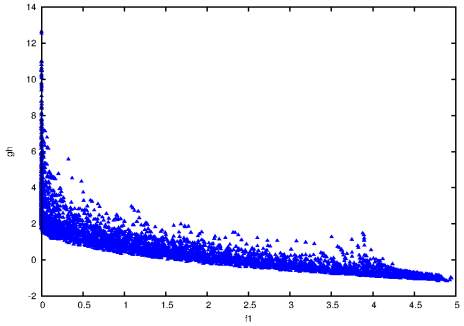


(e) DVEPSO - Zoomed

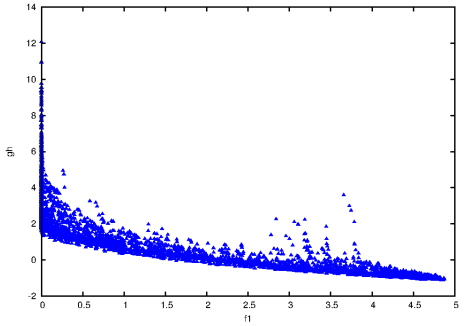


(f) *POF*

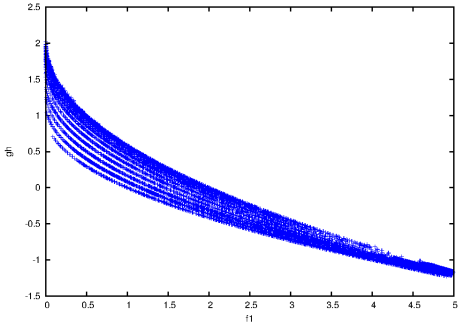
: POF^* for dMOP2$_{dec}$ for $n_t = 10$ and $\tau_t = 10$. $NS(DMOPSO) = 0$



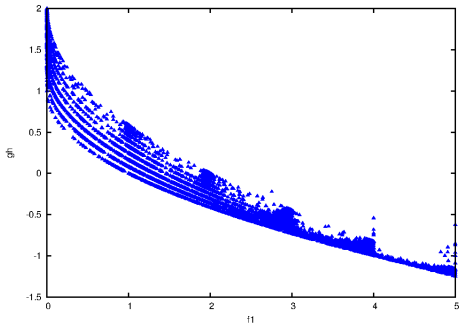
(a) DNSGA-II-A



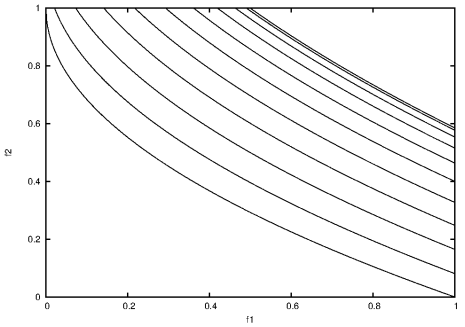
(b) DNSGA-II-B



(c) dCOEA

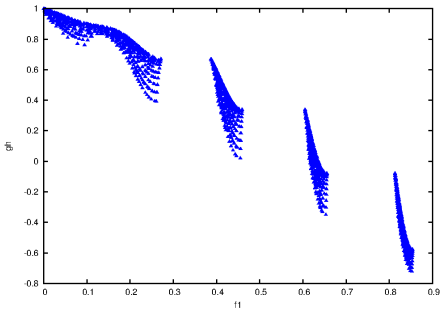


(d) DVEPSO

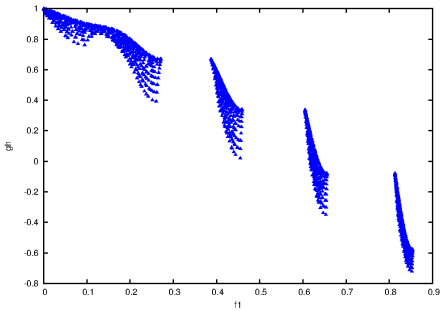


(e) *POF*

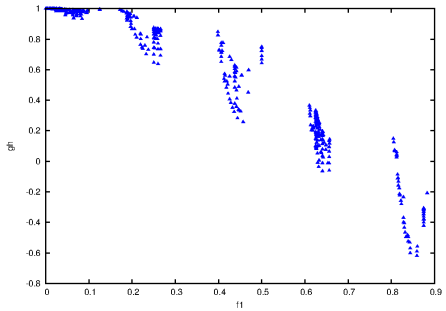
: POF^* for FDA3 for $n_t = 10$ and $\tau_t = 10$. $NS(DMOPSO) = 0$



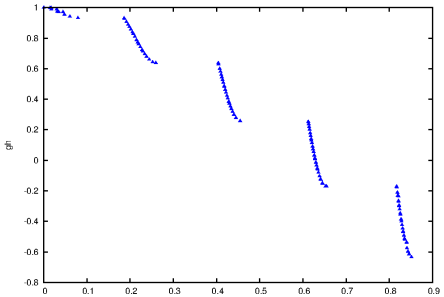
(a) DNSGA-II-A



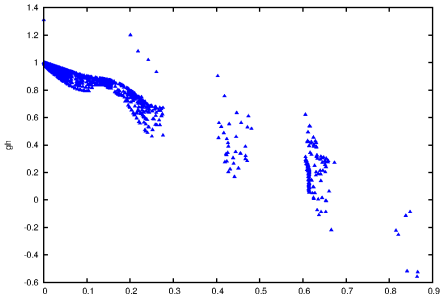
(b) DNSGA-II-B



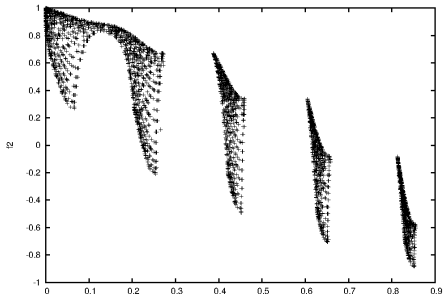
(c) dCOEA



(d) DMOPSO



(e) DVEPSO



(f) *POF*

: POF^* for HE2 for $n_t = 10$ and $\tau_t = 50$

Heterogeneous Dynamic Vector-Evaluated Particle Swarm Optimization

HDEVEPSO

The heterogeneous DVEPSO

- Each sub-swarm is an HPSO
- Behavior selected when change in sub-swarm detected, or
- pbest stagnated for a number of iterations

Heterogeneous Dynamic Vector-Evaluated Particle Swarm Optimization

Results

DMOOP Type	PM	Results	DMOO Algorithm							
			$DVEPSO_d$	$HDVEPSO_{ws5}$	$HDVEPSO_{ws10}$	$HDVEPSO_{ws15}$	$HDEVPSO_{ws20}$	$HDVEPSO_{ws25}$	$HDVEPSO_{ws50}$	$HDVEPSO_c$
Type I	*acc*	Wins	47.16	49.22	50.52	36.14	47.34	44.78	14.0	22.28
Type I	*acc*	Losses	33.84	28.78	28.48	40.86	30.66	32.22	64.44	52.16
Type I	*acc*	Diff	13.32	20.44	22.04	-4.72	16.68	12.56	-50.44	-29.88
Type I	*acc*	Rank	4.0	2.0	**1.0**	6.0	3.0	5.0	8.0	7.0
Type I	*stab*	Wins	4.3	0.77	0.04	1.76	0.31	0.03	1.2	0.29
Type I	*stab*	Losses	0.65	0.77	0.64	1.68	1.49	0.26	1.3	1.91
Type I	*stab*	Diff	3.65	0.0	-0.6	0.08	-1.18	-0.23	-0.1	-1.62
Type I	*stab*	Rank	**1.0**	3.0	6.0	2.0	7.0	5.0	4.0	8.0
Type III	*acc*	Wins	18.72	10.55	22.0	26.72	28.08	29.29	46.21	47.43
Type III	*acc*	Losses	59.28	55.45	36.0	23.28	18.92	19.71	8.79	7.57
Type III	*acc*	Diff	-40.56	-44.9	-14.0	3.44	9.16	9.58	37.42	39.86
Type III	*acc*	Rank	7.0	8.0	6.0	5.0	4.0	3.0	2.0	**1.0**
Type III	*stab*	Wins	27.5	5.94	4.93	2.67	2.49	3.17	1.82	1.72
Type III	*stab*	Losses	2.79	5.23	4.36	3.81	4.06	4.27	9.44	16.28
Type III	*stab*	Diff	24.71	0.71	0.57	-1.14	-1.57	-1.1	-7.62	-14.56
Type III	*stab*	Rank	**1.0**	2.0	3.0	5.0	6.0	4.0	7.0	8.0
all	*acc*	Wins	65.88	59.77	72.52	62.86	75.42	74.07	60.21	69.71
all	*acc*	Losses	93.12	84.23	64.48	64.14	49.58	51.93	73.23	59.73
all	*acc*	Diff	-27.24	-24.46	8.04	-1.28	25.84	22.14	-13.02	9.98
all	*acc*	Rank	8.0	7.0	4.0	5.0	**1.0**	2.0	6.0	3.0
all	*stab*	Wins	31.8	7.2	5.46	4.92	3.29	3.69	3.51	2.5
all	*stab*	Losses	6.87	6.0	5.0	5.49	5.55	4.53	10.74	18.19
all	*stab*	Diff	24.93	1.2	0.46	-0.57	-2.26	-0.84	-7.23	-15.69
all	*stab*	Rank	**1.0**	2.0	3.0	4.0	6.0	5.0	7.0	8.0

Summary

Research in PSO has been very active over the past two decades, and there are still much scope for more research, to focus on:

- Further theoretical analyses of PSO
- PSO for dynamic multi-objective optimization problems
- PSO for static and dynamic many-objective optimization problems
- PSO for dynamically changing constraints
- Set-based PSO algorithms
- Application of PSO to new real-world problems not yet solved