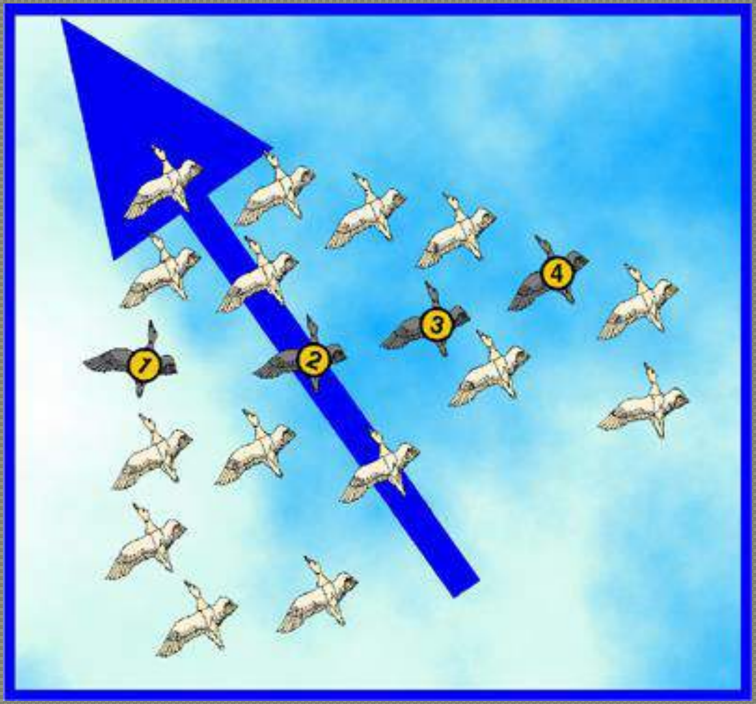
The Particle Swarm Optimization Algorithm



Decision Support
2010-2011

Andry Pinto
Hugo Alves
Inês Domingues
Luís Rocha
Susana Cruz

Summary

- Introduction to Particle Swarm Optimization (PSO)
  - Origins
  - Concept
  - PSO Algorithm

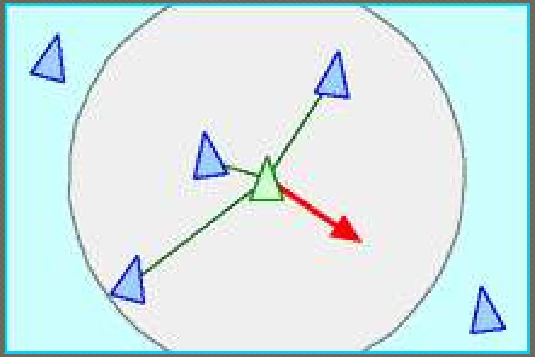
- PSO for the Bin Packing Problem (BPP)
  - Problem Formulation
  - Algorithm
  - Simulation Results

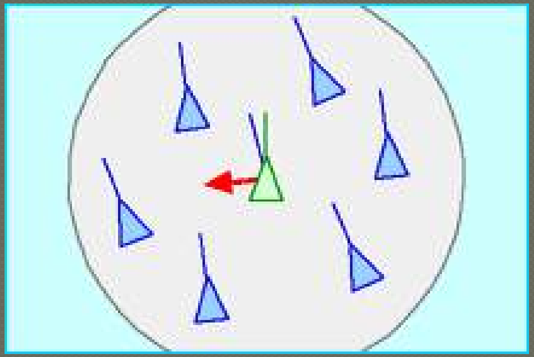
Introduction to the PSO: Origins

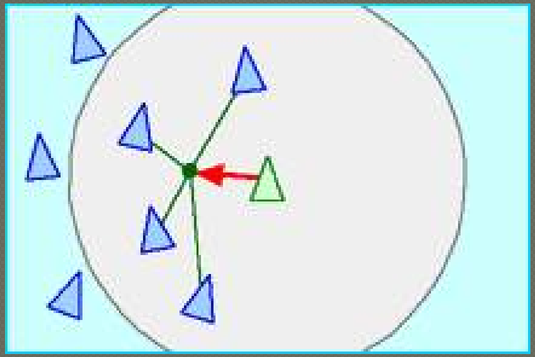
- Inspired from the nature social behavior and dynamic movements with communications of insects, birds and fish

Introduction to the PSO: **Origins**

- In 1986, Craig Reynolds described this process in 3 simple behaviors:







Separation

avoid crowding local flockmates

Alignment

move towards the average heading of local flockmates

Cohesion

move toward the average position of local flockmates

Introduction to the PSO: Origins

- Application to optimization: Particle Swarm Optimization
- Proposed by James Kennedy & Russell Eberhart (1995)
- Combines self-experiences with social experiences

Introduction to the PSO: <u>**Concept**</u>

- Uses a number of agents (**particles**) that constitute a swarm moving around in the search space looking for the best solution

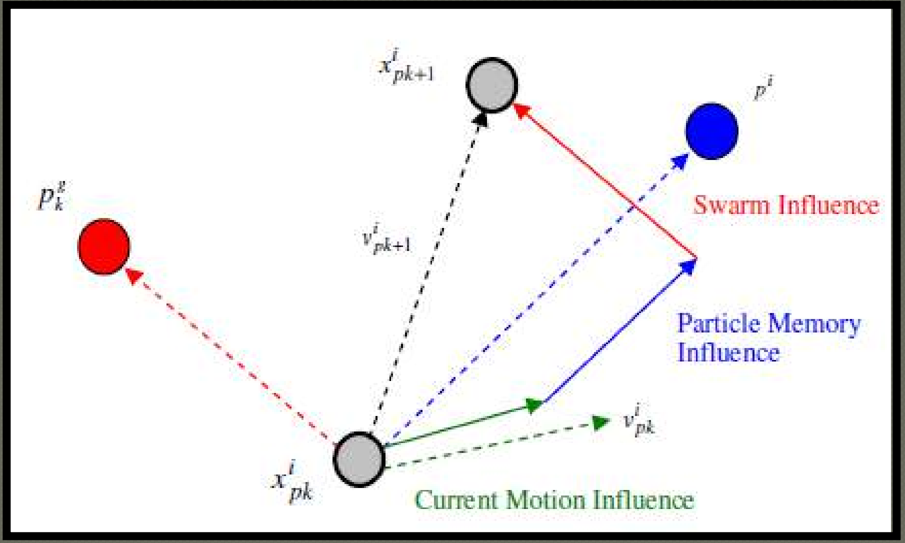
- Each particle in search space adjusts its "flying" according to its own flying experience as well as the flying experience of other particles

Introduction to the PSO: **Concept**

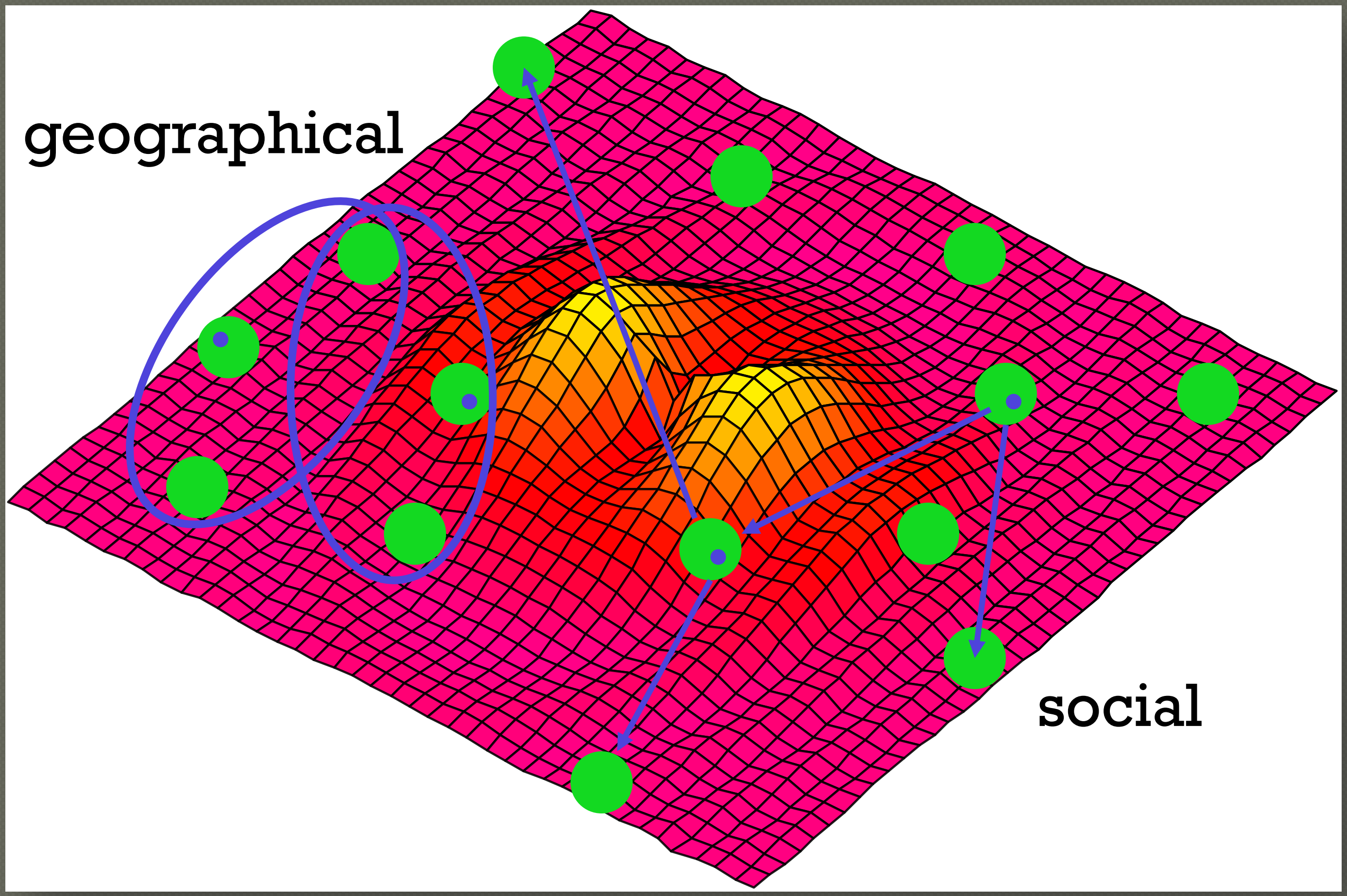
- Collection of flying particles (swarm) - Changing solutions
- Search area - Possible solutions
- Movement towards a promising area to get the global optimum
- Each particle keeps track:
  - its best solution, personal best, *pbest*
  - the best value of any particle, global best, *gbest*

Introduction to the PSO: <u>**Concept**</u>

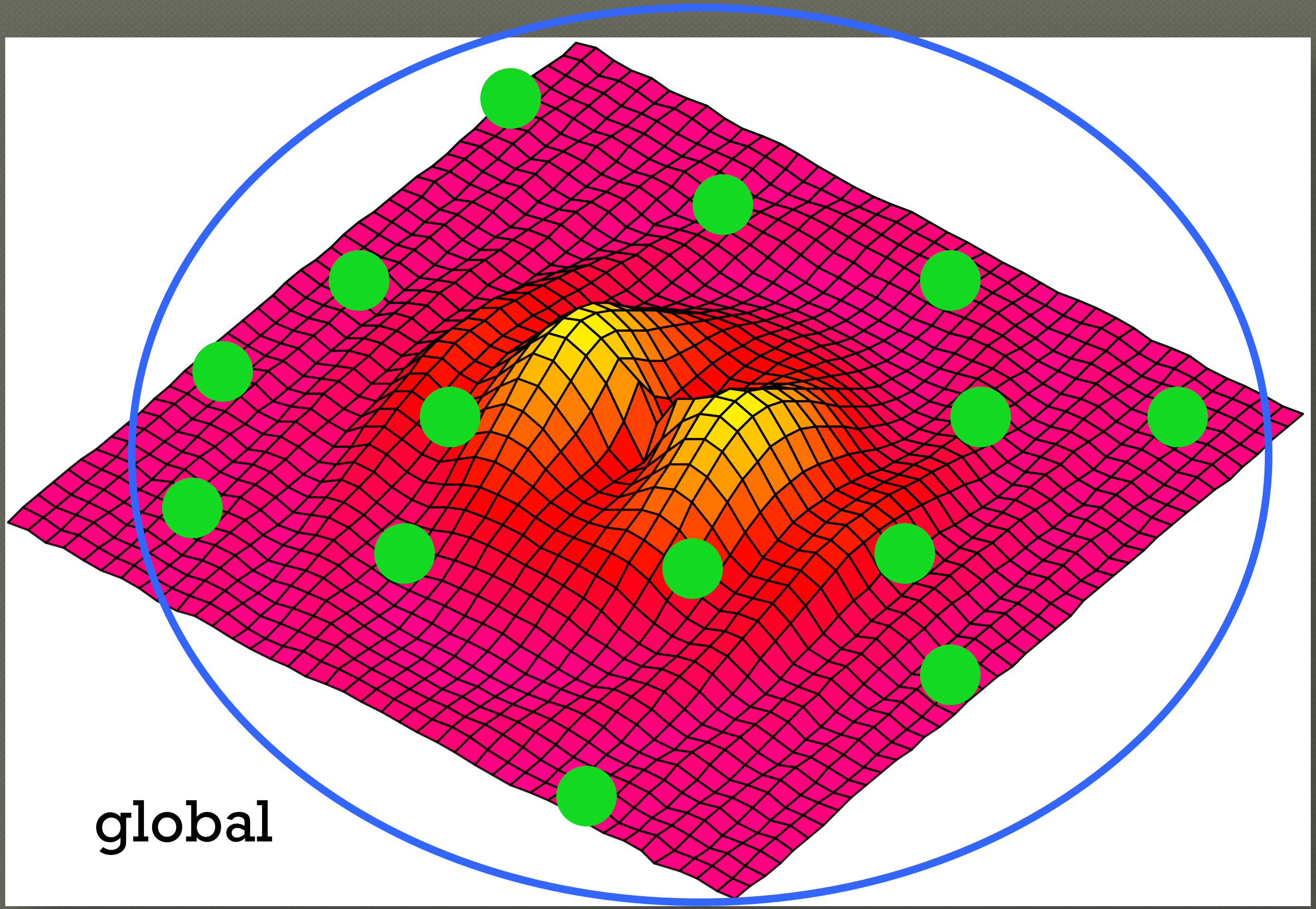
- Each particle adjusts its travelling speed dynamically corresponding to the flying experiences of itself and its colleagues
- Each particle modifies its position according to:
  - its current position
  - its current velocity
  - the distance between its current position and <u>*pbest*</u>
  - the distance between its current position and <u>*gbest*</u>



Introduction to the PSO: <u>Algorithm - Neighborhood</u>



Introduction to the PSO:
Algorithm - Neighborhood



Introduction to the PSO: Algorithm - Parameterss

- Algorithm parameters
  - ***A*** : Population of agents
  - $\boldsymbol{p_i}$: Position of agent $\boldsymbol{a_i}$ in the solution space
  - ***f*** : Objective function
  - $\boldsymbol{v_i}$: Velocity of agent's $\boldsymbol{a_i}$
  - $\boldsymbol{V(a_i)}$: Neighborhood of agent a_i (fixed)
- The neighborhood concept in PSO is not the same as the one used in other meta-heuristics search, since in PSO each particle's neighborhood never changes (is fixed)

Introduction to the PSO: **Algorithm**

```
[x*] = PSO()
P = Particle_Initialization();
For i=1 to it_max
   For each particle p in P do
      fp = f(p);
      If fp is better than f(pBest)
            pBest = p;
      end
   end
   gBest = best p in P;
   For each particle p in P do
      v = v + c1*rand*(pBest – p) + c2*rand*(gBest – p);
      p = p + v;
   end
end
```

Introduction to the PSO: Algorithm

- Particle update rule

$$p = p + v$$

  - with

$$v = v + c_1 * rand * (pBest - p) + c_2 * rand * (gBest - p)$$

  - where

- p: particle's position
- v: path direction
- c_1: weight of local information
- c_2: weight of global information
- *pBest*: best position of the particle
- *gBest*: best position of the swarm
- *rand*: random variable

Introduction to the PSO: **Algorithm - Parameters**

- Number of particles usually between 10 and 50
- C_1 is the importance of personal best value
- C_2 is the importance of neighborhood best value
- Usually $C_1 + C_2 = 4$ (empirically chosen value)
- If velocity is too low → algorithm too slow
- If velocity is too high → algorithm too unstable

Introduction to the PSO: **Algorithm**

1. Create a ‘population’ of agents (particles) uniformly distributed over 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)

Introduction to the PSO: **Algorithm**

5. Update particles' velocities:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{inertia} + \underbrace{c_1 \mathbf{U}_1^t (\mathbf{pb}_i^t - \mathbf{p}_i^t)}_{personal\ influence} + \underbrace{c_2 \mathbf{U}_2^t (\mathbf{gb}^t - \mathbf{p}_i^t)}_{social\ influence}$$

6. Move particles to their new positions:

$$\boxed{\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \mathbf{v}_i^{t+1}}$$

7. Go to step 2 until stopping criteria are satisfied

Introduction to the PSO: **Algorithm**

Particle's velocity:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^{t}}_{\textit{inertia}} + \underbrace{c_1\mathbf{U}_1^{t}(\mathbf{pb}_i^{t} - \mathbf{p}_i^{t})}_{\textit{personal influence}} + \underbrace{c_2\mathbf{U}_2^{t}(\mathbf{gb}^{t} - \mathbf{p}_i^{t})}_{\textit{social influence}}$$

1. Inertia

- Makes the particle move in the same direction and with the same velocity

2. Personal Influence

- Improves the individual
- Makes the particle return to a previous position, better than the current
- Conservative

3. Social Influence

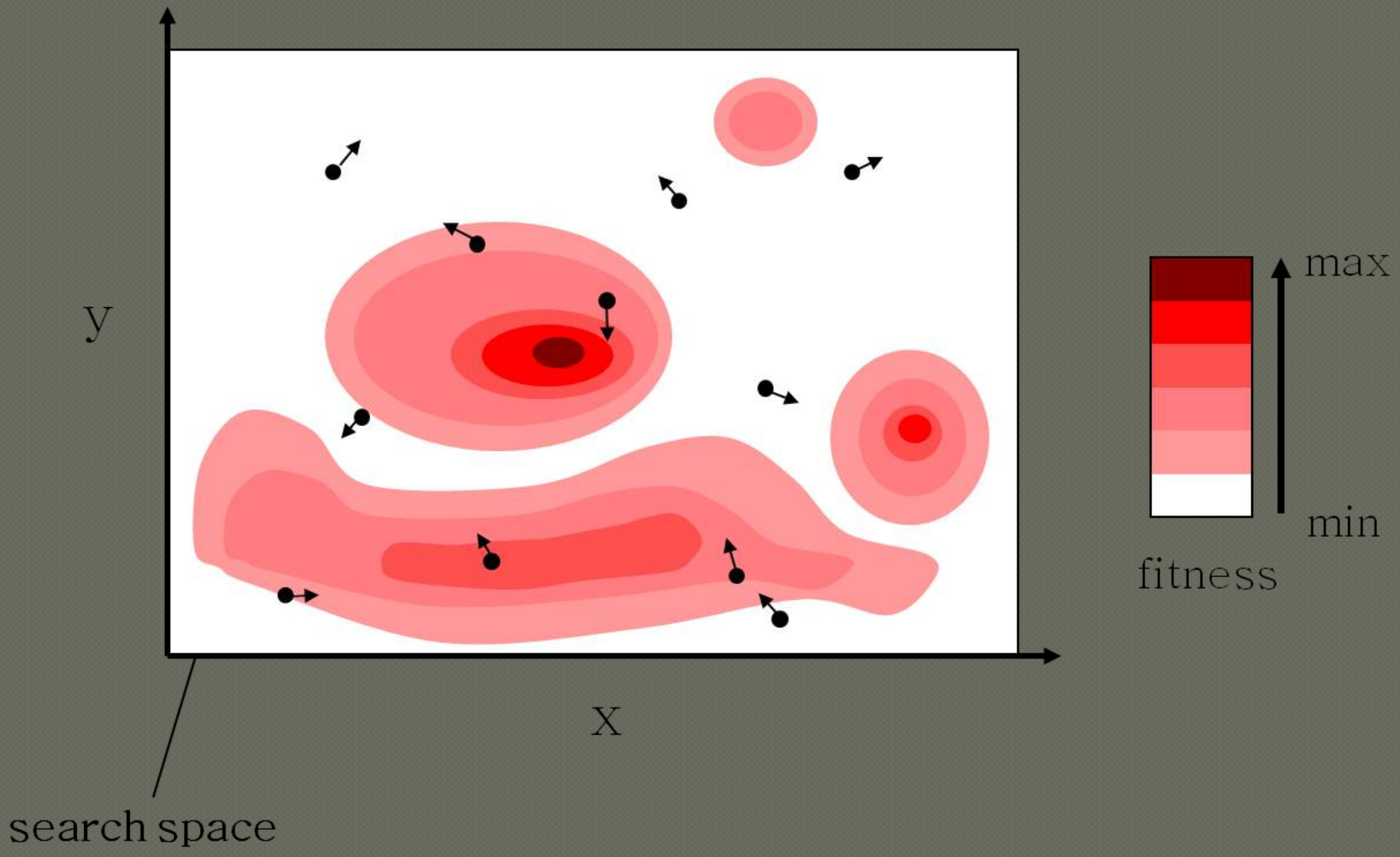
- Makes the particle follow the best neighbors direction

Introduction to the PSO: <u>Algorithm</u>

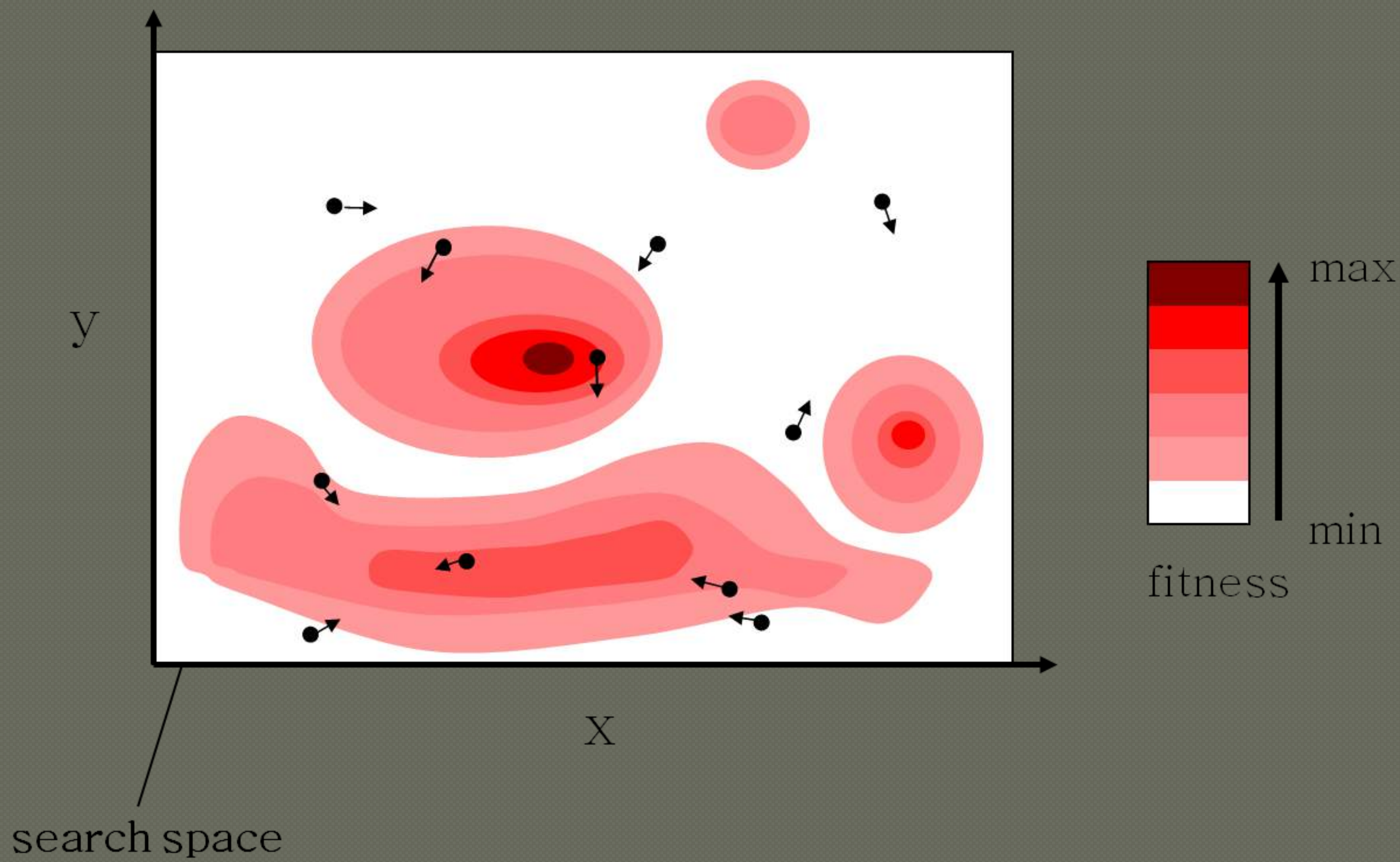
- <u>Intensification</u>: explores the previous solutions, finds the best solution of a given region
- <u>Diversification</u>: searches new solutions, finds the regions with potentially the best solutions
- In PSO:

$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\textbf{Diversification}} + \underbrace{c_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{p}_i^t) + c_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{p}_i^t)}_{\textbf{Intensification}}$$

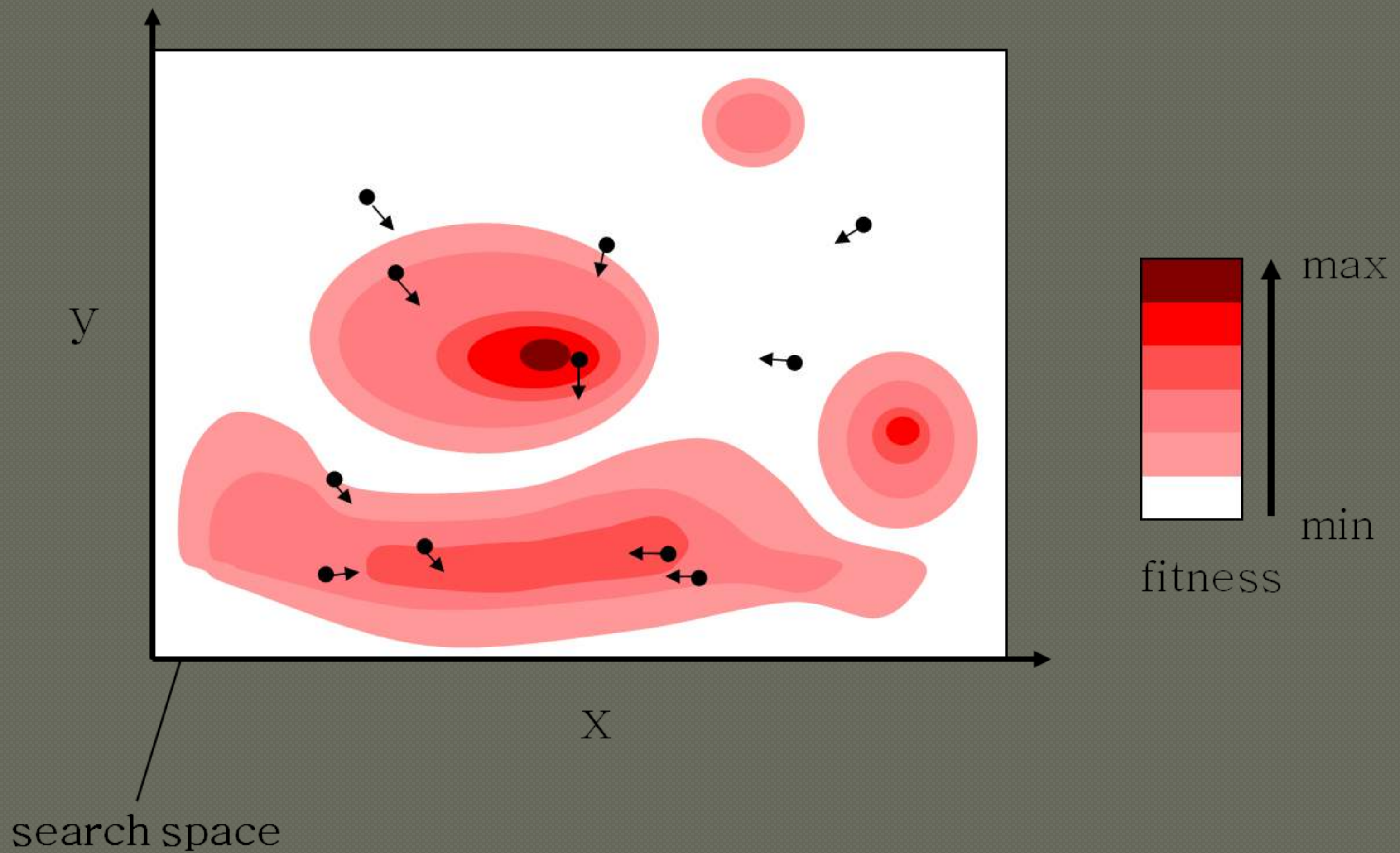
Introduction to the PSO: **Algorithm - Example**



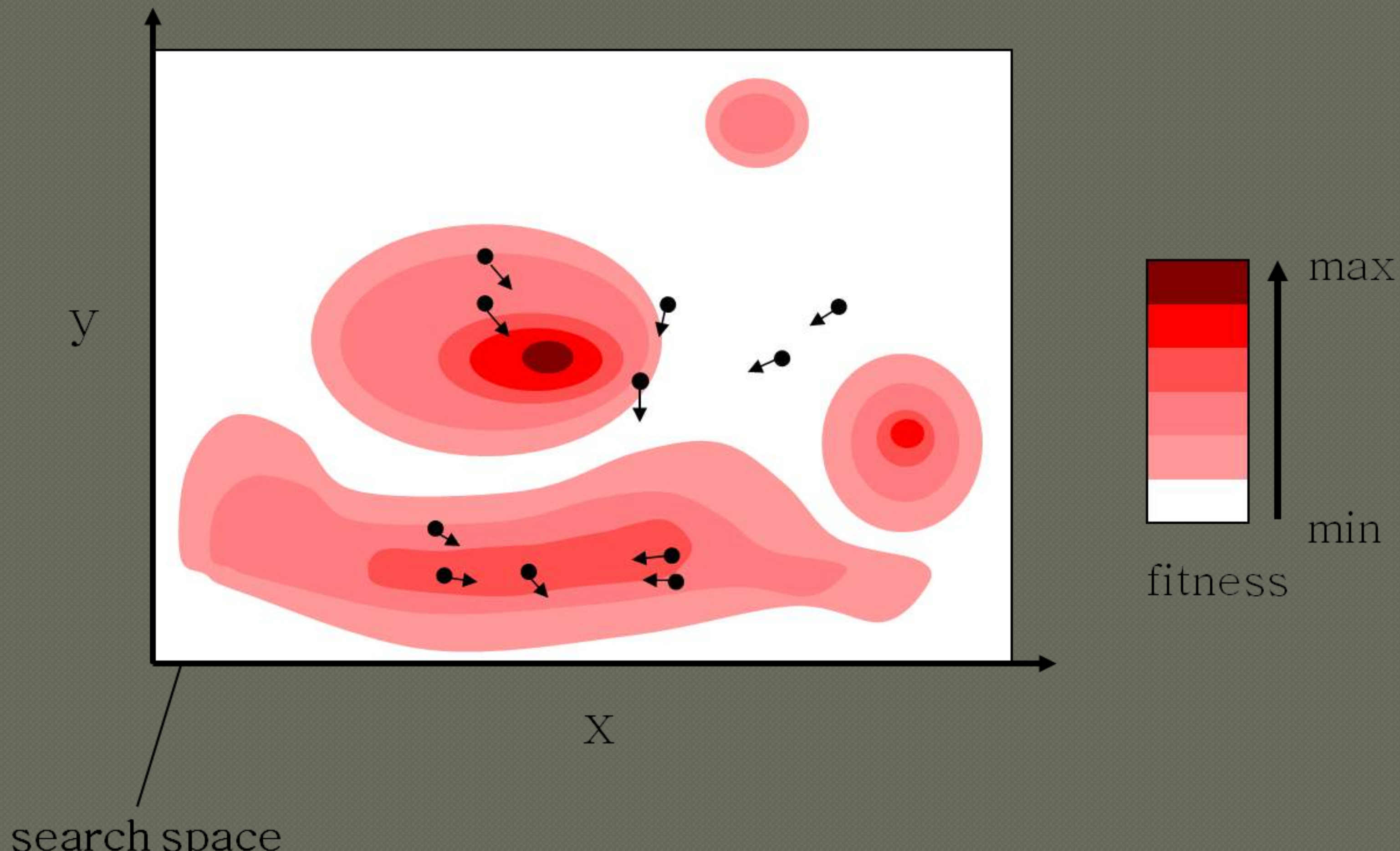
Introduction to the PSO:
Algorithm - Example



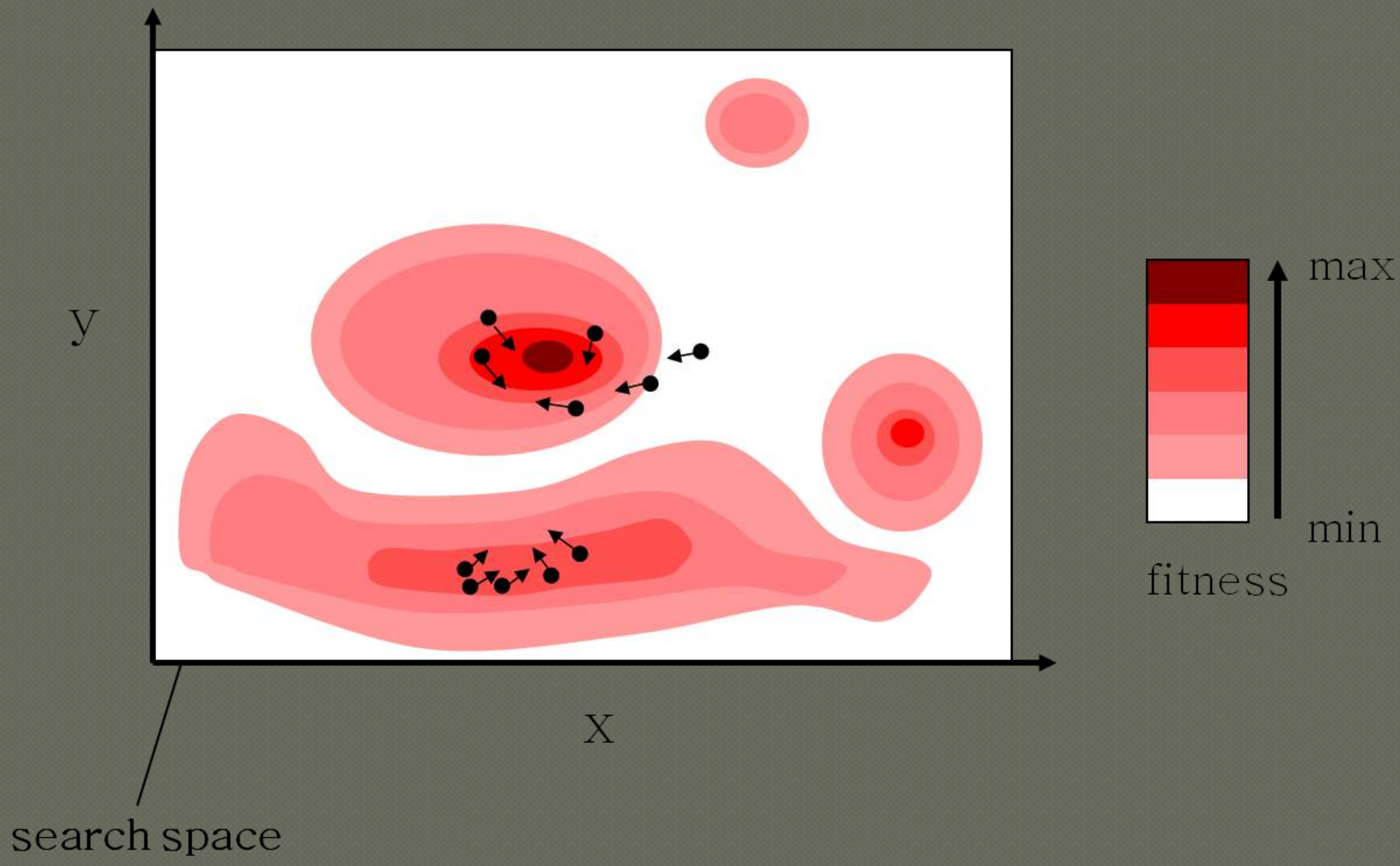
Introduction to the PSO: **Algorithm - Example**



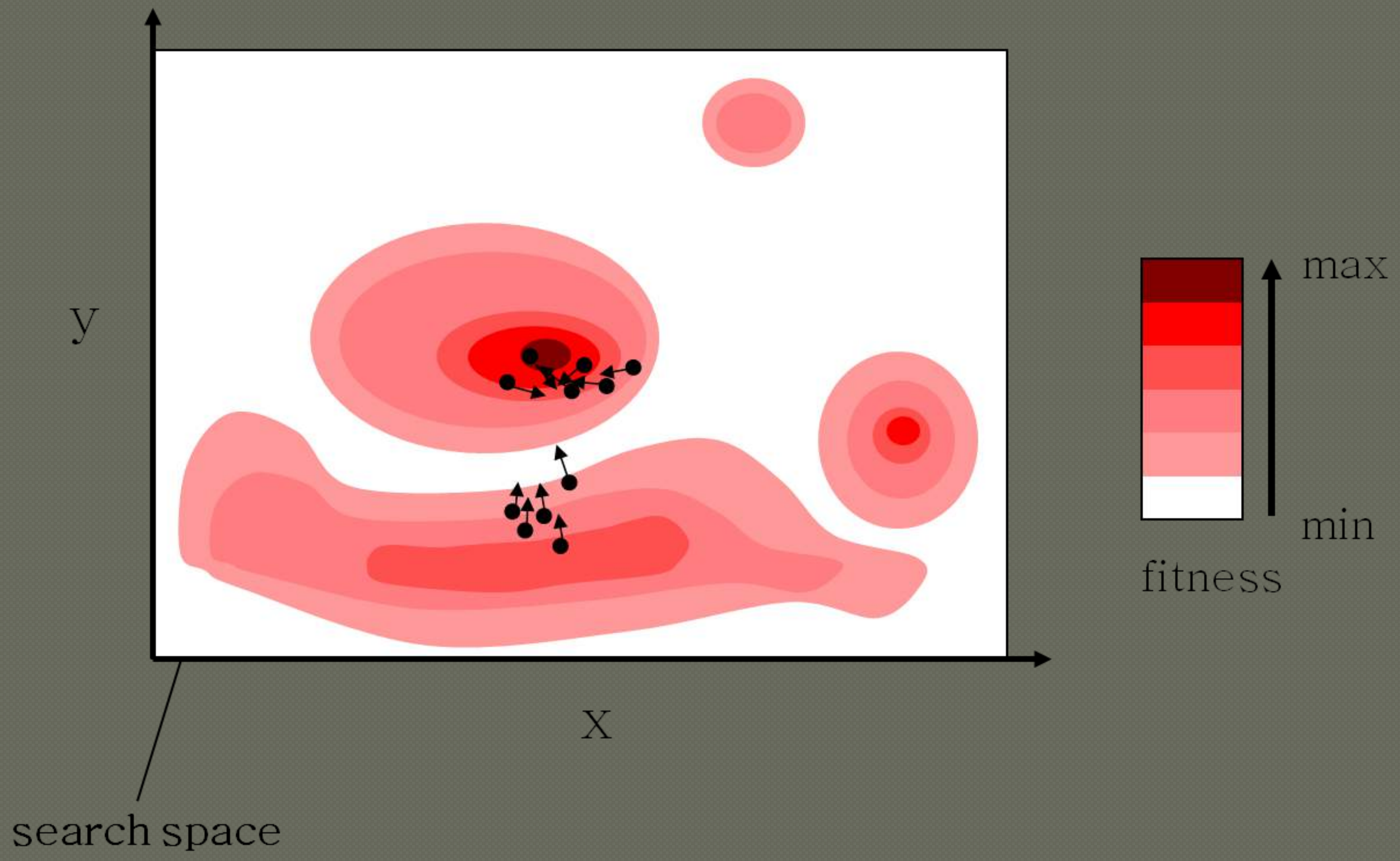
Introduction to the PSO: **Algorithm - Example**



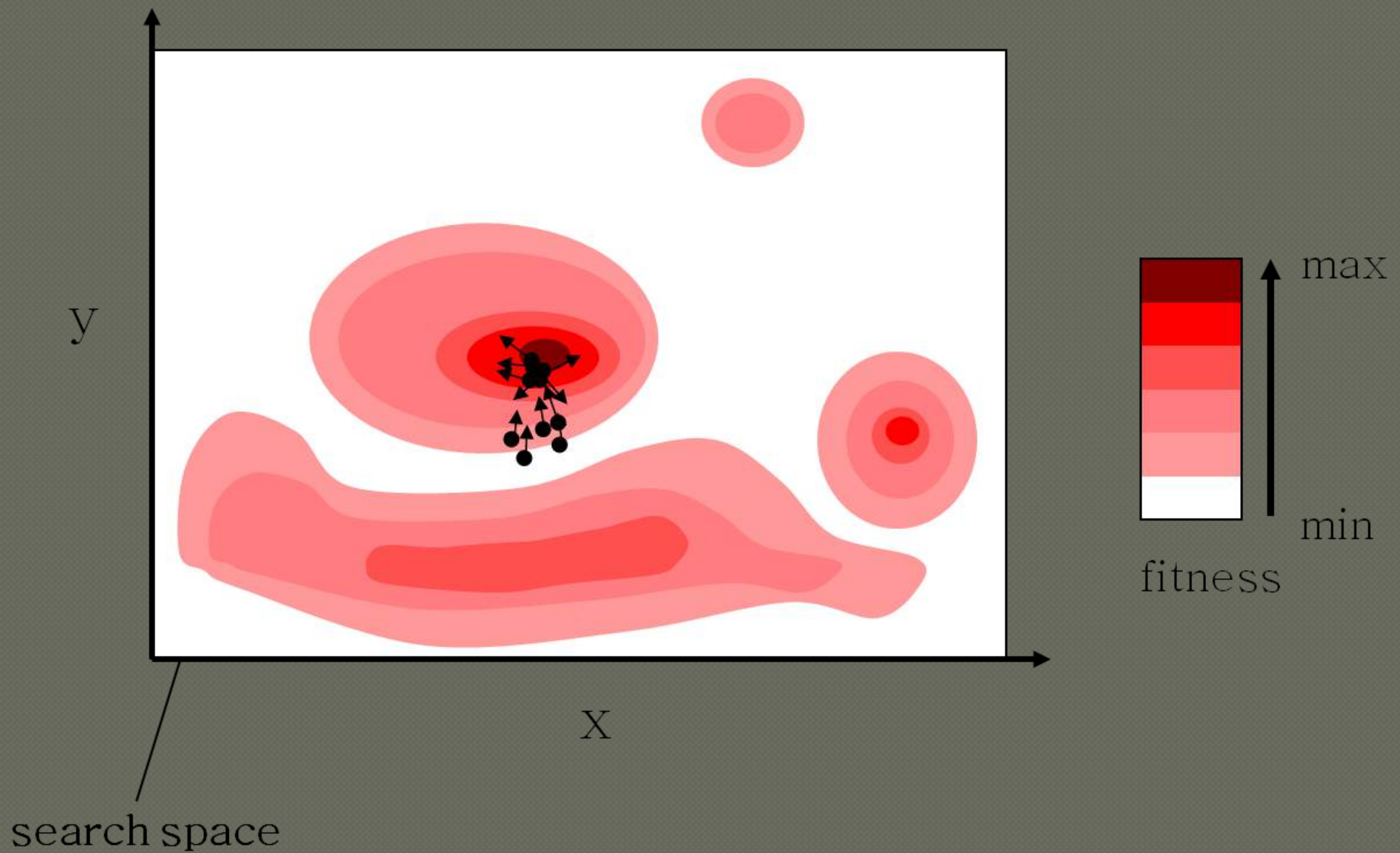
Introduction to the PSO: **Algorithm - Example**



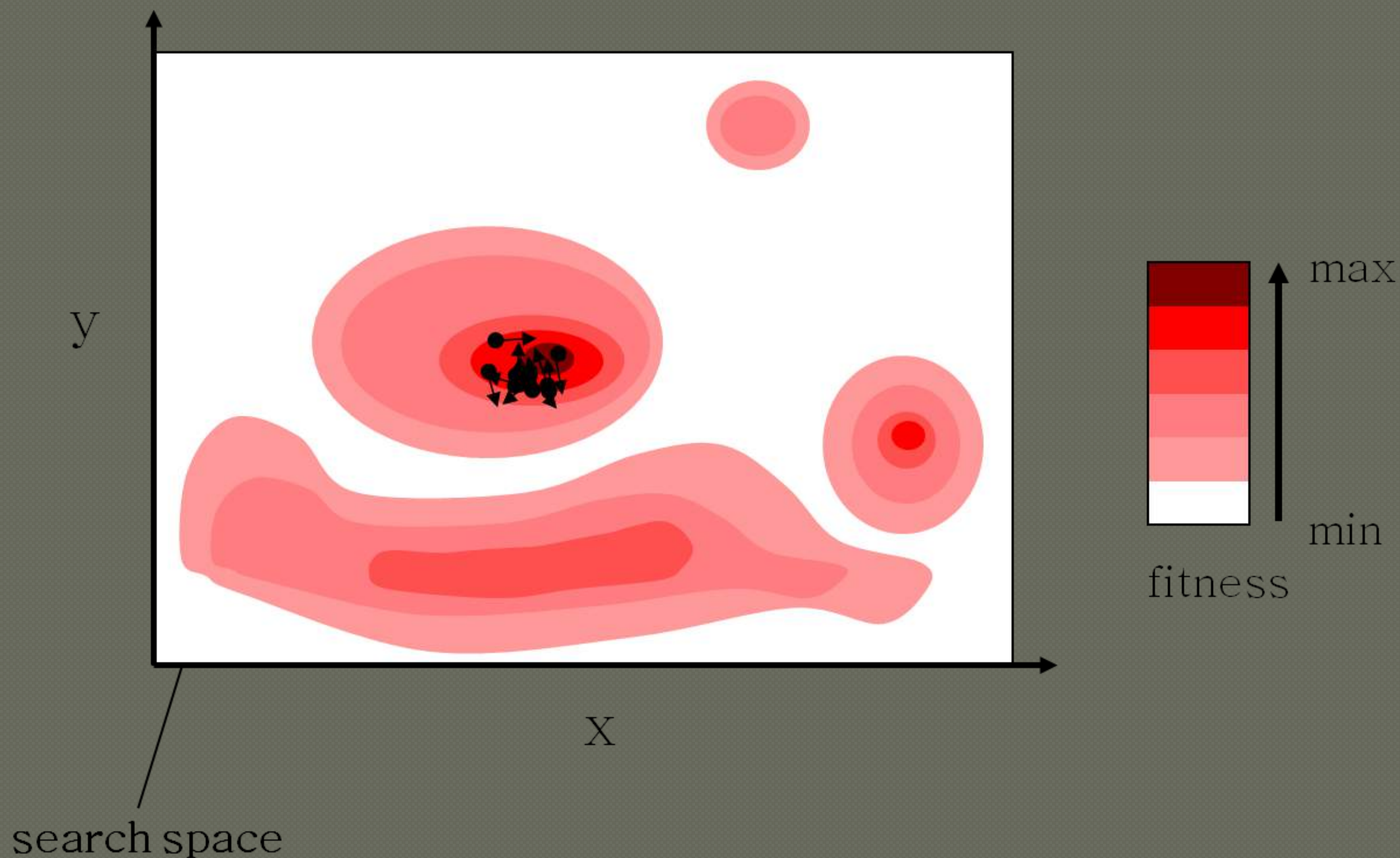
Introduction to the PSO: <u>Algorithm - Example</u>



Introduction to the PSO: **Algorithm - Example**



Introduction to the PSO: **Algorithm - Example**



Introduction to the PSO: Algorithm Characteristics

- **Advantages**
  - Insensitive to scaling of design variables
  - Simple implementation
  - Easily parallelized for concurrent processing
  - Derivative free
  - Very few algorithm parameters
  - Very efficient global search algorithm
- **Disadvantages**
  - Tendency to a fast and premature convergence in mid optimum points
  - Slow convergence in refined search stage (weak local search ability)

Introduction to the PSO: Different Approaches

- **Several approaches**
  - *2-D Otsu PSO*
  - *Active Target PSO*
  - *Adaptive PSO*
  - *Adaptive Mutation PSO*
  - *Adaptive PSO Guided by Acceleration Information*
  - *Attractive Repulsive Particle Swarm Optimization*
  - *Binary PSO*
  - *Cooperative Multiple PSO*
  - *Dynamic and Adjustable PSO*
  - *Extended Particle Swarms*
  - *…*

Davoud Sedighizadeh and Ellips Masehian, "Particle Swarm Optimization Methods, Taxonomy and Applications". International Journal of Computer Theory and Engineering, Vol. 1, No. 5, December 2009

PSO for the BPP: **Introduction**

On solving Multiobjective Bin Packing Problem Using Particle Swarm Optimization

D.S Liu, K.C. Tan, C.K. Goh and W.K. Ho
2006 - IEEE Congress on Evolutionary Computation

- First implementation of PSO for BPP

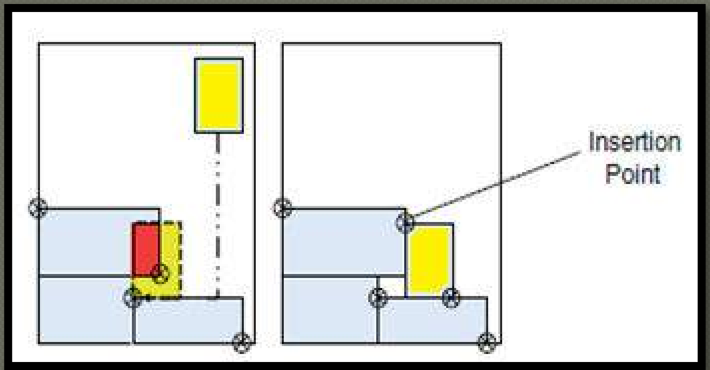
PSO for the BPP: **Problem Formulation**

- Multi-Objective 2D BPP
- Maximum of I bins with width W and height H
- J items with $w_j \leq W, h_j \leq H$ and weight ψ_j
- Objectives
  - Minimize the number of bins used K
  - Minimize the average deviation between the overall centre of gravity and the desired one

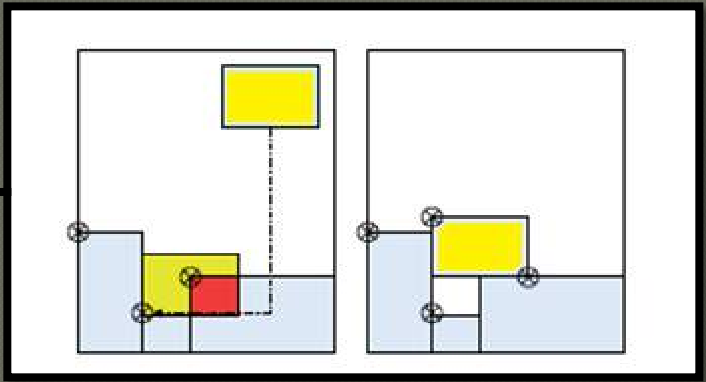
PSO for the BPP: <u>Initialization</u>

- Usually generated randomly
- In this work:
  - Solution from Bottom Left Fill (BLF) heuristic
  - To sort the rectangles for BLF:
    - Random
    - According to a criteria (width, weight, area, perimeter..)

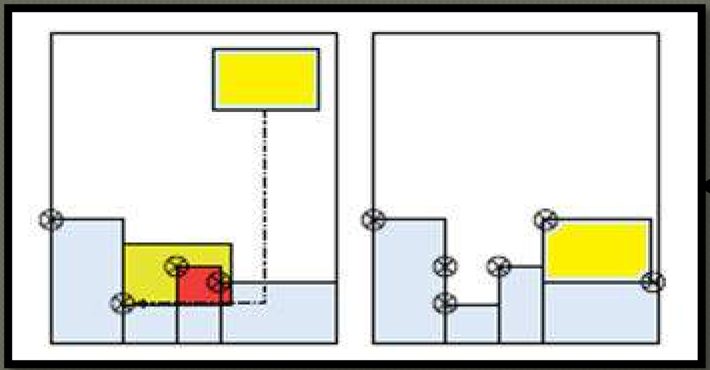
PSO for the BPP: **Initialization BLF**



Item moved to the right if intersection detected at the top



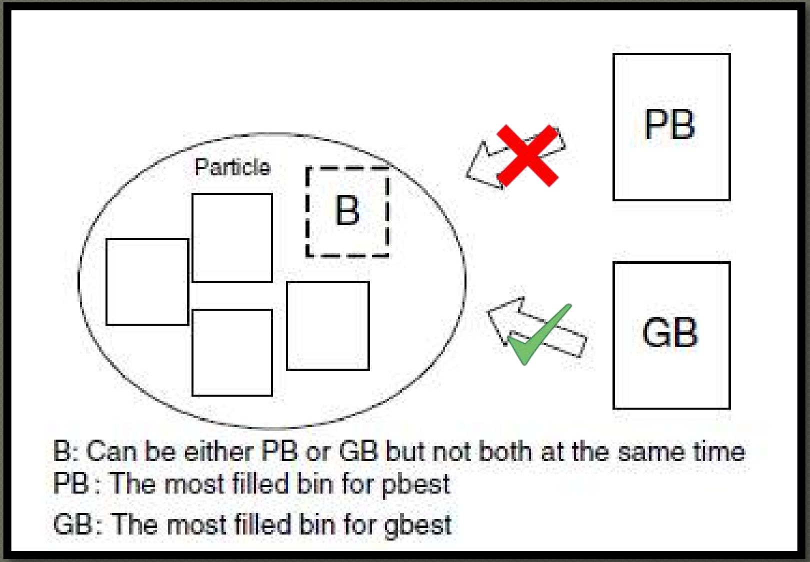
Item moved to the top if intersection detected at the right

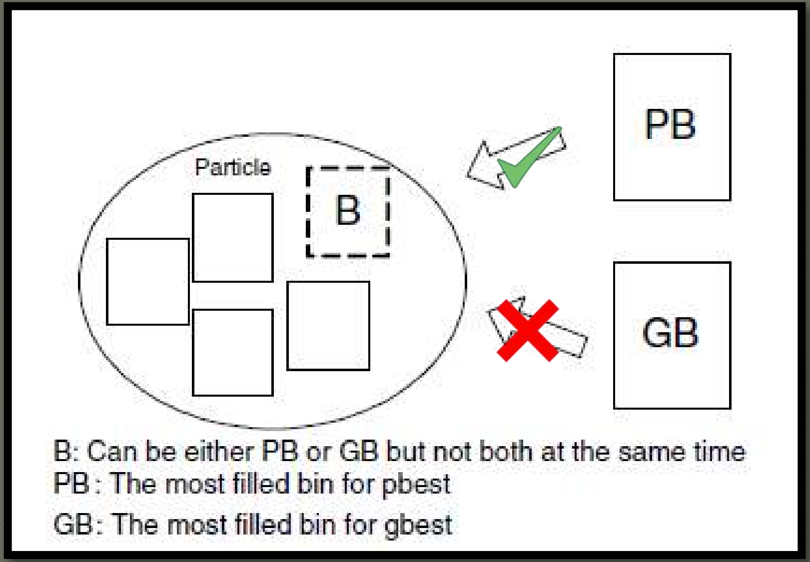


Item moved if there is a lower available space for insertion

PSO for the BPP: <u>**Algorithm**</u>

- Velocity depends on either *pbest* **or** *gbest*: never both at the same time





OR

PSO for the BPP: **Algorithm**

1st Stage:

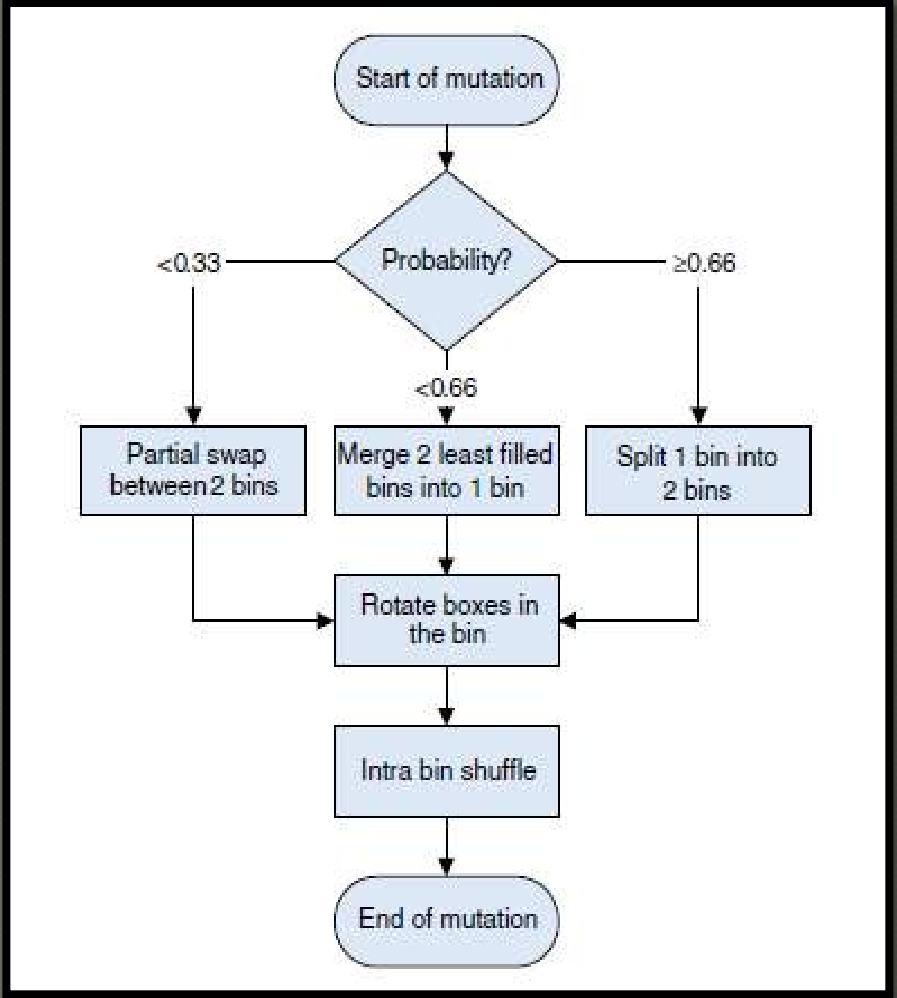
- Partial Swap between 2 bins
- Merge 2 bins
- Split 1 bin

2nd Stage:

- Random rotation

3rd Stage:

- Random shuffle



Mutation modes for a single particle

PSO for the BPP: Algorithm

H hybrid

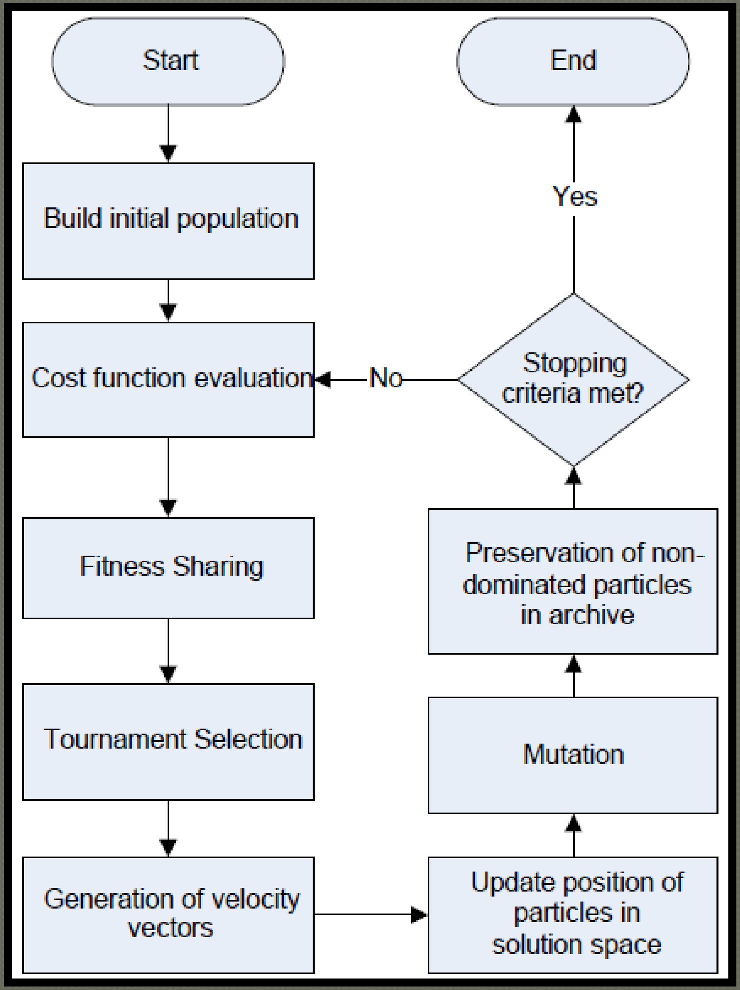
M multi

O objective

P particle

S swarm

O optimization



The flowchart of HMOPSO

PSO for the BPP: Problem Formulation

- 6 classes with 20 instances randomly generated
- Size range:
  - Class 1: [0, 100]
  - Class 2: [0, 25]
  - Class 3: [0, 50]
  - Class 4: [0, 75]
  - Class 5: [25, 75]
  - Class 6: [25, 50]
- Class 2: small items → more difficult to pack

PSO for the BPP: Simulation Results

- Comparison with 2 other methods
  - MOPSO (Multiobjective PSO) from [1]
  - MOEA (Multiobjective Evolutionary Algorithm) from [2]
- Definition of parameters:

Parameter	MOPSO	MOEA	HMOPSO
crossover rate	-	0.7	-
PSO update rate	-	-	0.7
mutation rate	-	0.4	0.4
population size	500	500	500
generation size	200 generations		
niche radius	0.1	0.1	0.1

[1] Wang, K. P., Huang, L., Zhou C. G. and Pang, W., "Particle Swarm Optimization for Traveling Salesman Problem," *International Conference on Machine Learning and Cybernetics, vol. 3, pp.* 1583-1585, 2003.

[2] Tan, K. C., Lee, T. H., Chew, Y. H., and Lee, L. H., "A hybrid multiobjective evolutionary algorithm for solving truck and trailer vehicle routing problems," *IEEE Congress on Evolutionary Computation, vol. 3, pp. 2134-2141, 2003.*

PSO for the BPP: **Simulation Results**

- Comparison on the performance of metaheuristic algorithms against the branch and bound method (BB) on single objective BPP
- Results for each algorithm in 10 runs
- Proposed method (HMOPSO) capable of evolving more optimal solution as compared to BB in 5 out of 6 classes of test instances

PSO for the BPP: Simulation Results

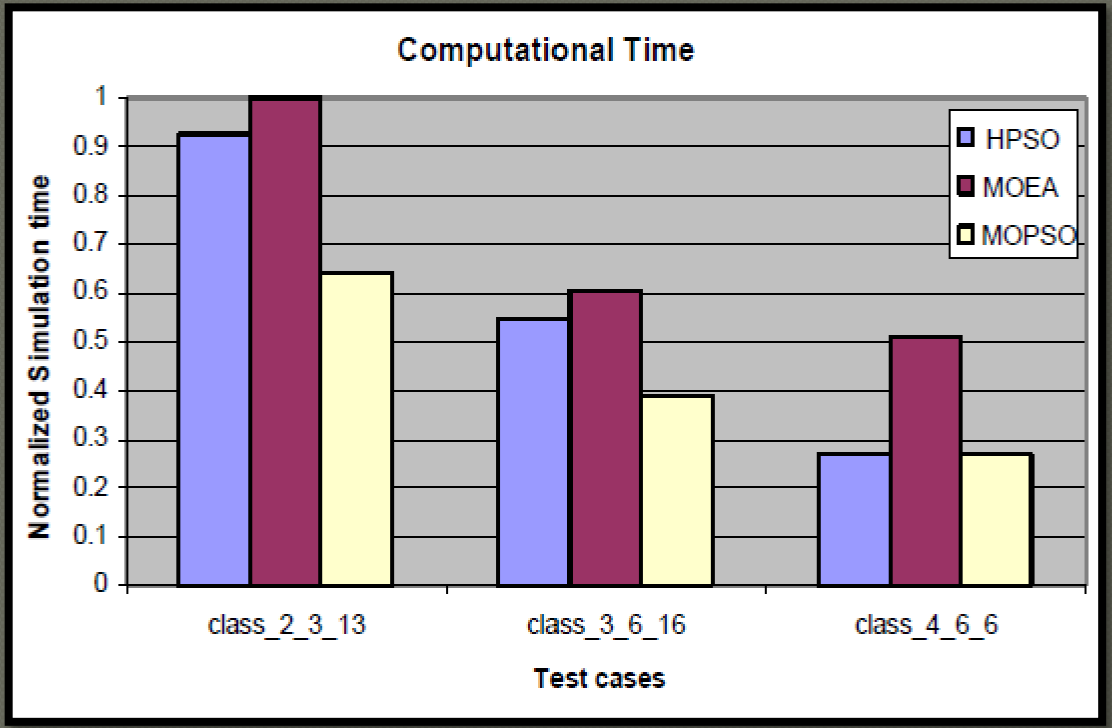
Class 1	BB	MOEA	MOPSO	HMOPSO
n=20	10	10	10	10
n=40	10	10	9	10
n=60	7	6	5	6
n=80	3	10	5	10
n=100	1	6	3	7
Total	31	**42**	**32**	**43**
Class 2				
n=20	10	10	10	10
n=40	10	9	9	9
n=60	4	10	9	10
n=80	10	9	8	9
n=100	10	10	9	10
Total	44	**48**	**45**	**48**
Class 3				
n=20	9	10	10	10
n=40	9	6	5	6
n=60	5	5	1	6
n=80	0	4	1	4
n=100	0	5	1	5
Total	23	**30**	**18**	**31**

Class 4	BB	MOEA	MOPSO	HMOPSO
n=20	10	10	10	10
n=40	10	10	10	10
n=60	7	8	8	8
n=80	10	7	7	7
n=100	10	8	7	8
Total	47	43	42	43
Class 5				
n=20	10	9	10	9
n=40	10	8	9	10
n=60	8	5	5	6
n=80	0	3	3	3
n=100	1	3	0	2
Total	29	28	27	**30**
Class 6				
n=20	10	10	10	10
n=40	5	6	6	9
n=60	10	9	9	6
n=80	10	10	10	10
n=100	2	8	7	8
Total	37	**43**	**42**	**43**

Number of optimal solution obtained

PSO for the BPP: <u>Simulation Results</u>

- Computational Efficiency
  - stop after 1000 iterations or no improvement in last 5 generations
  - MOPSO obtained inferior results compared to the other two



PSO for the BPP: Conclusions

- Presentation of a mathematical model for MOBPP-2D
- MOBPP-2D solved by the proposed HMOPSO
- BLF chosen as the decoding heuristic
- HMOPSO is a robust search optimization algorithm
  - Creation of variable length data structure
  - Specialized mutation operator
- HMOPSO performs consistently well with the best average performance on the performance metric
- Outperforms MOPSO and MOEA in most of the test cases used in this paper

The Particle Swarm Optimization Algorithm