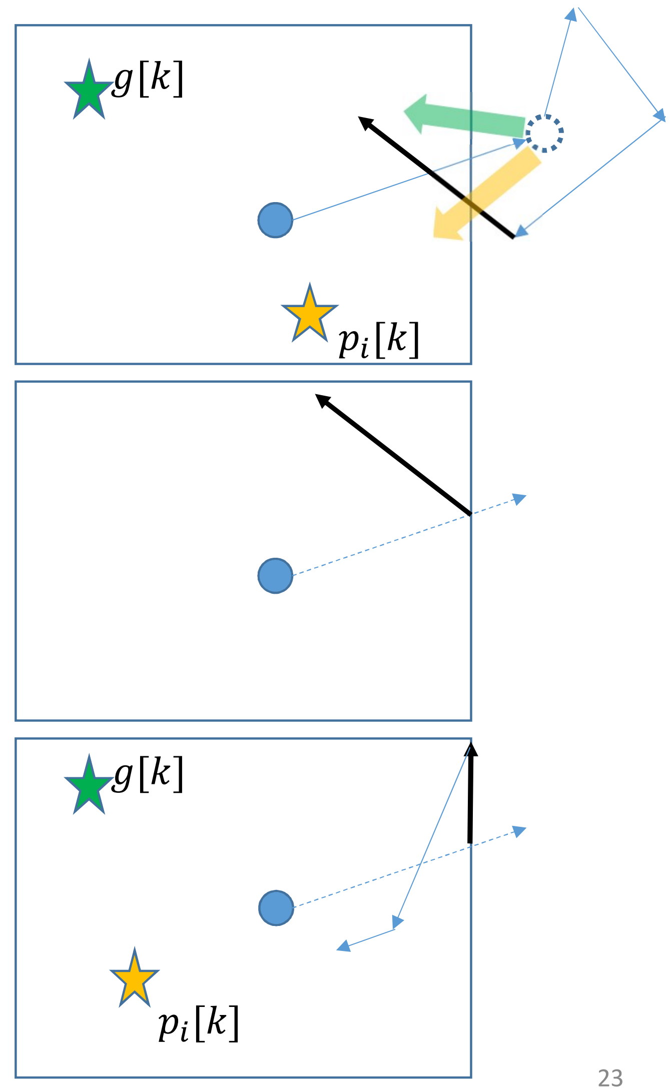
# Basic algorithm settings

$$v_i^j[k+1] = w\, v_i^j[k] + c_1 r_{1,j}(p_i^j[k] - x_i^j[k]) + c_2 r_{2,j}(g^j[k] - x_i^j[k])$$
$$x_i^j[k+1] = x_i^j[k] + v_i^j[k+1]$$

- Velocity clamping: $\left|v_i^j[k+1]\right| \leq v_{max}$
- Inertia weight decreases linearly with time
- Acceleration constants stay fixed
- Very important: Independent random number for each dimension
  - Common mistake made by beginners!

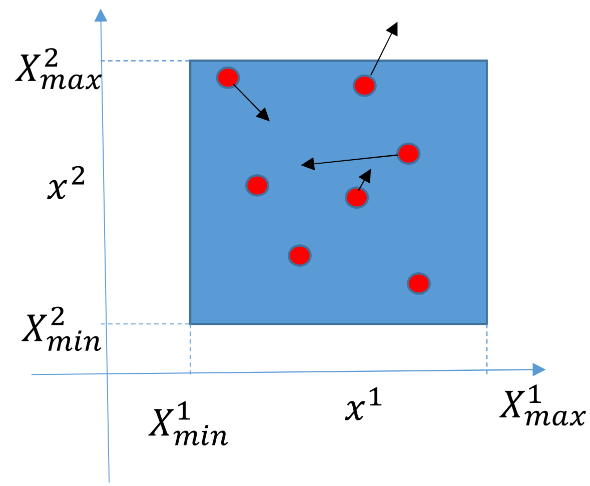
# Boundary conditions

- **"Let them fly":** set the fitness value to $+\infty$ outside the boundary and continue to iterate the dynamical equations
  - pbest and gbest are always in the search space and eventually pull the particle back
- "Reflecting walls" : Change the sign of the velocity component perpendicular to the boundary surface
- "Absorbing Walls": zero the velocity component perpendicular to the boundary surface



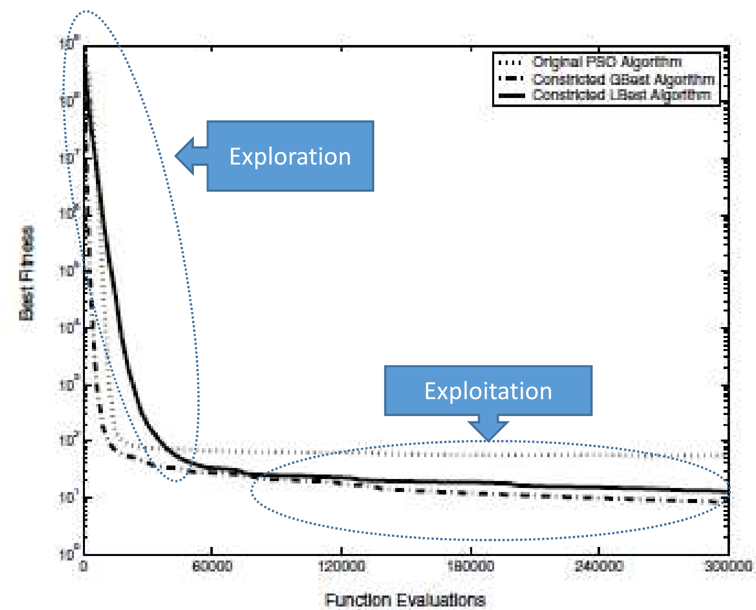
# Initialization and termination

- $x_i^j[0]$ is picked from a uniform distribution over $[X_{max}^j, X_{min}^j]$
    - Search space assumed to be a hypercube
- Initial velocity (variations):
    - Uniform distribution with velocity clamping
    - Zero initial velocities
    - Boundary constrained:
        - $v_i^j[0] \sim U(X_{min}^j - x_i^j[0], X_{max}^j - x_i^j[0])$ & velocity clamping
- Termination condition (variations):
    - Number of iterations (simplest)
    - Density of swarm
    - …



# Exploration and Exploitation in PSO

- $v_i^j[k+1] = w\, v_i^j[k] + c_1 r_{1,j}(p_i^j[k] - x_i^j[k]) + c_2 r_{2,j}(g[k] - x_i^j[k])$
- The inertia term and randomization promotes exploration
- Social and cognitive terms promote exploitation
- Inertia weight $w < 1 \Rightarrow$ velocity decreases if inertia term is the only term: Promotes exploitation
    - Position converges to a limiting value
    - Emulates "friction"
- Decay of inertia weight favors more exploitation at later stages
    - Linear decay: $w \rightarrow w[k] = w_{max} - (w_{max} - w_{min}) * k/(N-1)$
- Before inertial weight was introduced, Instability and divergence of swarm ("explosion") required velocity clamping
    - Velocity clamping : Smaller $v_{max}$ means less exploration
- Generous velocity clamping still recommended to prevent too many particles leaking out of the search space

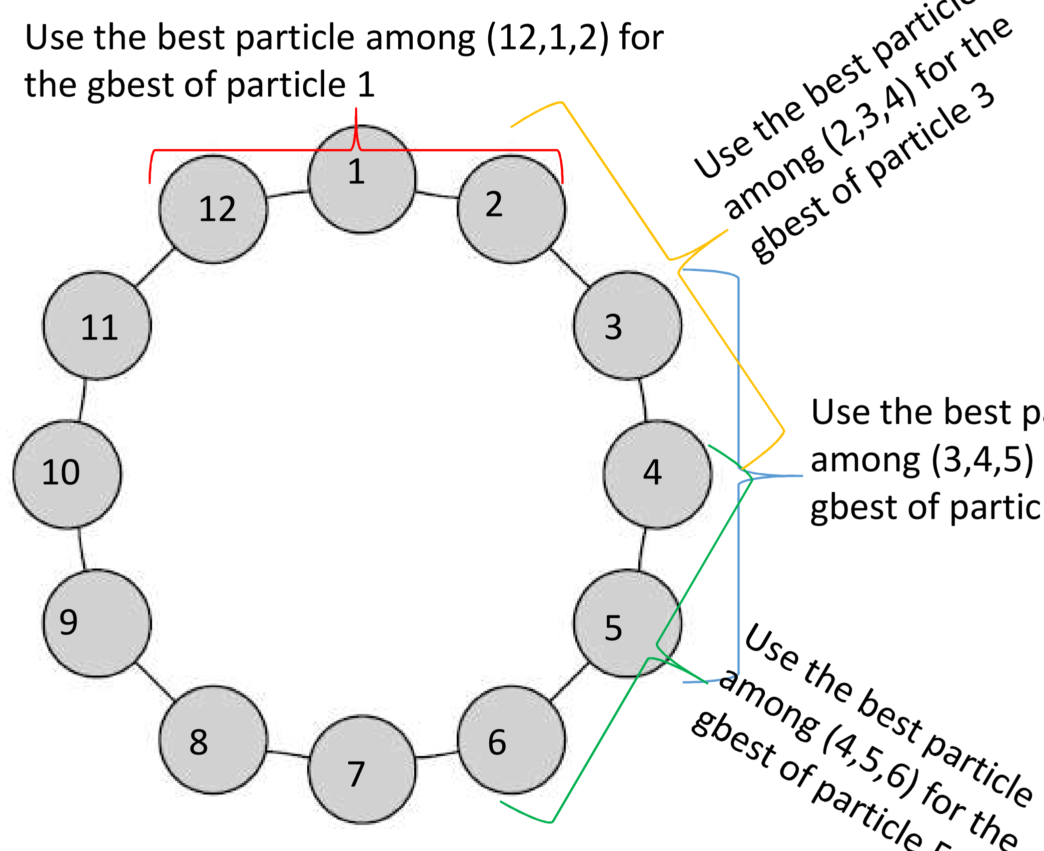


# Alternative algorithms under the PSO metaheuristic

See textbook by Engelbrecht for a more comprehensive review

# Local best (lbest) PSO

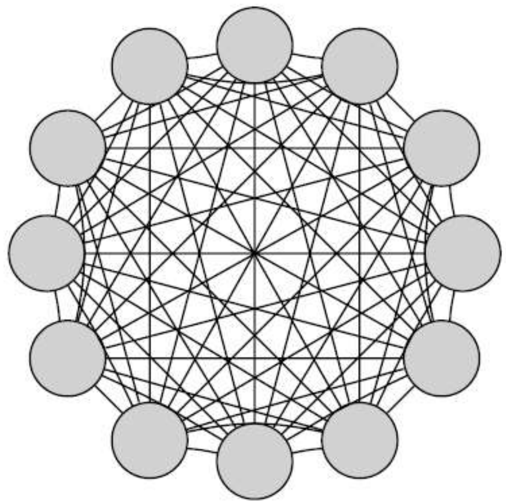
- PSO dynamical equations:
  - $v_i^j[k+1] = w\, v_i^j[k] + c_1 r_{1,j}(p_i^j[k] - x_i^j[k]) + c_2 r_{2,j}(g[k] - x_i^j[k])$
- Local best PSO:
  - $g[k] \rightarrow l_i[k]$ : best value among particles in the neighborhood of the $i^{\text{th}}$ particle.
  - Information about global best propagates more slowly through the swarm
  - More time spent in exploration



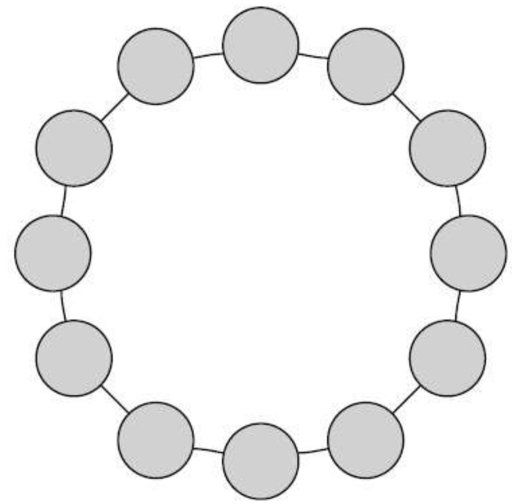

(b) The lbest ring topology

# Topologies

- Gbest
- Ring
- Star
- Wheel
- Pyramid
- Four Clusters
- Von-Neumann
- …



(a) The gbest topology

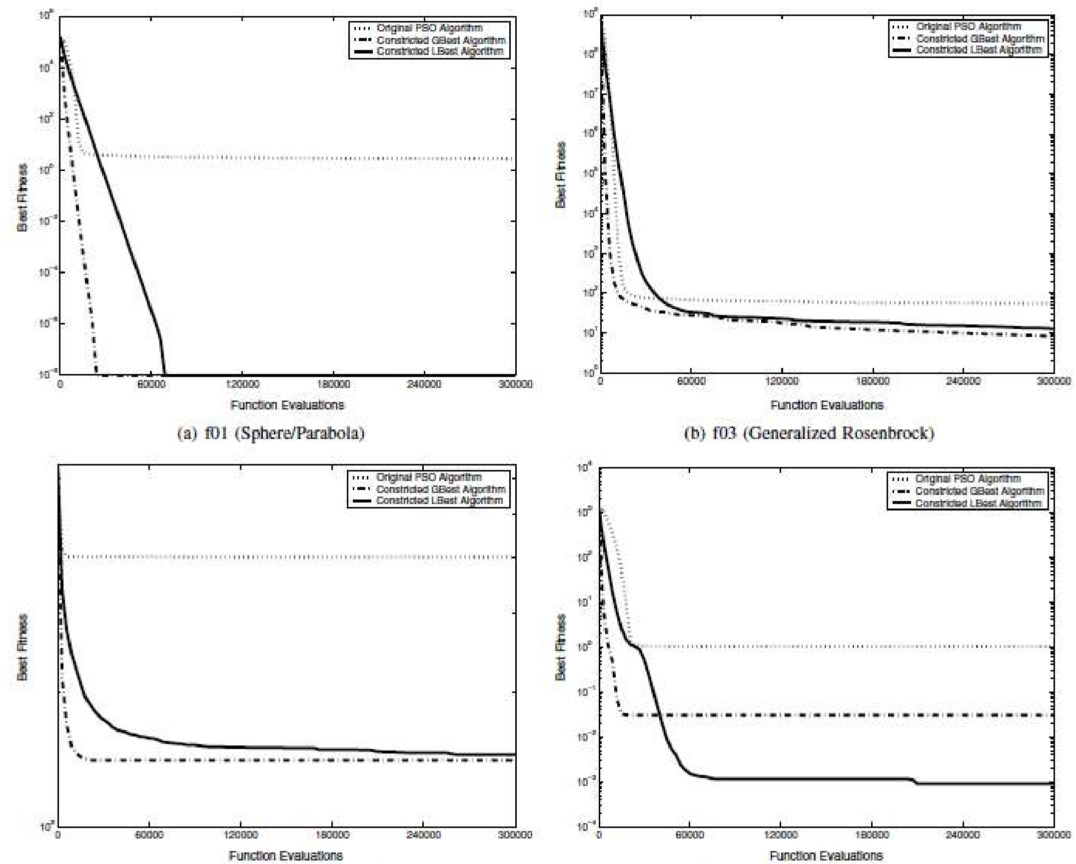


(b) The lbest ring topology

# Lbest PSO

- Increases the time spent in exploration
- Tends to become better than gbest PSO as:
  - the problem dimensionality increases and/or
  - fitness function becomes more rugged
- Penalties:
  - Slower convergence and
  - More fitness function evaluations (main computational cost)

From Bratton & Kennedy, 2007




(a) f01 (Sphere/Parabola)
(b) f03 (Generalized Rosenbrock)
(c) f05 (Generalized Rastrigin)
(d) f07 (Generalized Griewank)

# Termination (Variations)

- Stop when maximum number of iterations reached
- Stop when an acceptable solution has been found
  - If $x^*$ is known (e.g., benchmark functions), stop when $|f(x_k) - f(x^*)| < \epsilon$
  - Stop if the average change in particle positions is small
  - Stop if the average velocity over a number of iterations is close to zero
  - Stop if there is no significant improvement in the fitness value over a number of iterations
- Stop when the normalized swarm radius is close to zero
  - $R_{norm} = \frac{R_{max}}{initial\ R_{max}}$; $R_{max} = \max_i \|x_i[k] - g[k]\|$
- Stop when the best particle does not move out of a small box over a specified number of iterations (Wang, Mohanty, 2010)
- Load balancing considerations in parallel implementations may prefer termination by fixed number of iterations

# Inertia Weight Decay (Variations)

- Random adjustments: different inertia is randomly selected at each iteration
- Non-linear decrease
- Increasing inertia (!)
- Jumpstart (Wang, Mohanty, PRD, 2010): Increase inertia whenever gbest goes outside of a small box.

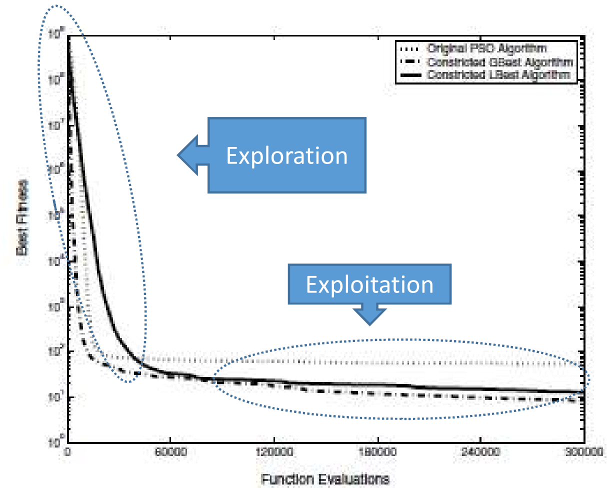
# Velocity constriction

- Velocity clamping is needed to keep particles from moving out of the search space ("explosion")
- Velocity constriction is another way to contain a particle explosion
- $v_i^j[k+1] = K\left[v_i^j[k] + c_1 r_{1,j}\left(p_i^j[k] - x_i^j[k]\right) + c_2 r_{2,j}\left(g^j[k] - x_i^j[k]\right)\right]$
- $K$ is called the constriction factor
  - $K = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|}$
  - $c = c_1 + c_2 > 4$
- Standard choice for $K$ is 0.729 corresponding to $c = 4.01$
- Normally, $c_1 = c_2 \Rightarrow 2.05$
- Even without velocity constriction, $c_1 = c_2 \simeq 2$ is widely adopted in the literature

# Memetic search

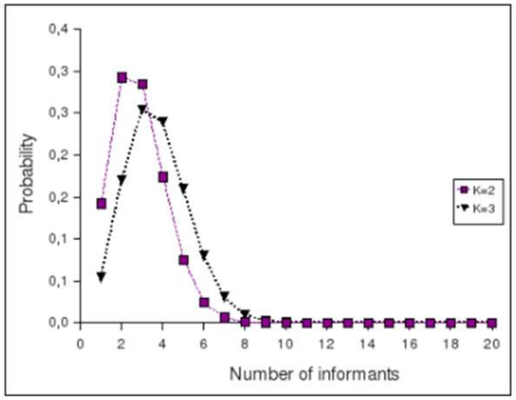
- Global optimizers like PSO lack exploitation ability in the late stages of optimization → slow convergence to optimum
- Local optimizers, e.g., steepest descent, can find a local optimum much faster

1. Use local optimizer to refine the gbest or lbest solutions
2. Use stochastic search in a neighborhood of gbest or lbest (GCPSO: increase search area if a better solution is found)

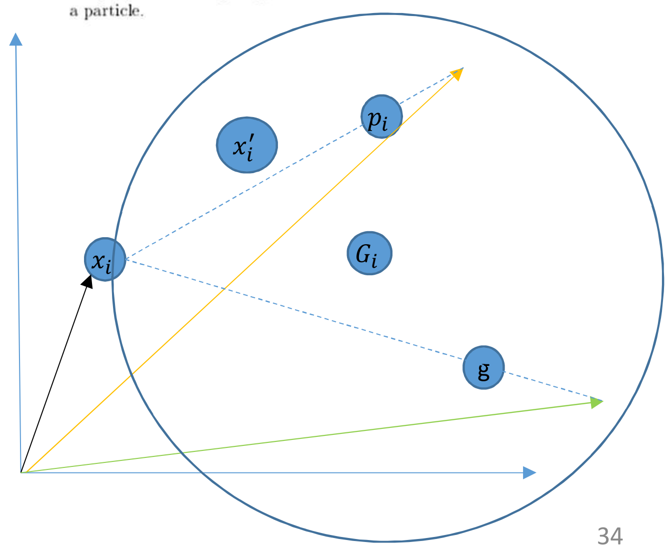


# Standard PSO (SPSO)

- SPSO '06, '07, '11: [http://clerc.maurice.free.fr/pso/SPSO_descriptions.pdf](http://clerc.maurice.free.fr/pso/SPSO_descriptions.pdf)
- Number of particles = 40
- Ring topology (2 nearest neighbors) / neighborhood sizes picked randomly
- Velocity initialization:
  - $v_i^j[0] \sim U(X_{min}^j - x_i^j[0], X_{max}^j - x_i^j[0])$
- Velocity update:
  - $3G_i = x_i + \left(x_i + c(p_i - x_i)\right) + \left(x_i + c(g - x_i)\right)$
  - $x_i'$: Point picked randomly in the sphere centered on $G_i$ with radius $\|G_i - x_i\|$
  - $v_i[k+1] = w\, v_i[k] + (x_i' - x_i)$
- Reflecting inelastic walls: $v_i[\mathrm{k}+1] = -0.5\, v_i[k+1]$

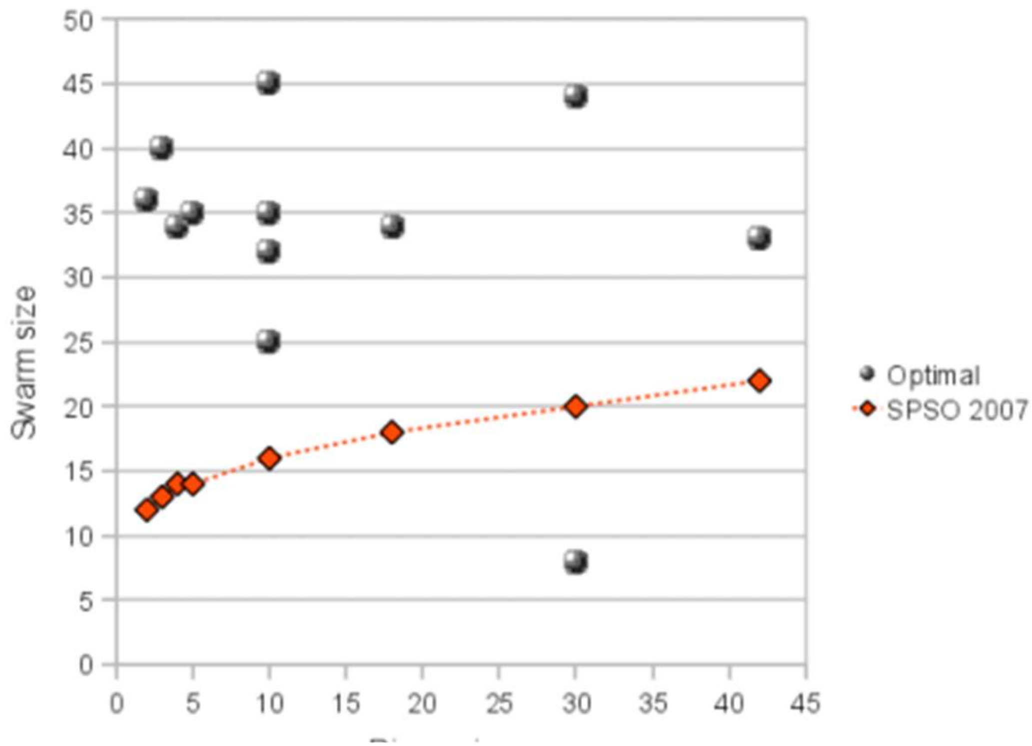



3.2: Adaptive random topology. Distribution of the number of informants of a particle.



# Recommended PSO parameter settings

- Bratton and Kennedy, 2007
- $\sim 40$ particles
  - Too many can cause premature convergence to a local optimum!
  - Too few and the space is not explored properly
  - Actual number will depend on computational costs
- Local Best (lbest) with ring topology (2 nearest neighbors)
  - Increases exploration
  - Slower convergence but often better probability of success
- Linearly decaying inertia weight (> 1 to <1)
- "Let them fly" boundary condition



## TABLE I
BENCHMARK FUNCTIONS

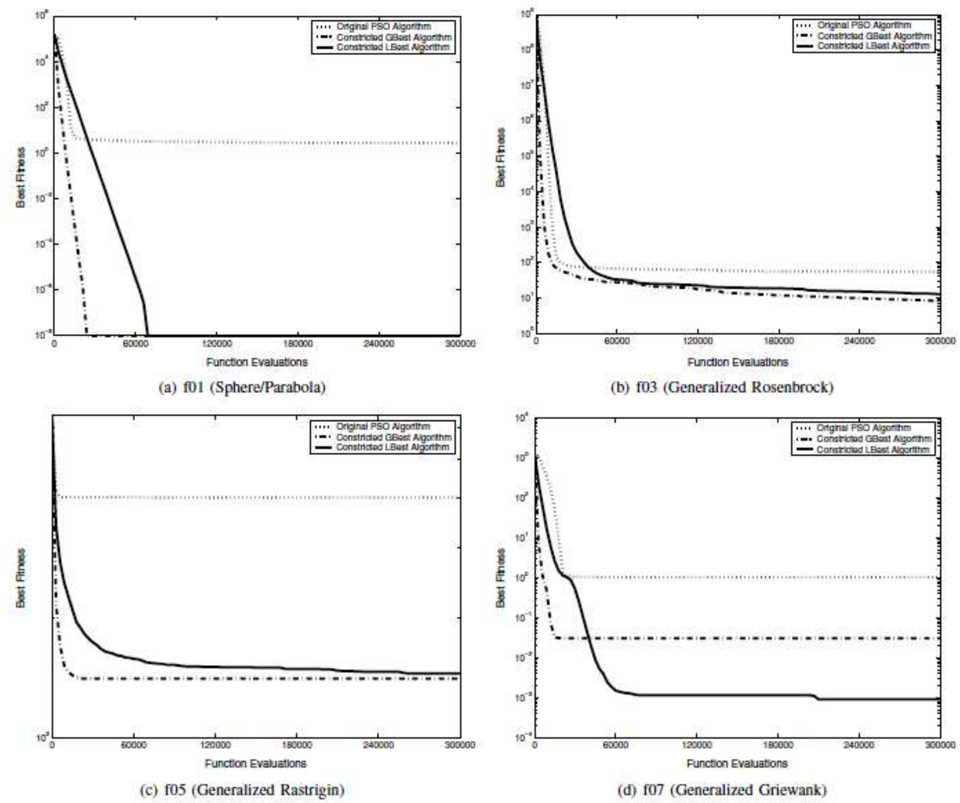
| Equation | Name | D | Feasible Bounds |
|---|---|---|---|
| $f_1 = \sum_{i=1}^{D} x_i^2$ | Sphere/Parabola | 30 | $(-100, 100)^D$ |
| $f_2 = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$ | Schwefel 1.2 | 30 | $(-100, 100)^D$ |
| $f_3 = \sum_{i=1}^{D-1} \left\{ 100 \left( x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right\}$ | Generalized Rosenbrock | 30 | $(-30, 30)^D$ |
| $f_4 = -\sum_{i=1}^{D} x_i \sin \left( \sqrt{x_i} \right)$ | Generalized Schwefel 2.6 | 30 | $(-500, 500)^D$ |
| $f_5 = \sum_{i=1}^{D} \left\{ x_i^2 - 10 \cos (2\pi x_i) + 10 \right\}$ | Generalized Rastrigin | 30 | $(-5.12, 5.12)^D$ |
| $f_6 = -20 \exp \left\{ -0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right\} - \exp \left\{ \frac{1}{D} \sum_{i=1}^{D} \cos (2\pi x_i) \right\} + 20 + e$ | Ackley | 30 | $(-32, 32)^D$ |
| $f_7 = \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1$ | Generalized Griewank | 30 | $(-600, 600)^D$ |
| $f_8 = \frac{\pi}{D} \left\{ 10 \sin^2 (\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 \left\{ 1 + 10 \sin^2 (\pi y_{i+1}) \right\} + (y_D - 1)^2 \right\}$ $+ \sum_{i=1}^{D} \mu (x_i, 10, 100, 4)$ <br> $y_i = 1 + \frac{1}{4} (x_i + 1)$ <br> $\mu (x_i, a, k, m) = \begin{cases} k (x_i - a)^m & x_i > a \\ 0 & -a \leq x_i \leq a \\ k (-x_i - a)^m & x_i < -a \end{cases}$ | Penalized Function P8 | 30 | $(-50, 50)^D$ |
| $f_9 = 0.1 \left\{ \sin^2 (3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 \left\{ 1 + \sin^2 (3\pi x_{i+1}) \right\} + (x_D - 1)^2 \times \right.$ $\left. \left\{ 1 + \sin^2 (2\pi x_D) \right\} \right\} + \sum_{i=1}^{D} \mu (x_i, 5, 100, 4)$ | Penalized Function P16 | 30 | $(-50, 50)^D$ |
| $f_{10} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3} x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$ | Six-hump Camel-back | 2 | $(-5, 5)^D$ |
| $f_{11} = \left\{ 1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2) \right\} \times$ $\left\{ 30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2) \right\}$ | Goldstein-Price | 2 | $(-2, 2)^D$ |
| $f_{12} = -\sum_{i=1}^{5} \left\{ \sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i \right\}^{-1}$ | Shekel 5 | 4 | $(0, 10)^D$ |
| $f_{13} = -\sum_{i=1}^{7} \left\{ \sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i \right\}^{-1}$ | Shekel 7 | 4 | $(0, 10)^D$ |
| $f_{14} = -\sum_{i=1}^{10} \left\{ \sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i \right\}^{-1}$ | Shekel 10 | 4 | $(0, 10)^D$ |

From Bratton & Kennedy, 2007

# Benchmarking metrics

From Bratton & Kennedy, 2007

- Average best fitness over multiple independent runs of the method
  - Independent initialization of particle position and velocities
- Plot average best fitness vs iteration
  - Exploration vs exploitation behavior




(a) f01 (Sphere/Parabola)

(b) f03 (Generalized Rosenbrock)

(c) f05 (Generalized Rastrigin)

(d) f07 (Generalized Griewank)

# Best of M runs

- Stochastic search algorithms like PSO are not guaranteed to find the global optimum (or even be close to it!)
- We can only ensure that the probability of finding the global optimum is sufficiently high
- Let the probability of "success" in one run of PSO on a fixed fitness function be $p$
- Then the probability of failure over M independent runs of PSO (independent initial positions and velocities) is $\sim (1-p)^M$
  - Decreases exponentially fast with $M$: If $p = 0.1$, failure probability over $M = 10$ runs is $\simeq 0.35$, which is significantly less than failure probability of $0.9$ for a single run.
  - If $p = 0.5$, failure probability is $10^{-3}$!
- Strategy: tune PSO such that single run success probability is reasonably high, then pick best of $M$ runs
  - Use fixed random number seeds for the different runs, so that pseudo-random number cycle length is not exhausted

# Best of $M$ runs

- Spend more effort on tuning or …
- Get something sufficiently good and do best of $M$ runs?
- Recommendation for problems that admit modeling and multiple realizations of optimization problem (i.e., Regression)
  - Do tuning until probability of success $p$ is $\sim 30\%$ to $50\%$ …
    - Often requires only minor adjustments in the case of PSO
    - Need to run optimization method over multiple realization of optimization problem
  - Then switch to best of $M$ runs for tuned algorithm
    - Choice of M is easy if $p$ is known!
  - Probability of failure decreases exponentially with $M$

# Parallelization

- Best of $M$ runs: Parallelization over runs is easy
- Parallelization is quite cheap these days
  - Multi-core processors
  - Breakdown of single core Moore's law: Expect cheaper multi-core computing
  - Intel Xeon Phi (2016) launched: 68 cores / 272 threads
- Easy to implement
  - Matlab: Parallel Computing Toolbox
  - Open MP
- For costly fitness functions: parallelize further over particles
  - MPI allows multi-level parallelization

# PSO resources

- References
  - *Original paper:* Eberhart, R. C. and Kennedy, J. A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995
- Public domain codes at: http://www.particleswarm.info/Programs.html
- “Standard PSO” (SPSO) codes (developed by M. Clerc) available from the above site
- Matlab: `particleswarm` function
- SPSO and Matlab implement different algorithms within the PSO metaheuristic
- http://www.swarmintelligence.org/ is another reference site