Particle Swarm Optimization

Introduction

Marco A. Montes de Oca

IRIDIA-CoDE, Université Libre de Bruxelles (U.L.B.)

May 7, 2007

Presentation overview

- Origins
- The idea
- Continuous optimization
- The basic algorithm
- Main variants
- Parameter selection
- Research issues
- Our work at IRIDIA-CoDE

Particle swarm optimization: Origins

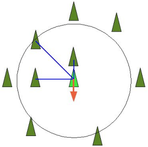
How can birds or fish exhibit such a coordinated collective behavior?

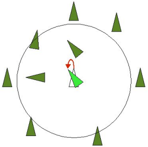
Reynolds [12] proposed a behavioral model in which each agent follows three rules:

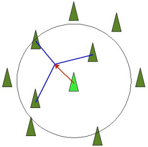
Separation. Each agent tries to move away from its neighbors if they are too close.

Alignment. Each agent steers towards the average heading of its neighbors.

Cohesion. Each agent tries to go towards the average position of its neighbors.



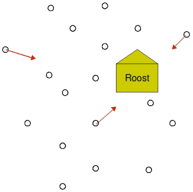


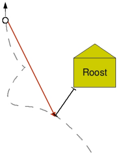


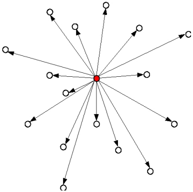
Particle swarm optimization: Origins

Kennedy and Eberhart [6] included a ‘roost’ in a simplified Reynolds-like simulation so that:

- Each agent was attracted towards the location of the roost.
- Each agent ‘remembered’ where it was closer to the roost.
- Each agent shared information with its neighbors (originally, all other agents) about its closest location to the roost.







Particle swarm optimization: The idea

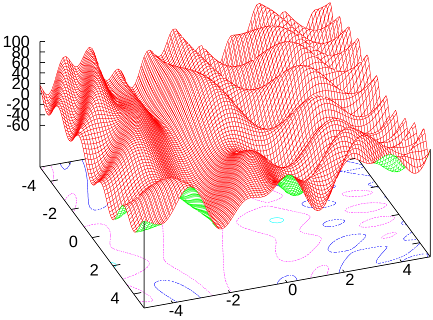
Eventually, all agents ‘landed’ on the roost.

What if the notion of distance to the roost is changed by an unknown function? Will the agents ‘land’ in the minimum?

J. Kennedy

R. Eberhart

The continuous optimization problem can be stated as follows:

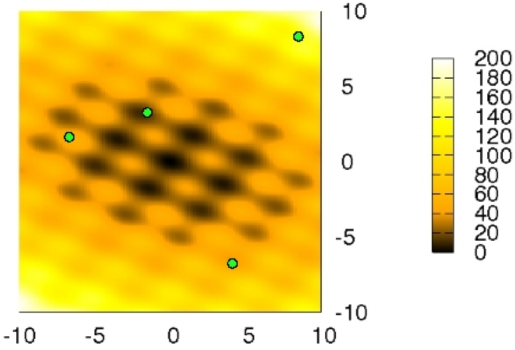


Find $\mathcal{X}^* \subseteq \mathcal{X} \subseteq \mathbb{R}^n$ such that

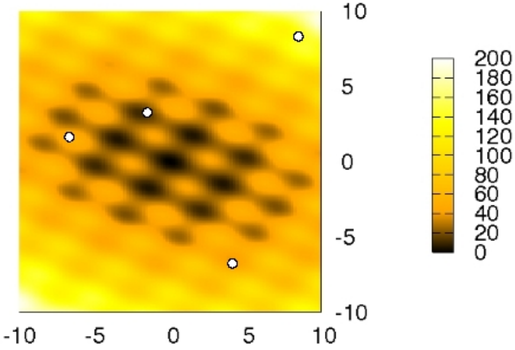
$$\mathcal{X}^* = \underset{\mathbf{x} \in \mathcal{X}}{\operatorname{argmin}} f(\mathbf{x}) = \{\mathbf{x}^* \in \mathcal{X} \,:\, f(\mathbf{x}^*) \leq f(\mathbf{x}) \ \forall \mathbf{x} \in \mathcal{X}\}$$

Particle swarm optimization: The basic algorithm

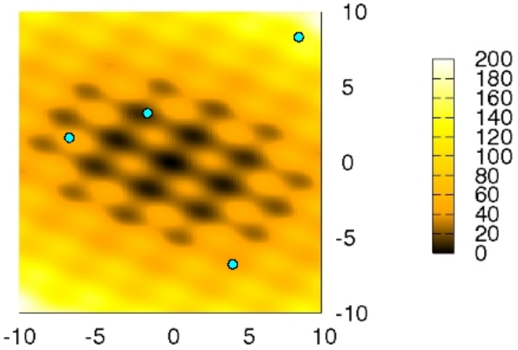
1. Create a 'population' of agents (called *particles*) uniformly distributed over $\mathcal{X}$.



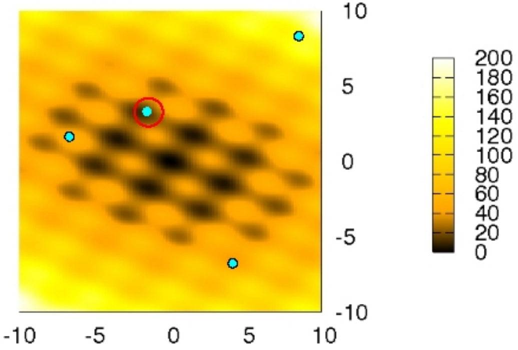
2. Evaluate each particle's position according to the objective function.



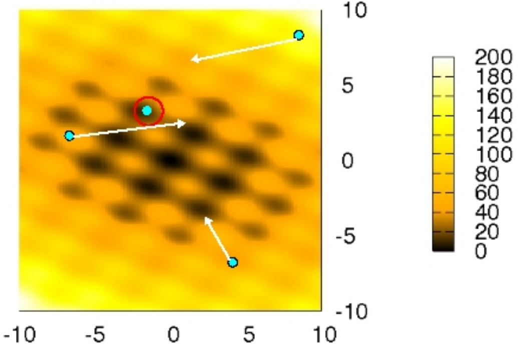
3. If a particle's current position is better than its previous best position, update it.



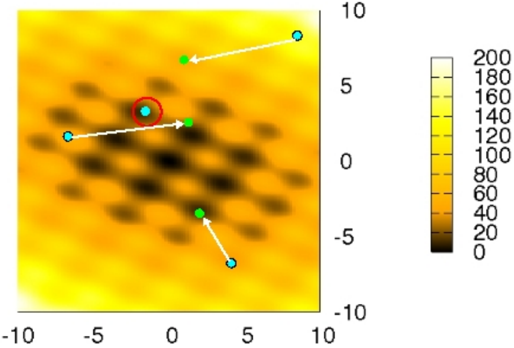
4. Determine the best particle (according to the particle's previous best positions).



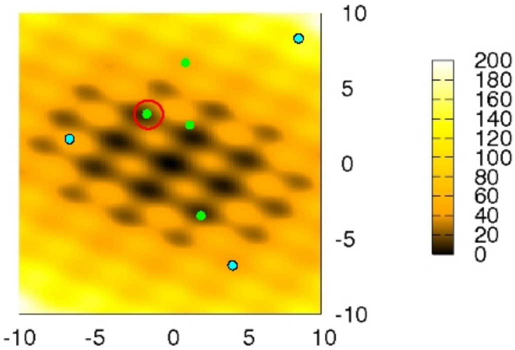
5. Update particles' velocities according to
$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \varphi_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{x}_i^t)$.



6. Move particles to their new positions according to $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}$.

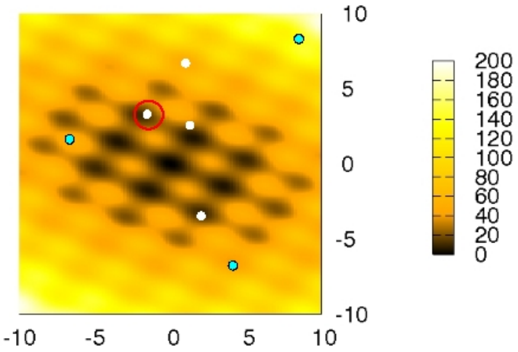


7. Go to step 2 until stopping criteria are satisfied.

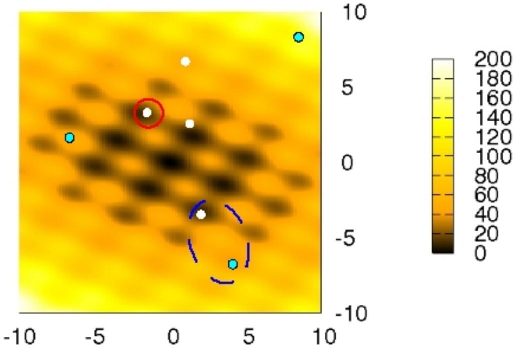


Particle swarm optimization: The basic algorithm

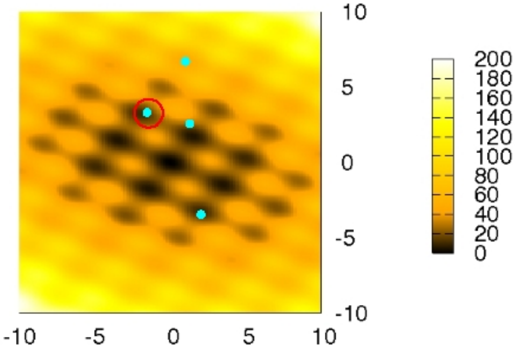
2. Evaluate each particle's position according to the objective function.



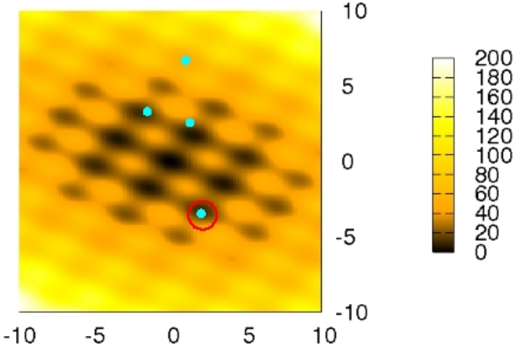
3. If a particle's current position is better than its previous best position, update it.



3. If a particle's current position is better than its previous best position, update it.

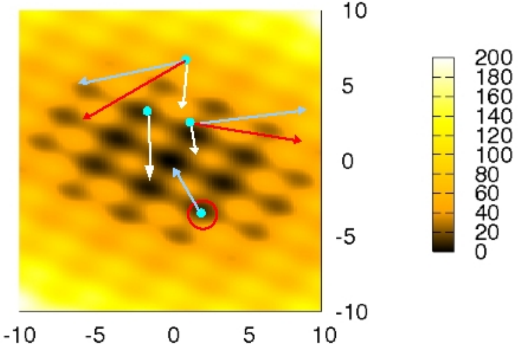


4. Determine the best particle (according to the particle's previous best positions).

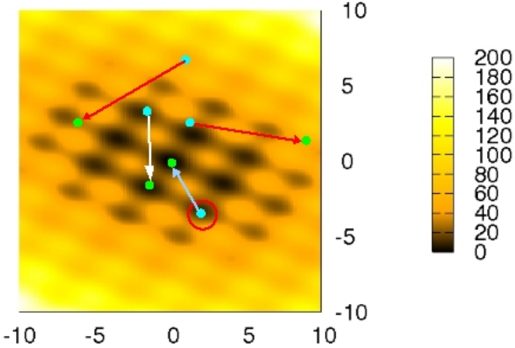


Particle swarm optimization: The basic algorithm

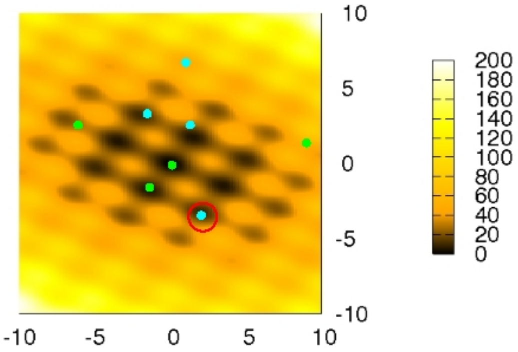
5. Update particles' velocities according to
$\mathbf{v}_i^{t+1} = \mathbf{v}_i^{t} + \varphi_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{x}_i^t)$.



6. Move particles to their new positions according to $\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}$.



7. Go to step 2 until stopping criteria are satisfied.



Particle swarm optimization: Main variants

Almost all modifications vary in some way the velocity-update rule:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \varphi_1 \mathbf{U}_1^t (\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t (\mathbf{gb}^{\,t} - \mathbf{x}_i^t)$$

Almost all modifications vary in some way the velocity-update rule:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^{t}}_{inertia} + \varphi_1 \mathbf{U}_1^t (\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t (\mathbf{gb}^t - \mathbf{x}_i^t)$$

Almost all modifications vary in some way the velocity-update rule:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \underbrace{\varphi_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{x}_i^t)}_{\textit{personal influence}} + \varphi_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{x}_i^t)$$

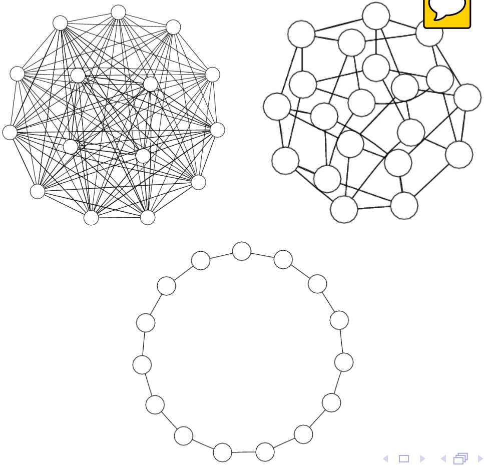
Almost all modifications vary in some way the velocity-update rule:

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \varphi_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{x}_i^t) + \underbrace{\varphi_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{x}_i^t)}_{\textit{social influence}}$$

Particle swarm optimization: Different population topologies

Every particle i has a neighborhood N_i.

$$\mathbf{v}_i^{t+1} = \mathbf{v}_i^t + \varphi_1 \mathbf{U}_1^t (\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t (\mathbf{lb}_i^t - \mathbf{x}_i^t)$$



Particle swarm optimization: Inertia weight

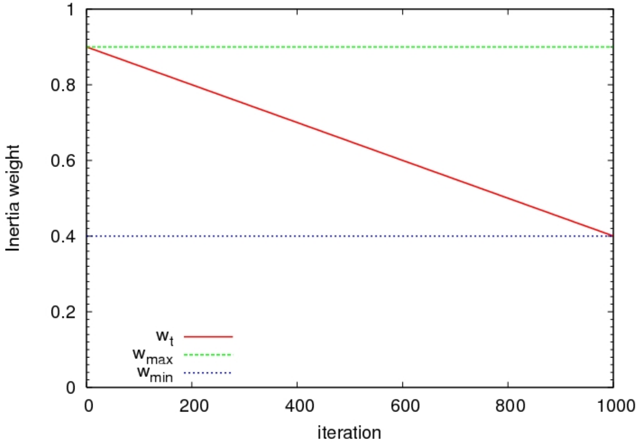
It adds a parameter called *inertia weight* so that the modified rule is:

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + \varphi_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t(\mathbf{lb}_i^t - \mathbf{x}_i^t)$$

It was proposed by Shi and Eberhart [14].

Particle swarm optimization: Time-decreasing inertia weight

The value of the inertia weight is decreased during a run



It was proposed by Shi and Eberhart [15].

Particle swarm optimization: Canonical PSO

It is a special case of the inertia weight variant derived from:

$$\mathbf{v}_i^{t+1} = \chi \left[\mathbf{v}_i^t + \varphi_1 \mathbf{U}_1^t (\mathbf{pb}_i^t - \mathbf{x}_i^t) + \varphi_2 \mathbf{U}_2^t (\mathbf{lb}_i^t - \mathbf{x}_i^t)\right] ,$$

where χ is called a "constriction factor" and is fixed.

It has been very influential after its proposal by Clerc and Kennedy [3].

Particle swarm optimization: Fully Informed PSO

In the Fully Informed PSO, a particle is attracted by every other particle in its neighborhood:

$$\mathbf{v}_i^{t+1} = \chi \left[\mathbf{v}_i^t + \sum_{p_k \in \mathcal{N}_i} \varphi_k \mathbf{U}_k^t (\mathbf{pb}_k^t - \mathbf{x}_i^t) \right] .$$

It was proposed by Mendes et al. [9].

Particle swarm optimization: Other variants

There are many other variants reported in the literature. Among others:

- with dynamic neighborhood topologies (e.g., [16], [10])
- with enhanced diversity (e.g., [2], [13])
- with different velocity update rules (e.g., [11], [8])
- with components from other approches (e.g., [1], [5])
- for discrete optimization problems (e.g., [7], [18])
- ...

Consider a one-particle one-dimensional particle swarm. This particle's velocity-update rule is

$$v^{t+1} = av^{t} + b_1 U_1^t(pb^{t} - x^{t}) + b_2 U_2^t(gb^{t} - x^{t})$$

Additionally, if we make

$$E[U_*^t(0,1)] = \frac{1}{2}\,,$$

$$b = \frac{b_1 + b_2}{2}\,,$$

$$pb^{\,t+1} = pb^{\,t+1}\,, gb^{\,t+1} = gb^{\,t}\,,$$

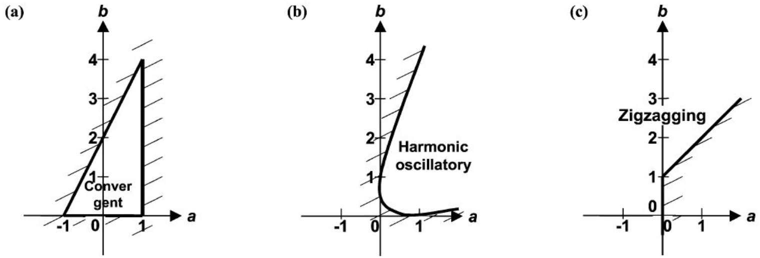
and

$$r = \frac{b_1}{b_1 + b_2} pb^{\,t} + \frac{b_2}{b_1 + b_2} gb^{\,t}\,.$$

Then, we can say that

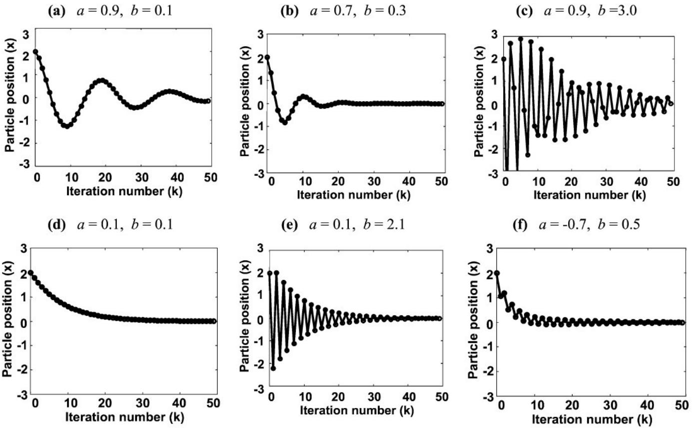
$$v^{t+1} = av^{t} + b(r - x^{t}) \,.$$

It can be shown that this system will behave in different ways depending on the value of a, b.



Graph taken from Trelea [17].

Some examples



Graph taken from Trelea [17].

Factors to consider when choosing a particular variant and/or a parameter set:

- The characteristics of the problem ("modality", search ranges, dimension, etc)
- Available search time (wall clock or function evaluations)
- The solution quality threshold for defining a satisfactory solution

Particle swarm optimization: Research issues

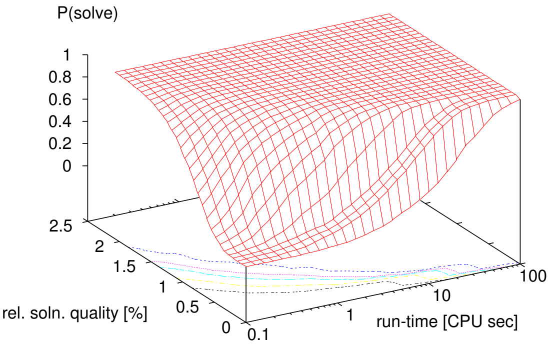
A number of research directions are currently pursued:

- Matching algorithms (or algorithmic components) to problems
- Application to different kind of problems (dynamic, stochastic, combinatorial)
- Parameter selection. (How many particles, which topology?)
- Identification of "state-of-the-art" PSO algorithms (comparisons)
- New variants (modifications, hybridizations)
- Theoretical aspects (particles behavior, stagnation)

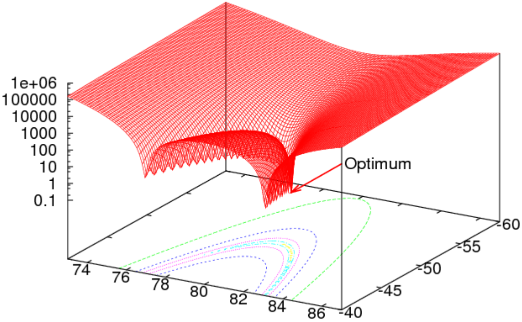
We have been working on three of the previously mentioned directions:

- Identification of "state-of-the-art" PSO algorithms (comparisons)
- Matching algorithms (or algorithmic components) to problems
- New variants (modifications, hybridizations)

We used *run-time and solution-quality distributions* [4] to compare several PSO variants.

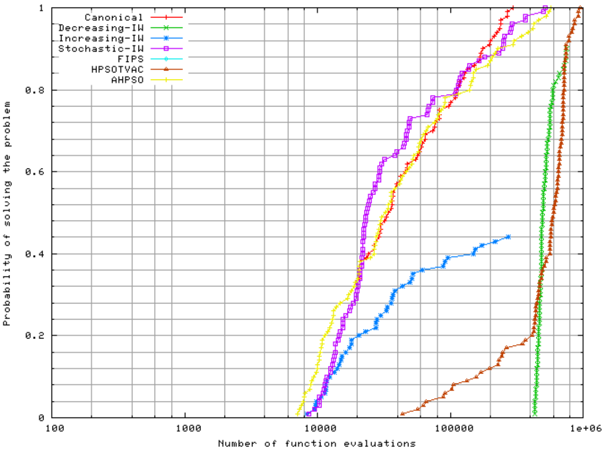


Rosenbrock function



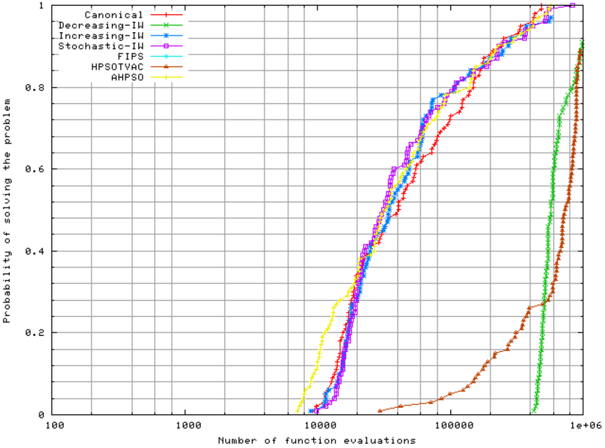
Particle swarm optimization: Comparisons: Different Topologies

(20 particles, Rosenbrock) : Fully connected topology



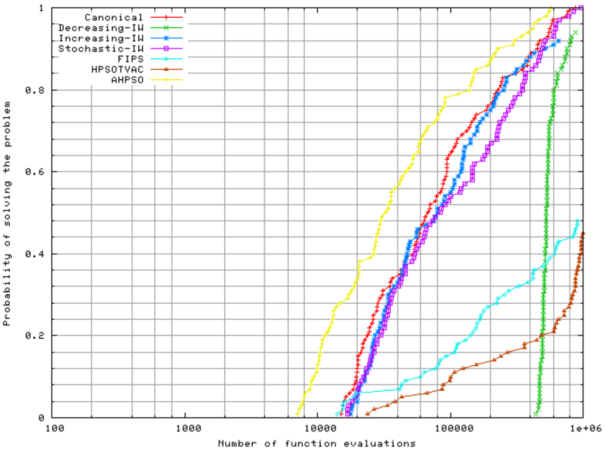
Particle swarm optimization: Comparisons: Different Topologies

(20 particles, Rosenbrock) : Square topology



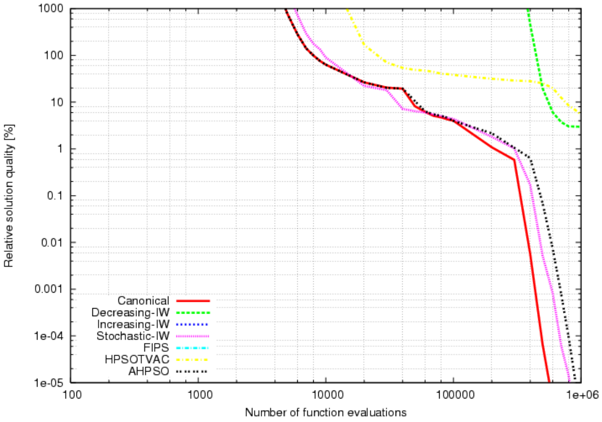
Particle swarm optimization: Comparisons: Different Topologies

(20 particles, Rosenbrock) : Ring topology



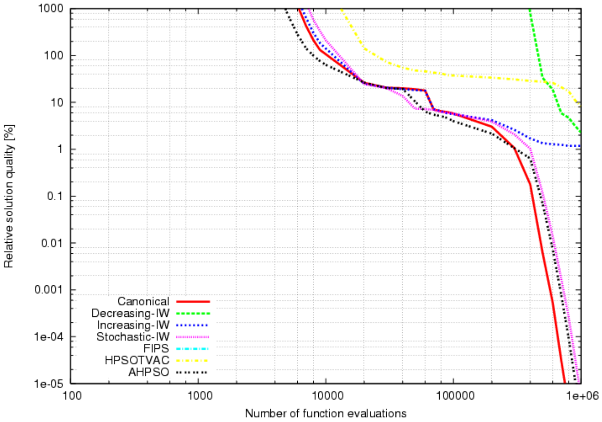
Particle swarm optimization: Comparisons: Different Topologies

(20 particles, Rosenbrock) : Fully connected topology



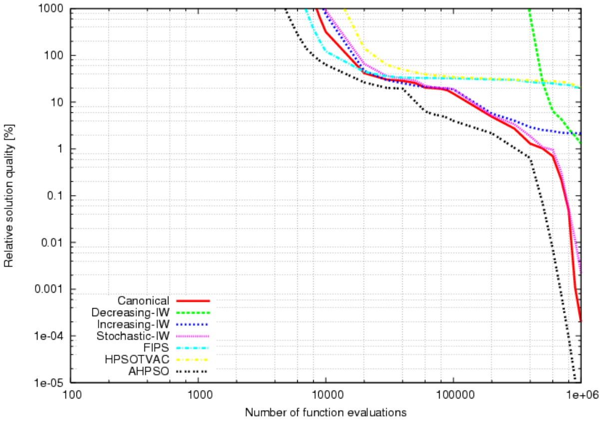
Particle swarm optimization: Comparisons: Different Topologies

(20 particles, Rosenbrock) : Square topology

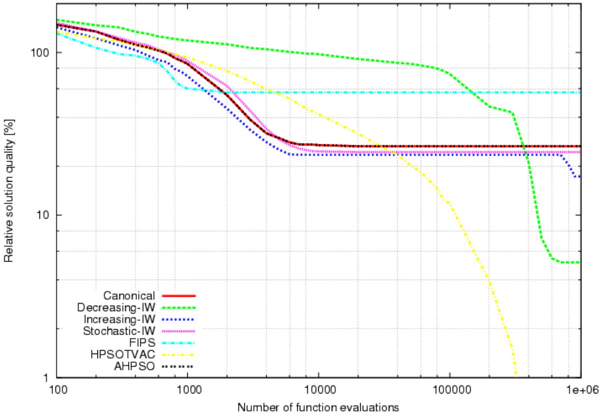


Particle swarm optimization: Comparisons: Different Topologies

(20 particles, Rosenbrock) : Ring topology

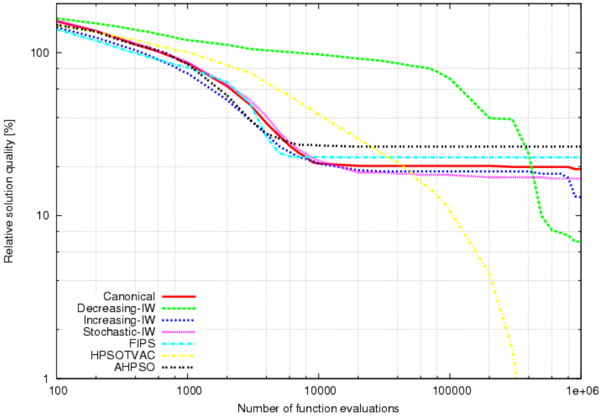


(20 particles, Rastrigin) : Fully connected topology



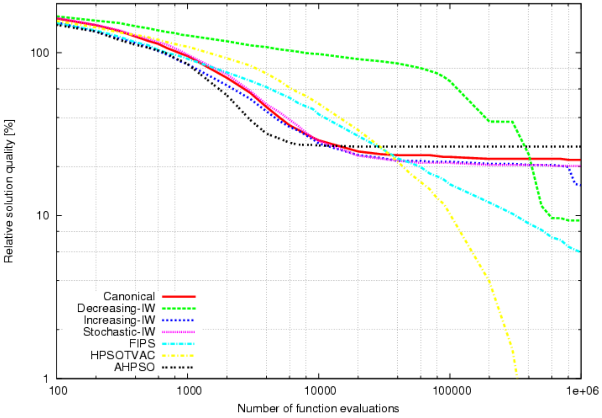
Particle swarm optimization: Comparisons: Different Topologies

(20 particles, Rastrigin) : Square topology



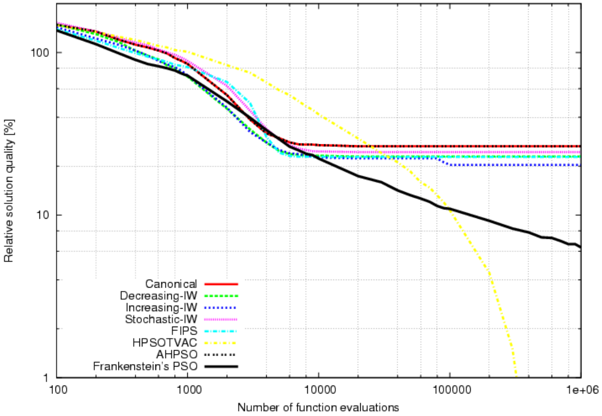
Particle swarm optimization: Comparisons: Different Topologies

(20 particles, Rastrigin) : Ring topology



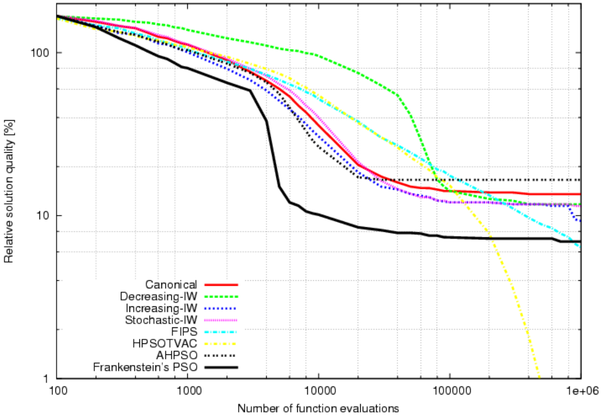
Particle swarm optimization: New variants (Frankenstein's PSO)

Rastrigin (best configurations for speed)



Particle swarm optimization: New variants (Frankenstein's PSO)

Rastrigin (best configurations for quality)



Thank you

(More) Questions?

http://iridia.ulb.ac.be/~mmontes/slidesCIL/slides.pdf

Peter J. Angeline.

Using selection to improve particle swarm optimization.

In *Proceedings of the 1998 IEEE Congress on Evolutionary Computation*, pages 84–89, Piscataway, NJ, USA, 1998. IEEE Press.

Tim M. Blackwell and Peter J. Bentley.

Dońt push me collision-avoiding swarms.

In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, pages 1691–1696, Piscataway, NJ, USA, 2002. IEEE Press.

Maurice Clerc and James Kennedy.

The particle swarm–explosion, stability, and convergence in a multidimensional complex space.

IEEE Transactions on Evolutionary Computation, 6(1):58–73, 2002.

Holger H. Hoos and Thomas Stützle.

Stochastic Local Search: Foundations and Applications.

Morgan Kaufmann, San Francisco, CA, USA, 2004.

Mudassar Iqbal and Marco A. Montes de Oca.

An estimation of distribution particle swarm optimization algorithm.

In Marco Dorigo, Luca M. Gambardella, Mauro Birattari, Alcherio Martinoli, Riccardo Poli, and Thomas Stützle, editors, *LNCS 4150. Ant Colony Optimization and Swarm Intelligence. 5th International Workshop, ANTS 2006*, pages 72–83, Berlin, Germany, 2006. Springer-Verlag.

James Kennedy and Russell Eberhart.

Particle swarm optimization.

In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ, USA, 1995. IEEE Press.

James Kennedy and Russell Eberhart.

A discrete binary version of the particle swarm algorithm.

In *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics*, pages 4104 – 4108, Piscataway, NJ, USA, 1997. IEEE Press.

Jing Liu, Wenbo Xu, and Jun Sun.

Quantum-behaved particle swarm optimization with mutation operator.
In *Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence. ICTAI 05*, pages 237 – 240, Piscataway, NJ, USA, 2005. IEEE Press.

Rui Mendes, James Kennedy, and José Neves.
The fully informed particle swarm: Simpler, maybe better.
IEEE Transactions on Evolutionary Computation, 8(3):204–210, 2004.

Arvind Mohais, Rui Mendes, Christopher Ward, and Christian Postoff.
Neighborhood re-structuring in particle swarm optimization.
In Shichao Zhang and Ray Jarvis, editors, *LNCS 3809. Proceedings of the 18th Australian Joint Conference on Artificial Intelligence*, pages 776–785, Berlin, 2005. Springer.

Riccardo Poli, Cecilia Di Chio, and William B. Langdon.
Exploring extended particle swarms: A genetic programming approach.

In *Proceedings of the 2005 conference on Genetic and Evolutionary Computation*, pages 169–176, New York, NY, USA, 2005. ACM Press.

Craig W. Reynolds.

Flocks, herds, and schools: A distributed behavioral model.

ACM Computer Graphics, 21(4):25–34, 1987.

J. Riget and J. Vesterstroem.

A diversity-guided particle swarm optimizer - the arpso.

Technical Report 2002-02, Department of Computer Science, University of Aarhus, 2002.

Yuhui Shi and Russell Eberhart.

A modified particle swarm optimizer.

In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 69–73, Piscataway, NJ, USA, 1998. IEEE Press.

Yuhui Shi and Russell Eberhart.

Empirical study of particle swarm optimization.

In *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, pages 1945–1950, Piscataway, NJ, USA, 1999. IEEE Press.

Ponnuthurai N. Suganthan.
Particle swarm optimiser with neighbourhood operator.
In *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, pages 1958–1962, Piscataway, NJ, USA, 1999. IEEE Press.

Ioan C. Trelea.
The particle swarm optimization algorithm: Convergence analysis and parameter selection.
Information Processing Letters, 85(6):317–325, 2003.

Kang-Ping Wang, Lan Huang, Chun-Guang Zhou, and Wei Pang.
Particle swarm optimization for traveling salesman problem.
In *Proceedings of the 2003 IEEE International Conference on Machine Learning and Cybernetics*, pages 1583 – 1585, Piscataway, NJ, USA, 2003. IEEE Press.