Distributed Particle Swarm Optimization

Salman Kahrobaee

CSCE 990 Seminar

References:

Main Reference:

- A Comparative Study of Four Parallel and Distributed PSO Methods

Leonardo VANNESCHI, Daniele CODECASA and Giancarlo MAURI

New Generation Computing, Volume 29, Issue 2, pp 129-161, April 2011

Other References:

- Applying Multi-Swarm Accelerating Particle Swarm Optimization to Dynamic Continuous Functions

Yi Jiang, Wei Huang, Li Chen

Second International Workshop on Knowledge Discovery and Data Mining, WKDD 2009

- Distributed Adaptive Particle Swarm Optimizer in Dynamic Environment

Xiaohui Cui and Thomas E. Potok

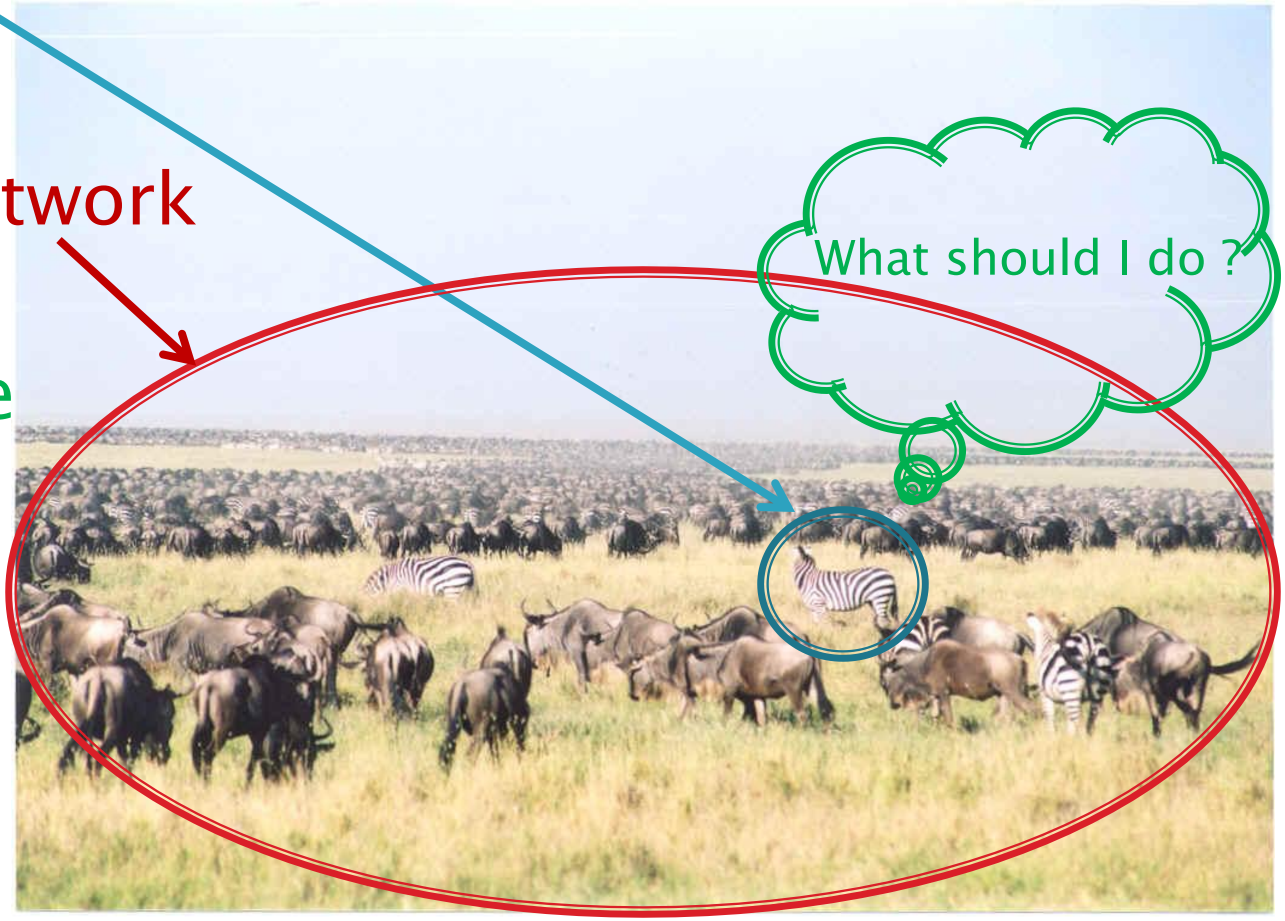
IEEE International Parallel and Distributed Processing Symposium, IPDPS 2007

Outline

- Introduction
- Particle Swarm Optimization
- Extensions of PSO
- Particle Swarm Evolver (PSE)
- Repulsive PSE (RPSE)
- Multi-swarm PSO (MPSO)
- Multi-swarm Repulsive PSO (MRPSO)
- Case studies and Results
- Conclusion
- Praises
- Critiques
- PSO methods for dynamic environments
- Proposed PSO application in a Smart Grid

Introduction

- Particle
- Social network
- Objective



Introduction

Particle Swarm Optimization

- An iterative computation technique developed by Dr. Eberhart and Dr. Kennedy in 1995

- Inspired by social behavior of animals e.g. bird flocking and fish schooling

Particle Swarm Optimization:

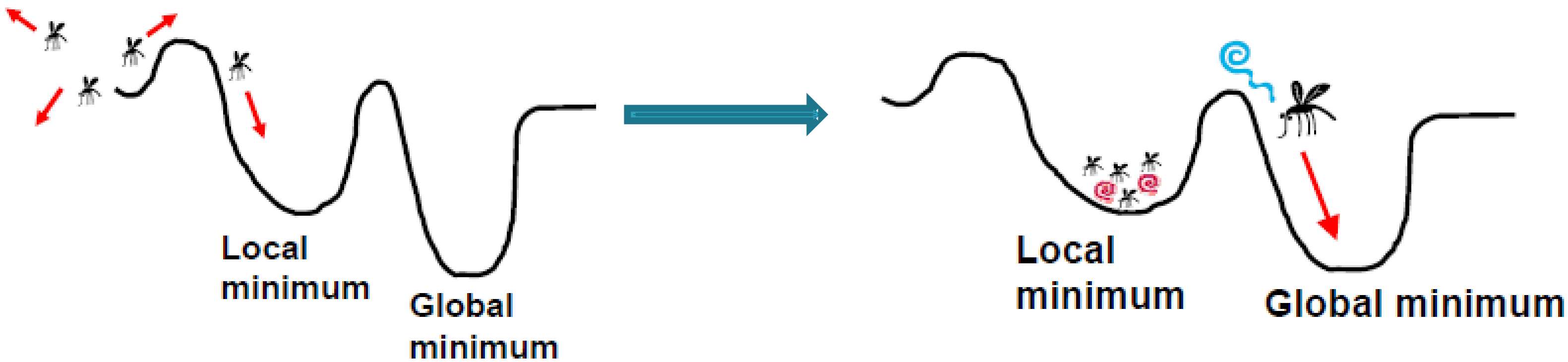
- Particles positions: candidate solution
- Environment: problem search space
- Solution evaluation: fitness function
- Own best solution
- Other’s best solution

- Movement of particles:

  exploration vs. exploitation

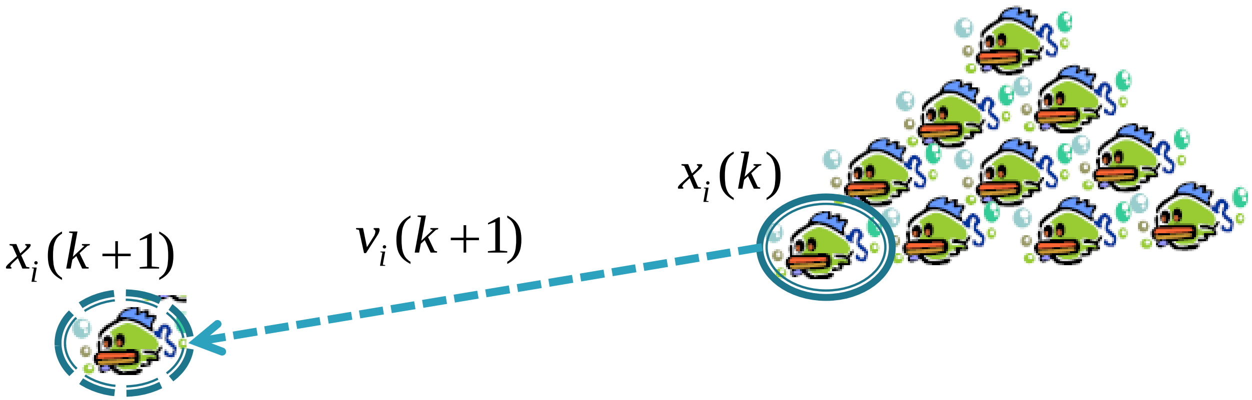
Particle Swarm Optimization:

- Exploration vs. Exploitation



- Exploration:
  - Global minimum
  - Adaptability
- Exploitation:
  - Stability

Particle Swarm Optimization



- Particle's position update

$$x_i(k+1) = x_i(k) + v_i(k+1)$$

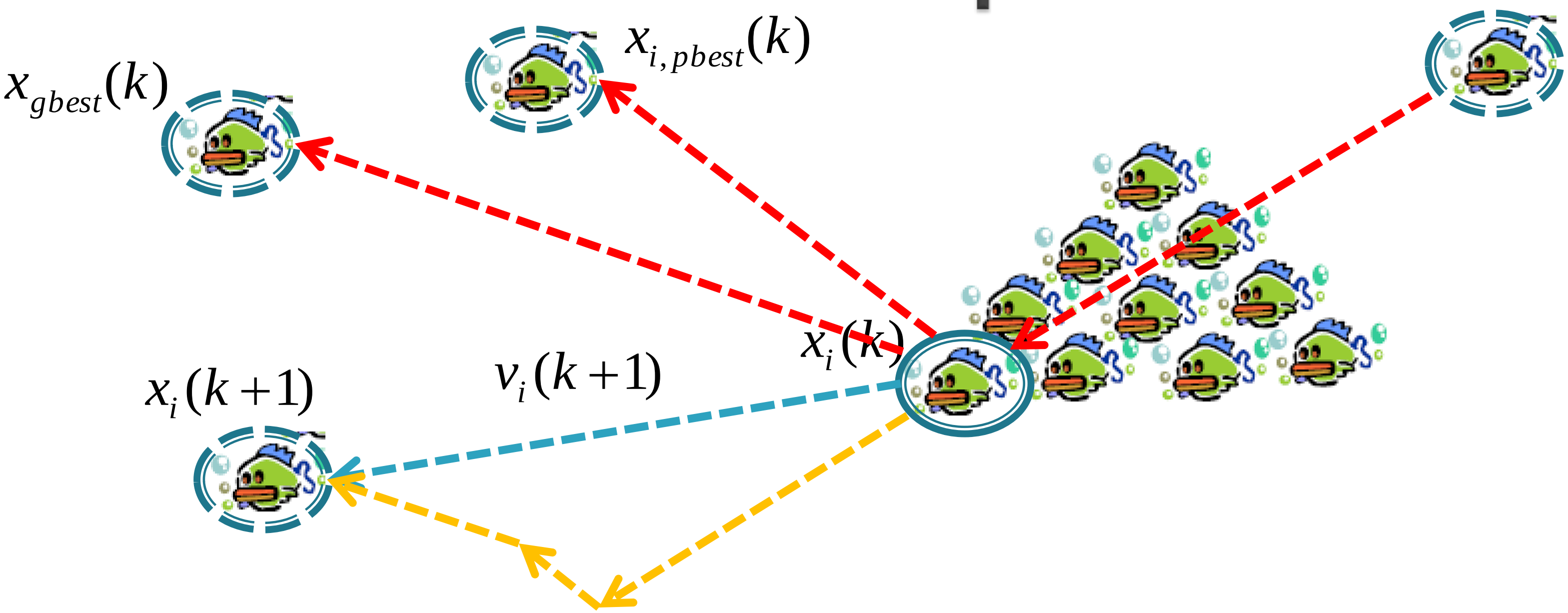
Particle index

Position

Iteration

Velocity

Particle Swarm Optimization



- Particle's velocity update

$$v_i(k+1) = w \cdot v_i(k) + c_1\phi_1(x_{i,pbest}(k) - x_i(k)) + c_2\phi_2(x_{gbest}(k) - x_i(k))$$

Inertia constant

Acceleration constant

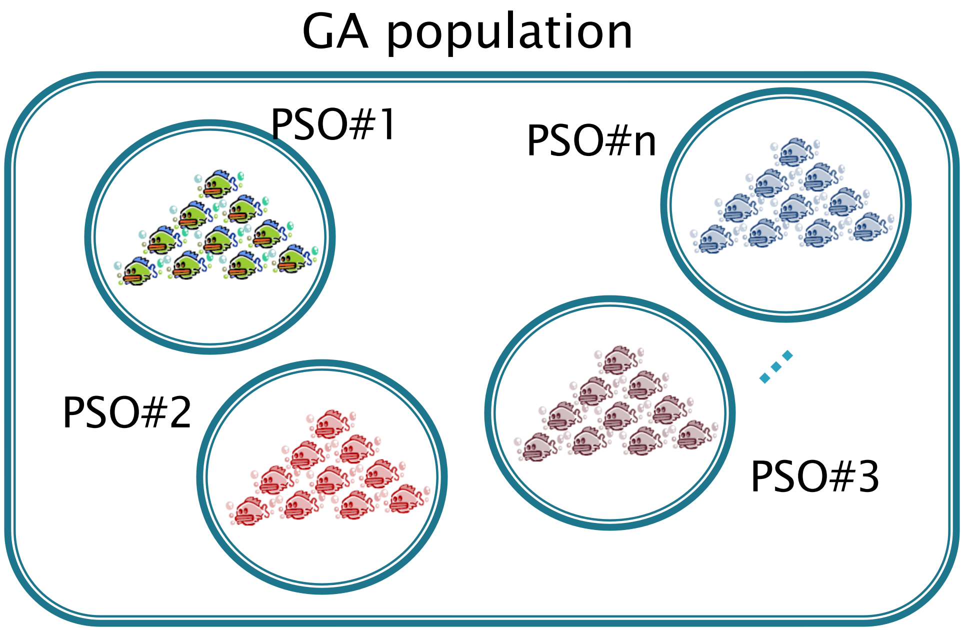
Random numbers in U{0,1}

Extensions of PSO

- Optimization of PSO parameters
- Different topologies of swarm
- Conjunction of PSO with Evolutionary Algorithms; e.g. GA
- Multi-swarm PSO
- Master-Slave PSO
- Attractive/Repulsive PSO

Particle Swarm Evolver (PSE)

- Hybrid PSO-GA method

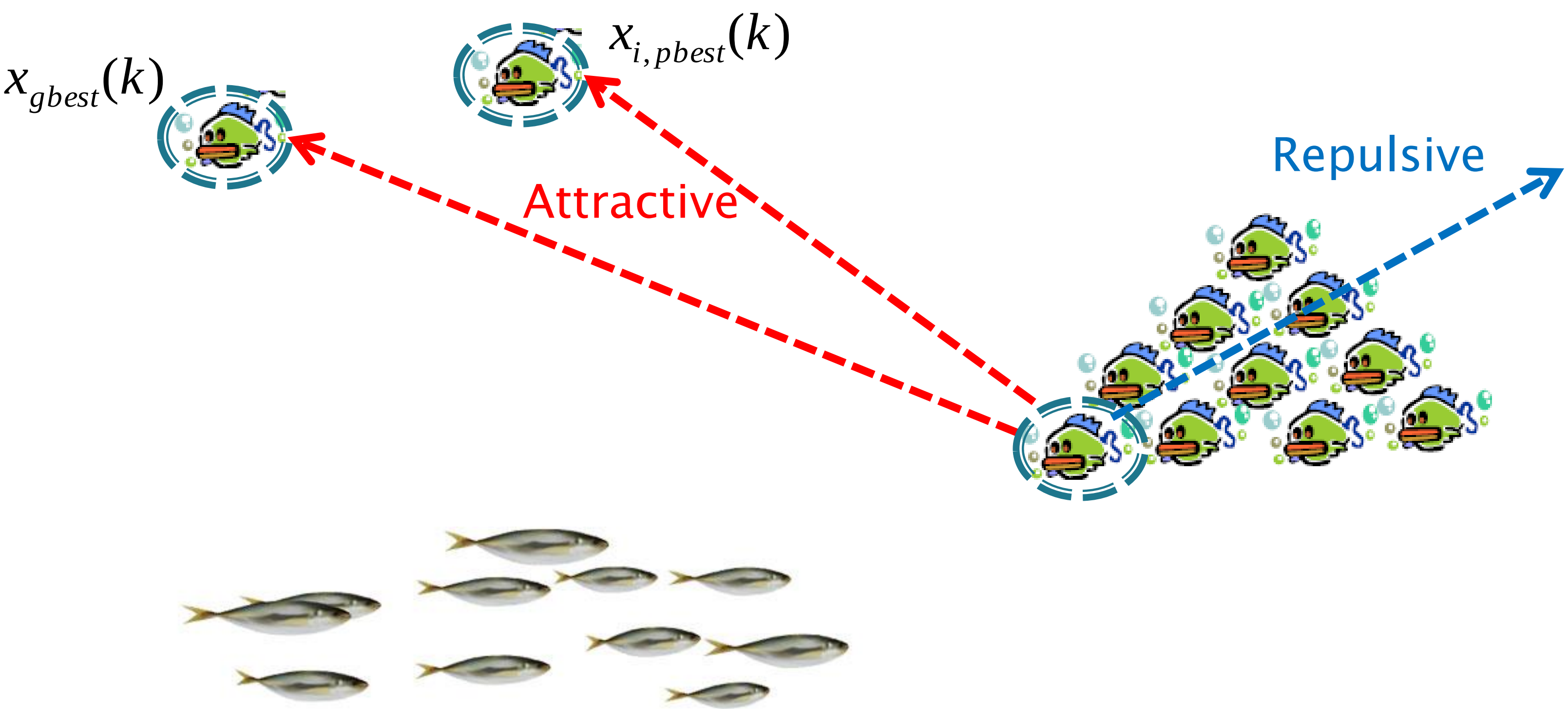


Particle Swarm Evolver (PSE)

- Each chromosome = independent PSO
- Method:
  - Perform P number of iterations for each PSO
  - Choose the best PSOs based on their x_{gbest} as parents
  - Perform cross over by randomly mixing their particles
  - Perform mutation by replacing a random particle in PSO with a completely random particle
  - Repeat the process for the convergence
- Crossover probability=0.95
- Mutation probability=0.01

Repulsive PSE (RPSE)

- A repulsive component is added to PSE



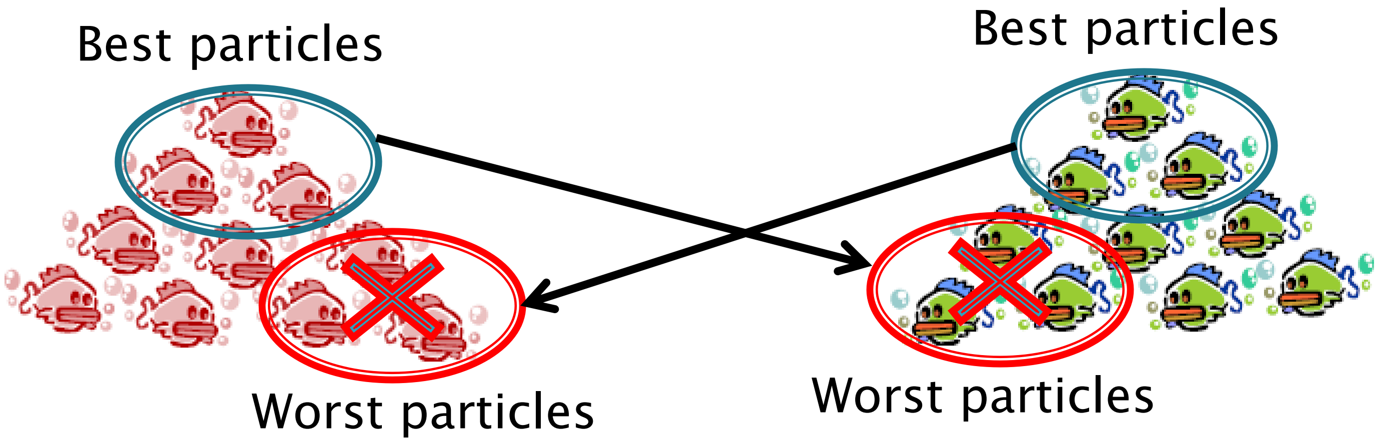
Repulsive PSE (RPSE)

- Particle of each swarm is *attracted* by the local/global best of its own swarm
- Particle of each swarm is *repulsed* by the global best of all other swarms

$$v_i(k+1) = v_{i,PSO}(k+1) + c_3\phi_3 f(x_{foreign-gbest}(k), x_{gbest}(k), x_i(k))$$

Multi-swarm PSO (MPSO)

- An alternative to the PSE algorithm



Multi-swarm PSO (MPSO)

- A set of independent swarms
- Communicate using a ring topology
- Method:
  - Run PSO for a number of iterations
  - Have an interaction
  - *k* best particles in the sender swarm is sent to the receiver swarm
  - The new particles replace the worst *k* ones in the receiver swarm

Multi-swarm Repulsive PSO (MRPSO)

- A repulsive components is added to MPSO for half of the swarms
- The exchange of particles is between one PSO with repulsive component and one without
- Migrated particles from the sender are very different from those in the receiver due to repulsive effect

Case studies and Results

- Parameters:
  - 100 total number of particles in all PSO methods
  - 200 independent run for each PSO method

- Evaluation metrics:
  - Number of successful runs
  - Average best fitness

Case studies and Results

- Test function:

$$cosff(\mathbf{x}) = (\sum_{i=1}^{n} f_i(x_i, M_i))/n$$

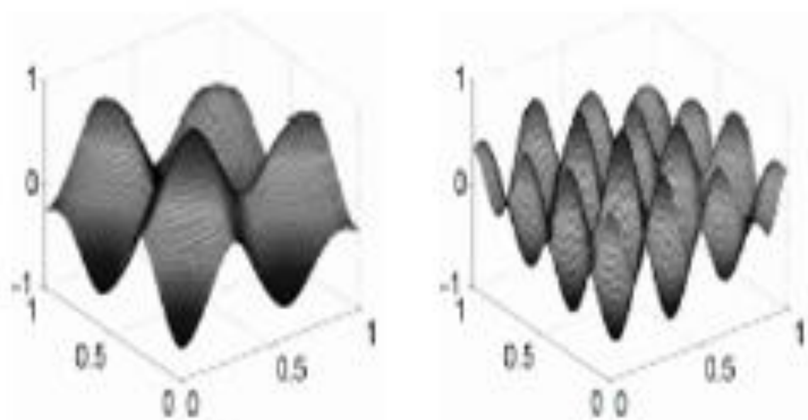
$$f_i(x, M) = \begin{cases} \cos(K * (x - M)) * (1.0 - (M - x)), & \text{if } x \le M \\ \cos(K * (x - M)) * (1.0 - (x - M)), & \text{otherwise} \end{cases}$$

- n : dimension of the problem
- $\{M_1, M_2, \ldots M_n\}$: coordinates of maximum value of the function
- K : ruggedness constant of the environment

Case studies and Results

- Example of a 2-dimension cosff(x):

$$cosff(\mathbf{x}) = (\sum_{i=1}^{n} f_i(x_i, M_i))/n$$



(a) (b)

$K = 10$ $K = 20$

$n = 2$

$M_1 = M_2 = 0.3$

Case studies and Results

- Results for a 20-dimension cosff(x):

$$n = 20 \quad K = 10$$

Number of successful runs:

Method	no. succ.	std. dev.
PSO	74	6.82
PSE	44	5.58
RPSE	1	0.99
MPSO	180	4.24
MRPSO	188	3.35

(a)

$M_i = 0.1$

Method	no. succ.	std. dev.
PSO	8	2.77
PSE	2	1.40
RPSE	1	0.99
MPSO	6	2.41
MRPSO	30	5.04

(b)

$M_i = 0.2$

Method	no. succ.	std. dev.
PSO	2	1.40
PSE	0	0
RPSE	1	0.99
MPSO	2	1.40
MRPSO	16	3.83

(c)

$M_i = 0.3$

Method	no. succ.	std. dev.
PSO	6	2.41
PSE	8	2.77
RPSE	0	0
MPSO	8	2.77
MRPSO	72	6.78

(d)

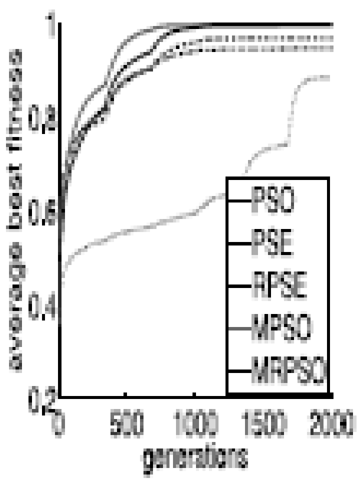
$M_i = 0.4$

$$i = \{1, 2, \ldots, n\}$$

Case studies and Results

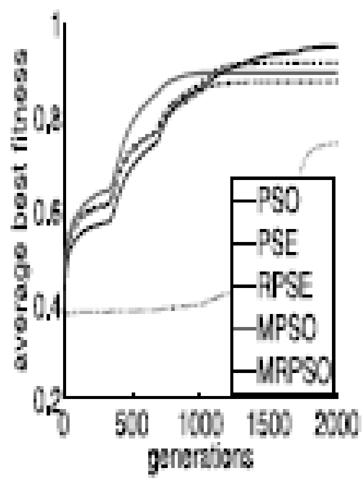
- Results for a 20-dimension cosff(x):

$$n = 20 \quad K = 10$$



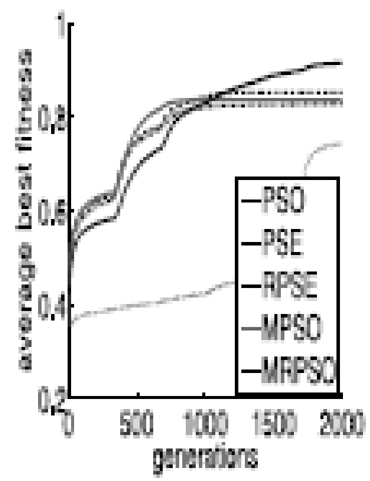
(i)

$M_i = 0.1$



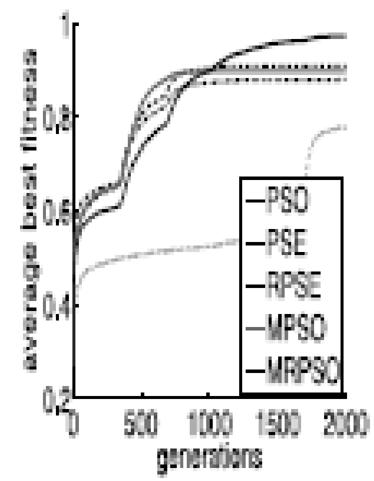
(l)

$M_i = 0.2$



(m)

$M_i = 0.3$



(n)

$M_i = 0.4$

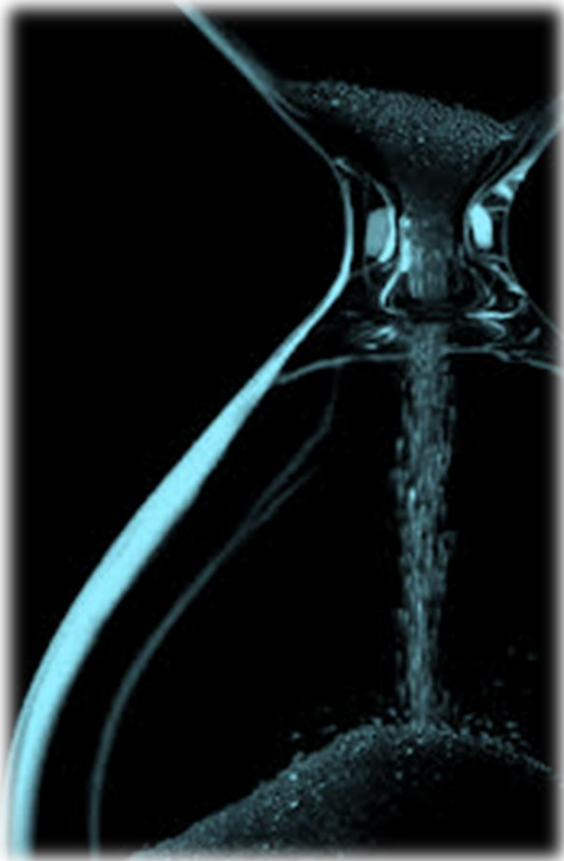
$$i = \{1, 2, \ldots, n\}$$

Case studies and Results

- Results for a 20-dimension cosff(x):
  - CPU times in milliseconds

Method	avg.time	std. dev.
PSO	1'899	30
PSE	2'075	29
RPSE	10'724	506
MPSO	1'838	9
MRPSO	2'503	21

(Cosff)



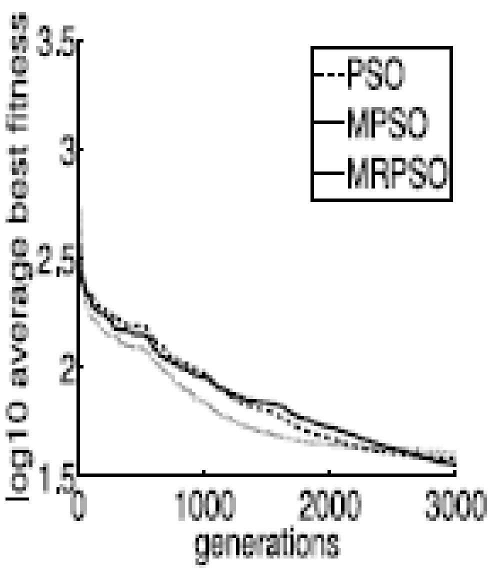
Case studies and Results

- Prediction of Pharmacokinetic Parameters :
  - %F: the percentage of the initial orally submitted drug dose that effectively reaches the systemic blood circulation after the passage from the liver
  - Prediction of %F for different molecular structures identifying the drugs
  - 70% of the molecules as the training set and 30% as the test set
  - Use PSO to obtain the coefficients with a linear regression analysis
  - Fitness = root mean squared error (RMSE) between outputs and targets

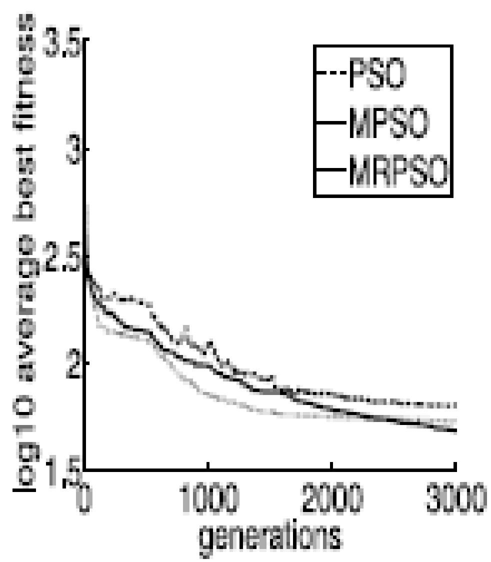
Case studies and Results

- Prediction of Pharmacokinetic Parameters :

Results on the training set:



Results on the test set:

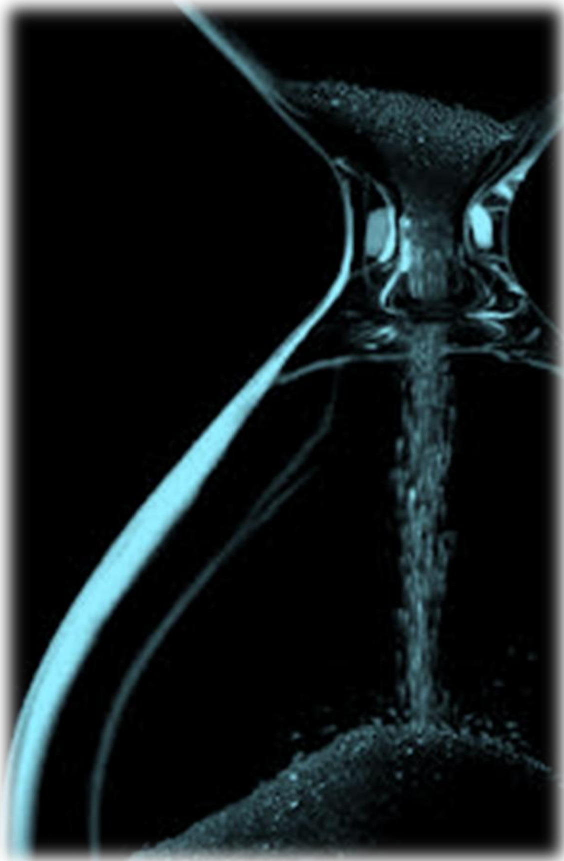


Case studies and Results

- Prediction of Pharmacokinetic Parameters :
  - CPU times in milliseconds

Method	avg.time	std. dev.
PSO	2'128'144	196'887
MPSO	2'158'310	90'705
MRPSO	2'265'062	42'753

(%F)



Conclusion

- Comparison of four parallel and distributed particle swarm optimization methods
- variants of multi-swarm and attractive/repulsive PSO.
- MRPSO outperforms the other considered PSO methods.
  - Probably because it maintains a higher diversity degree in the whole system
- Poor performances of PSE and RPSE
  - Probably because individuals of the GA are swarms and the complicated structure limits the exploration ability

Praises

- Distributed PSO methods applicable to MAS
- Several case studies and a number of sensitivity analysis
- Simplicity of the methods
- Applicable to a variety of problems

Critiques

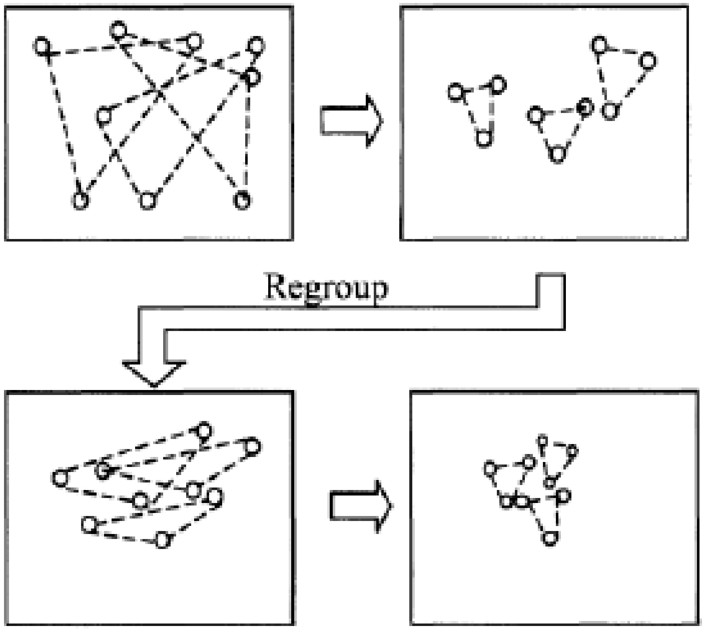
- Not enough reasoning over the parameters selected for the proposed methods.
- The PSO methods may not be comparable as changing the parameters and environment can alter the performance of the methods.
- Static environment
- No scalability evaluation
- Weak justification and implication of the results based on the characteristics of the methods

PSO Methods for Dynamic Environments

- Multi-Swarm Accelerating PSO (MSA-PSO)
  - Small neighborhood
  - Small-sized swarms
  - Randomized regrouping every R iterations
  - Accelerating operation

Exploration

Exploitation



PSO Methods for Dynamic Environments

- Distributed Adaptive PSO (DAPSO)
  - Particle's memory of fitness value will gradually evaporate at a constant rate $0 < T < 1$.
  - Same evaporation constant for all particles.
  - Particles' updating frequency may be different.
  - Similar to the human's knowledge/experience learning and updating

$$F(x_{i,pbest}(k+1)) = \begin{cases} T \cdot F(x_{i,pbest}(k)) & if \quad F(x_i(k+1)) < T \cdot F(x_{i,pbest}(k)) \\ F(x_i(k+1)) & if \quad F(x_i(k+1)) > T \cdot F(x_{i,pbest}(k)) \end{cases}$$

Proposed PSO Application in a Smart Grid

- Negotiation between the self-interested customer agents
  - To join coalitions for buying/selling electricity.
  - To invest in community-based distributed generation/storage systems. (team formation)