Particle Swarm Optimization (PSO) #1

粒子群聚最佳化法

➢ 粒子群聚最佳化法(Particle swarm optimization, PSO)是一種較新的技術，起因是出於對魚群(fish schooling)或鳥群(bird flocking)的群聚互動特性所引發之社會行為(Social behavior)的觀察，藉由群聚移動(Movement of swarms)的模擬，發展出一種以族群為基礎(Population-based stochastic search)之最佳化法(Optimization)。

➢ 因為PSO的強健性(robustness)及簡易性(simplicity)，近年來被廣泛地應用在各種領域上，並且有**運算簡單(simple operation)、執行快速(execution efficiency)、本質上可以並行處理(parallel processing)**等優點，因此已逐漸被所採用，但是還是有缺點的存在。

Outline

- Swarm Intelligence (SI)
- Introduction to Particle swarm optimization (PSO) (粒子群聚最佳化法、粒子群優法)
  - PSO real-world applications
  - PSO variants
  - Communication topologies
- Velocities updating
- Comparison between PSO and EC variants
- Hybrid PSO (with local or global optimization methods)
- PSO for multi-objective optimization (MOO)

Swarm Intelligence

Swarm intelligence (SI) is an artificial intelligence technique based around the study of **collective behavior** in decentralized, self-organized systems.

SI systems are typically made up of a population of simple **agents** interacting locally with one another and with their environment. Although there is normally no centralized control structure dictating how individual agents should behave, local interactions between such agents often lead to the emergence of global behavior. Examples of systems like this can be found in nature, including ant colonies (蟻群), bird flocking, animal herding, bacteria molding, and fish schooling.

➔ Nature Inspired Cooperative Strategies for Optimization

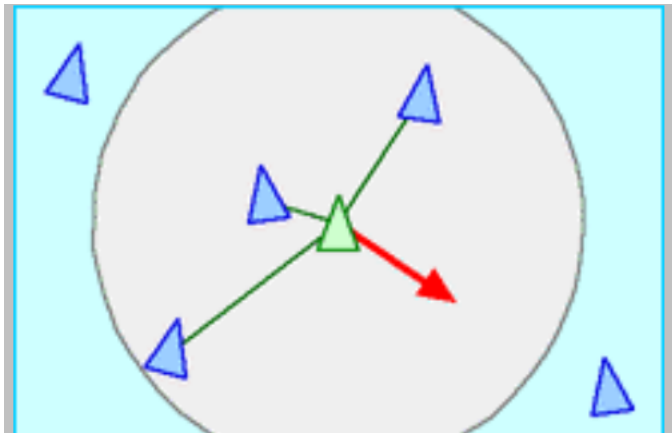
"In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food. This advantage can become decisive, outweighing the disadvantages of competition for food items, whenever the resource is unpredictably distributed in patches"

~ by Sociobiologist E. O. Wilson

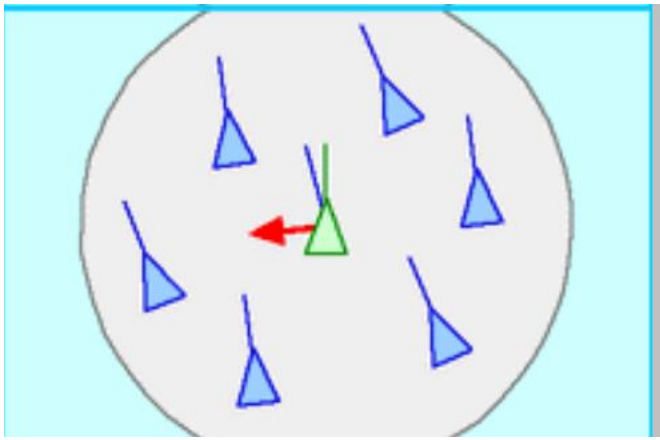
BOIDS

BOIDS

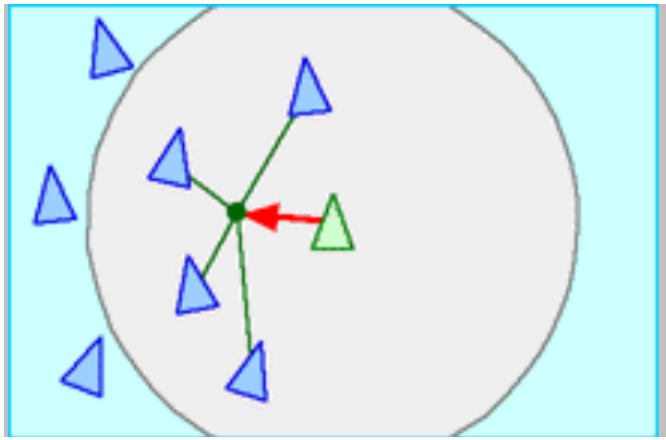
Including: Separation, Alignment, and Cohesion (凝聚)
Separation: steer to avoid crowding local flockmates;
Alignment: steer towards the average heading of local flockmates; Cohesion: steer to move toward the average position of local flockmates



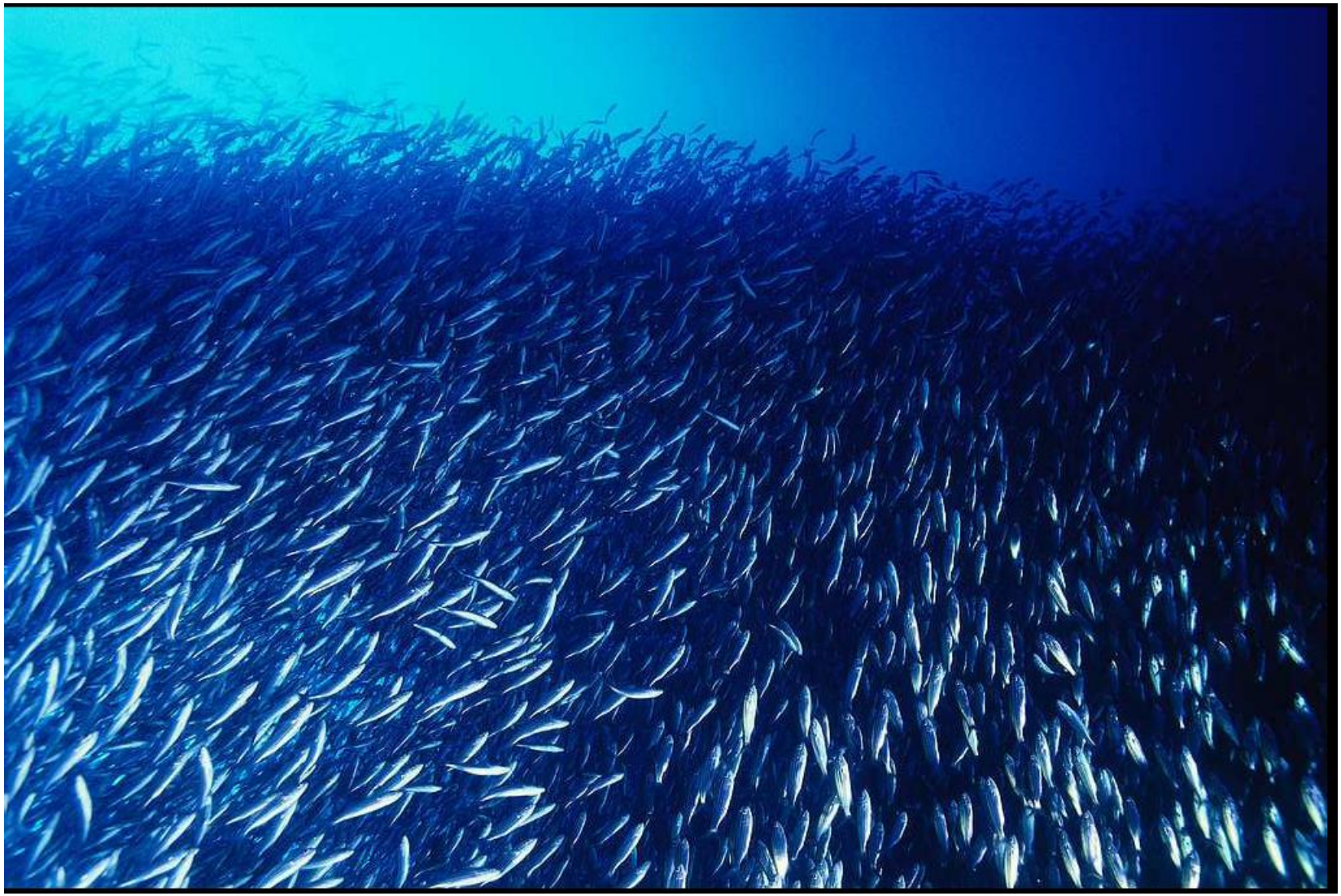
Separation: steer to avoid crowding local flockmates



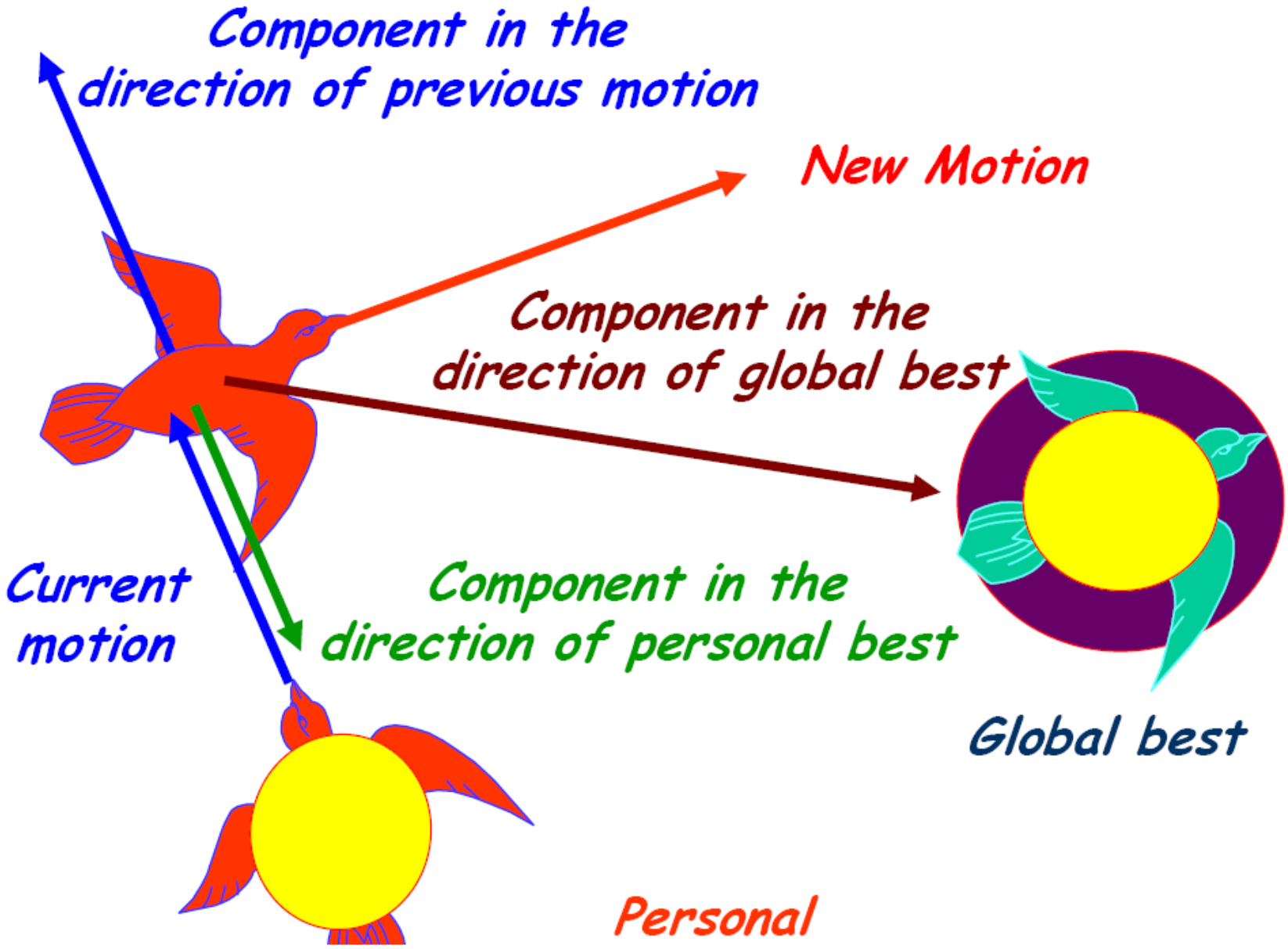
Alignment: steer towards the average heading of local flockmates

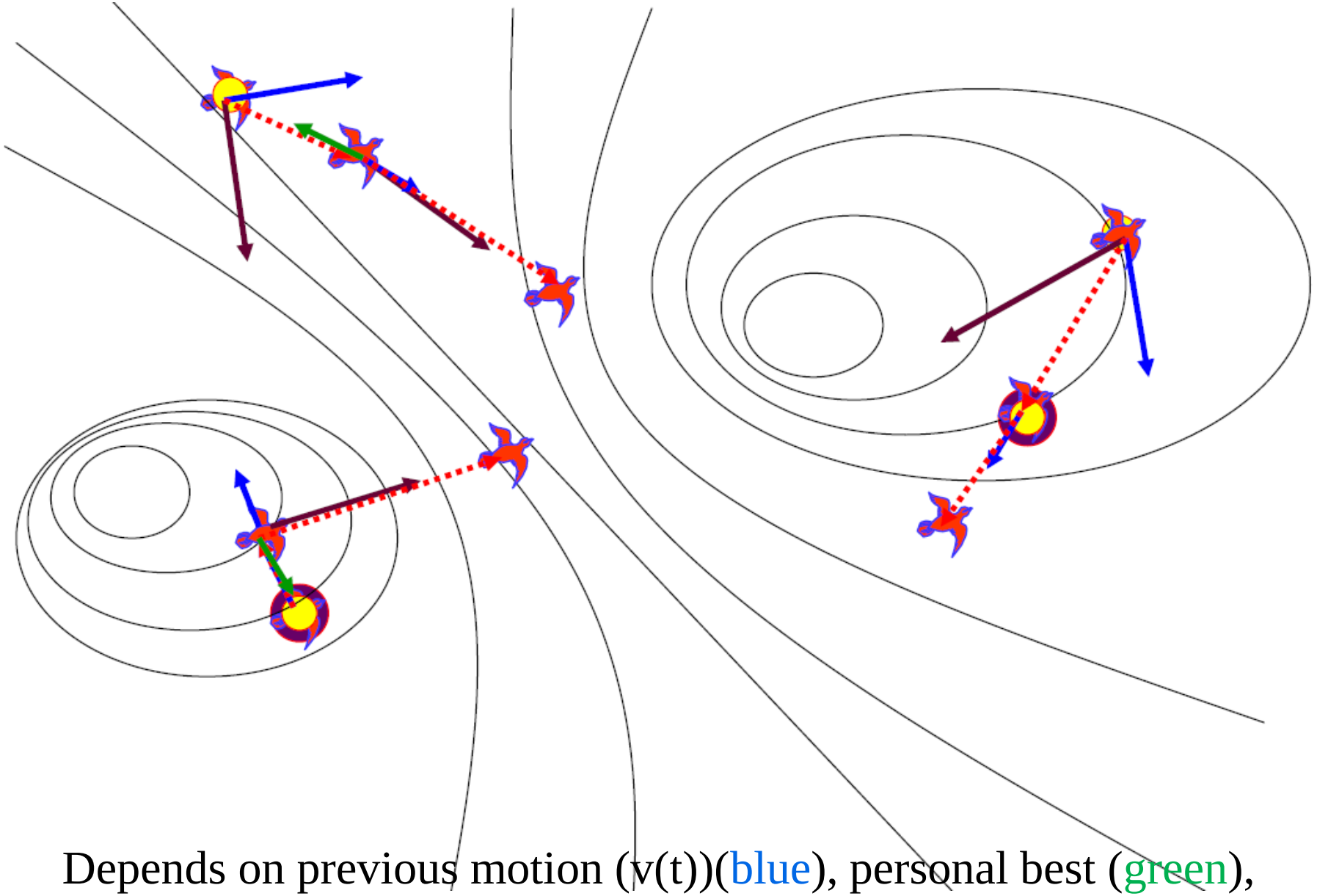


Cohesion: steer to move toward the average position of local flockmates



particle swarm optimization (**PSO**) is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. It solves a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formula over the particle's position and velocity. Each particle's movement is influenced by its local best known position, but is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions.


Component in the
direction of previous motion
New Motion
Component in the
direction of global best
Current
motion
Component in the
direction of personal best
Global best
Personal



Depends on previous motion (v(t))(blue), personal best (green), and global best (brown)， to obtain a new position (dashed red)

Introduction to Particle Swarm Optimization (PSO)

- Inventors: James Kennedy and Russell Eberhart
- Developed in 1995 (a relatively new member in EC in general term)
- An evolution-based (population-based) heuristic method (i.e., an evolutionary computation technique)
- An algorithm originally developed to imitate the motion of a flock of birds, or insects.
- Assumes information exchange (social interactions) among the search agents.
- Basic idea: keep track of
  - Global Best
  - Self (personal) Best

The original version

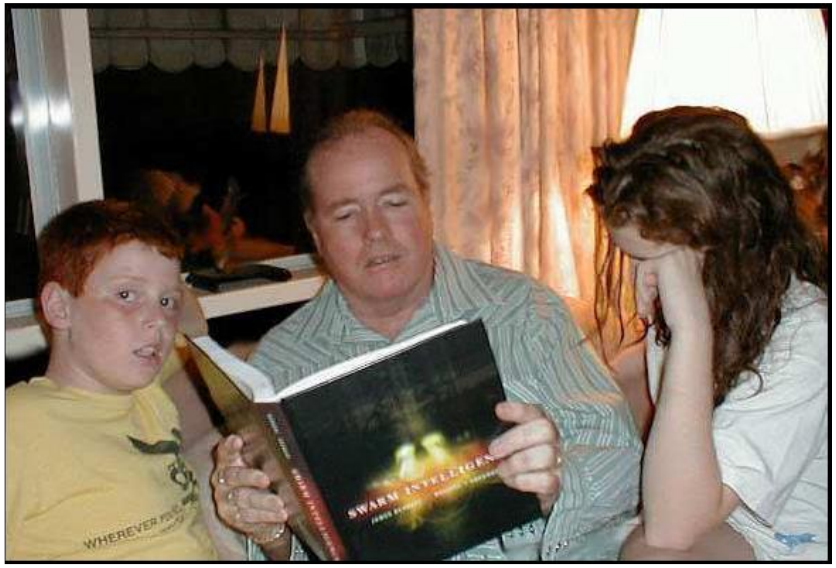
• The particle swarm concept originated as a simulation of a simplified social system, such as bird flocking or fish schooling.

• The original intent was to graphically simulate the graceful but unpredictable choreography (舞蹈、編舞) of a bird flock. (Boid 3cr)

• Initial simulations were modified to incorporate nearest-neighbor velocity matching, eliminate ancillary variables, and incorporate multi-dimensional search and acceleration by distance.

➔ During the evolution of the algorithm, it was realized that the conceptual model was, in fact, an optimizer.

Particle Swarm Optimization

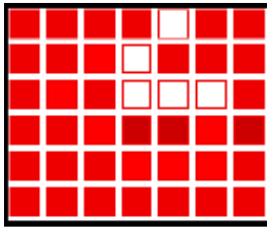


J. Kennedy and R. Eberhart, “Particle Swarm Optimization”, *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942-1948.

Particle Swarm Optimization

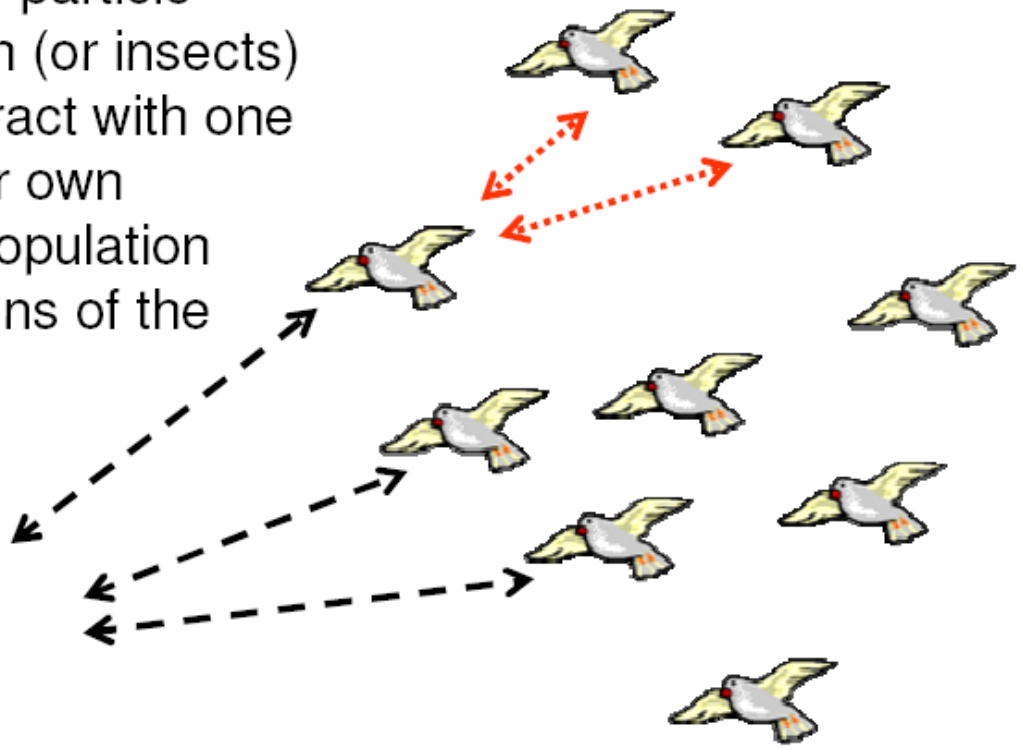
- PSO has its roots in Artificial Life and social psychology, as well as engineering and computer science. (not well explored yet!)
- The particle swarms in some way are closely related to cellular automata (CA):格構自動機、細胞自動機

  - ➢a) individual cell updates are done in parallel
  - ➢b) each new cell value depends only on the old values of the cell and its neighbors, and
  - ➢c) all cells are updated using the same rules (Rucker, 1999).
  - ➔Artificial Life



Individuals in a particle swarm can be conceptualized as cells in a CA, whose states change in many dimensions simultaneously.

As described by the inventers James Kennedy and Russell Eberhart, “particle swarm algorithm imitates human (or insects) social behavior. Individuals interact with one another while learning from their own experience, and gradually the population members move into better regions of the problem space”.

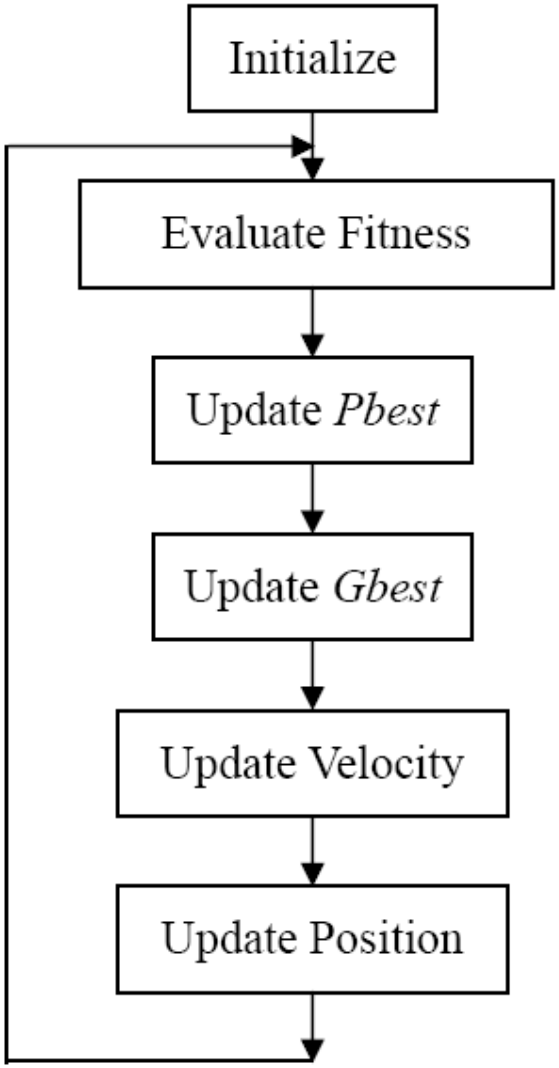


Why named as “Particle”, not “points”? Both Kennedy and Eberhart felt that velocities and accelerations are more appropriately applied to particles.

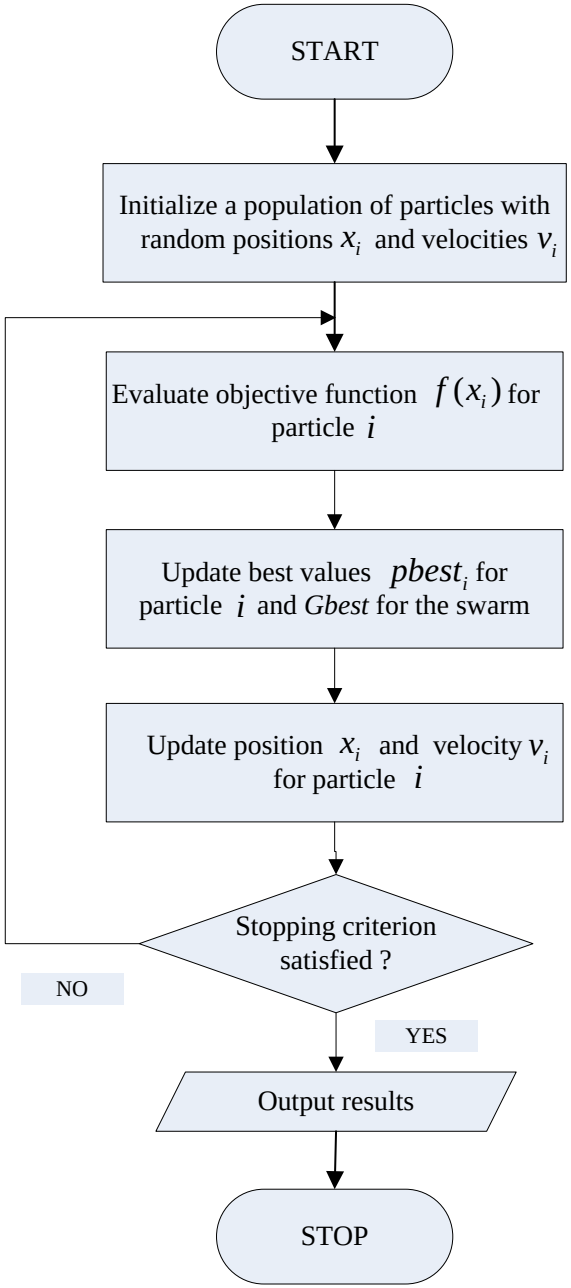
PSO演算法介绍

- 每個尋優（optimization）的問題解都被想像成一隻鳥在解空間中飛行(flying)，也稱為 “Particle”。
- 所有的Particle 都有一個fitness function 以判斷目前的位置之好壞。
- 每一個Particle必須賦予記憶性(memory)，能記得所搜尋到最佳位置。
- 每一個Particle 有一個速度(velocity)以決定飛行的距離與方向。

How does it work?



Flow chart of a basic PSO algorithm



$$v_i(t+1) = w \times v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t))$$

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

c_1, c_2: acceleration constants (Eberhart and Shi suggested using 0.2~2 with a typical value of 2 [23]);

rand : random number between 0 and 1;

$x_i(t)$: the position of particle i at iteration t ;

$v_i(t)$: the velocity of particle i at iteration t ;

w : inertia weight factor (Hu and Eberhart suggested using 0.4~0.9);

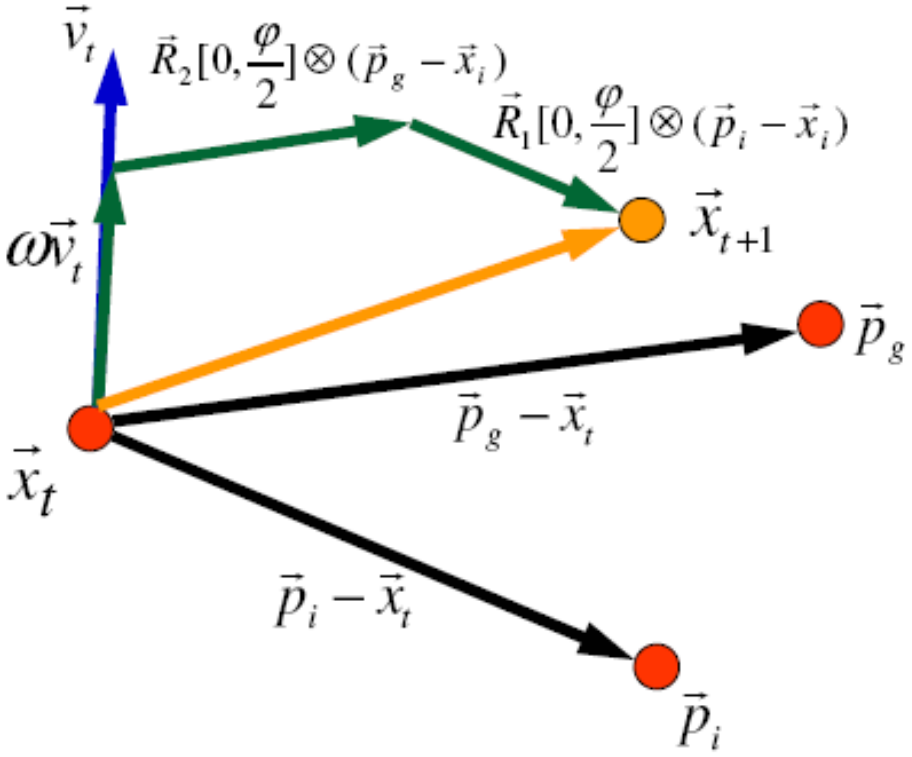
Gbest: the best previous position among all the particles;

Pbest$_i$: the best previous position of particle i;

演算法流程

Step 1. ***Initialize***
將群族做初始化，以隨機的方式求出每一Particle 之初始位置與速度。
Step 2. ***Evaluation*****：**
依據fitness function 計算出其fitness value 以作為判斷每一Particle之好壞。
Step 3. ***Find the Pbest$_i$*****：**
找出每一Particle 到目前為止的搜尋過程中最佳解，這個最佳解稱為Pbest。
Step 4. ***Find the Gbest*****：**
找出所有Particle 到目前為止所搜尋到的整體最佳解，此最佳解稱為Gbest。
Step 5. ***Update the Velocity and Position*****：**
依據(1) 與式(2) 更新每一Particle之速度與位置。
Step 6. **回到步驟 2. 繼續執行，直到獲得一個令人滿意的結果或符合終止條件為止。Go back to Step 2 until termination condition is satisfied.**

Visualizing PSO



$$v_i(t+1) = w \times v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t))$$

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

PSO Modeling

- Each solution vector is modeled as:
  - The coordinates of a bird or a ‘particle’ in a ‘swarm’ flying through the search space
  - All the particles have a **non-zero velocity** and thus never stop flying and are always sampling new regions.
- Each ‘particle’ remembers
  - where the global best (Gbest) and where the local best ($pbest_i$) are.

The search is guided by

- The collective consciousness of the swarm
- Introducing randomness into the dynamics in a controlled manner

$$v_i(t+1) = w \times v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t))$$

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

Note that the first term on the right-hand side of the velocity-updating rule in Eq. (1) represents the previous velocity, which provides the necessary momentum for particles to roam across the search space. The second term, known as the "cognitive" component, represents the personal thinking of each particle, which encourages the particles to move toward their own best positions found so far. The third term is known as the "social" component, which represents the collaborative effect of the particles, in finding the global optima.

每一個粒子 (Particle) 在群體中會有以下代表特徵：

x_i：粒子的移動位置

v_i：粒子的移動速度

$Pbest_i$：粒子在自己歷史紀錄中最好的位置

$Gbest$：粒子在整個群體中最好的位置

如圖 2-5 所示為粒子群聚最佳化法的演化流程圖，一開始用亂數產生粒子位置 x_i 及每個粒子的初始速度 v_i，粒子 $x_i \in [a_i\ b_i]$，初始速度 v_i 通常限定在 $\pm(a_i - b_i)/2$，接著將粒子帶入適應函數 (Fitness function) 中得到適應值 (Fitness value)，更新每一粒子在自己歷史紀錄中最好的位置 $Pbest_i$，

$$Pbest_i(t+1) = \begin{cases} x_i(t+1),\ f(x_i(t+1)) < f(Pbest_i(t)) \\ Pbest_i(t),\ f(x_i(t+1)) \geq f(Pbest_i(t)) \end{cases} \tag{11}$$

接著更新全域最佳解的粒子，

$$\begin{aligned} &Gbest(t) \in \{Pbest_1(t), Pbest_2(t), \cdots, Pbest_n(t)\} \\ &= \min(f(Pbest_1(t)),\ f(Pbest_2(t)), \cdots,\ f(Pbest_n(t))) \end{aligned} \tag{12}$$

其中，n 是指族群大小。然後將每一粒子帶入速度公式計算新的速度 v_i'，

$$\begin{aligned} v_i(t+1) = w \times v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) \\ + c_2 \times rand \times (Gbest - x_i(t)) \end{aligned} \tag{13}$$

其中，w 是介於 $0.4 \sim 0.9$ 的權重常數[2]，c_1、$c_2 = 2$，$rand \in [01]$。從公式得知，每一粒子的最新速度是參考上一次的速度、自己歷史紀錄中最好的位置、及整個族群中最好的位置。

若該粒子移動速度超出訂定的範圍，則直接限速為極限速度$\pm(a_i - b_i)/2$。最後將每一粒子之原本位置再加上新的速度得到新的位置，

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{14}$$

這就完成一次粒子移動。若該粒子位置超出制定的上下限範圍，則表示最佳解有可能在邊界上，這時候便將已經出界的粒子訂定為上或下界值。

Velocity-updating rules

為了改善以上問題，已經有許多研究對PSO提出改變與調整，特別是速度更新法則(Velocity-updating rules)以及各項運作參數之調整。

- Inertia Weight by Shi: 藉由慣性權重改變粒子的速度，以達到粒子在空間搜尋能更快找到全域最佳解。
- Constriction Factor by Clerc: 利用壓縮因子來調整粒子的多樣性，以得到較佳的收斂性。

參數選取

綜合既有文獻之作法[8-11]發現 PSO 搜尋法最關鍵之處在於速度的更新與參數之選擇：

(a) 動態慣性權重 w_i：

原始的PSO 並沒有使用權重，之後有人加上一固定權重值，以控制先前速度對現在速度的影響，因此可以獲得較佳的性能[1]，甚至於將權重值設計成隨著演化代數增加而減少[9,28,29]，以提升最佳化收斂的穩定度，因此將速度 V_i' 及權重 w_i 公式設計為[30]：

$$v_i(t+1) = w_t \times v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t)) \tag{15}$$

$$w_t = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times t \tag{16}$$

(b) 收縮係數K[30,31]：

文獻[28]宣稱使用慣性權重w_i之速度更新法較使用收縮係數K法具有較快的收斂速度，由於收縮係數K係用以控制粒子速度，現有之速度更新公式V_i'及收縮係數K訂為：

$$v_i(t+1) = K \cdot [v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t))] \tag{17}$$

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \qquad \varphi = c_1 + c_2, \varphi > 4 \tag{18}$$

其中，φ為收縮係數，由於有文獻[29]提出$c_1 + c_2$必需大於4，在選定$c_1 + c_2 = 4.1$時，可以得到$K = 0.729$[28]。

(c) 極限速度之控制：

雖然有各種控制速度的方法，但是有些時候難免會因為速度太大造成粒子超出設定範圍 $x_i \in [a_i\ b_i]$，為了避免粒子移動太快，通常會限制 $V_i^{'}$ 的極限速度，常見的法則係將極限速度訂為粒子初始條件上下限的一半 $\pm(a_i - b_i)/2$ 。

$$v_{j,g}^{(t+1)} = w \cdot v_{j,g}^{(t)} + c_1^* rand()^* (\text{pbest}_{j,g} - x_{j,g}^{(t)}) + c_2^* Rand()^* (\text{gbest}_g - x_{j,g}^{(t)}) \quad (10)$$

$$x_{j,g}^{(t+1)} = x_{j,g}^{(t)} + v_{j,g}^{(t+1)}$$

$$j = 1, 2, \ldots, n$$

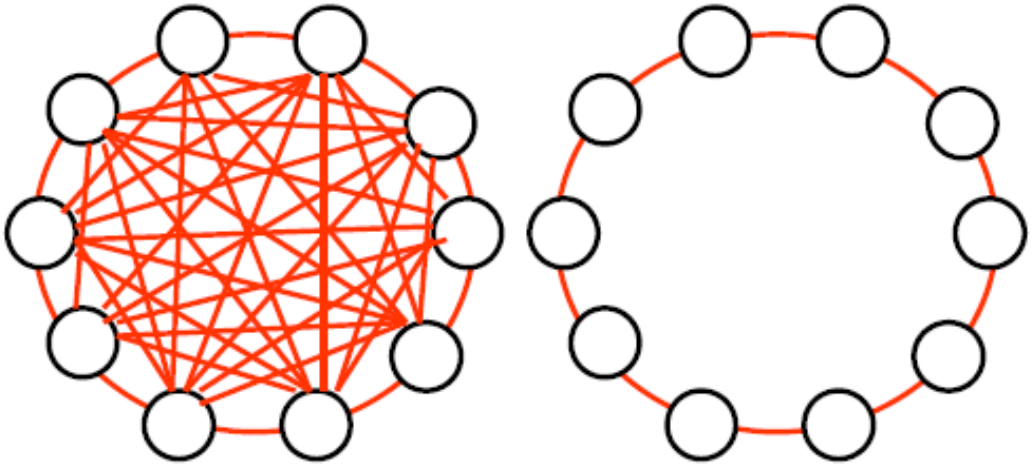
$$g = 1, 2, \ldots, m \quad (11)$$

where

n	number of particles in a group;
m	number of members in a particle;
t	pointer of iterations (generations);
$v_{j,g}^{(t)}$	velocity of particle j at iteration t, $V_g^{\min} \leq v_{j,g}^{(t)} \leq V_g^{\max}$;
w	inertia weight factor;
$c1$, $c2$	acceleration constant;
$rand()$, $Rand()$	random number between 0 and 1;
$x_{j,g}^{(t)}$	current position of particle j at iteration t;
pbest_j	pbest of particle j;
gbest	gbest of the group.

PSO topologies

Star Topology
(global neighborhood)



Ring Topology
(neighborhood
of 3)

Two most common models:

- **gbest**: each particle is influenced by the best found from the entire swarm.
- **lbest**: each particle is influenced only by particles in local neighbourhood.

pbest: each particle's personal best

Neighborhood best / local best: *Lbest*

由於在計算速度時，通常只用到粒子本身之歷史最佳紀錄，以及當代最佳紀錄，有時候會流失區域資訊，因此可將鄰近區域最好的粒子(Neighborhood best)也考慮進來，以產生更新速度公式：

$$v_i(t+1) = w \times v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t)) + c_3 \times rand \times (Lbest - x_i(t))$$

Particle Swarm Dynamics

non-zero velocity

PS never stop flying

$$\vec{x}(k+1) = \vec{x}(k) + \vec{v}(k)$$

Self consciousness of the swarm

Controlled randomness

Inertia

$$\vec{v}(k+1) = w.\vec{v}(k) + r(0,a_1).(\vec{x}_{SelfBest}(k) - \vec{x}(k)) + r(0,a_2).(\vec{x}_{GroupBest}(k) - \vec{x}(k))$$

The collective consciousness of the swarm

Exploration versus Exploitation

- Inertia constant *w*

$$v_i(t+1) = w \times v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t))$$

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

  - *w* controls exploration and exploitation
  - $w \geq 1$
    - velocities increases (accelerates) over time
    - swarm diverges
    - particles fail to change direction towards more promising regions

Exploration versus Exploitation

$0 < w < 1$

- particles decelerate
- convergence also depends on values of c1 and c2

- Large w → favor exploration
  Small w → favor exploitation
- Problem-dependent

•If w=0, the velocity of the particle is only determined by the $pbest_i$ and *Gbest* positions; this means that the particle may change its velocity instantly if it is moving far from the best positions in its knowledge. Thus, low inertia weights favor exploitation (local search).

•If w is high, the rate at which the particle may change its velocity is lower (it has an "inertia" that makes it follow its original path) even when better fitness values are known. Thus, high inertia weights favor exploration (global search).

Exploration versus Exploitation

$$v_i(t+1) = K \cdot [v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t))]$$

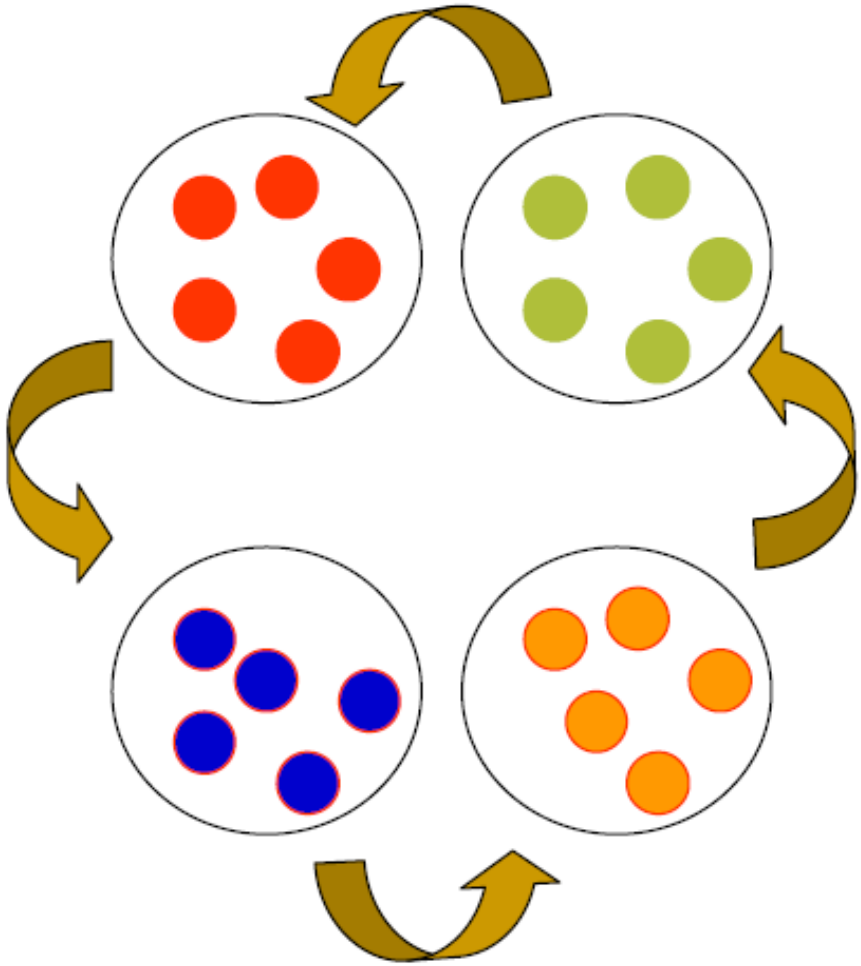
$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \qquad \varphi = c_1 + c_2, \varphi > 4$$

- If $\varphi \geq 4$, the swarm is guaranteed to converge
- When choosing $\varphi = 4.1$, $K = 0.729$
- $K \approx 0.729$ ➔ slow convergence, high degree of exploration
- $K \approx 0$ ➔ fast convergence, local exploitation

特點 (Features)

- Distributed search (Island model)
- memory (具記憶性)
- few components, easy to realize (組件較少，容易實現)
- suitable for continuous optimization (適合在連續性的範圍內搜尋)
- Good computation efficiency(具計算效率)
- Stable convergence characteristics(收斂特性)

Island model



Related Issues: Performance

- There are a number of related issues concerning PSO:
  - Controlling velocities (determining the best value for Vmax),
  - Swarm Size,
  - Neighborhood Size,
  - Updating X and Velocity Vectors,
  - Robust Settings for ($\varphi 1$ and $\varphi 2$),
  - An Off-The-Shelf PSO

- Carlisle, A. and Dozier, G. (2001). "An Off-The-Shelf PSO", *Proceedings of the 2001 Workshop on Particle Swarm Optimization*, pp. 1-6, Indianapolis, IN. (http://antho.huntingdon.edu/publications/Off-The-Shelf_PSO.pdf)

Controlling Velocities

- When using PSO, it is possible for the magnitude of the velocities to become very large.
- Performance can suffer if Vmax is inappropriately set.
- Two methods were developed for controlling the growth of velocities:
  - A dynamically adjusted inertia factor *w*, and
  - A constriction coefficient *K*.

$$v_i(t+1) = w \times v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t))$$

$$v_i(t+1) = K \cdot [v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t))]$$

Swarm and Neighborhood Size

- Concerning the swarm size for PSO, as with other ECs there is a trade-off between solution quality and cost (in terms of function evaluations (FEs)).
- Global neighborhoods seem to be better in terms of computational costs. The performance is similar to the ring topology (or neighborhoods greater than 3).
- There has been little research on the effects of swarm topology on the search behavior of PSO.

Suggestions on the selection of PSO parameters (An Off-The-Shelf PSO)

- Explicit and implicit parameters of PSO, which affects its performance (efficiency and reliability)
  - Social learning rate c2
  - Cognitive learning rate c1
  - Population size
  - Neighborhood size
  - Synchronous vs asynchronous update
  - Inertia weight w
  - Constriction factor *K*

$$v_i(t+1) = w \times v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t))$$

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

Original PSO by Kennedy:

$$v_{id} = v_{id} + \varphi_1 * \text{rand}() * (p_{id} - x_{id}) + \varphi_2 * \text{rand}() * (p_{gd} - x_{id})$$
$$x_{id} = x_{id} + v_{id}$$

A constant V_{max} was used to limit the velocities of the particles and improve the resolution of the search

➔Original PSO searches wide areas effectively, but tends to lack local search precision.

PSO by Eberhart and Shi:

$$v_{id} = \omega * v_{id} + \varphi_1 * rand() * (p_{id} - x_{id}) + \varphi_2 * rand() * (p_{gd} - x_{id})$$

An inertia, *w* , was introduced to dynamically adjust the velocities over time, gradually focusing the PSO into a local search

PSO by Maurice Clerc:

$$v_{id} = K(v_{id}+\varphi_1*\text{rand}()*(p_{id} - x_{id})+\varphi_2*\text{rand}()*(p_{gd} - x_{id}))$$

A constriction factor, ***K***, was introduced to improve PSO's ability to constrain and control velocities.

K combined with V_{max} significantly improved the PSO performance, by choosing ***K*** as:

$$K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}$$

Testing of various PSO parameters

• Setting: 30 particles, c1=c2=2.05, Vmax=Xmax, neighborhood is global, particles are updated synchronously, and incorporating Clerc constriction factor ***K***.

• 20 repetitions (runs), with an upper limit of 100,000 iterations (generations).

Benchmark functions

Function	Equation	Parameters
Sphere (De Jong F1)	$f(x) = \sum_{i=1}^{n} x_i^2$	Dimensions: 30 Xmax: 100 Acceptable error: <0.01
Rosenbrock	$f(x) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	Dimensions: 30 Xmax: 30 Acceptable error: <100
Rastrigrin	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	Dimensions: 30 Xmax: 5.12 Acceptable error: <100
Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Dimensions: 30 Xmax: 600 Acceptable error: <0.1
Schaffer F6	$f(x) = 0.5 - \frac{\left(\sin\sqrt{x^2 + y^2}\right)^2 - 0.5}{\left(1.0 + 0.001(x^2 + y^2)\right)^2}$	Dimensions: 2 Xmax: 100 Acceptable error: <0.00001

Figure 1. Functions in the test suite

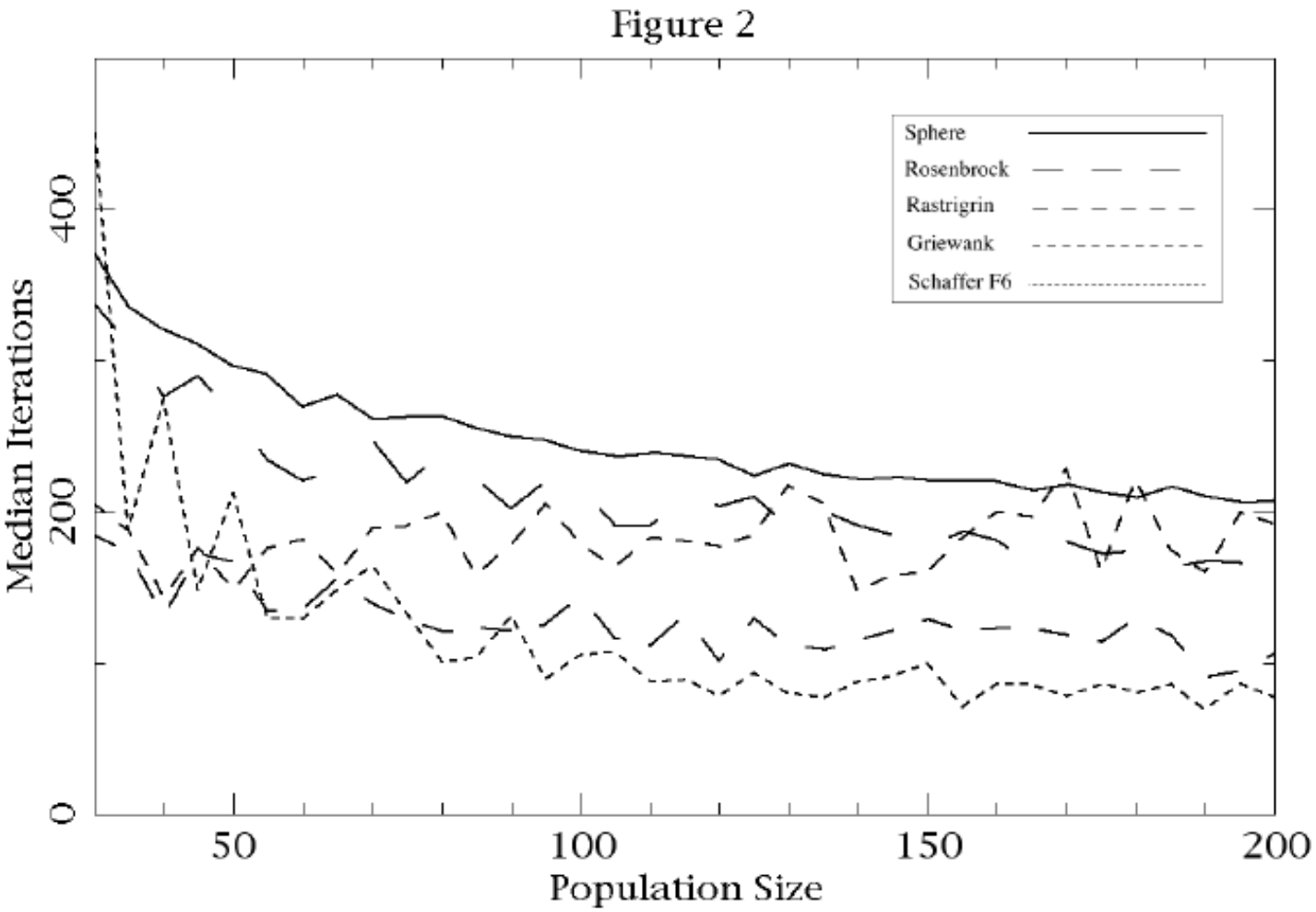
Population size

Eberhart reports:

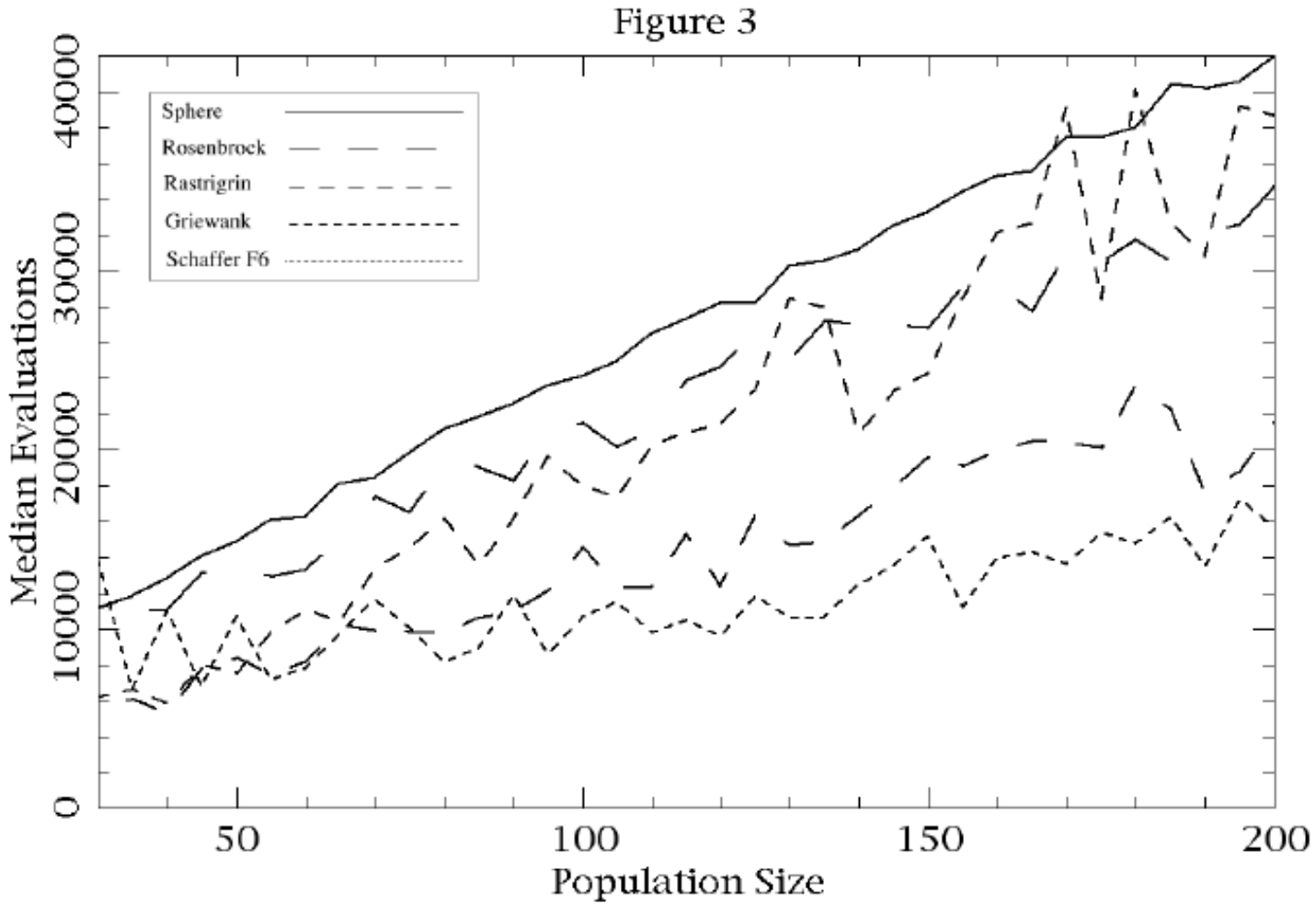
"PSO is **not sensitive** to the population size"

➔ The statement is generally true in terms of performance, but not in terms of cost (function evaluations)

Population size



Population size



Population size

Conclusion:

A population size of **30** appears to be a good choice. It is small enough to be efficient, yet large enough to produce reliable results.

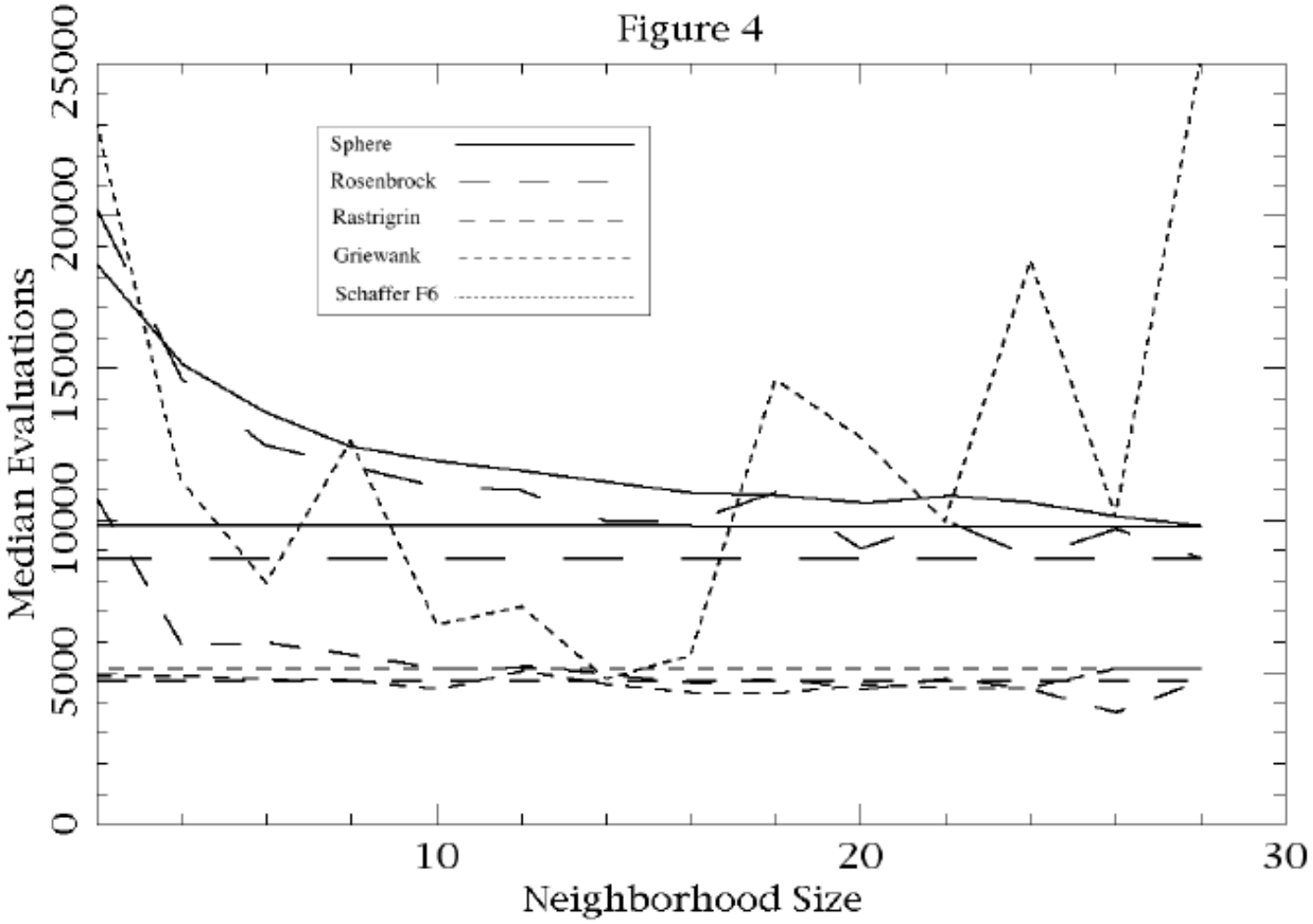
Neighborhood size

Conclusion:

•A small local neighborhood is better at avoiding local minimum; a global neighborhood converges faster.

•The global neighborhood appears to be a better general choice, as it seems to require less work (function evaluations) for the same results (success rates).

Neighborhood size



Synchronous vs Asynchronous updating

Conclusion:

Asynchronous updates are generally less costly.

$$v_i(t+1) = w \times v_i(t) + c_1 \times rand \times (pbest_i - x_i(t)) + c_2 \times rand \times (Gbest - x_i(t))$$

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

	Median Iterations					
	Neighborhood of 2		Neighborhood of 14		Global Neighborhood	
	Async	Sync	Async	Sync	Async	Sync
Sphere	622.5	628.0	376.5	401.5	331.5	368.5
Rosenbrock	685.0	645.0	365.0	367.5	288.0	327.0
Rastrigrin	312.0	314.5	162.5	162.0	155.0	206.5
Griewank	152.0	181.5	154.5	148.5	173.0	171.0
Schaffer F6	624.0	993.5	310.0	499.5	255.5	522.5

Figure 5 Asynchronous vs. Synchronous updates

Cognitive (c1) / Social (c2) ratio

Background:

- The sum of c_1 and c_2 is about 4.1
- Usual setting of 2.05 each

Conclusion:

- Social component tends to lead to more local minima trapping! i.e. smaller c2 is preferred.
- A reasonable comprise for the cognitive and social component values appear to be c1=2.8 and c2=1.3, respectively.

	Best Performance	Median Evaluations	Acceptable Range for φ_1
Sphere	2.90	7,679.0	2.05-3.00
Rosenbrock	2.80	7,318.0	1.60-3.10
Rastrigrin	2.50	4,213.5	2.05-3.50
Griewank	2.90	4,026.0	1.00-3.60
Schaffer F6	2.90	5,470.0	2.05-3.60

Figure 6 Cognitive Component Values

Magnitude of φ

Background:

- Usual setting of 4.1 for $\varphi=c_1+c_2$
- Varying φ from 1.0~6.0 by maintaining the same Cognitive / Social ratio of 2.8:1.3

$$K = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|}$$

Conclusion:

- The setting of $\varphi=c_1+c_2=4.1$ is the best in general cases.
- φ=4.1, K=0.729

V_{max} Selection

Background:

- Reducing V_{max} from a very large value to $V_{max}=X_{max}$ significantly improved PSO performance
- In general, performance improves as V_{max} shrinks (but with a low limit).
- V_{max} is the step size of the swarm, the maximum distance a particle can travel in an iteration
- Reducing V_{max} too much impedes the ability of the swarm to search.

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

Conclusion:

For problems where X_{max} must be enforced, when a particle reaches X_{max}, set velocity to zero. If X_{max} is not enforced, V_{max} need not be enforced, either.

	Median Iterations				
	No *Xmax* enforced	*Vmax*= *Xmax*	*Vmax*= *Xmax*/2	*Vmax*= *Xmax*/4	V_{id}=0.0 at *Xmax*
Sphere	226.0	226.5	210.0	195.5	231.5
Rosenbrock	199.5	207.5	187.0	152.5	205.0
Rastrigrin	131.0	122.0	94.5	69.5	124.0
Griewank	147.0	146.0	113.0	120.0	178.0
Schaffer F6	272.0	285.0	280.0	761.0	261.5

PSO applications

Problems with continuous, discrete, or mixed search space, with multiple local minima.

- Evolving neural networks:
- Human tumor analysis;
- Battery pack state-of-charge estimation;
- Real-time training of neural networks (Diabetes among Pima Indians);
- Servomechanism (time series prediction optimizing a neural network);
- Reactive power and voltage control;
- Ingredient mix optimization;
- Pressure vessel (design a container of compressed air, with many constraints);
- Compression spring (cylindrical compression spring with certain mechanical characteristics);
- Moving Peaks (multiple peaks dynamic environment); and more

➔ PSO can be tailor-designed to deal with specific real-world problems.

Some PSO variants

• Tribes (Clerc, 2006) – aims to adapt population size, so that it does not have to be set by the users;
• ARPSO (Riget and Vesterstorm, 2002) – uses a diversity measure to alternate between 2 phases;
• Dissipative PSO (Xie, et al., 2002) – increasing randomness;
• PSO with self-organized criticality (Lovbjerg and Krink, 2002) – aims to improve diversity;
• Self-organizing Hierachical PSO (Ratnaweera, et al. 2004);
• FDR-PSO (Veeramachaneni, et al., 2003) – using nearest neighbor interactions;
• PSO with mutation (Higashi and Iba, 2003; Stacey, et al., 2004)
• Cooperative PSO (van den Bergh and Engelbrecht, 2005) – a cooperative approach
• DEPSO (Zhang and Xie, 2003) – aims to combine DE (differential evolution) with PSO; (incorporating bell-shaped mutations borrowed from ES)
• CLPSO (Liang, et al., 2006) – incorporate learning from more previous best particles.

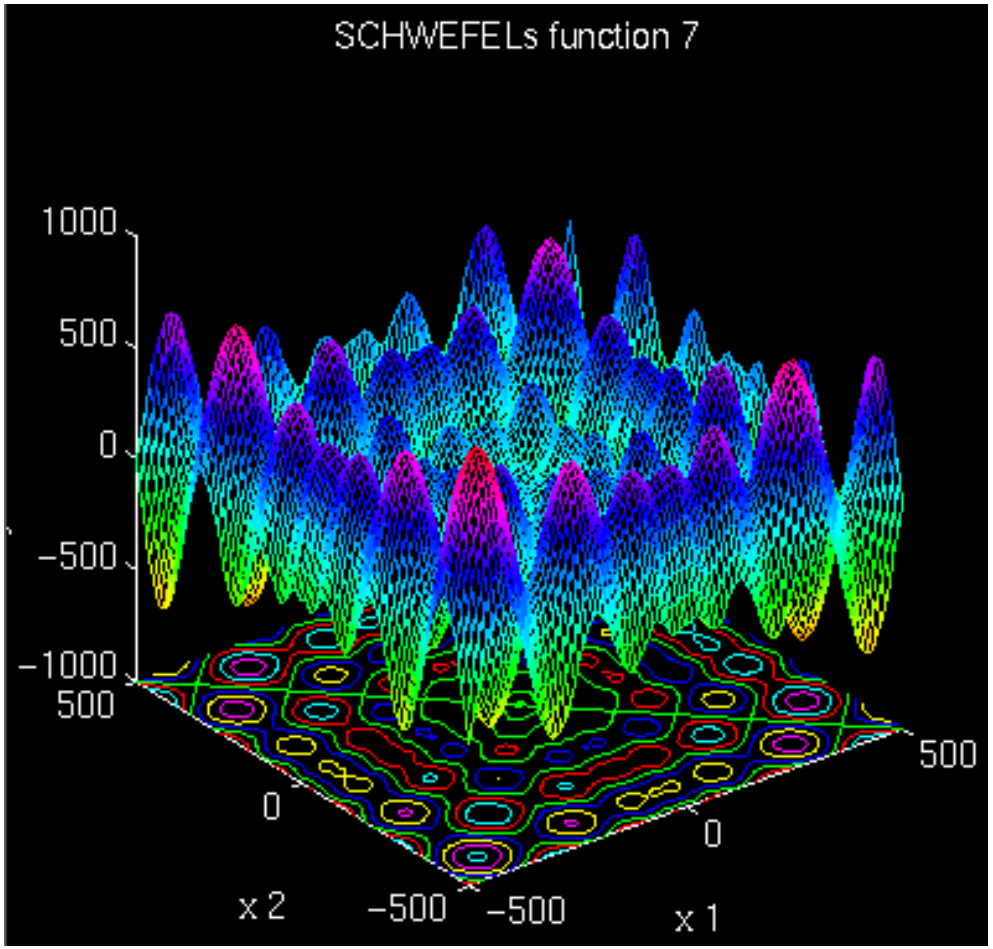
Its links to Evolutionary Computation

1. Both PSO and EC are population based.
2. PSO also uses the fitness concept, but, less-fit particles do not die. No "survival of the fittest".
3. No evolutionary operators such as crossover and mutation.
4. Each particle (candidate solution) is varied according to its past experience and relationship with other particles in the population.
5. Having said the above, there are hybrid PSOs, where some EC concepts are adopted, such as selection, mutation, etc.

Future development

- Incorporate evolution operators from other EC variants
- Hybridation with other heuristic methods
- Parallel computation
- Multi-objective optimization (MOO) as MOPSO
- Hardware implementation (SOPC-based)
- Theoretical development (not mature yet!)

Schwefel's function



$$f(x) = n \cdot 418.9829 - \sum_{i=1}^{n} (x_i) \cdot \sin(\sqrt{|x_i|})$$

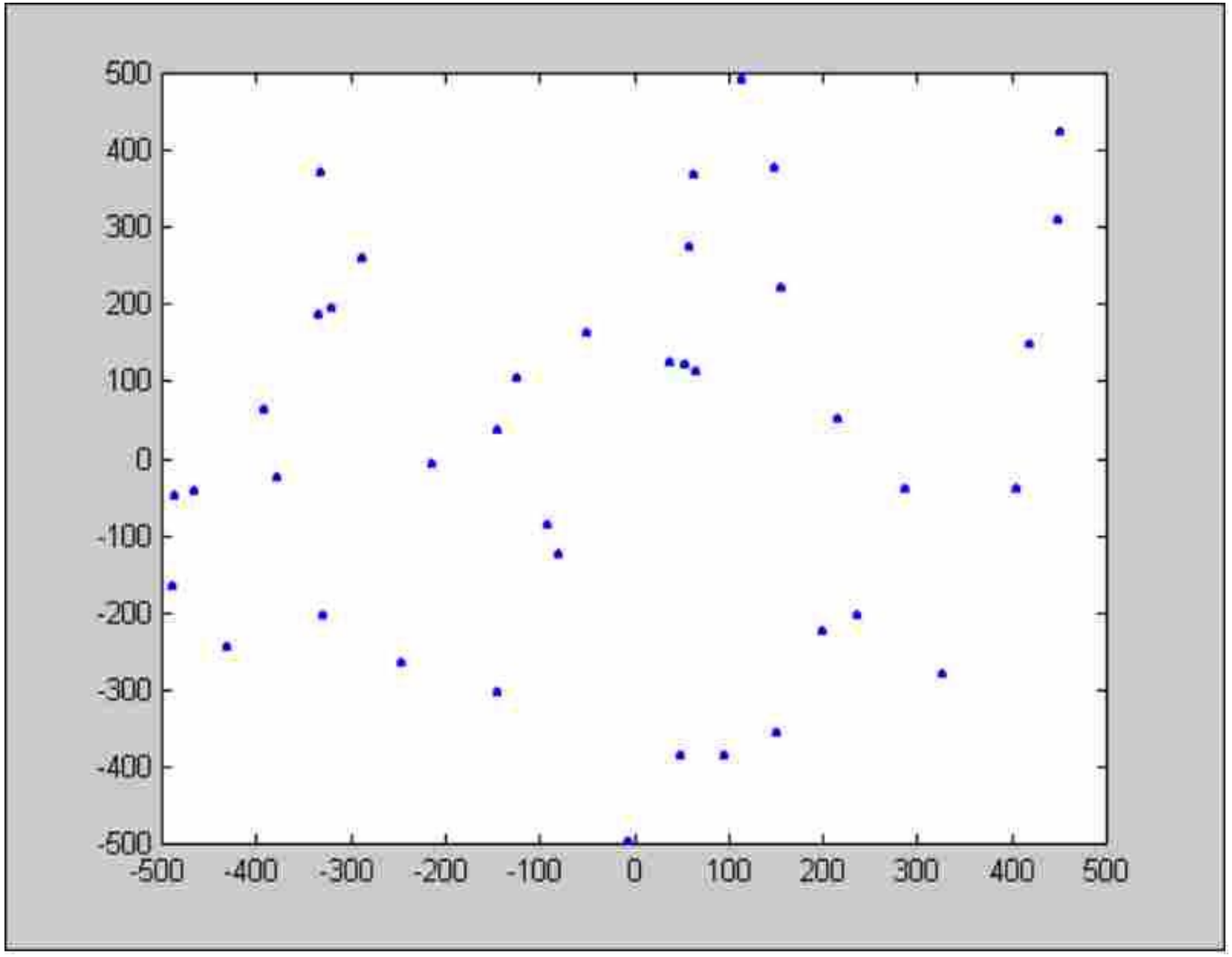
where

$$-500 \le x_i \le 500$$

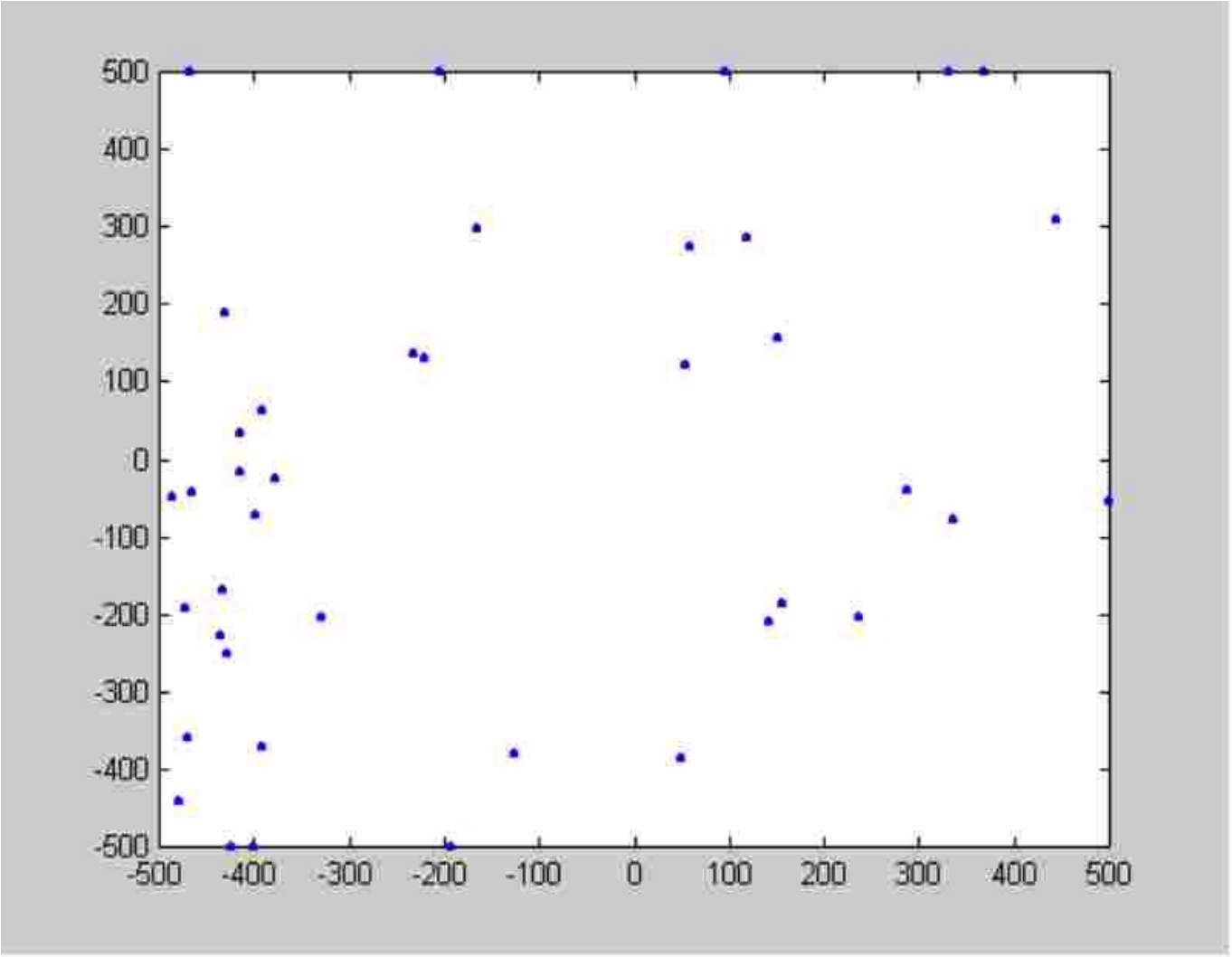
global minimum

$f(x) = 0,$ x_i=420.9687, i=1:n

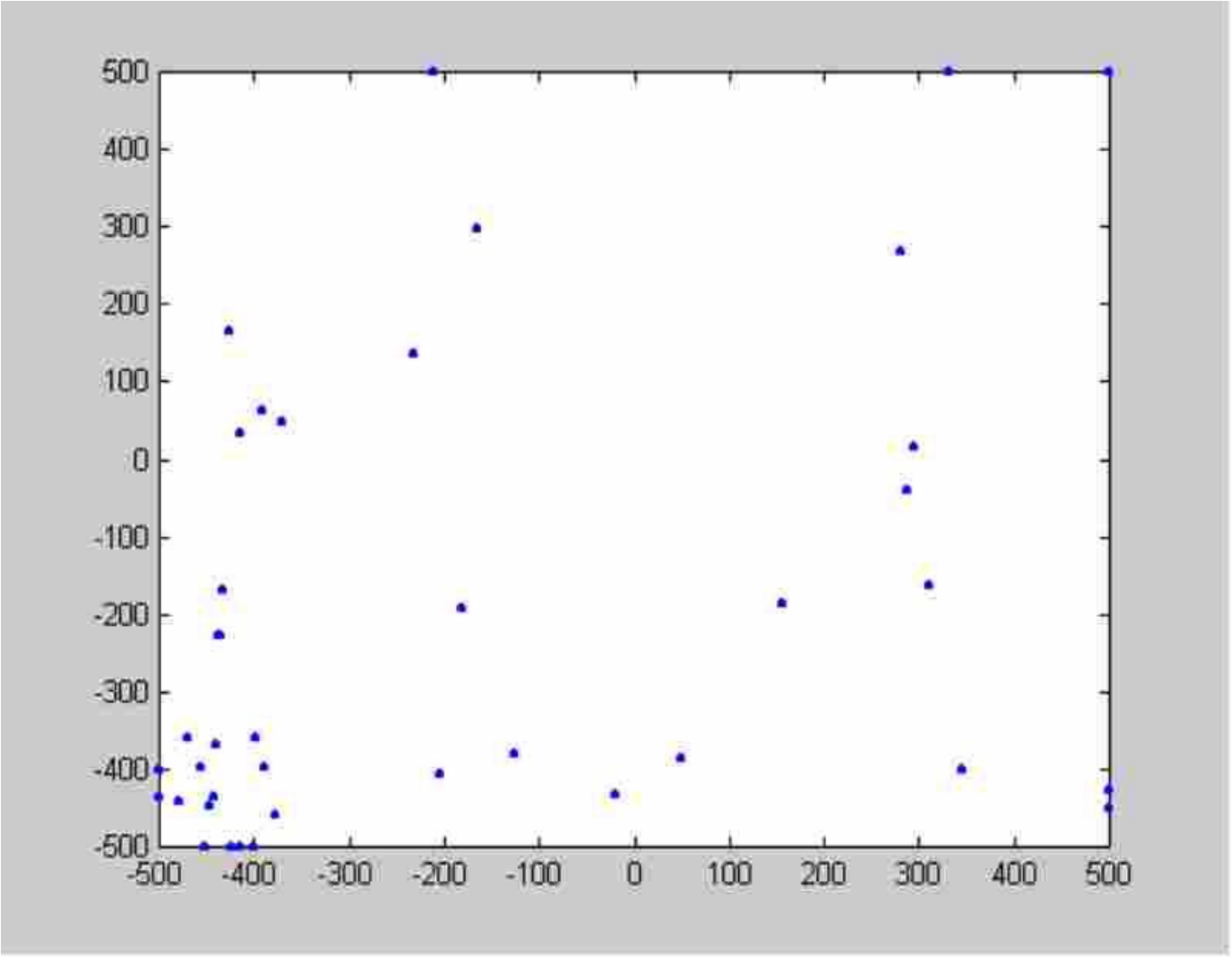
Initial Population



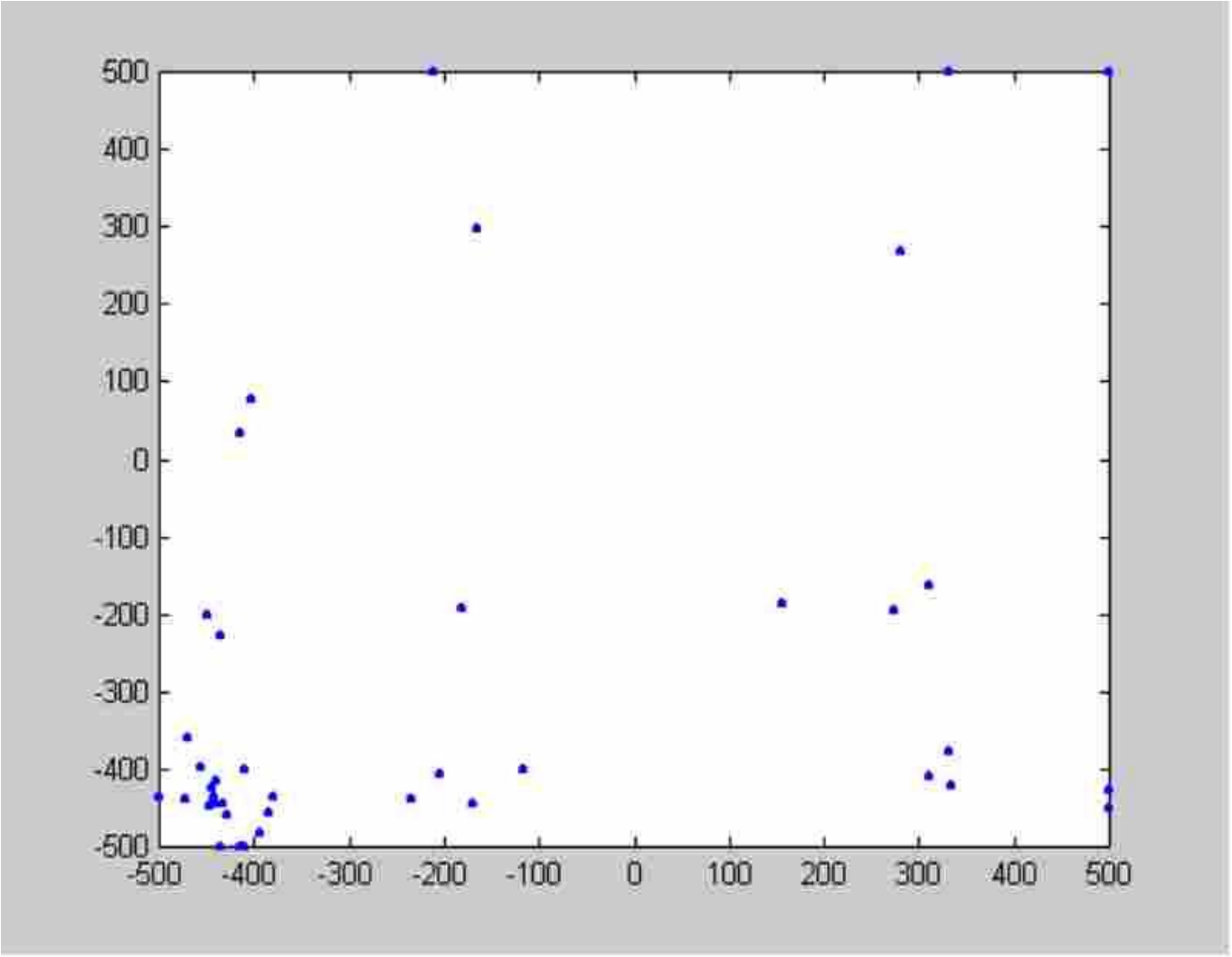
Search after 5 generations



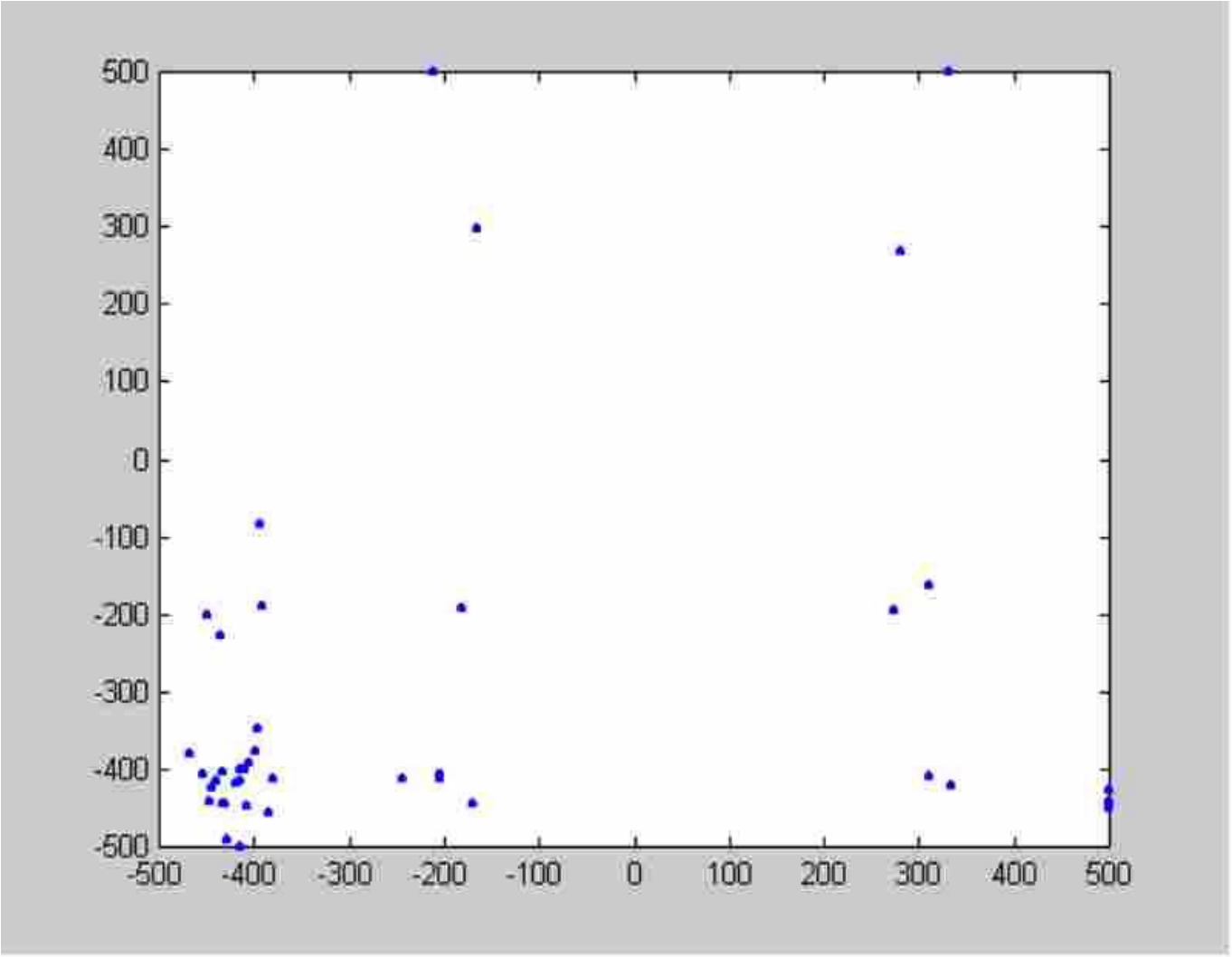
Search after 10 generations



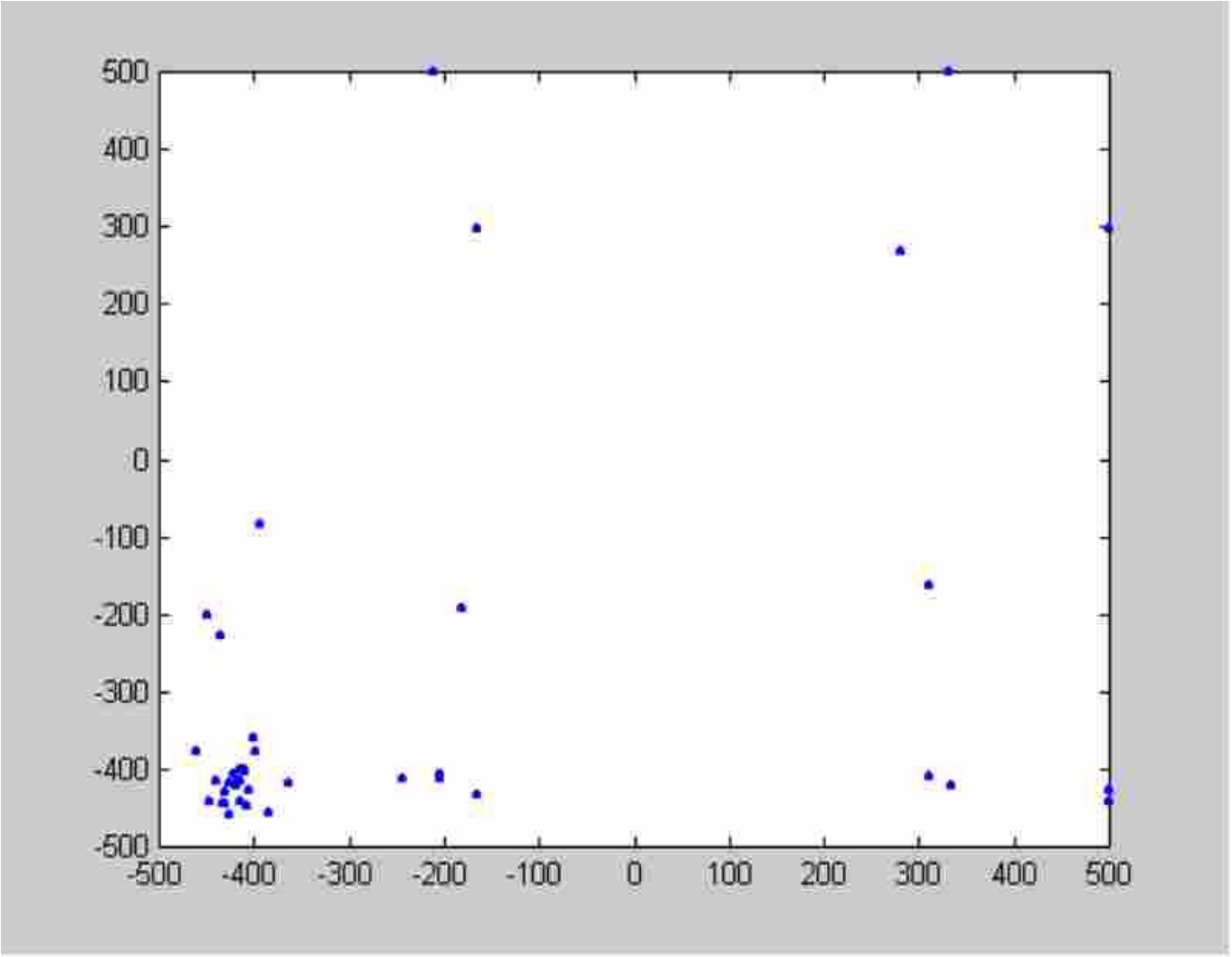
Search after 15 generations



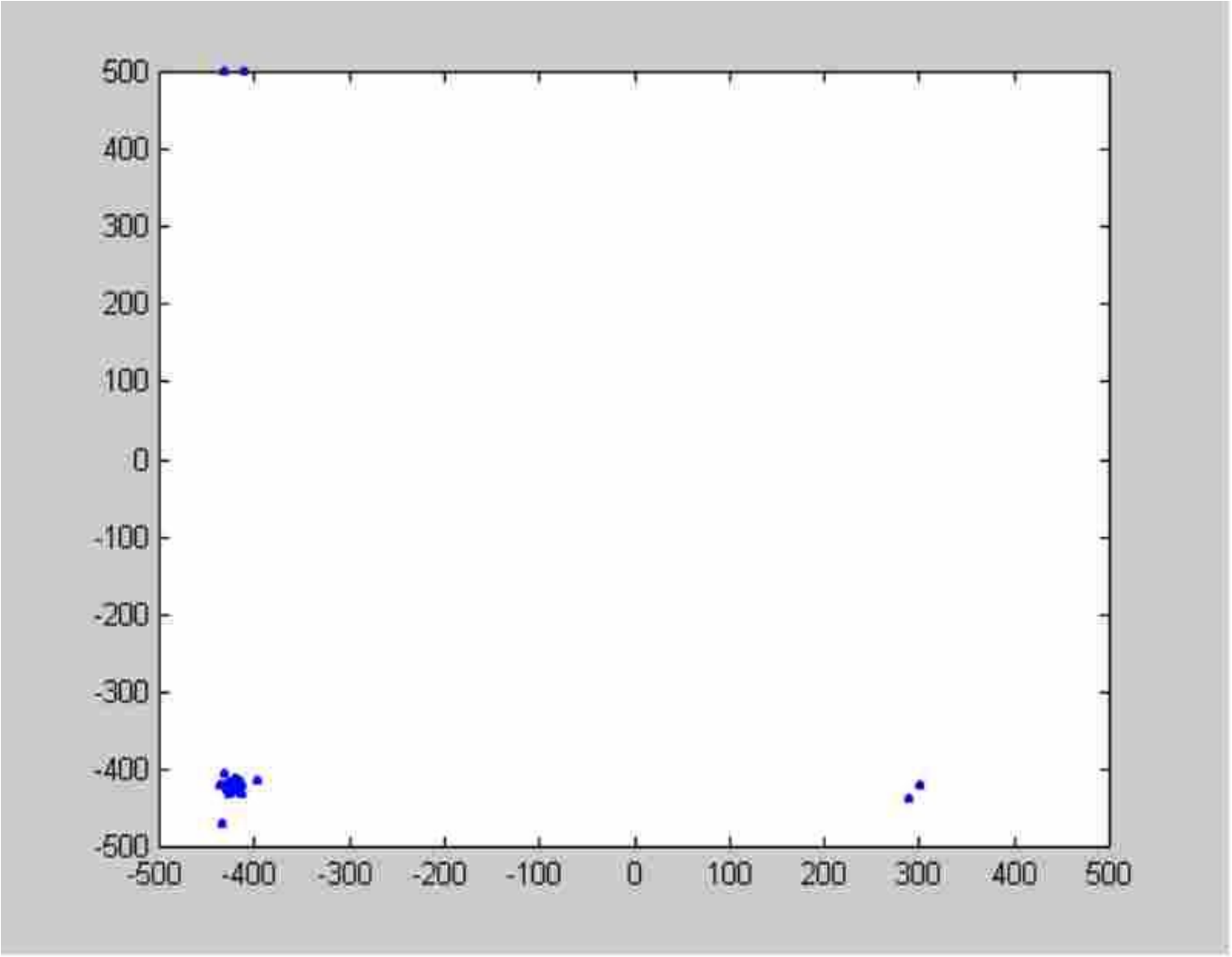
Search after 20 generations



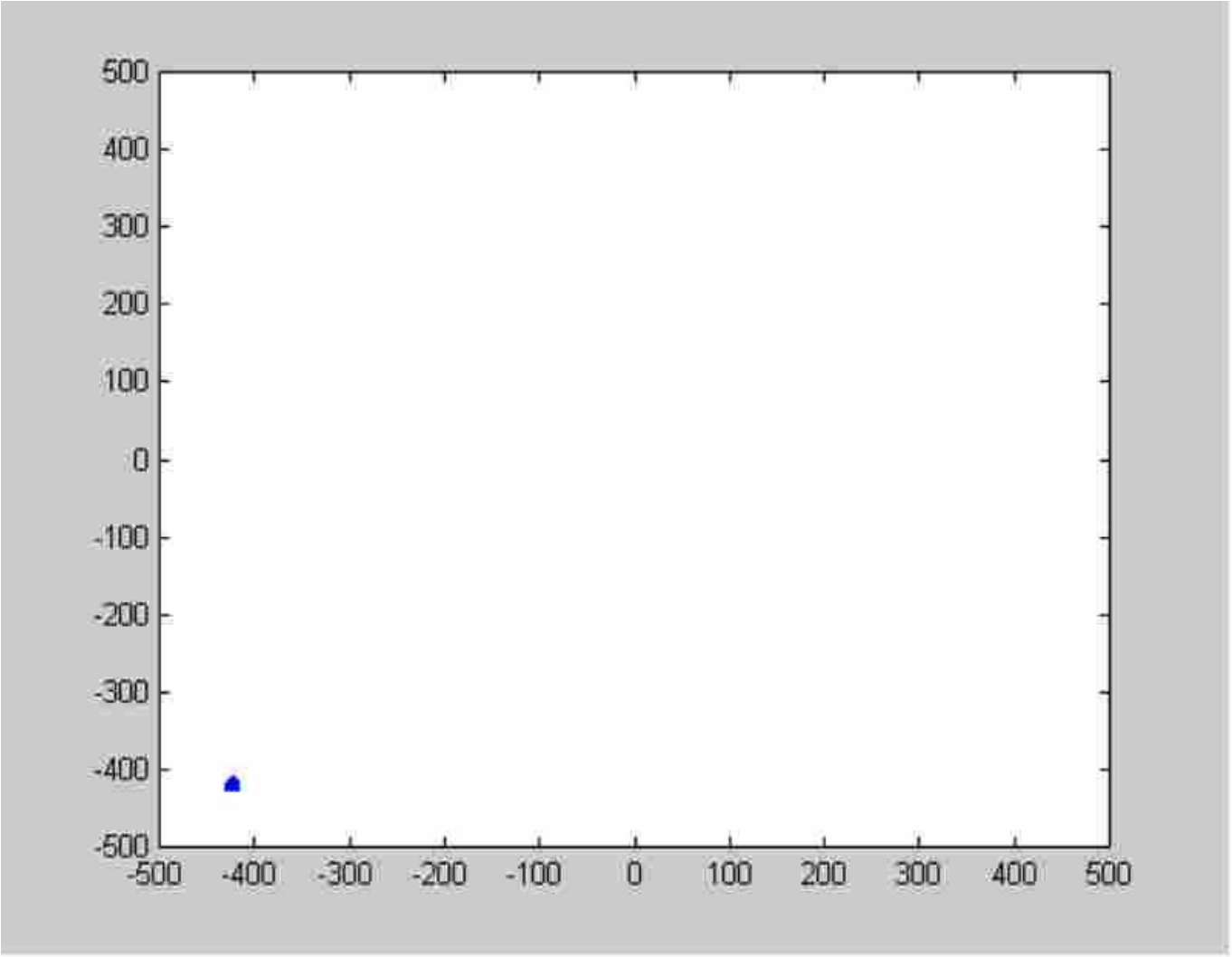
Search after 25 generations



Search after 100 generations



Search after 500 generations



PSO需要後續探討的問題

1. 在演化末期收斂速度會變的很緩慢(slow convergence at later stage)。
2. 演化進行需要較長的時間，特別是對於複雜的問題可能要花費的時間更久才能找出解答。 (longer computation time, particularly for complex optimization problems)
3. 如何在鑽研搜尋(Exploitation search)以及探索收尋(Exploration search)之間取得平衡點仍是一個重要且尚未能完全解決的問題。(balance between exploitation and exploration search is an important but not yet solved problem)
4. 對於參數的調整非常敏感。(sensitive to parameters)

Results

Gen. no.	Results
0	416.245599
5	515.748796
10	759.404006
15	793.732019
20	834.813763
100	837.911535
5000	837.965771
最佳解	837.9658