Particle Swarm Optimization Algorithm

Kuliah Kecerdasan Komputasional

04-10-2020

Pendahuluan

- PSO didasarkan pada perilaku sekawanan Serangga, burung atau ikan.
- Algoritma PSO meniru perilaku sosial organisme, yang terdiri dari tindakan individu dan pengaruh dari individu-individu lain dalam suatu kelompok.
- ***Perilaku seekor hewan dalam kawanan (swarm) dipengaruhi perilaku individu dan juga kelompoknya.***
- Kata partikel menunjukkan, misalnya, seekor burung dalam kawanan burung.

- Setiap individu/partikel berperilaku dengan cara menggunakan kecerdasannya sendiri dan juga dipengaruhi perilaku kelompok kolektifnya.
- Jika satu partikel/seekor burung menemukan jalan yang tepat/pendek menuju ke sumber makanan, sisa kelompok yang lain juga akan dapat segera mengikuti jalan tersebut meskipun lokasi mereka jauh dalam kelompok tersebut

- Dalam PSO, kawanan diasumsikan mempunyai ukuran tertentu dengan setiap partikel posisi awalnya terletak di suatu lokasi yang acak dalam ruang multidimensi.
- Setiap partikel diasumsikan memiliki dua karakteristik: posisi dan kecepatan.
- Setiap partikel bergerak dalam ruang/space tertentu dan mengingat posisi terbaik yang pernah dilalui/ditemukan terhadap sumber makanan/nilai fungsi objektif.

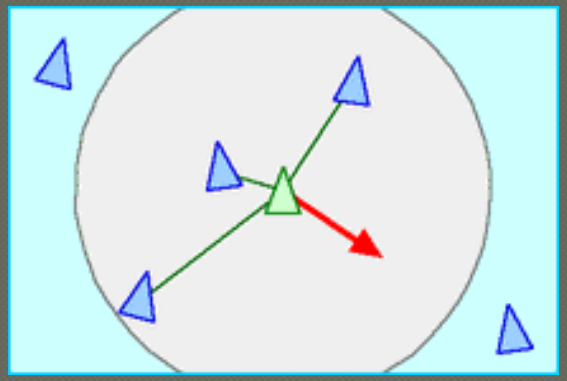
- Setiap partikel menyampaikan informasi atau posisi terbaiknya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi tersebut.
- Sebagai contoh, misalnya perilaku burung-burung dalam dalam kawanan burung.

- Setiap burung mempunyai kebiasaan (rule) seperti berikut :
  - Seekor burung tidak berada terlalu dekat dengan burung yang lain
  - Burung tersebut akan mengarahkan terbangnya ke arah rata-rata keseluruhan burung
  - Akan memposisikan diri dengan rata-rata posisi burung yang lain dengan menjaga sehingga jarak antar burung dalam kawanan itu tidak terlalu jauh

- Perilaku kawanan burung akan didasarkan pada kombinasi dari 3 faktor simpel berikut:

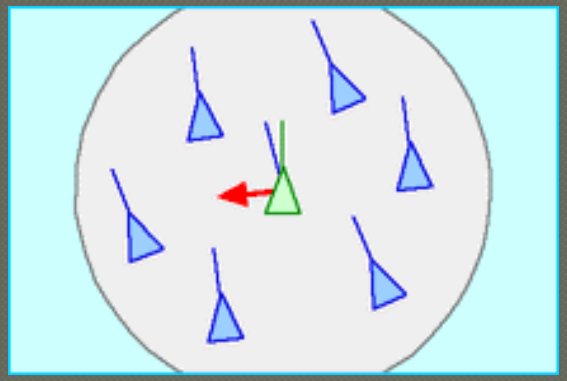
1. Kohesi - terbang bersama
2. Separasi - jangan terlalu dekat
3. Penyesuaian (alignment) - mengikuti arah bersama

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



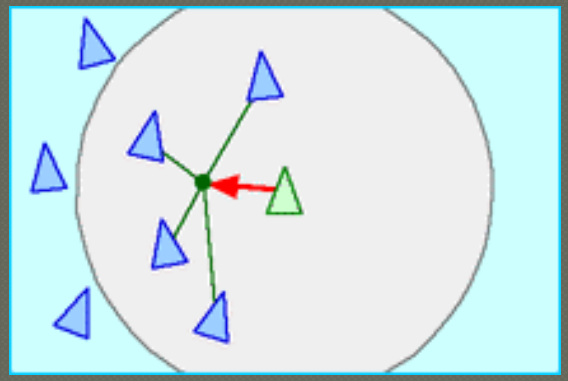
Separation

avoid crowding local flockmates



Alignment

move towards the average heading of local flockmates



Cohesion

move toward the average position of local flockmates

- PSO dikembangkan dengan berdasarkan pada model berikut:
  1. Ketika seekor burung mendekati target atau makanan (atau bisa minimum atau maximum suatu fungsi tujuan) secara cepat mengirim informasi kepada burung-burung yang lain dalam kawanan tertentu
  2. Burung yang lain akan mengikuti arah menuju ke makanan tetapi tidak secara langsung
  3. Ada komponen yang tergantung pada pikiran setiap burung, yaitu memorinya tentang apa yang sudah dilewati pada waktu sebelumnya.

Introduction to the PSO: **Concept**

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

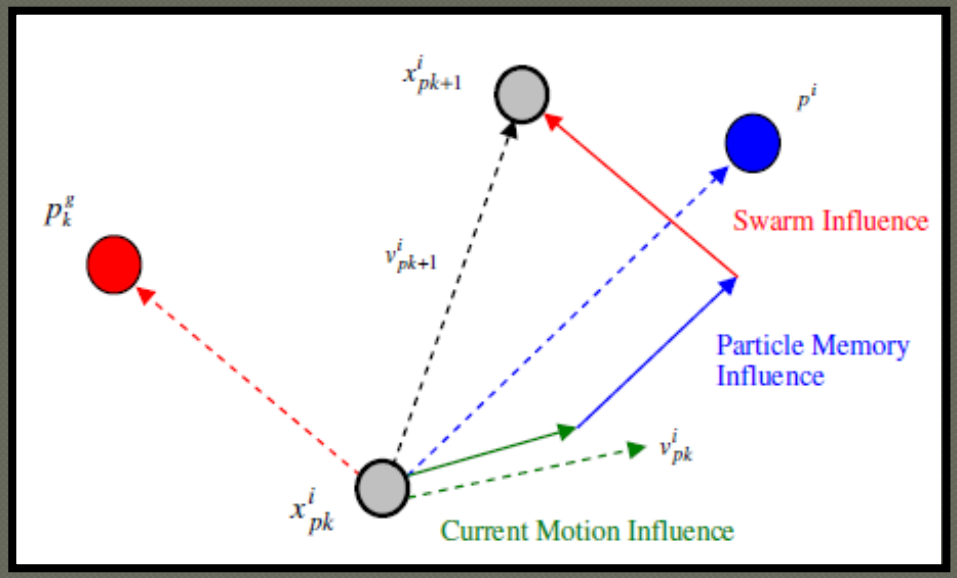
- Each particle in search space adjusts its "flying" according to its own flying experience as well as the flying experience of other particles

Introduction to the PSO: **Concept**

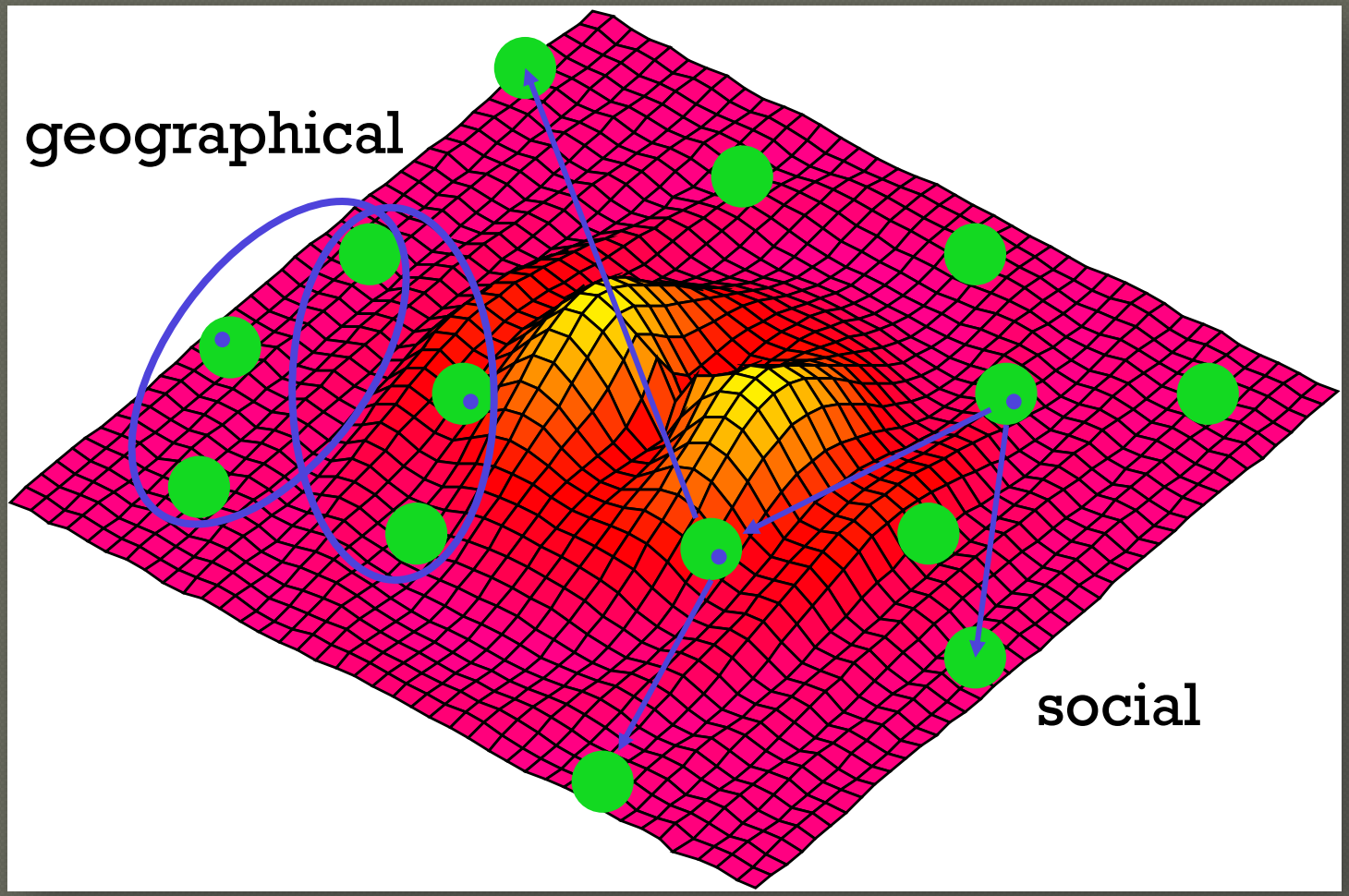
- Collection of flying particles (swarm) - Changing solutions
- Search area - Possible solutions
- Movement towards a promising area to get the global optimum
- Each particle keeps track:
  - its best solution, personal best, *pbest*
  - the best value of any particle, global best, *gbest*

Introduction to the PSO: **Concept**

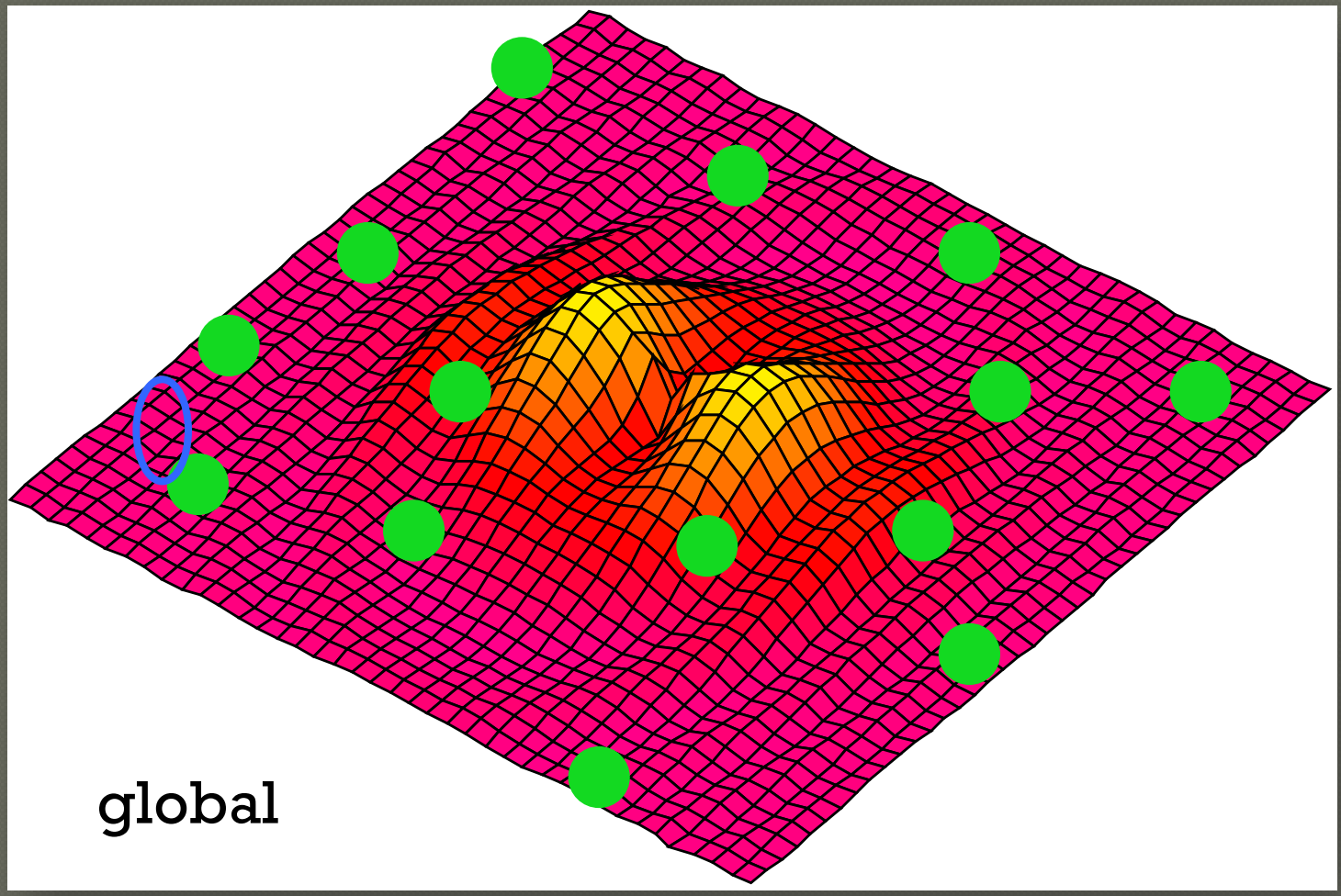
- Each particle adjusts its travelling speed dynamically corresponding to the flying experiences of itself and its colleagues
- Each particle modifies its position according to:
  - its current position
  - its current velocity
  - the distance between its current position and *pbest*
  - the distance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



Introduction to the PSO: Algorithm - Neighborhood



Introduction to the PSO: Algorithm - Neighborhood



Introduction to the PSO: Algorithm - Parameterss

- Algorithm parameters
  - A : Population of agents
  - p_i : Position of agent a_i in the solution space
  - f : Objective function
  - v_i : Velocity of agent's a_i
  - $V(a_i)$: 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:

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

6. Move particles to their new positions:

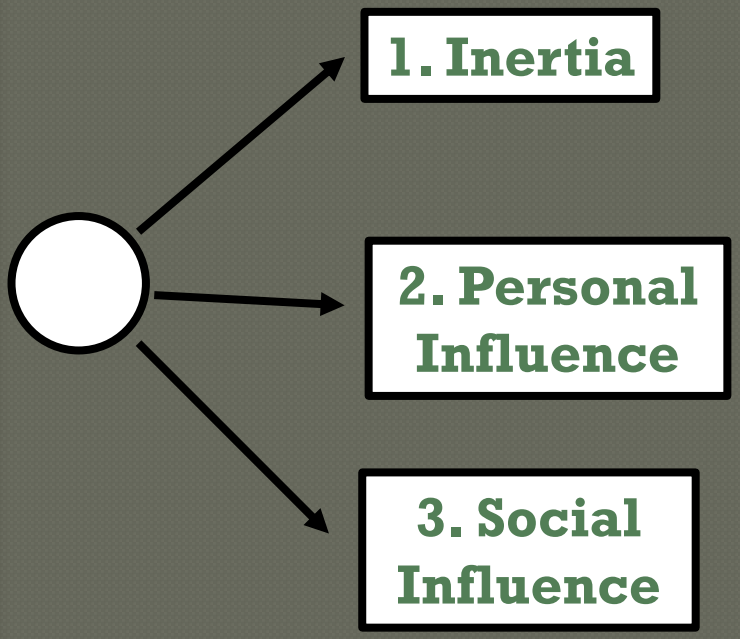
$$\mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \mathbf{v}_i^{t+1}$$

7. Go to step 2 until stopping criteria are satisfied

Introduction to the PSO: **Algorithm**

Particle's velocity:

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



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

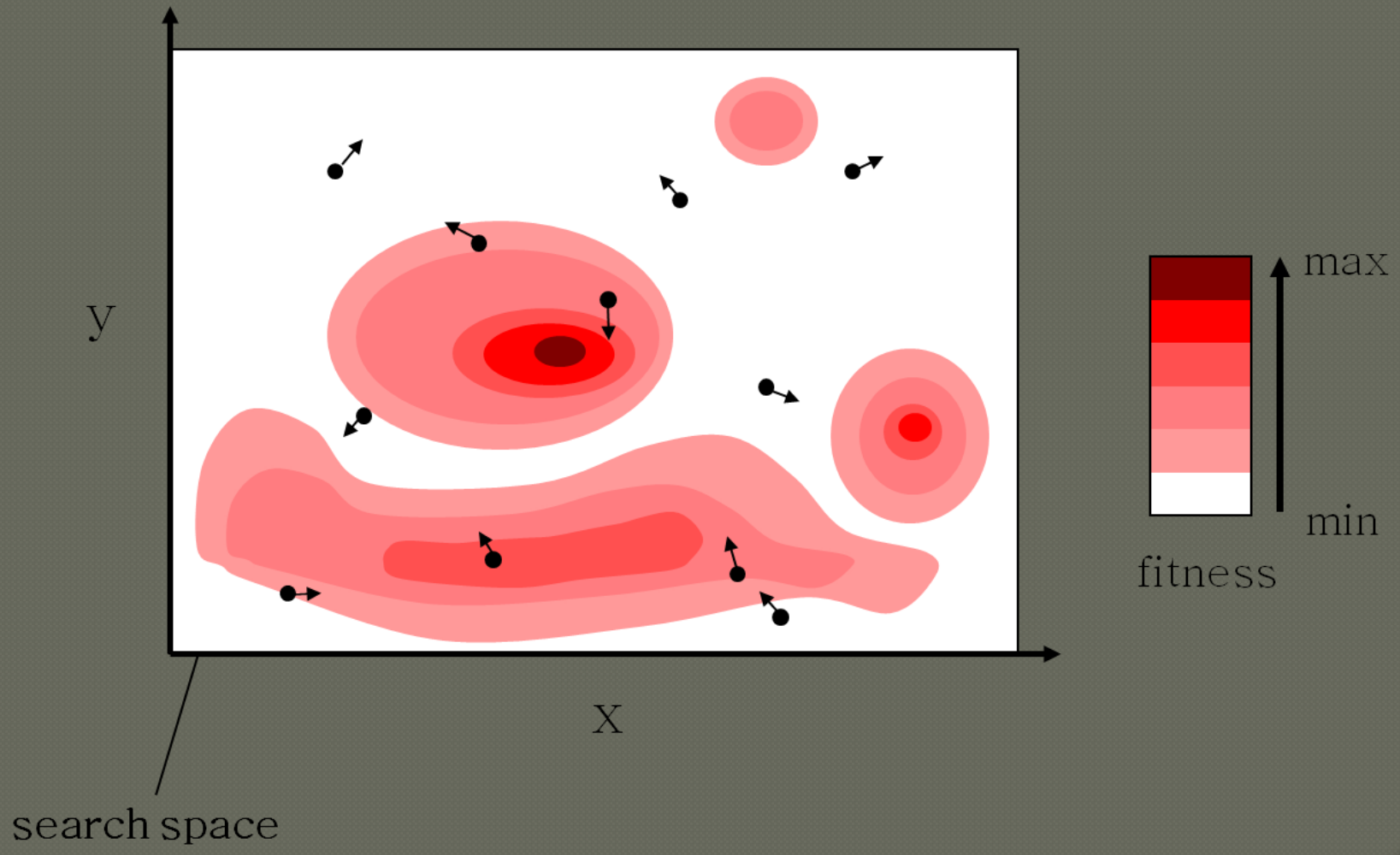
- Makes the particle follow the best neighbors direction

Introduction to the PSO: **Algorithm**

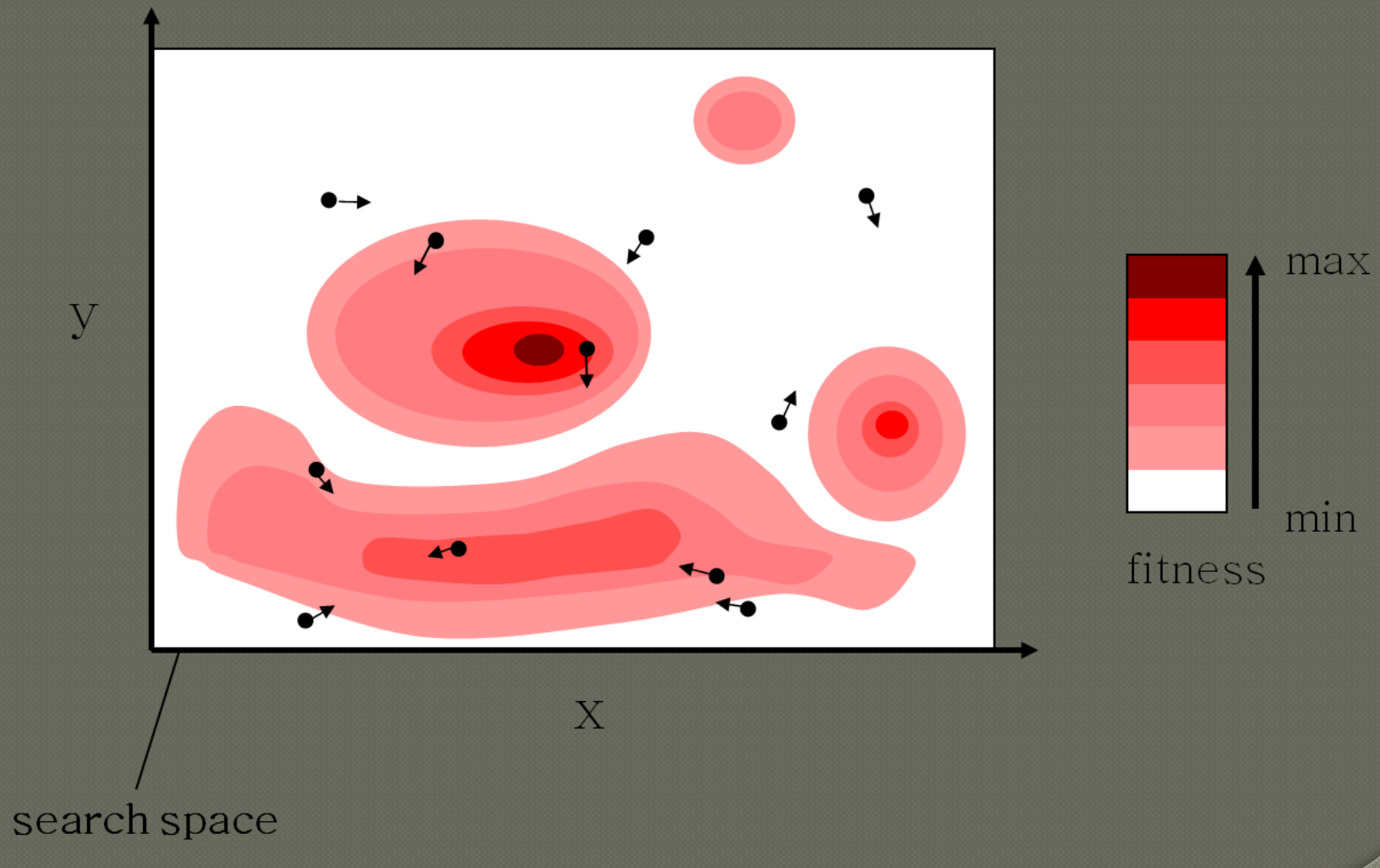
- Intensification: explores the previous solutions, finds the best solution of a given region
- Diversification: searches new solutions, finds the regions with potentially the best solutions
- In PSO:

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

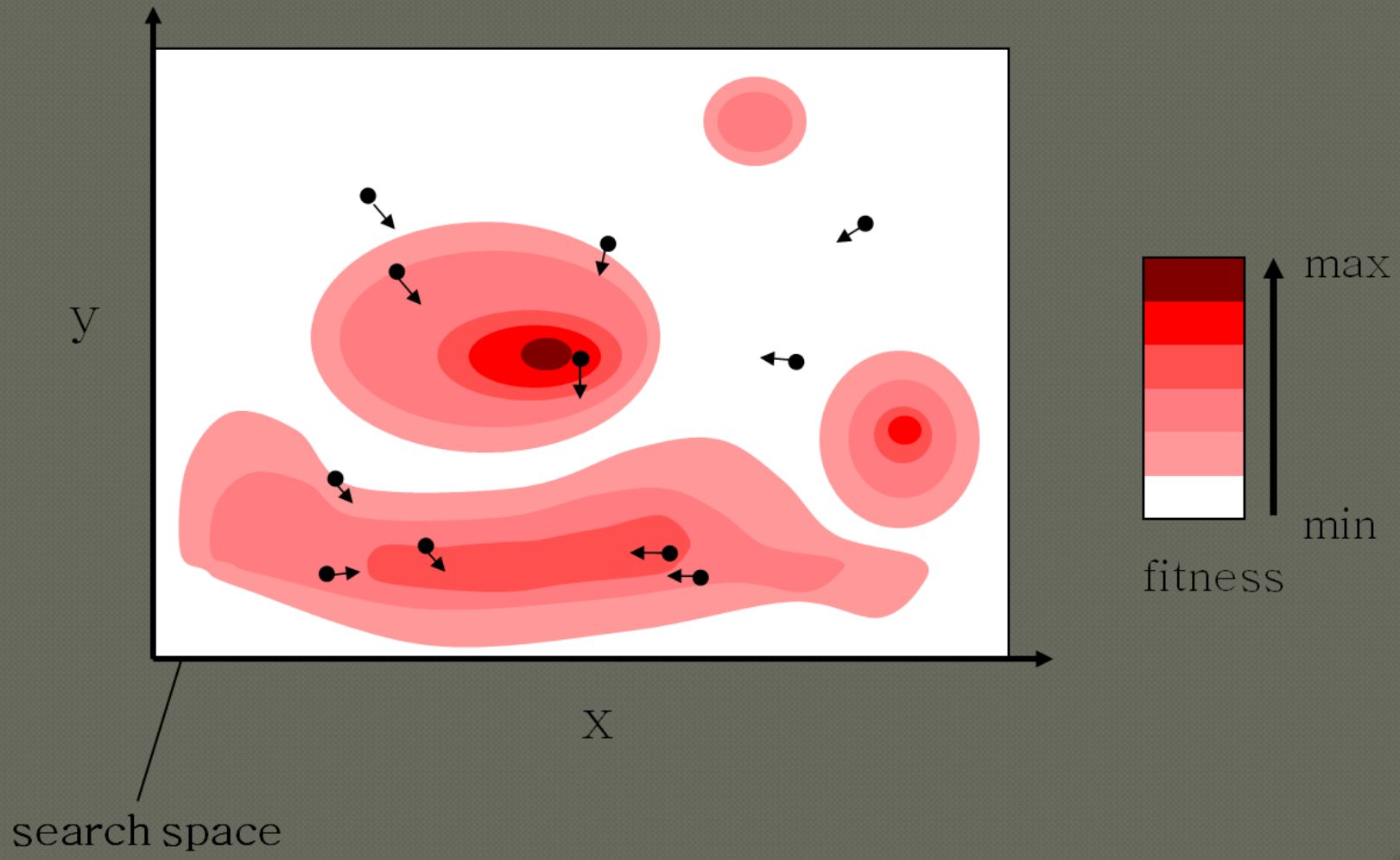
Introduction to the PSO: **Algorithm - Example**



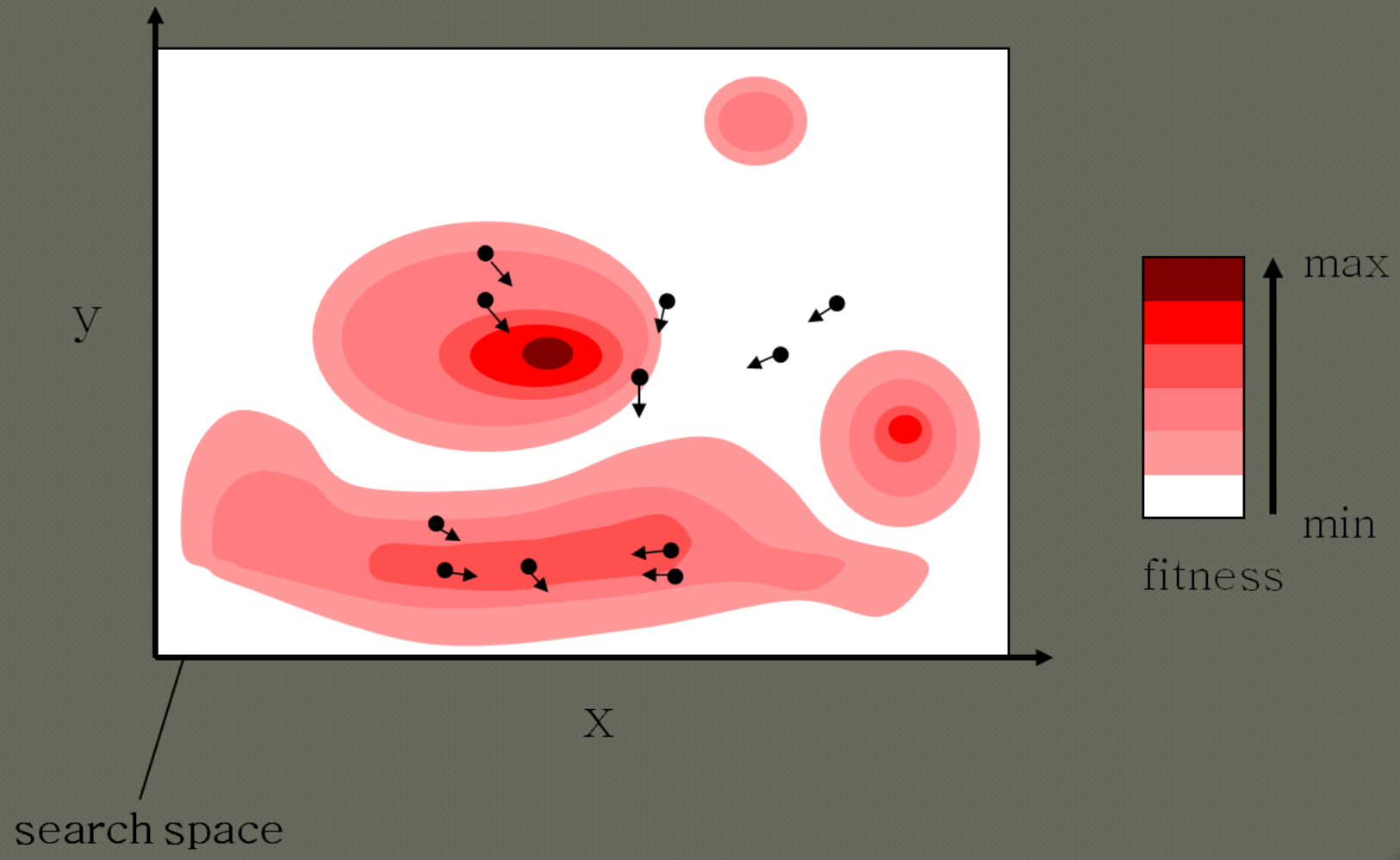
Introduction to the PSO: **Algorithm - Example**



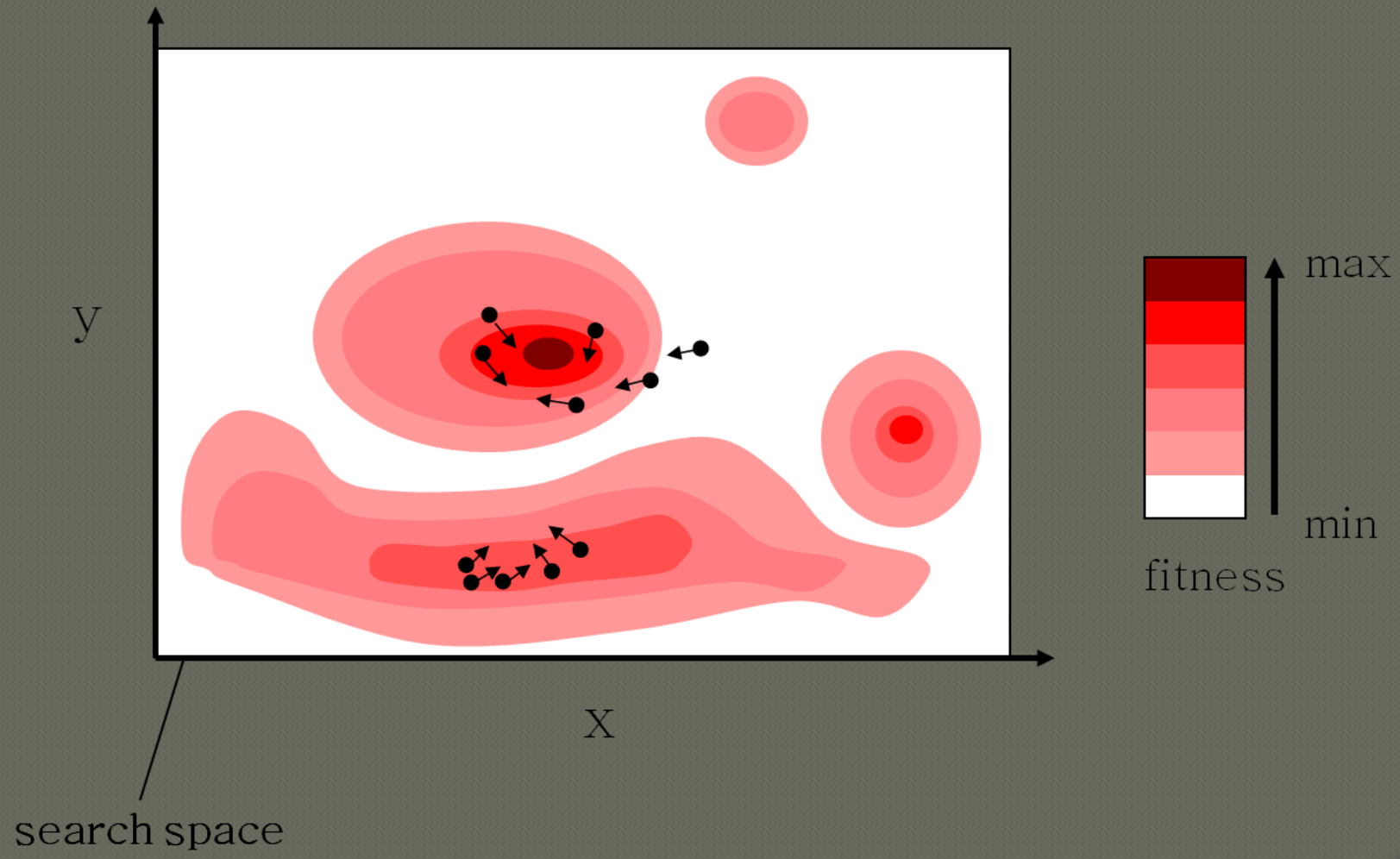
Introduction to the PSO: **Algorithm - Example**



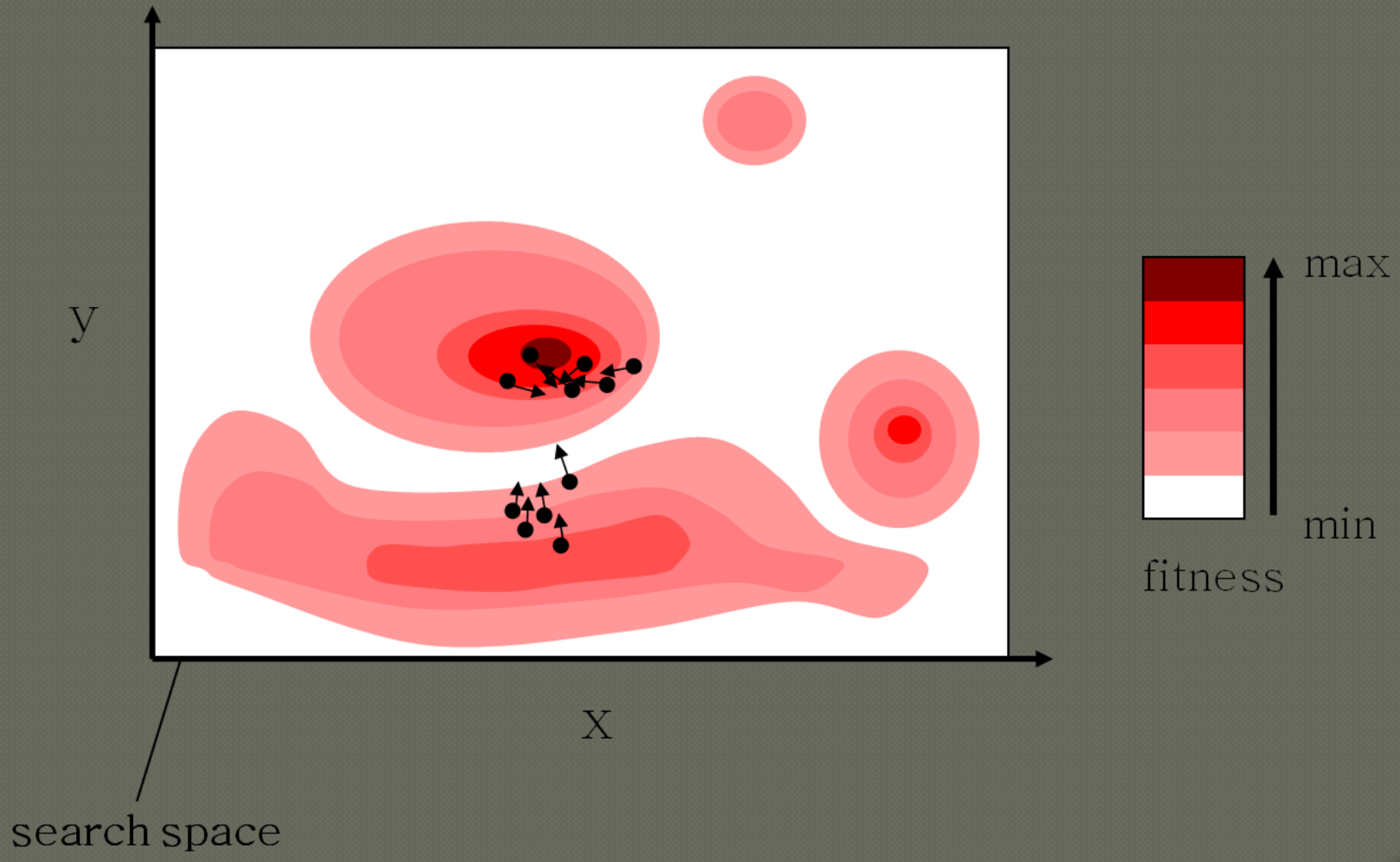
Introduction to the PSO: **Algorithm - Example**



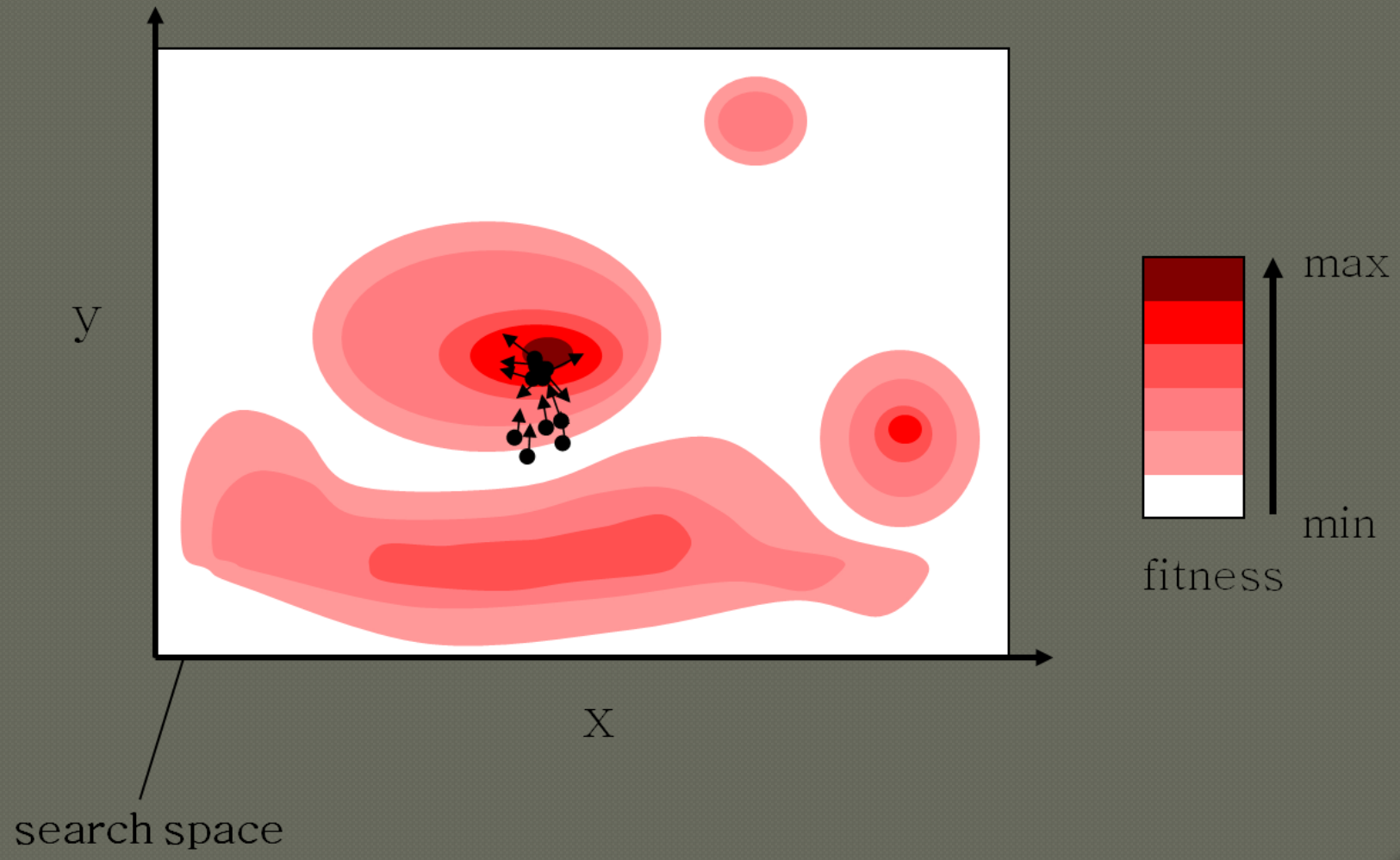
Introduction to the PSO: **Algorithm - Example**



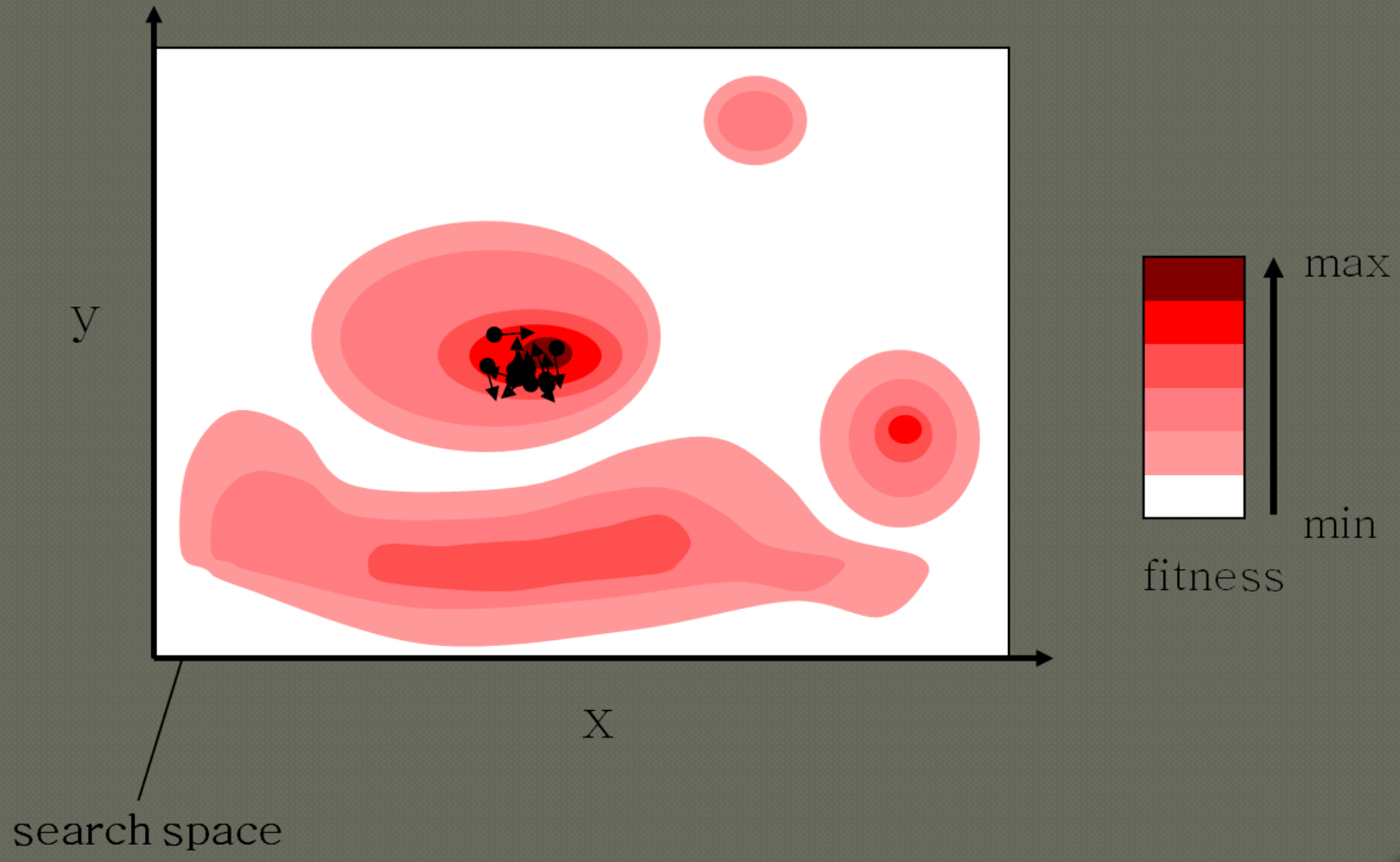
Introduction to the PSO: **Algorithm - Example**



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)

The Particle Swarm Optimization Algorithm