

Particle Swarm Optimization: A Guide to Effective, Misconception Free, Real World Use

AP Engelbrecht and CW Cleghorn

Computational Intelligence Research Group (CIRG)
Department of Computer Science
University of Pretoria
South Africa

`{engel,ccleghorn}@cs.up.ac.za`

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

GECCO'18 Companion, July 15–19, 2018, Kyoto, Japan

©2018 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-5764-7/18/07.

<https://doi.org/10.1145/3205651.3207877>

Presenter

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 Director: Institute for Big Data and Data Science. 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*.



Presenter

Christopher Cleghorn



Received his Masters and PhD degrees in Computer Science from the University of Pretoria, South Africa, in 2013 and 2017 respectively. He is lecturer in Computer Science at the University of Pretoria, and a member of the Computational Intelligence Research Group. His research interests include swarm intelligence, evolutionary computation, and machine learning, with a strong focus of theoretical research. Dr Cleghorn annually serves as a reviewer for numerous international journals and conferences in domains ranging from swarm intelligence and neural networks to mathematical optimization.





- 1 Introduction
- 2 Standard Particle Swarm Optimization
- 3 Neighbourhood Topologies
- 4 Velocity Initialization
- 5 Iteration Strategies
- 6 Control Parameters

Introduction

Purpose



TBC



What is particle swarm optimization (PSO)?

- a simple, computationally efficient optimization method
- population-based, stochastic search
- individuals follow very simple behaviors:
 - emulate the success of neighboring individuals,
 - but also bias towards own experience of success
- emergent behavior: discovery of optimal regions within a high dimensional search space



What are the 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
 - reflects experiential knowledge of the particles and socially exchanged information about promising areas in the search space



- used either the star (gbest PSO) or social (lbest PSO) topology
- velocity update per dimension:

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

- $v_{ij}(0) = 0$ (preferred)
- w is the inertia weight
- 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)

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

- $\hat{\mathbf{y}}_i(t)$ is the neighborhood best position calculated as the best personal best position in particle i 's neighborhood

Particle Swarm Optimization

PSO Algorithm



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

for each particle \hat{i} with particle i in its neighborhood **do**

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

$S.\hat{y}_{\hat{i}} = S.y_i$;

end

end

end

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

 update the velocity and position;

end

until *stopping condition is true*;



Neighborhood topologies are used to determine the best positions, or attractors, which guide the search trajectories of particles:

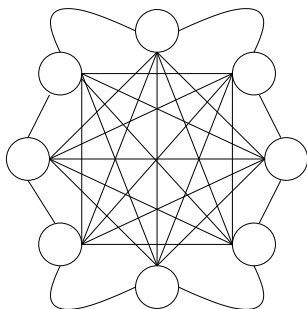
- topologies determine extend of the search space used to determine best positions
- topologies regulate the speed at which information about best positions is transferred through the swarm
- neighborhoods are based on particle indices, not spatial information
- neighborhoods overlap to facilitate information exchange

Neighborhood Topologies

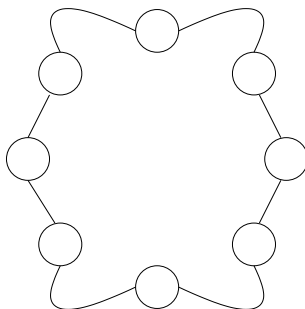
Popular Topologies



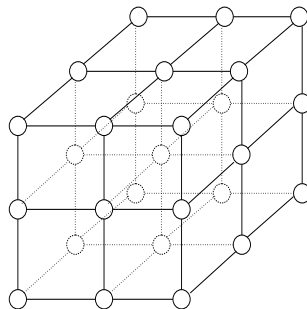
While many neighborhood topologies have been proposed, the most popular ones are



Star Topology
(gbest PSO)



Ring Topology
(lbest PSO)



Von Neumann Topology

gbest PSO versus lbest PSO

Problem Statement



Original PSO came in two versions, differing in the neighborhood topology used to exchange information about best found positions, i.e.

- **gbest PSO**, using a star neighborhood topology, and
- **lbest PSO**, using a ring neighborhood topology

A general opinion emerged from the PSO community that gbest PSO should not be used, and that lbest PSO should be used due to lbest PSO's

- better exploration ability,
- diminished susceptibility of being trapped in local minima, and
- because it does not suffer from premature convergence.

These opinions are based on very limited empirical evidence and intuitive beliefs about particle behavior.

gbest PSO versus lbest PSO

Two Topologies



gbest PSO and lbest PSO differ in the way that neighborhood best positions are updated:

- gbest PSO uses a star neighborhood topology
 - each particle has the entire swarm as its neighborhood
 - $\hat{\mathbf{y}}_i = \hat{\mathbf{y}}$ for all particles $i = 1, \dots, n_s$
 - consequence: all particles are attracted to one global best position.
- lbest PSO uses a ring topology
 - each particle's neighborhood consists of itself and its immediate two neighbours
 - neighborhoods overlap
 - consequence: each particle is attracted to a (initially) different neighborhood best position.

gbest PSO versus lbest PSO

General Opinions



Much has been said about the advantages and disadvantages of these two topologies:

- gbest PSO should not be used due to premature convergence to local optima
- gbest PSO converges fast due to faster transfer of best position throughout the swarm, therefore a strong attraction to one best position
- lbest PSO converges more slowly, and therefore explores more as it maintains diversity for longer
- gbest PSO is more susceptible to being trapped in local minima
- gbest PSO is best suited to unimodal problems and should not be used for multimodal problems
- gbest PSO does not perform well for non-separable problems
- lbest PSO is superior to gbest PSO in terms of solution accuracy for the majority of problems



Objective: To conduct an extensive empirical analysis to test these general opinions

Two algorithms were implemented to differ only in the neighborhood topology used

- synchronous position updates
- memory-based personal best position update
- zero initial velocities
- no velocity clamping
- personal best positions updated only if they remain within bounds

Control parameter values:

- $w = 0.729844$
- $c_1 = c_2 = 1.49618$
- 30 particles
- 5000 iterations



Performance was quantified over 50 independent runs using

- **Accuracy:**

- average quality of best solution over 50 runs after 5000 iterations

- **Success Rate:**

- percentage of the 50 independent runs that converged to specific accuracy levels
- 1000 accuracy levels have been considered, from best obtain accuracy, logarithmically scaled to the worst obtained accuracy

- **Efficiency:**

- average number of iterations to reach the different accuracy levels

- **Consistency:**

- deviation from the average best value



- **Accuracy:**

- paired Mann-Whitney U tests at 0.05 significance level
- wins and losses calculated per function class

- **Success rate:**

- Mann-Whitney U test applied on success rates over all of the accuracy levels
- indicates success rate profile, over all accuracy levels
- a win indicates that corresponding algorithm had most successful runs for most of the accuracy levels

- **Efficiency:**

- average number of iterations to reach accuracy levels over all accuracy levels
- a win indicates that the corresponding algorithm converged faster to most accuracy levels



59 boundary constrained problems, of the following types

- uni-modal
- multi-modal
- separable, rotated
- non-separable
- shifted
- noisy
- composition functions

gbest PSO versus lbest PSO

Empirical Analysis: Results (cont)



'>' indicates gbest better than lbest, '<' gbest worse than lbest, and '=' no statistically significant difference

Function Class		Number of Functions	Accuracy			Success Rate			Efficiency			Diversity		
			>	=	<	>	=	<	>	=	<	>	=	<
UM	S	7	5	0	2	6	0	1	2	0	5	5	0	2
	NS	3	2	1	0	2	1	0	2	1	0	2	0	1
	N	2	1	0	1	1	1	0	2	0	0	1	0	1
	Sh	5	2	3	0	2	3	0	2	3	0	1	0	4
	R	1	1	0	0	1	0	0	0	1	0	0	0	1
MM	S	6	1	2	3	2	2	2	3	1	2	6	0	0
	NS	9	4	1	4	3	4	2	4	3	2	1	0	8
	Sh	10	3	4	3	5	5	0	8	1	1	1	0	9
	R	4	0	3	1	1	2	1	2	1	1	0	0	4
	N	1	0	1	0	0	1	0	0	1	0	0	0	1
	C	11	1	2	8	0	4	7	1	5	5	0	0	11
Overall		59	20	17	22	23	23	13	26	17	16	11	0	48
Overall UM		18	11	4	3	12	5	1	8	5	5	9	0	9
Overall MM		41	13	19	14	11	18	12	18	12	11	2	0	39
Overall S		17	7	4	6	9	5	3	12	1	4	5	0	12
Overall NS		42	13	13	16	14	18	10	11	16	9	6	0	36



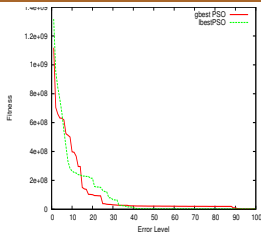
With reference to consistency:

- For 21.7% of the functions did gbest PSO have a significantly smaller deviation than lbest PSO
- For 31.6% of the functions did lbest PSO have a significantly smaller deviation than gbest PSO

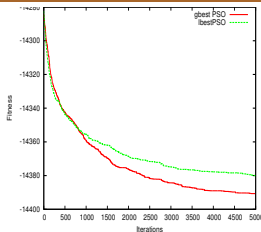
No one of the two topologies can be said to be more consistent than the other

gbest PSO versus lbest PSO

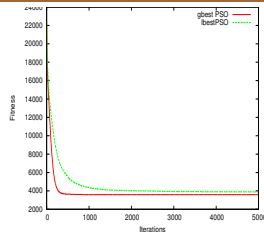
Empirical Analysis: Fitness Profiles



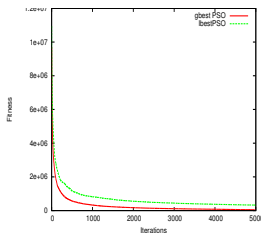
(a) Elliptic



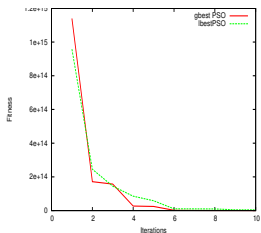
(b) Shifted Schaffer 6



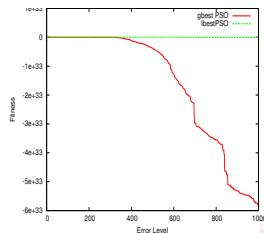
(c) Rastrigin



(d) Noisy Shifted Schwefel 1.2



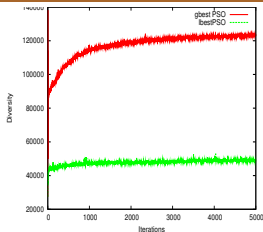
(e) Schwefel 2.22



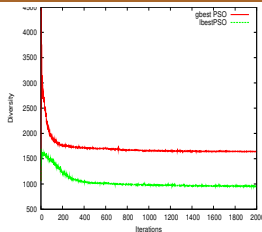
(f) Shubert

gbest PSO versus lbest PSO

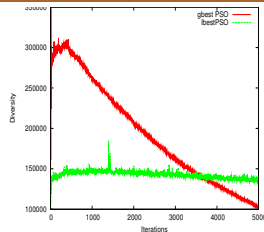
Empirical Analysis: Diversity Profiles



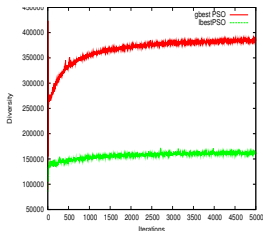
(h) Shifted Rotated Ackley



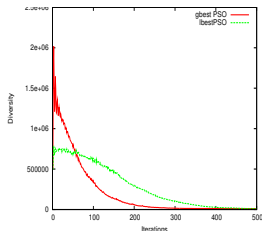
(i) De Jong F4



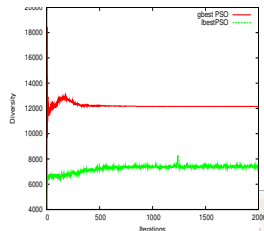
(j) Rotated Elliptic



(k) Shifted Schaffer 6



(l) Griewank



(m) Bastingin



The following observations can be made over all the functions:

- gbest and lbest PSO performed similar with respect to accuracy
- gbest slightly better than lbest with respect to success rate and efficiency
- lbest slightly better than gbest with respect to consistency
- lbest PSO did not maintain diversity for longer than PSO for all functions
- despite the fact that gbest converges faster, it is not at the cost of accuracy nor success rate
- both gbest PSO and lbest PSO sometimes prematurely converge



Observations with respect to specific function classes:

- gbest and lbest equally good at separable and non-separable functions with respect to accuracy
- gbest obtained better success rates than lbest PSO for separable and non-separable functions
- for most of the non-separable functions, no significant difference in convergence speed
- lbest was more accurate for a number of unimodal functions
- lbest more accurate for less than half of the multi-modal functions
- lbest did converge faster for a number of functions unimodal and multi-modal functions



Which of gbest PSO or lbest PSO is best?

Based on an extensive empirical analysis, the main conclusions are that

- none of the two algorithms can be considered the preferred algorithm for any of the main function classes
- the best choice is very problem dependent



Velocities have been initialized using any of the following:

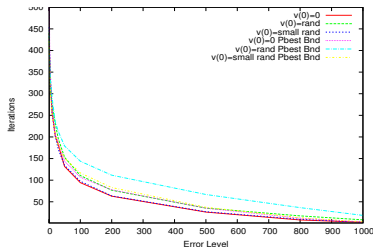
- $\mathbf{v}_i(0) = \mathbf{0}$
 - Critique: Limits exploration ability, therefore extent to which the search space is initially covered
 - Counter argument: Initial positions are uniformly distributed
 - Flocking analogy: Physical objects, in their initial state, do not have any momentum
- $\mathbf{v}_i(0)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

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

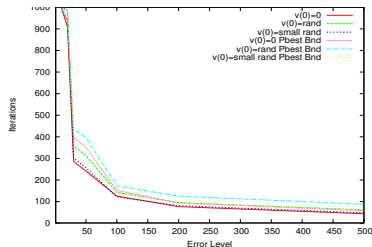
- Initialize to small random values

Velocity Initialization

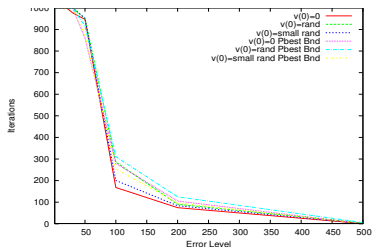
Fitness Profiles



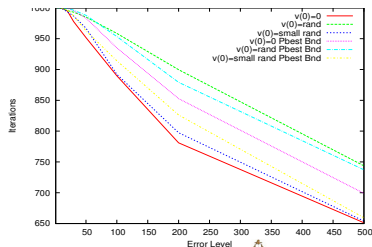
(p) Absolute Value



(q) Rosenbrock



(r) Rastrigin



(s) Quadric



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA



Velocity Initialization

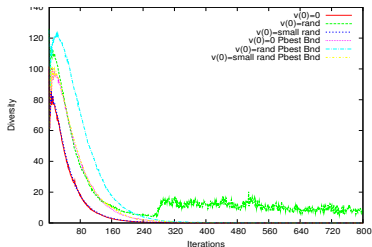
Fitness After 1000 Iterations



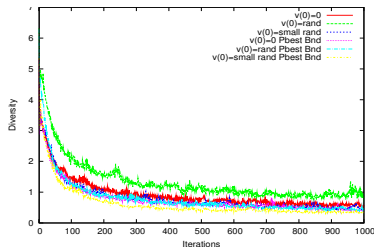
Function	Zero Init No Pbest Bound	Random Init No Pbest Bound
Absolute Value	$3.53\text{E-}001 \pm 2.87\text{E+}000$	$2.46\text{E-}001 \pm 1.47\text{E+}000$
Ackley	$2.49\text{E+}000 \pm 1.35\text{E+}000$	$2.68\text{E+}000 \pm 2.67\text{E+}000$
Bukin 6	$6.20\text{E-}002 \pm 4.50\text{E-}002$	$6.65\text{E-}002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

Velocity Initialization

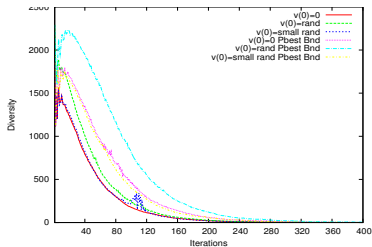
Diversity Profiles



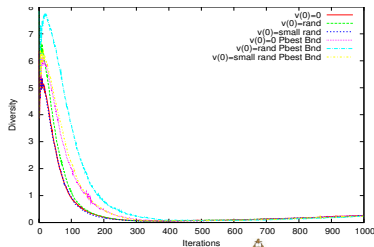
(a) Ackley



(b) Bukin 6



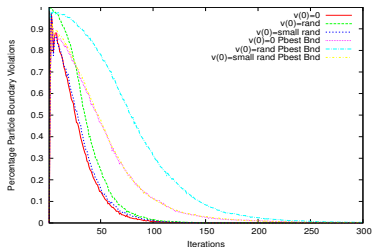
(c) Griewank



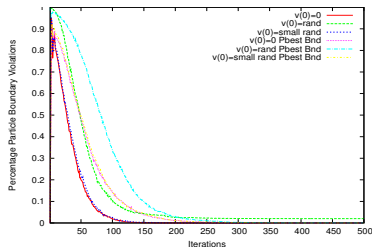
(d) Rosenbrock

Velocity Initialization

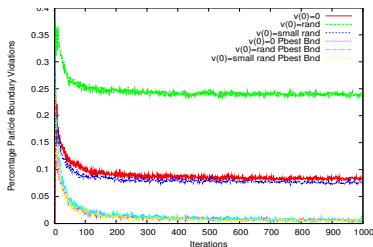
Roaming Behavior: Percentage of Infeasible Particles



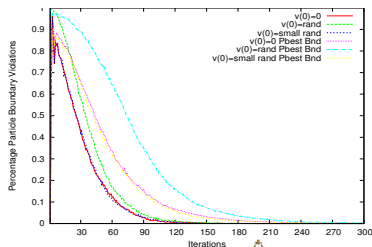
(a) Absolute Value



(b) Ackley



(c) Bukin 6



(d) Griewank

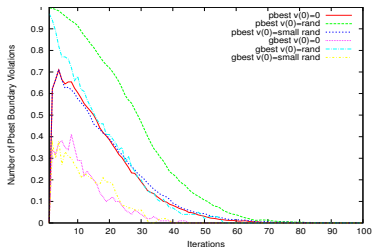


UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

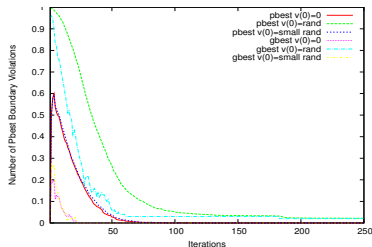


Velocity Initialization

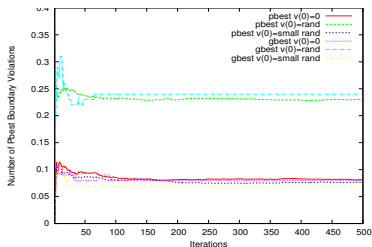
Roaming Behavior: Percentage of Infeasible Personal Bests



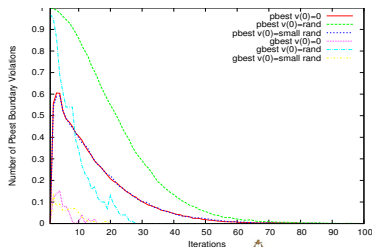
(a) Absolute Value



(b) Ackley



(c) Bukin 6



(d) Griewank



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA





The following general observations are made:

- Small random initialization and zero initialization have similar behaviors
- Random initialization
 - slower in improving fitness of best solution
 - resulted in larger diversity
 - had more roaming particles, roaming for longer
 - significantly more best positions left boundaries
 - took longer to reduce number of particle and best position violations
 - very slow in increasing number of converged dimensions
- Not much of a difference in final accuracies obtained for most of the problems, with random initialization performing poor for some functions



Two iteration strategies can be found for PSO:

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



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*;



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

Ranks based on Final Fitness Values

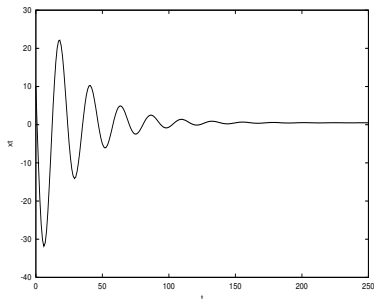
Function Class		Number of Functions	gbest PSO			lbest PSO			GCP SO			BBPSO		
			>	=	<	>	=	<	>	=	<	>	=	<
UM	Sep	7	0	0	7	0	1	6	0	0	7	0	1	6
	Non-sep	3	1	1	1	0	2	1	0	2	1	0	3	0
	Noisy	2	0	0	2	1	1	0	1	0	1	1	0	1
	Shifted	5	0	5	0	0	4	1	0	5	0	0	5	0
	Rotated	1	0	0	1	0	0	1	0	0	1	0	1	0
MM	Sep	6	0	5	1	0	6	0	0	4	2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



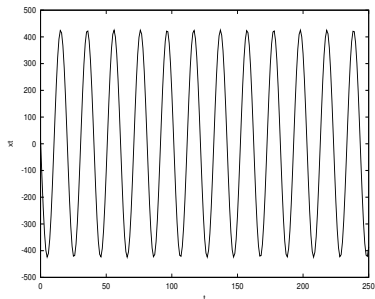
- Unimodal functions: AIS better accuracy for most functions
- Multimodal functions:
 - No significant difference for most of the functions
 - For the remainder of the functions, no clear winner
 - For lbest PSO not significant difference over all functions – insensitive to iteration strategy
- Separable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

Performance of PSO has been shown to be very sensitive to values assigned to its control parameters

$w = 0.5$ and $c_1 = c_2 = 1.4$



$w = 1.0$ and $c_1 = c_2 = 1.999$

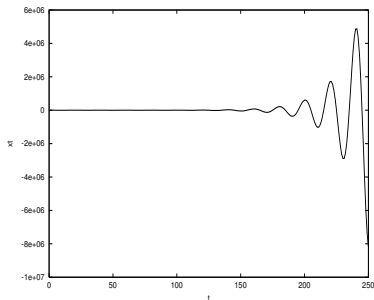


Control Parameters

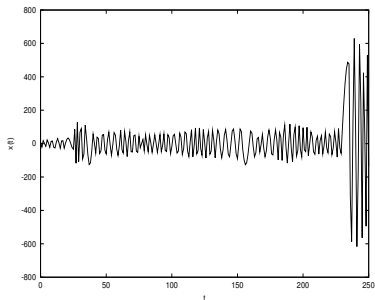
Introduction (cont)



$w = 0.7$ and $c_1 = c_2 = 1.9$



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





Where are these control parameters used?

- previous velocity, $w\mathbf{v}_i(t)$
 - 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



- Was introduced to control step sizes
- Can be used to balance exploration-exploitation trade-off
 - large values – favor exploration
 - small values – promote exploitation
 - (depending on the values of c_1 and c_2)
- for $w \geq 1$
 - velocities increase over time
 - swarm diverges
 - particles fail to change direction towards more promising regions
- for $0 < w < 1$
 - particles decelerate
 - convergence also dependent on values of c_1 and c_2



Weights the contributions of the cognitive and social components:

- $c_1 = c_2 = 0$?
- $c_1 > 0, c_2 = 0$:
 - particles are independent hill-climbers
 - local search by each particle
- $c_1 = 0, c_2 > 0$:
 - swarm is one stochastic hill-climber
- $c_1 = c_2 > 0$:
 - particles are attracted towards the average of \mathbf{y}_i and $\hat{\mathbf{y}}_i$
- $c_2 > c_1$:
 - promotes exploitation
- $c_1 > c_2$:
 - promotes exploration



Note that a strong dependence exists between w and $c_1 + c_2$:

- Guaranteed equilibrium state will be reached if

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

- The extend to which particles explores and exploits also influenced by this dependence



Approaches to find the best values for control parameters:

- Just use the values published in literature?
- Fine-tuned static values
- Dynamically changing values
- Self-adaptive control parameters

Many dynamic and self-adaptive approaches have recently been developed

But... more research is needed...



Issues with current self-adaptive approaches:

- Most, at some point in time, violate convergence conditions, and many do so for most of the search process
- Converge prematurely, with little exploration of control parameter space
- Introduce more control parameters
- Current empirical analysis shows that they do not really result in improved performance with reference to solution quality

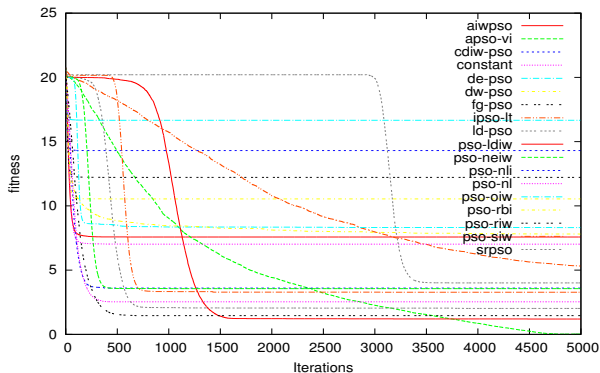
Control Parameters

Self-Adaptive Particle Swarm Optimization: Approaches

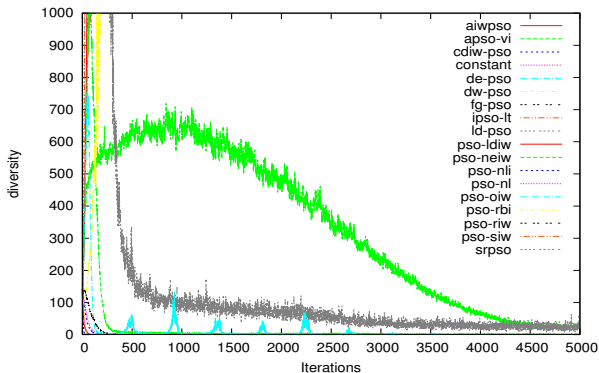


Optimizer	Parameters Tuned	Net Change
PSO-TVIW (Shi and Eberhart, 1998, 1999)	ω	+1
PSO-AIWF (Liu et al, 2005)	ω	+1
DAPSO (Yang et al, 2007)	ω	+2
IPSO-LT (Li and Tan, 2008)	ω	+1
SAPSO-LFZ (Li et al, 2008)	ω	-1 (0)
SAPSO-DWCY (Dong et al, 2008)	ω	-1 (+2)
PSO-RBI (Panigrahi et al, 2008)	ω	+1
IPSO-CLL (Chen et al, 2009)	ω	-1
AIWPSO (Nickabadi et al, 2011)	ω	+1
APSO-VI (Xu, 2013)	ω	+2
SRPSO (Tanweer et al, 2015)	ω	+2
PSO-SAIC (Wu and Zhou, 2007)	ω, c_2	+2 (+4)
PSO-RAC	ω, c_1, c_2	-3
PSO-TVAC (Ratnaweera et al, 2004)	ω, c_1, c_2	+3
PSO-ICSA (Jun and Jian, 2009)	ω, c_1, c_2	+3 (+31)
APSO-ZZLC (Zhan et al, 2009)	ω, c_1, c_2	-3 (+35)
UAPSO-A (Hashemi and Meybodi, 2011)	ω, c_1, c_2	+6
GPSO (Leu and Yeh, 2012)	ω, c_1, c_2	+3 (+($n_d + 3$))

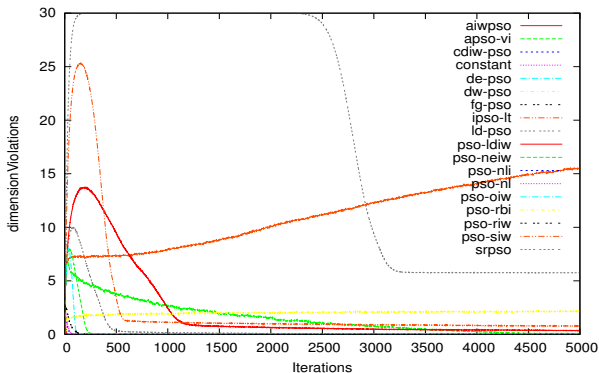
Average solution quality



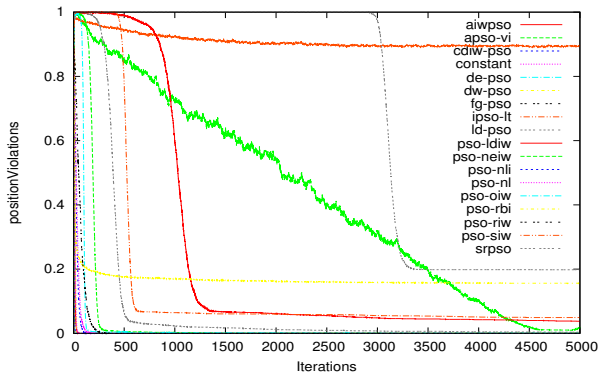
Average swarm diversity



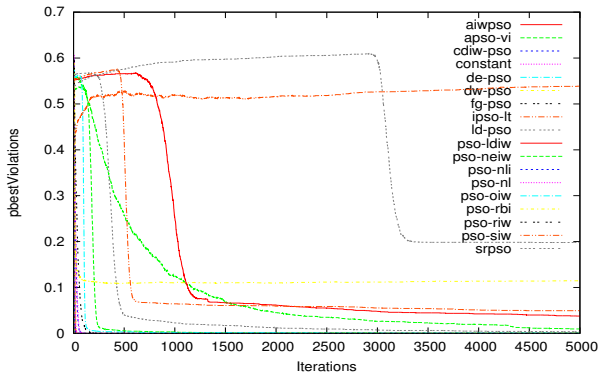
Average boundary violations per dimension



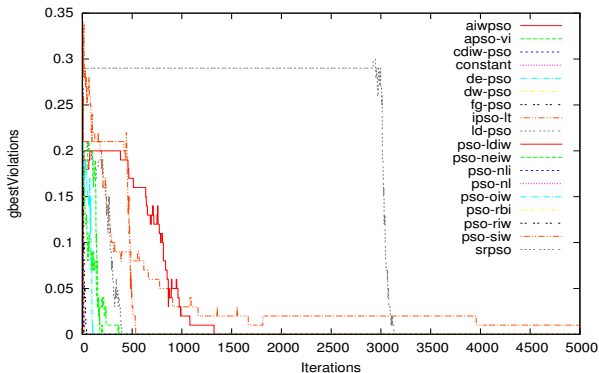
Average percentage particle position boundary violations



Average percentage personal best position boundary violations



Average global best position boundary violations

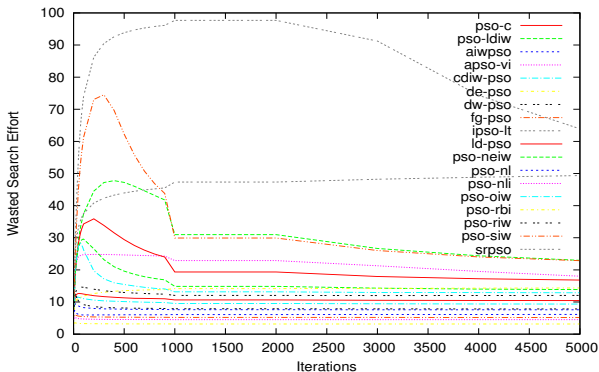


Control Parameters

Self-Adaptive Particle Swarm Optimization: Ackley (cont)



Wasted search effort over 60 functions, in dimensions 30, 40, 50, 60, 80, 90, and 100





Uses the specially-formulated function to study convergence behavior:

$$F(\mathbf{x}) \sim U(0, 2000)$$

such that

$$F(\mathbf{x}_1) = F(\mathbf{x}_2) \text{ if } \mathbf{x}_1 = \mathbf{x}_2$$

- the fitness value of each position in the search space is randomly sampled within the range $[0, 2000]$
- complete stagnation is highly unlikely
- provides a good benchmark function for studying convergence behavior



Performance measures:

- Average particle movement, Δ :
 - quantifies average particle step size
 - if value does not decrease, particles do not converge
- Percentage particles with convergent control parameters, CP :
 - measures algorithm's ability to generate convergent parameters
- Average parameter movement, Δ_p :
 - average step size in parameter space
 - quantifies stability of the control parameter values
- Percentage particles that violates boundaries, IP :
 - proportion of particles that violates boundary constraints in at least one dimension
 - quantification of wasted search effort

Control Parameters



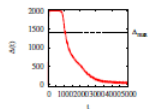
Self-Adaptive Particle Swarm Optimization: Analysis (cont)

After 5000 iterations:

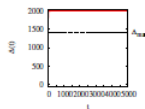
Algorithm	Δ	CP	Δ_p	IP
PSO	415.125	100%	0.0	70.7%
PSO-TVIW	56.489	100%	1.00e-4	9.6%
PSO-AIWF	2000.000	0%	0.0	96.7%
DAPSO	2000.000	0%	NaN	96.9%
IPSO-LT	2000.000	0%	0.0	96.7%
SAPSO-LFZ	2000.000	47.2%	0.0	53.5%
SAPSO-DWCY	1324.322	100%	0.0	96.2%
PSO-RBI	2000.000	76.7%	6.01e-2	41.5%
IPSO-CLL	2000.000	100%	0.0	100%
AIWPSO	45.521	100%	0.0	3.3%
APSO-VI	55.940	100%	0.0	6.1%
SRPSO	2000.000	96.7%	0.0	3.3%
PSO-SAIC	2000.000	0%	NaN	96.7%
PSO-RAC	165.544	100%	1.60e+0	44.2%
PSO-TVAC	32.354	100%	5.74e-4	6.5%
PSO-ICSA	2000.000	0%	4.00e-4	96.7%
APSO-ZZLC	1318.307	100%	4.51e-5	96.1%
UAPSO-A	124.467	70%	8.47e-1	38.1%
GPSO	2000.000	16.7%	8.35e-2	96.7%

Control Parameters

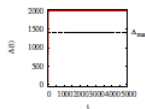
Self-Adaptive Particle Swarm Optimization: Average Particle Movement



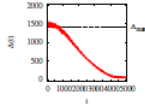
(a) PSO-TVIW



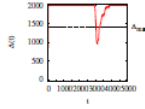
(b) PSO-AIWF



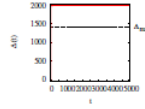
(c) DAPSO



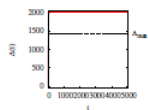
(j) APSO-VI



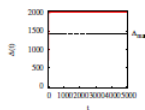
(k) SRPSO



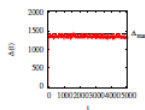
(l) PSO-SAIC



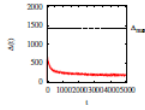
(d) IPSO-LT



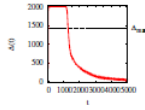
(e) SAPSO-LFZ



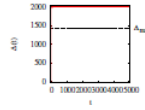
(f) SAPSO-DWCY



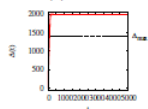
(m) PSO-RAC



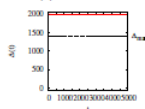
(n) PSO-TVAC



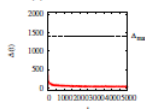
(o) PSO-ICSA



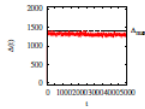
(g) PSO-RBI



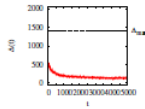
(h) IPSO-CLL



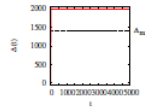
(i) AIWPSO



(p) APSO-ZZLC



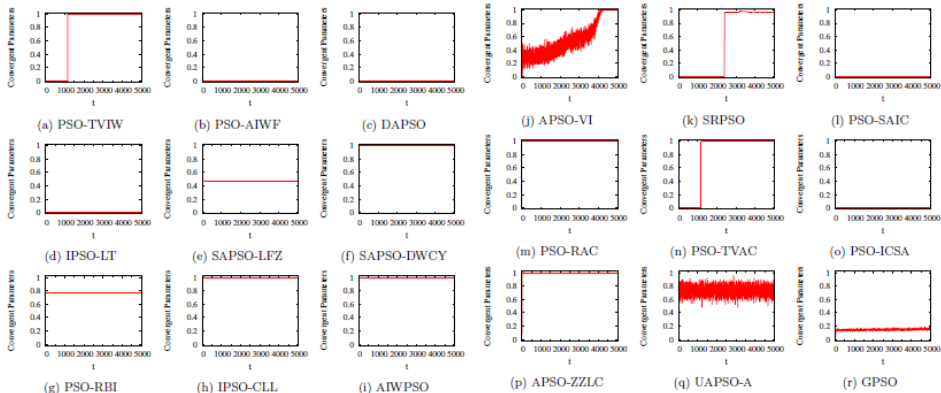
(q) UAPSO-A



(r) GPSO

Control Parameters

Self-Adaptive Particle Swarm Optimization: %Convergent Parameters

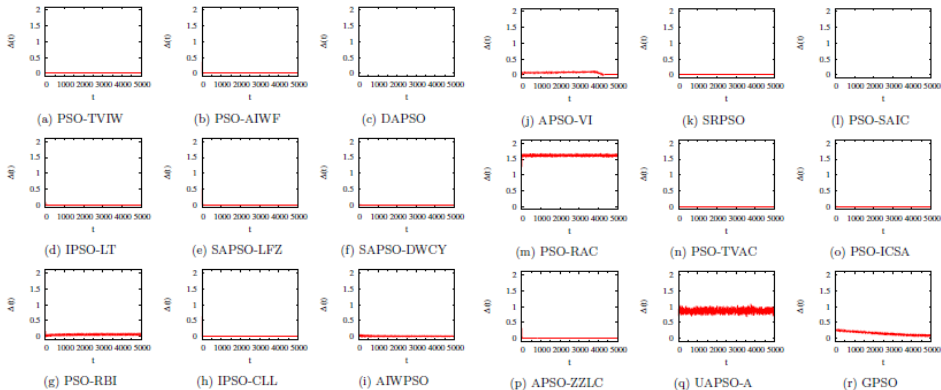


UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA



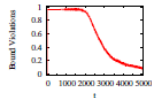
Control Parameters

Self-Adaptive Particle Swarm Optimization: Parameter Movement

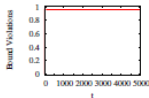


Control Parameters

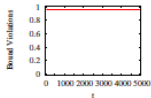
Self-Adaptive Particle Swarm Optimization: Boundary Violations



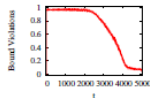
(a) PSO-TVIW



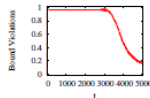
(b) PSO-AIWF



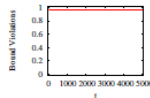
(c) DAPSO



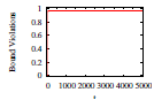
(j) APSO-VI



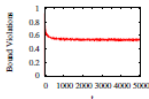
(k) SRPSO



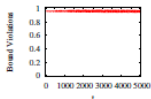
(l) PSO-SAIC



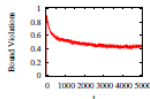
(d) IPSO-LT



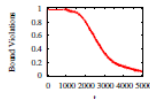
(e) SAPSO-LFZ



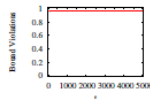
(f) SAPSO-DWCY



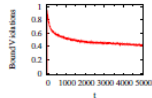
(m) PSO-RAC



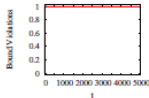
(n) PSO-TVAC



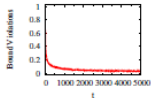
(o) PSO-ICSA



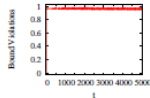
(g) PSO-RBI



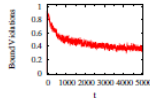
(h) IPSO-CLL



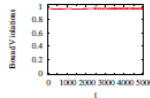
(i) AIWPSO



(p) APSO-ZZLC



(q) UAPSO-A



(r) GPSO



UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

