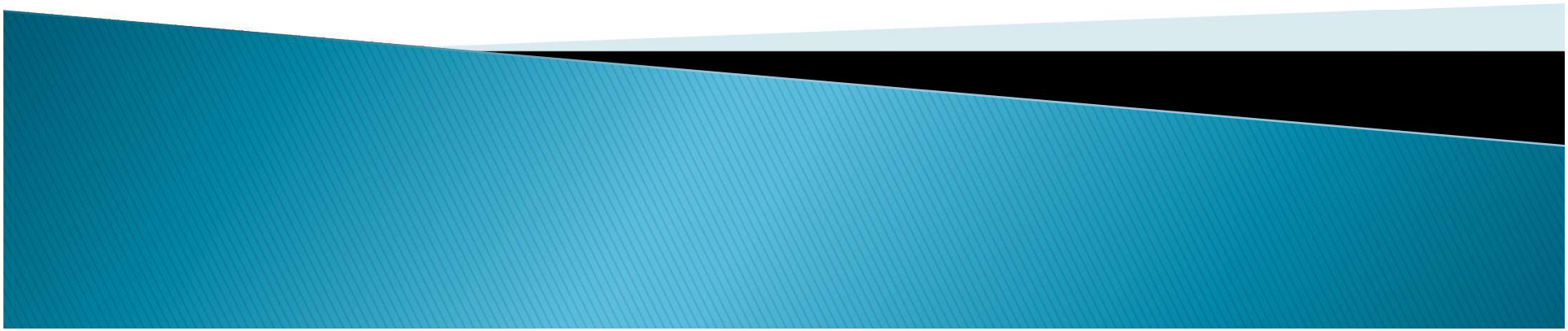
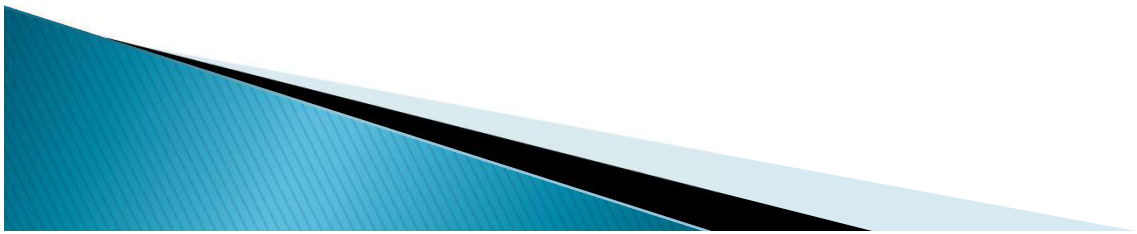
Particle Swarm Optimization

بهینه سازی جمعیت ذرات



Outline

- Introduction
- Particle swarm optimization
- PSO algorithm
- PSO solution update in 2-D
- Example
- BPSO



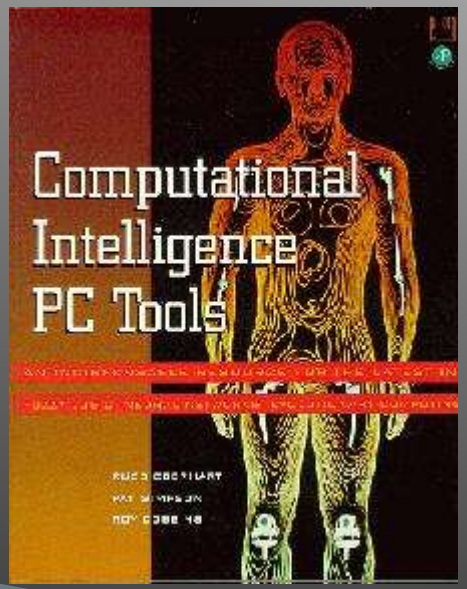
Introduction

- Particle Swarm Optimization(PSO)
  - Proposed by James Kennedy & Russell Eberhart in 1995
  - Inspired by social behavior of birds and fishes
  - Combines self-experience with social experience
  - Population-based optimization

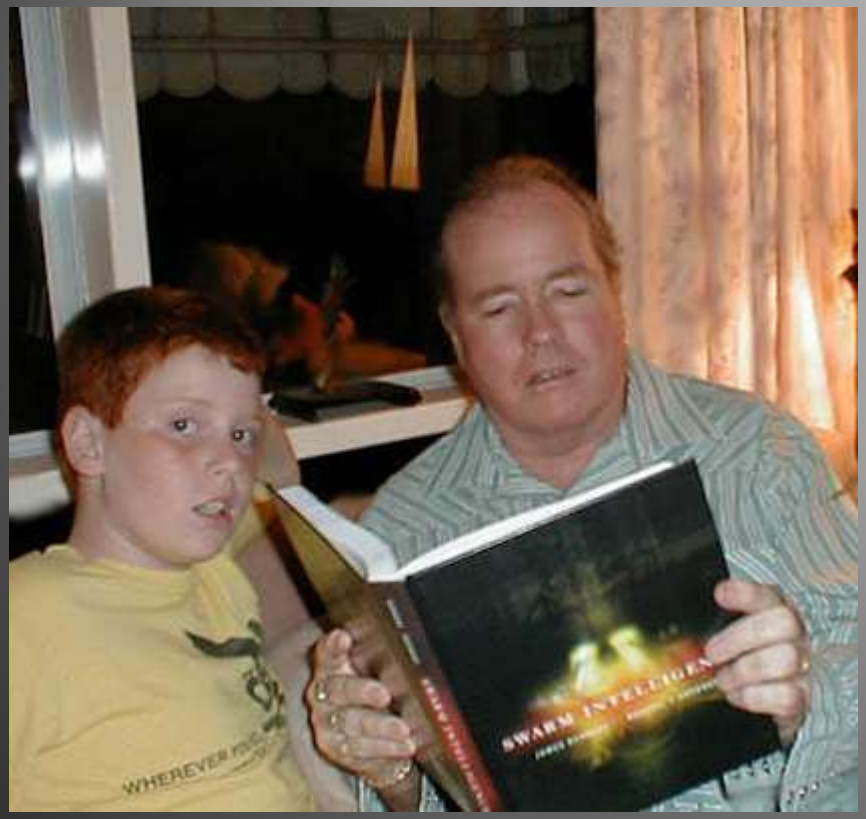
The “inventors”

Russell Eberhart

eberhart@engr.iupui.edu



The “inventors”



J. Kennedy, R.C. Eberhart, Particle swarm optimization, Proceedings of IEEE International Conference on Neural Networks 4, 1995.

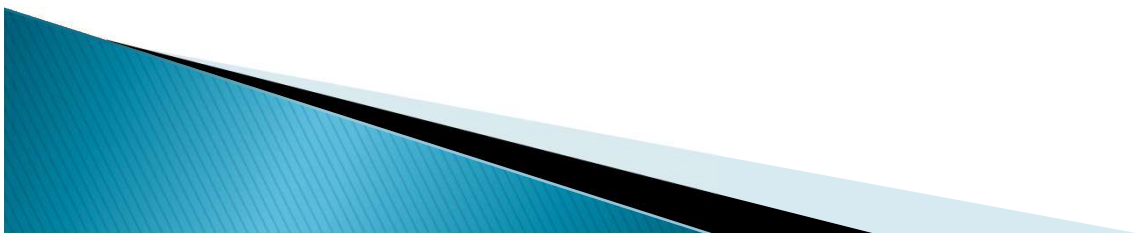
James Kennedy

Kennedy_Jim@bls.gov

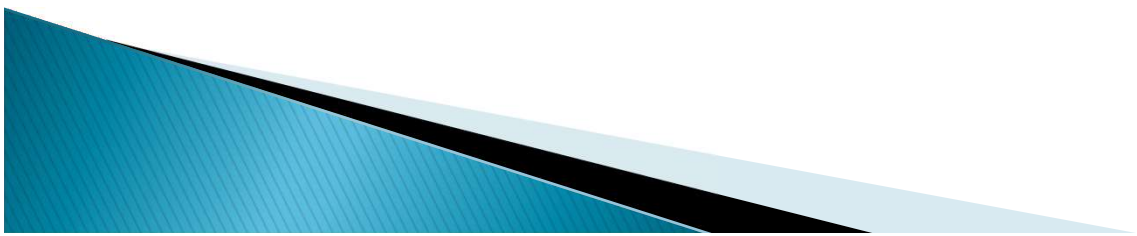
Concept

- Uses a number of 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



- رفتار گروهی در گونه های مختلف حیوانات قابل مشاهده است. بدون داشتن سردسته، کل گروه از درون سامان دهی می‌شود مثل انواعی از پرندگان و ماهیها.
- بهینه یاب جمعیت ذرات PSO در سال ۱۹۹۵ با الهام از رفتار پرندگان و ماهیها معرفی شد.
- بر پایه مدل فیزیولوژیکی تاثیرات و یادگیری اجتماعی می‌باشد.
- جیمز کندی، روانشانس اجتماعی و راسل سی ابرهارت، مهندس برق، صاحبان اصلی ایده‌ی الگوریتم PSO می باشند. در این الگوریتم اعضا رفتارهای بسیار ساده ای را دنبال می‌کنند اما نتیجه‌ای که در نهایت حاصل می‌شود کشف مناطق بهینه در یک فضای جستجوی بعد بالا است.
- الگوریتم بهینه سازی جمعیت ذرات از مجموعه ای از ذره ها تشکیل یافته است. هر ذره جوابی از مسئله است.



Optimization problem

- فرض کنید که هدف از بهینه‌سازی، پیدا کردن بیشینه تابع f در یک دامنه مشخص باشد:

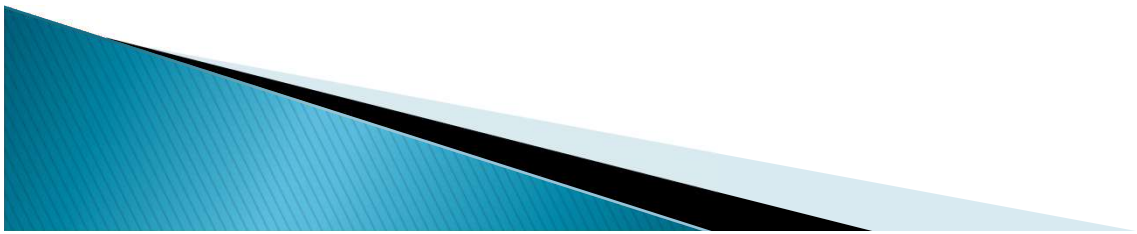
$$f\left(x^1, x^2, \ldots, x^m\right)$$

$$x^{d,low} \le x^d \le x^{d,hi} \quad \text{for } d = 1,2,\ldots,m$$

- در این وضعیت، پیدا کردن مقادیری برای متغیرهای x_1 تا x_m مد نظر است که تابع f، به ازای آنها بیشترین مقدار را به خود بگیرد.

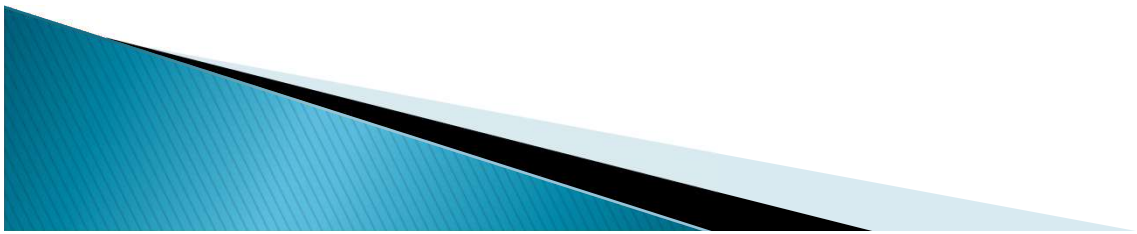
- به عبارتی هدف از بیشنه‌سازی، یافتن X^* است به گونه‌ای که

$$f(X^*) \ge f(X) \quad , \quad \forall X \in D_f$$



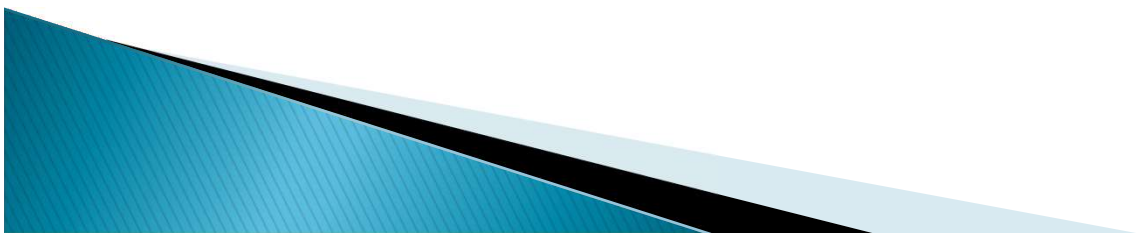
Particle Swarm Optimization

- Swarm: a set of particles (N)
- Particle: a potential solution
  - Position: $\mathbf{x}_i = (x_i^1, x_i^2, ..., x_i^m)$
  - Velocity: $\mathbf{v}_i = (v_i^1, v_i^2, ..., v_i^m)$
- Each particle maintains
  - Individual best position (PBest)
- Swarm maintains its global best (GBest)

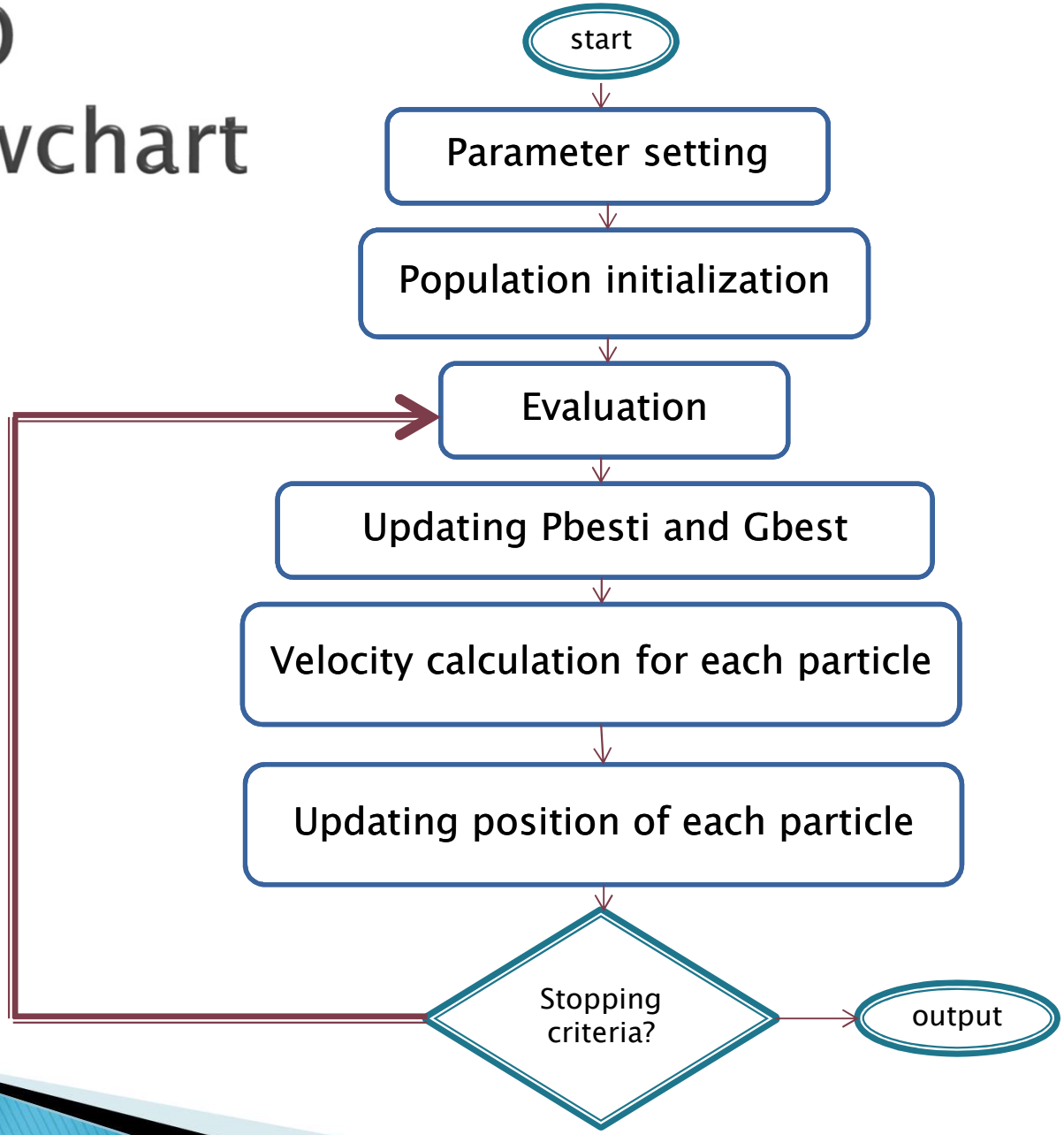


PSO Algorithm

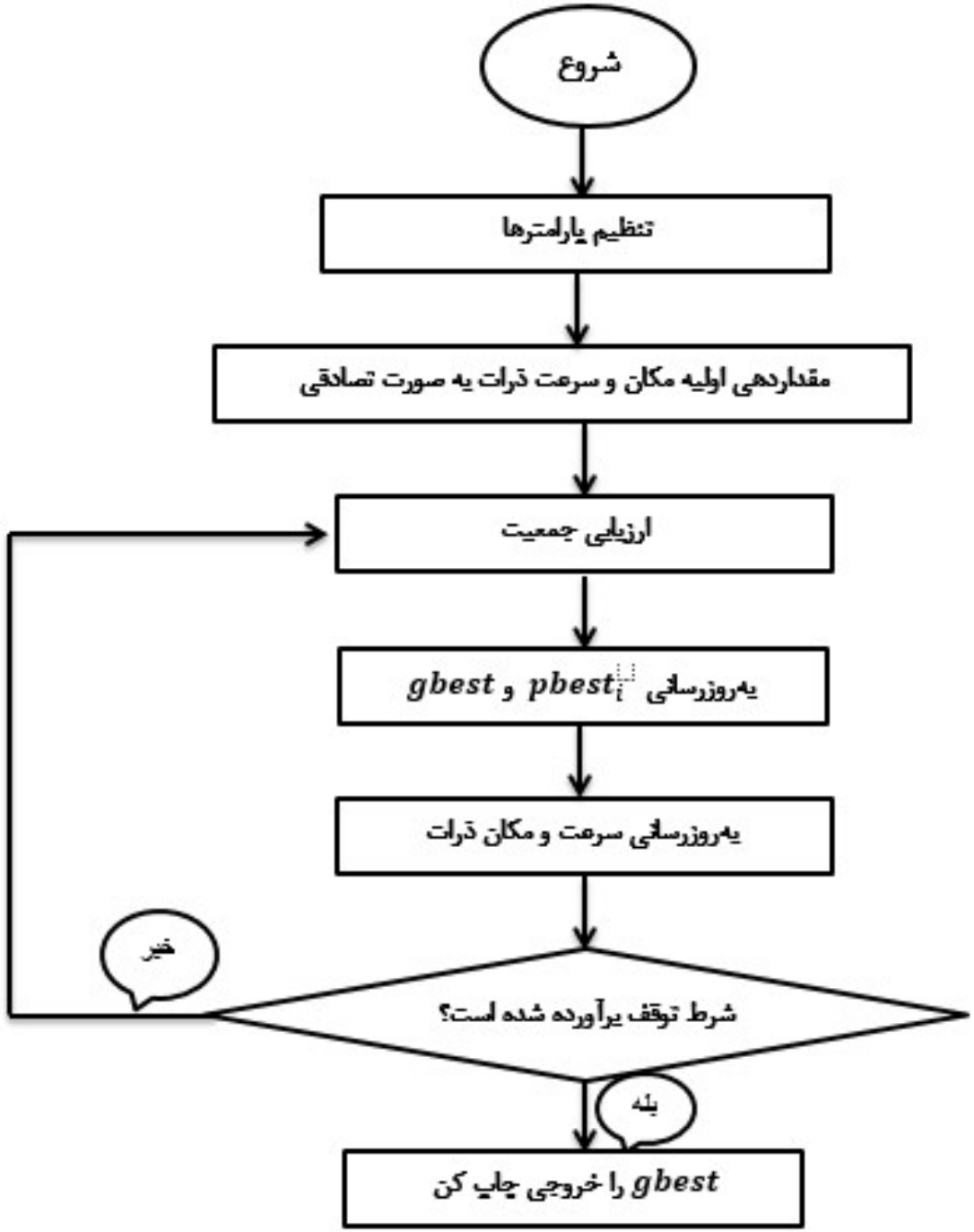
- Basic algorithm of PSO
  1. Initialize the swarm form the solution space
  2. Evaluate the fitness of each particle
  3. Update individual and global bests
  4. Update velocity and position of each particle
  5. Go to step2, and repeat until termination condition



PSO flowchart



PSO flowchart



Initialization

- موقعیت اولیه ذرات

موقعیت اولیه هر ذره به صورت تصادفی در فضای جستجو با یک توزیع یکنواخت در محـدوده تعریـف مسئله تعیین می‌شود

$x^{d,hi}$ و $x^{d,low}$ به ترتیب حد پایین و حد بالا از بعد d ام فضای جستجو می‌باشند.

$$x_i^d(t=1) \sim U(x^{d,low}, x^{d,hi})$$

$$x_i^d = x_i^{d,lo} + \left(x_i^{d,hi} - x_i^{d,lo}\right) \times x_i^n \qquad \text{for } d = 1, 2, \ldots, m$$

$$i = 1, 2, \ldots, N$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_N \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 \ldots\ldots\ldots x_1^m \\ x_2^1 & x_2^2 \ldots\ldots\ldots x_2^m \\ & \cdot \\ & \cdot \\ & \cdot \\ x_N^1 & x_N^2 \ldots\ldots\ldots x_N^m \end{bmatrix}$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_N \end{bmatrix} = \begin{bmatrix} 0.8 & 3 & 0.1 \ldots\ldots 10 \\ 1.2 & 2.9 & 0.3 \ldots\ldots 8 \\ & & \cdot \\ & & \cdot \\ & & \cdot \\ 0.7 & 3.2 & 0.2 \ldots\ldots 11 \end{bmatrix}$$

Initialization

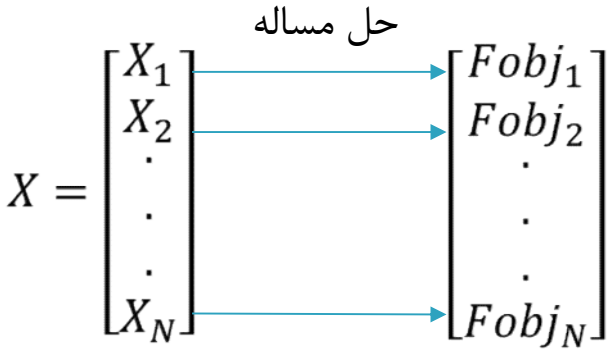
- سرعت اولیه ذرات

سرعت اولیه ذرات یا ۰ تنظیم می شود. یا اینکه مقادیر تصادفی حقیقی به آنها نسبت داده می شود.

$$V=\begin{bmatrix} V_1 \\ V_2 \\ \cdot \\ \cdot \\ \cdot \\ V_N \end{bmatrix}=\begin{bmatrix} v_1^1 & v_1^2 \dots\dots\dots v_1^m \\ v_2^1 & v_2^2 \dots\dots\dots v_2^m \\ & \cdot \\ & \cdot \\ & \cdot \\ v_N^1 & v_N^2 \dots\dots\dots v_N^m \end{bmatrix}$$

Evaluation

- در هر تکرار، به ازای هر عضو یکبار مساله حل می شود.



Update Gbest,Pbest

- به روزسانی:
- بهترین موقعیتی که کل جمعیت تا کنون (تا لحظه t) کشف کرده است. gbest(t)
- بهترین موقعیتی است که ذره i تا کنون (تا لحظه t) به آن دست پیدا کرده است.$Pbest_i(t)$

- $gbest = [g^1\ g^2 \ldots g^m]$
- $pbest_i = [p_i^1\ p_i^2 \ldots p_i^m]$
- $pbest = \begin{bmatrix} pbest_1 \\ pbest_2 \\ \cdot \end{bmatrix} = \begin{bmatrix} p_1^1 & p_1^2 & \ldots & p_1^m \\ p_2^1 & p_2^2 & \ldots & p_2^m \\ & \cdot & & \end{bmatrix}$

$$\text{pbest}(t) = \begin{bmatrix} pbest_1(t) \\ pbest_2(t) \\ \cdot \\ \cdot \\ \cdot \\ pbest_N(t) \end{bmatrix}$$

Velocity calculation

- به روز رسانی سرعت

$$v_i^d(t+1) = w(t)v_i^d(t) + c_1 r_1(t)[pbest_i^d - x_i^d(t)] + c_2 r_2(t)[gbest^d - x_i^d(t)]$$

$$V = \begin{bmatrix} V_1 \\ V_2 \\ \cdot \\ \cdot \\ \cdot \\ V_N \end{bmatrix} = \begin{bmatrix} v_1^1 & v_1^2 \dots\dots\dots v_1^m \\ v_2^1 & v_2^2 \dots\dots\dots v_2^m \\ & \cdot \\ & \cdot \\ & \cdot \\ v_N^1 & v_N^2 \dots\dots\dots v_N^m \end{bmatrix}$$

Velocity calculation...

c_1و c_2 ثوابت مثبتی هستند که برای وزن دهی به اجـزا خـودی و جمعـی استفاده می‌شود و ضرایب شتاب نامیده می‌شوند

- Acceleration constant.

- r_1 و r_2 اعداد تصادفی با توزیـع یکنواخـت بـین صـفر تـا یـک بـوده، خاصیت تصادفی بودن الگوریتم را حفظ می‌کنند.

$$r_1(t), r_2(t) \sim U(0,1)$$

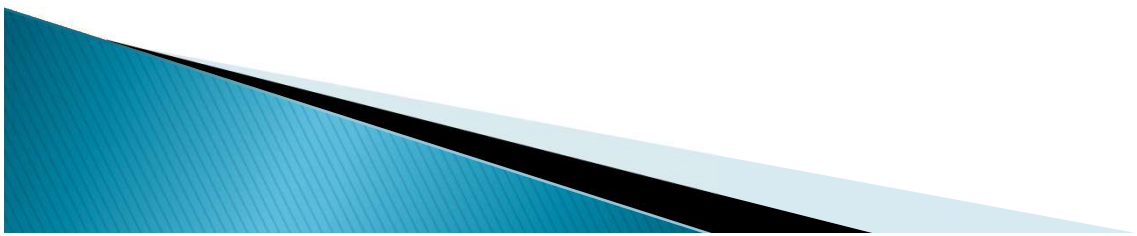
- w پارامتر وزن اینرسی می‌باشد.

- Inertia weight.

inertia factor

- inertia factor (w) is decreasing linearly from 0.9 to 0.2

$$w(t) = w_{max} - \frac{w_{max} - w_{min}}{T} \times t$$



Velocity calculation...

- این بردار سرعت است که ذرات را هدایت می‌کند. در بردار سرعت، هم نتیجه تجربه اجتماعی ذره های همسایه و هم تجربه فردی هر ذره دخیل است. در تعریف بردار سرعت، به جزیی که از دانش به دست آمده از تجربه شخصی استفاده می‌کند، جزء فردی و به قسمتی که از تجربیات همسایه ها استفاده می‌کند، جزء اجتماعی گفته می‌شود. هر ذره سرعت خود را با ترکیب خطی این دو به روز رسانی می‌کند.
- در جزء فردی از بهترین موقعیتی که ذره تا کنون به آن دست پیدا کرده و در جزء اجتماعی از بهترین موقعیتی که کل ذرات تا کنون بدان دست پیدا کرده اند، استفاده می‌شود
  - Cognitive component.
  - Social component.

Position updating

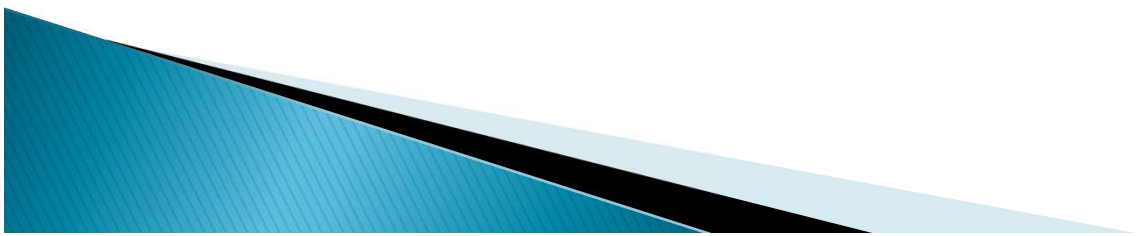
- به روزرسانی موقعیت

در هر لحظه، موقعیت هر ذره بر مبنای تجربه خودش و نیز همسایگانش تنظیم می‌شود.
اگر $x_i^d(t)$ موقعیت بعد d ذره i در زمان t باشد،
و $v_i^d(t)$ بعد d سرعت ذره i در زمان t باشد،
موقعیت ذره با جمع شدن با سرعت به روزرسانی می شود.

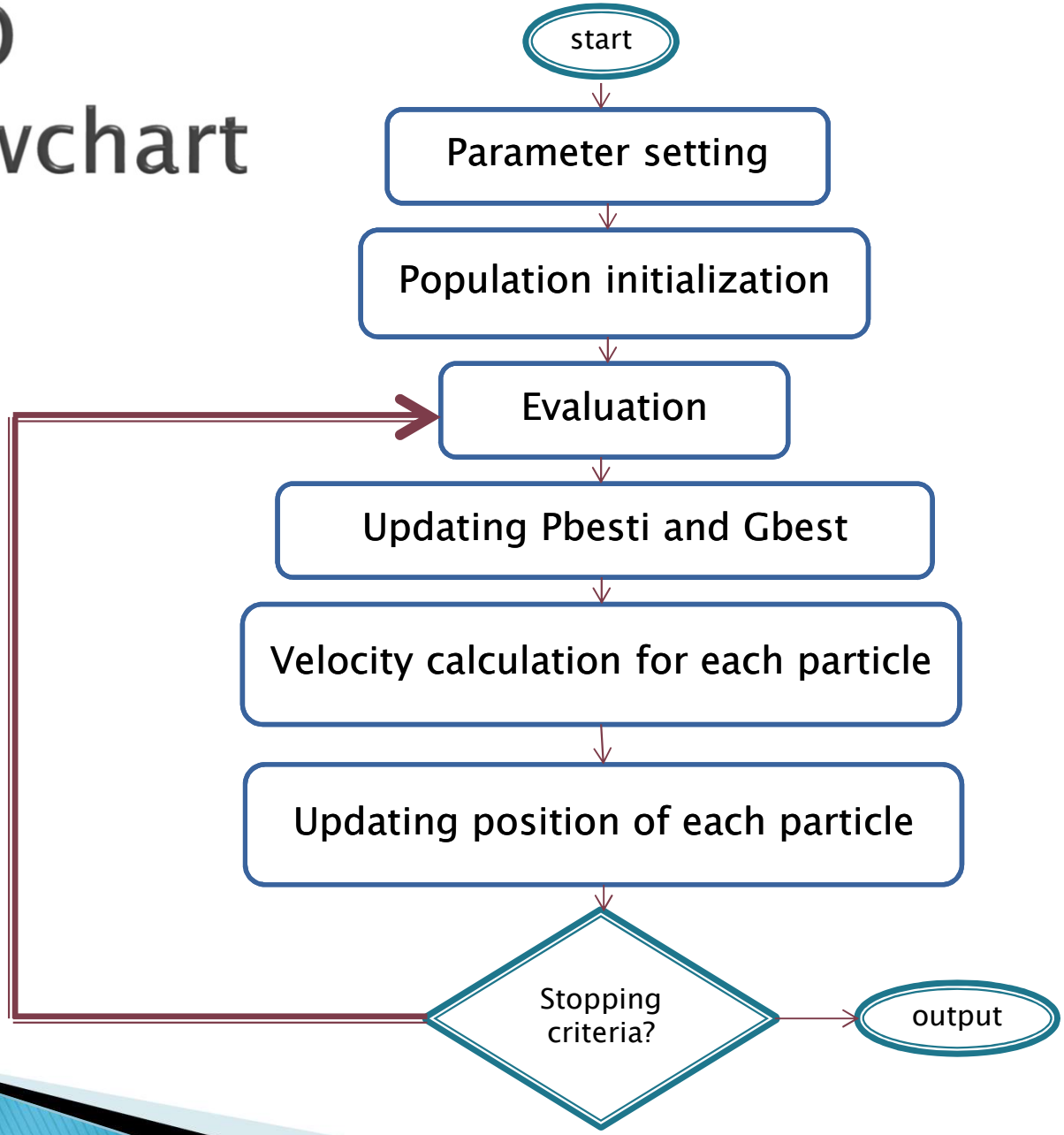
$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1)$$

Stopping criteria

- شرط توقف.
- الف- توقف پس از تعداد تکرارهای مشخص: در این مورد الگوریتم پس از تعداد مشخصی از تکرارها متوقف می‌شود.
- ب- توقف پس از رسیدن به یک دقت معین: در این مورد الگوریتم پس از کشف جواب با یک دقت معین متوقف می‌شود.
- ج- توقف پس از رکود جمعیت: الگوریتم پس از این که با تکرار آن هیچ بهبودی مشاهده نشود متوقف می‌شود.
- د- توقف پس از متمرکز شدن جمعیت حول بهینه موجود: تحت این شرط، چنانچه الگوریتم پس از اینکه فاصله دورترین ذره از بهینه کشف شده از حد معینی کمتر شود، متوقف می‌شود. این حد به صورت مقیاسی از ابعاد فضا قابل تعریف است.
- ه- توقف پس از رشد کم تابع هدف در محل بهینه کشف شده: در این حالت پس از اینکه نرخ تغییر مقدار بهینه در دو تکرار متوالی از حد ϵ کمتر شود، الگوریتم متوقف می‌شود.



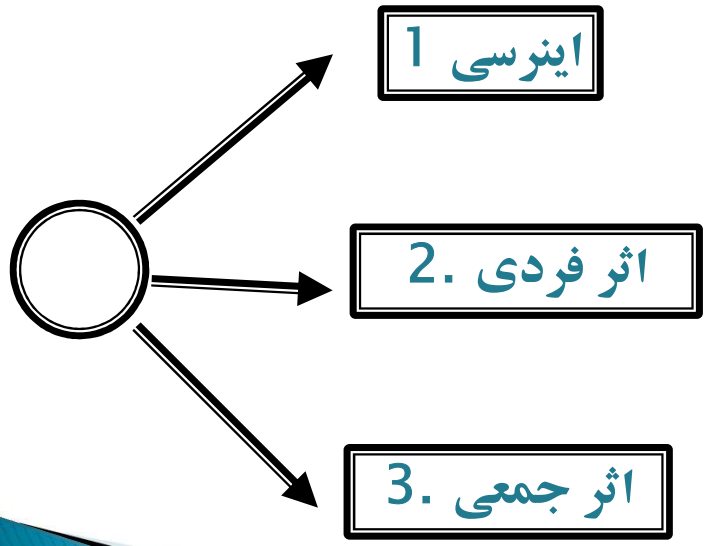
PSO flowchart



بررسی دقیق تر رابطه بروز رسانی سرعت ذرات

Particle's velocity:

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



- **Makes the particle move in the same direction and with the same velocity**
- **Improves the individual**
- **Makes the particle return to a previous position, better than the current**
- **Conservative**
- **Makes the particle follow the best neighbors direction**

کاوش و بهره وری در الگوریتم PSO

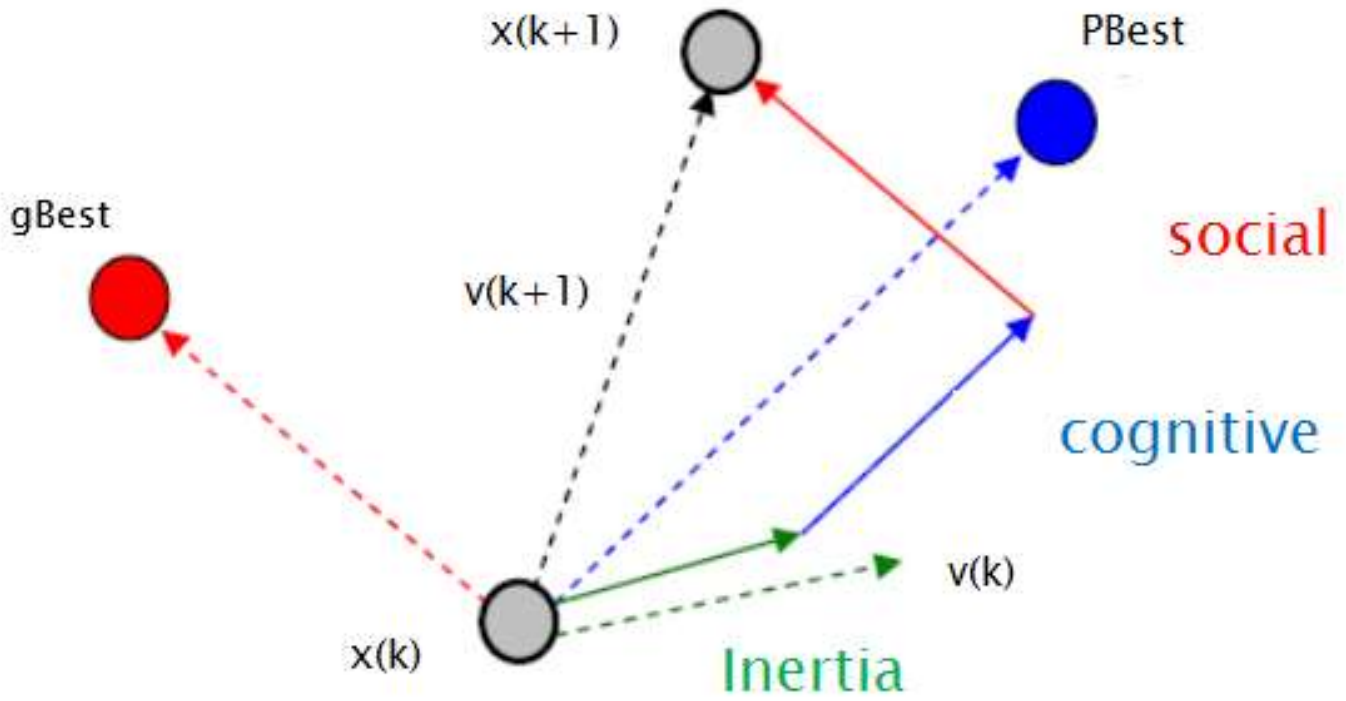
- **Diversification** کاوش (= **Exploration**): searches new solutions, finds the regions with potentially the best solutions
- **Intensification** بهره وری (= **Exploitation**): explores the previous solutions, finds the best solution of a given region.
- In PSO:

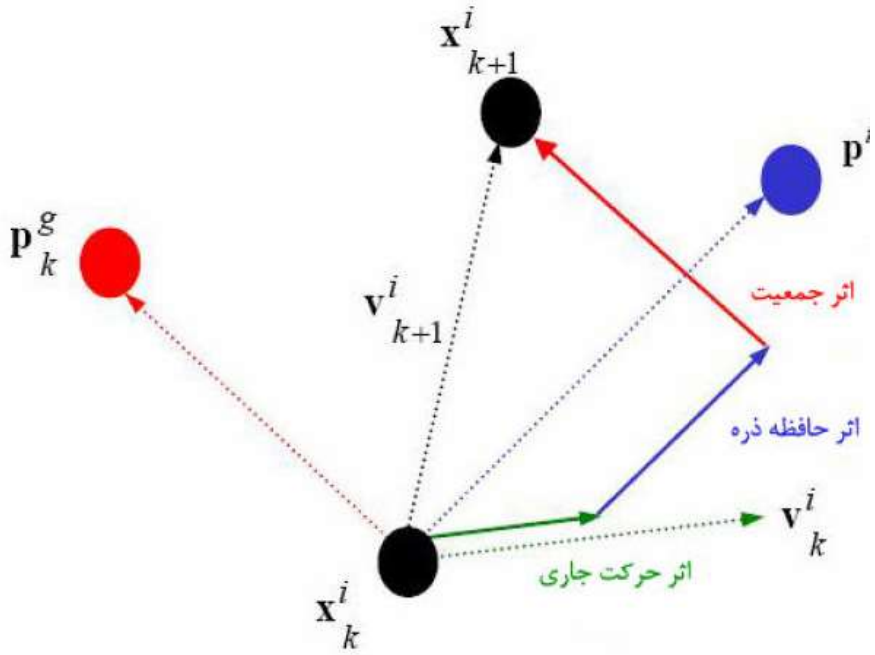
$$\mathbf{v}_i^{t+1} = \underbrace{\mathbf{v}_i^t}_{\text{Diversification}} + \underbrace{c_1 \mathbf{U}_1^t(\mathbf{pb}_i^t - \mathbf{p}_i^t) + c_2 \mathbf{U}_2^t(\mathbf{gb}^t - \mathbf{p}_i^t)}_{\text{Intensification}}$$

PSO Algorithm

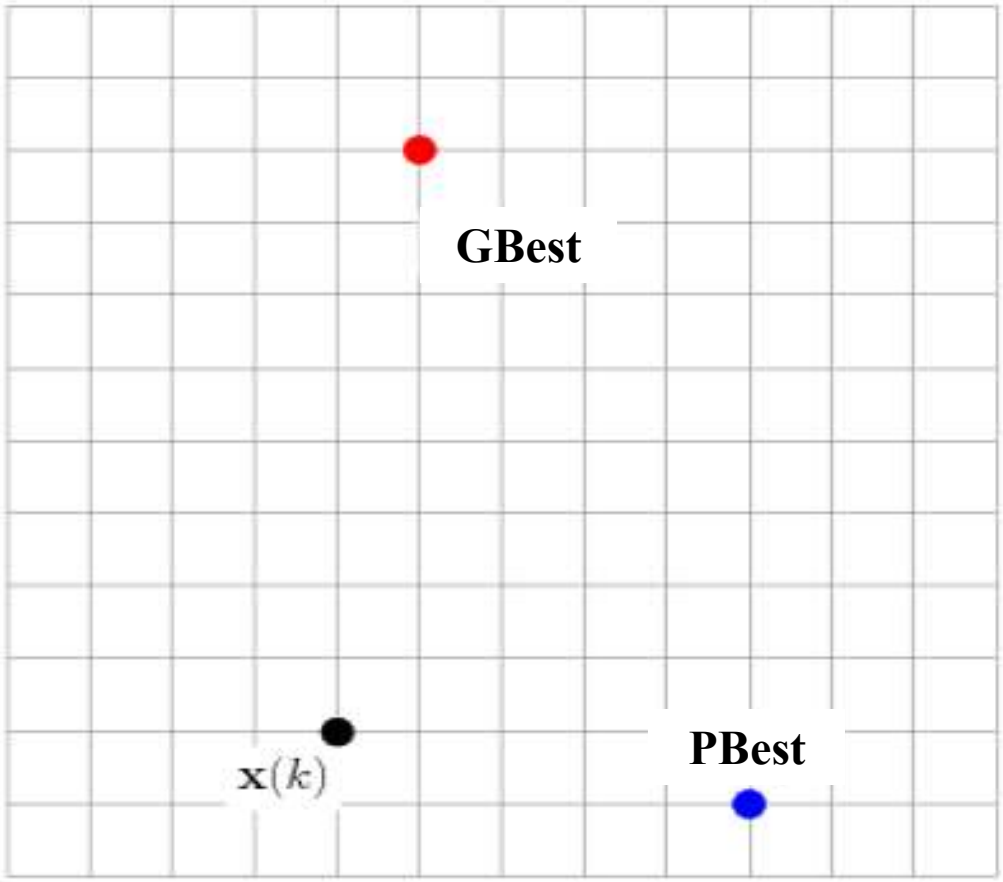
- Particle's velocity

$$\mathbf{v}_i(k+1) = \text{Inertia} + \text{cognitive} + \text{social}$$



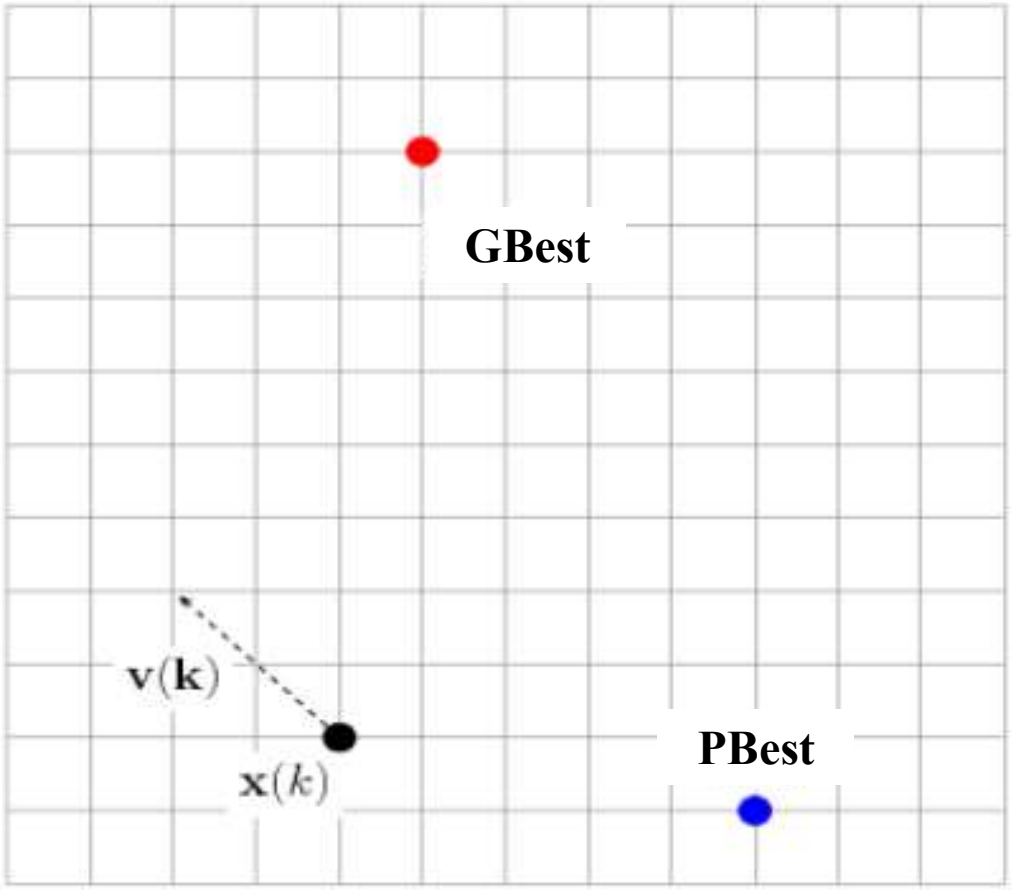

$\mathbf{x}^i_{k+1}$
$\mathbf{p}^g_k$
$\mathbf{v}^i_{k+1}$
اثر جمعیت
اثر حافظه ذره
$\mathbf{v}^i_k$
اثر حرکت جاری
$\mathbf{x}^i_k$

PSO solution update in 2D



- $\mathbf{x}(k)$ - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

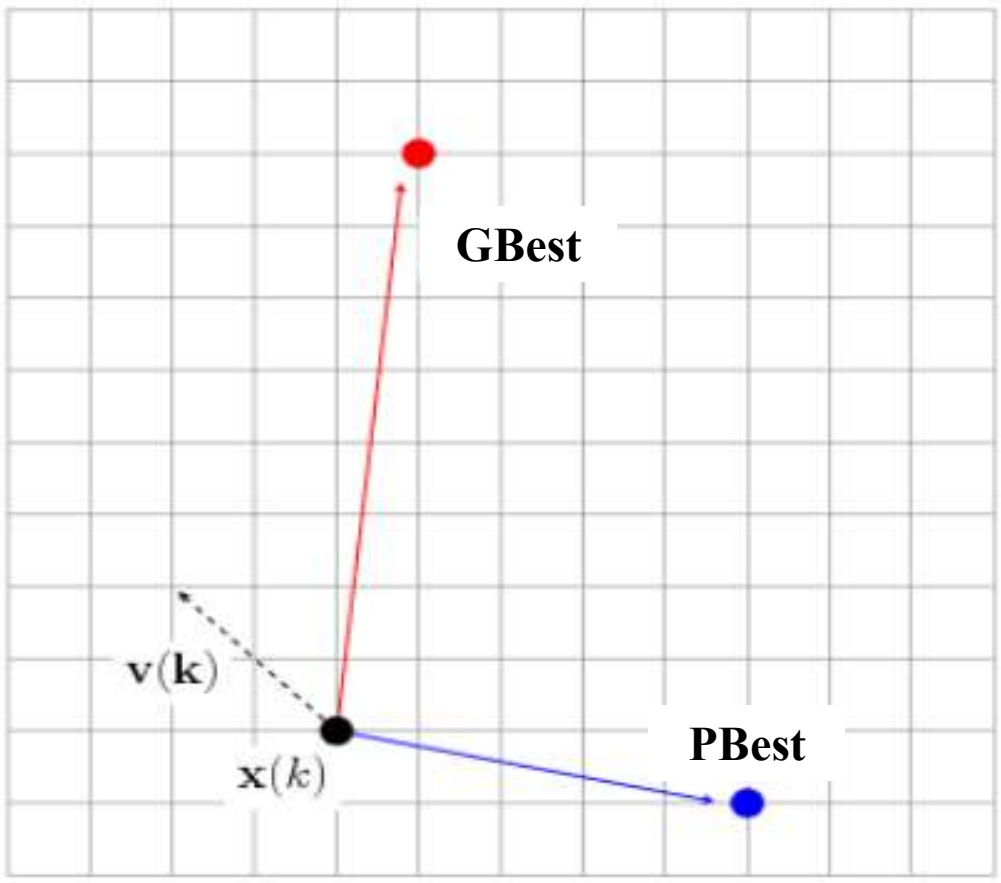
PSO solution update in 2D



Inertia: $\mathbf{v}(k)=(-2, 2)$

- $\mathbf{x}(k)$ - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

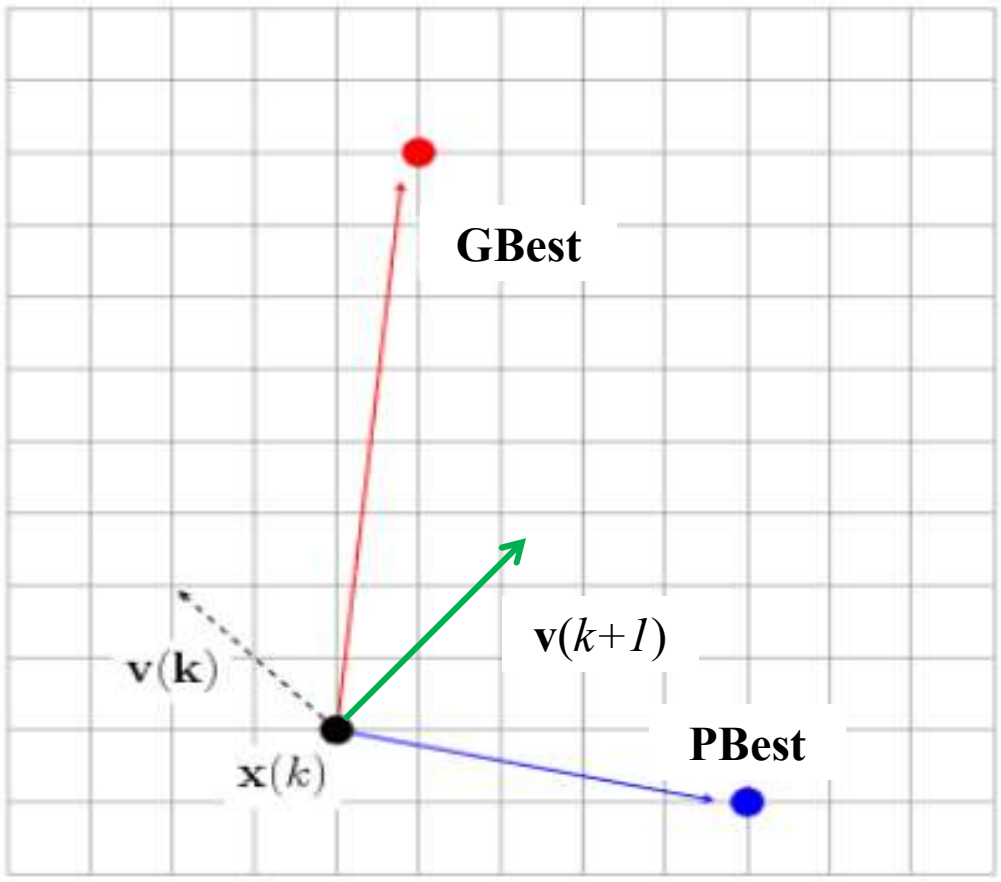
PSO solution update in 2D



- Inertia: $\mathbf{v}(k)=(-2,2)$
- Cognitive:
  PBest-$\mathbf{x}(k)=(9,1)-(4,2)=(5,-1)$
- Social:
  GBest-$\mathbf{x}(k)=(5,10)-(4,2)=(1,8)$

- $\mathbf{x}(k)$ - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

PSO solution update in 2D

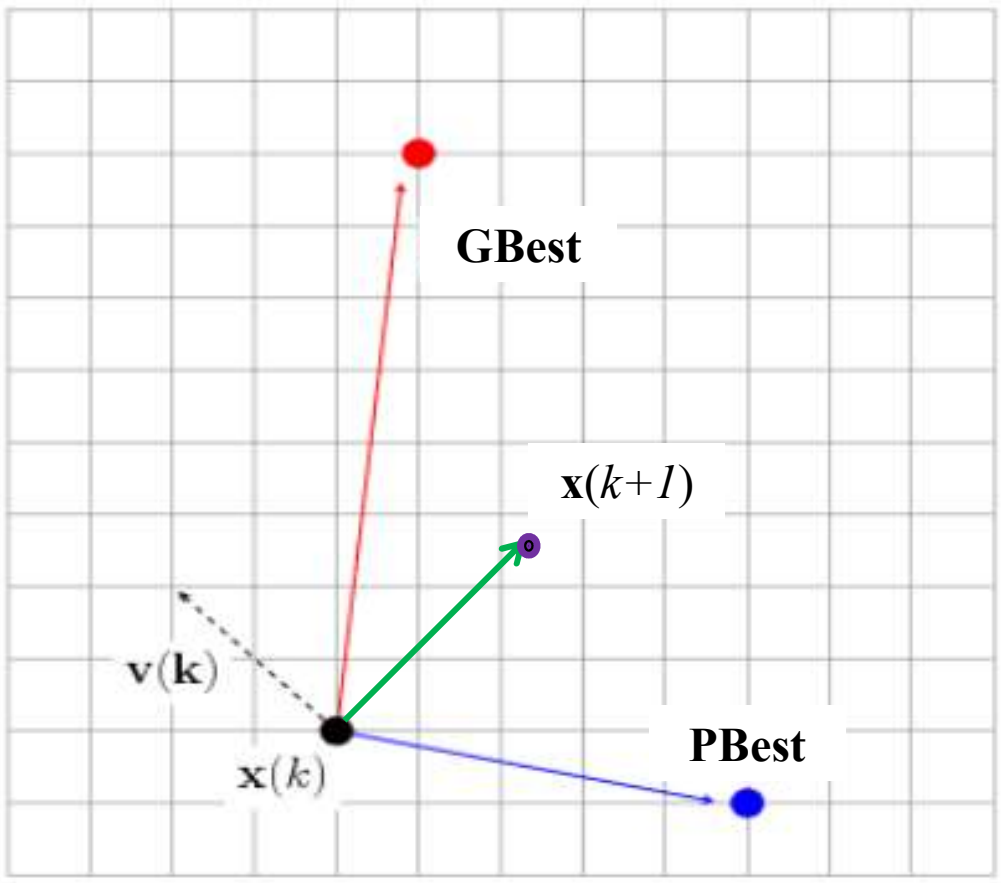


- Inertia: **v**(*k*)=(-2,2)
- Cognitive:
  PBest-**x**(*k*)=(9,1)-(4,2)=(5,-1)
- Social:
  GBest-**x**(*k*)=(5,10)-(4,2)=(1,8)

v(*k+1*)=(-2,2)+0.8*(5,-1)
+0.2*(1,8) = (2.2,2.8)

- **x**(*k*) - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

PSO solution update in 2D



- Inertia: $\mathbf{v}(k)=(-2,2)$
- Cognitive:
  PBest-$\mathbf{x}(k)=(9,1)-(4,2)=(5,-1)$
- Social:
  GBest-$\mathbf{x}(k)=(5,10)-(4,2)=(1,8)$
- $\mathbf{v}(k+1)=(2.2,2.8)$

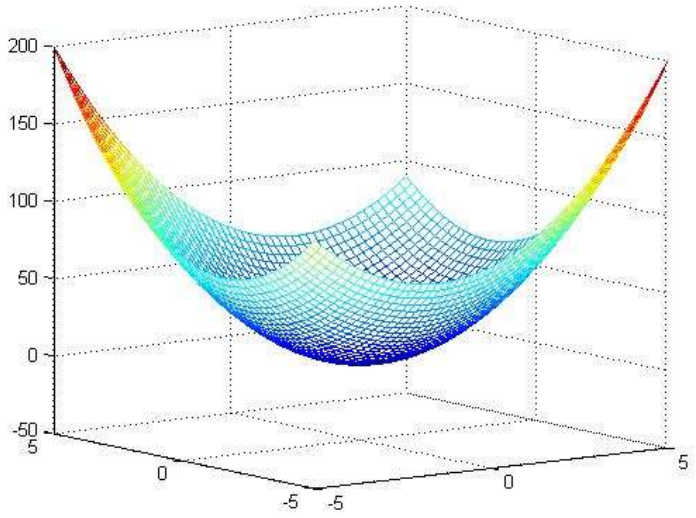
$\mathbf{x}(k+1)=\mathbf{x}(k)+\mathbf{v}(k+1)=$
$(4,2)+(2.2,2.8)=(6.2,4.8)$

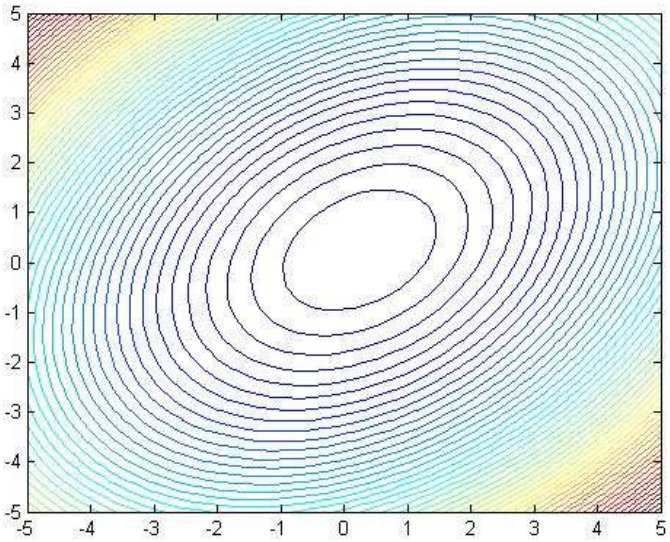
- $\mathbf{x}(k)$ - Current solution (4, 2)
- PBest - Particle's best solution (9, 1)
- GBest-Global best solution (5, 10)

Example

- Find the minimum of this function

$$f(\mathbf{x}) = 3x_1^2 - 2x_1x_2 + 3x_2^2 - x_1 - x_2$$





Example

$$\mathbf{x}_1 = \begin{bmatrix} 2.2824 & 0.6238 & 4.0005 & 3.1717 & -4.0058 \\ -0.4894 & -2.7580 & -2.7043 & -3.3118 & 1.5771 \end{bmatrix}$$

$$\mathbf{v}_1 = \begin{bmatrix} -0.6321 & 0.1712 & 0.6942 & 0.0264 & 0.2207 \\ 0.2133 & -0.5598 & -0.2500 & 0.6079 & 0.3122 \end{bmatrix}$$

$$\mathbf{x}_2 = \begin{bmatrix} 1.7767 & 1.4300 & 2.5656 & 2.2018 & 3.3541 \\ -0.3187 & -2.2903 & -0.3385 & 0.3199 & -0.5338 \end{bmatrix}$$

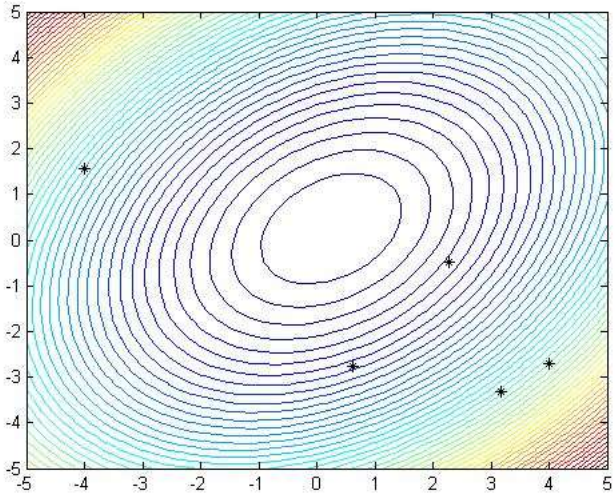
$$\mathbf{v}_2 = \begin{bmatrix} -0.5057 & 0.8063 & -1.4349 & -0.9700 & 7.3599 \\ 0.1706 & 0.4677 & 2.3657 & 3.6317 & -2.1109 \end{bmatrix}$$

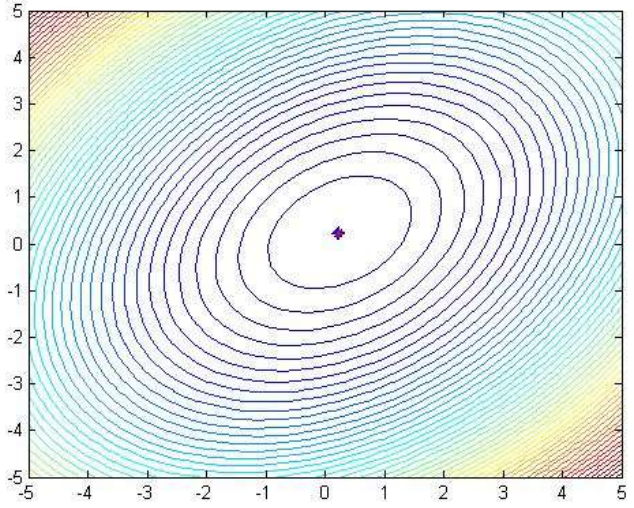
$$\mathbf{x}_3 = \begin{bmatrix} 1.3721 & 2.4464 & 1.0728 & 1.1350 & 7.9656 \\ -0.1822 & 0.1959 & 1.5627 & 2.7884 & -2.0485 \end{bmatrix}$$

$$\mathbf{v}_3 = \begin{bmatrix} -0.4046 & 1.0163 & -1.4928 & -1.0667 & 4.6114 \\ 0.1365 & 2.4862 & 1.9012 & 2.4685 & -1.5146 \end{bmatrix}$$

$$\vdots$$

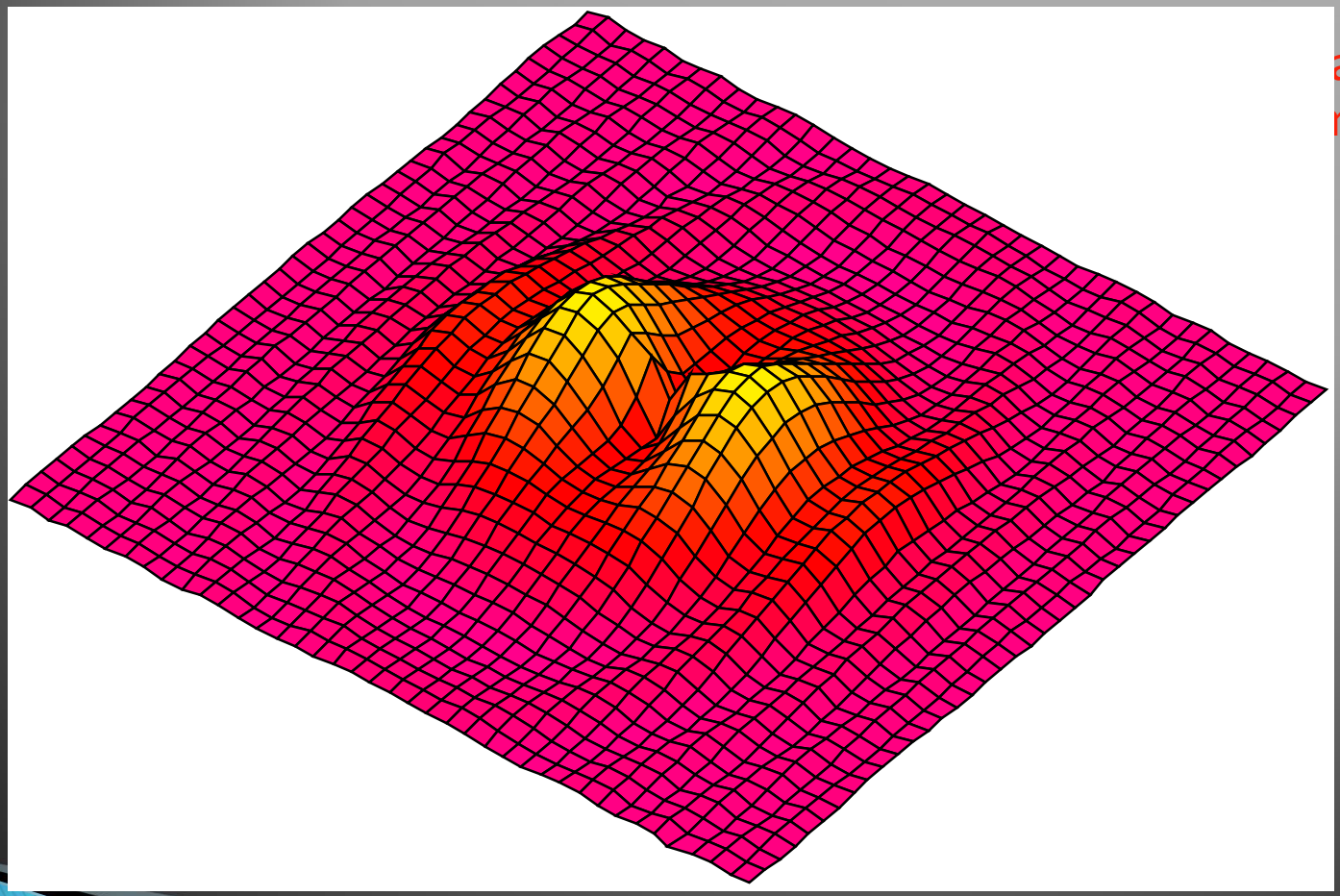
$$\mathbf{x}_t = \begin{bmatrix} 0.2230 & 0.2197 & 0.2400 & 0.2293 & 0.2167 \\ 0.2056 & 0.2436 & 0.2378 & 0.2156 & 0.2106 \end{bmatrix}$$





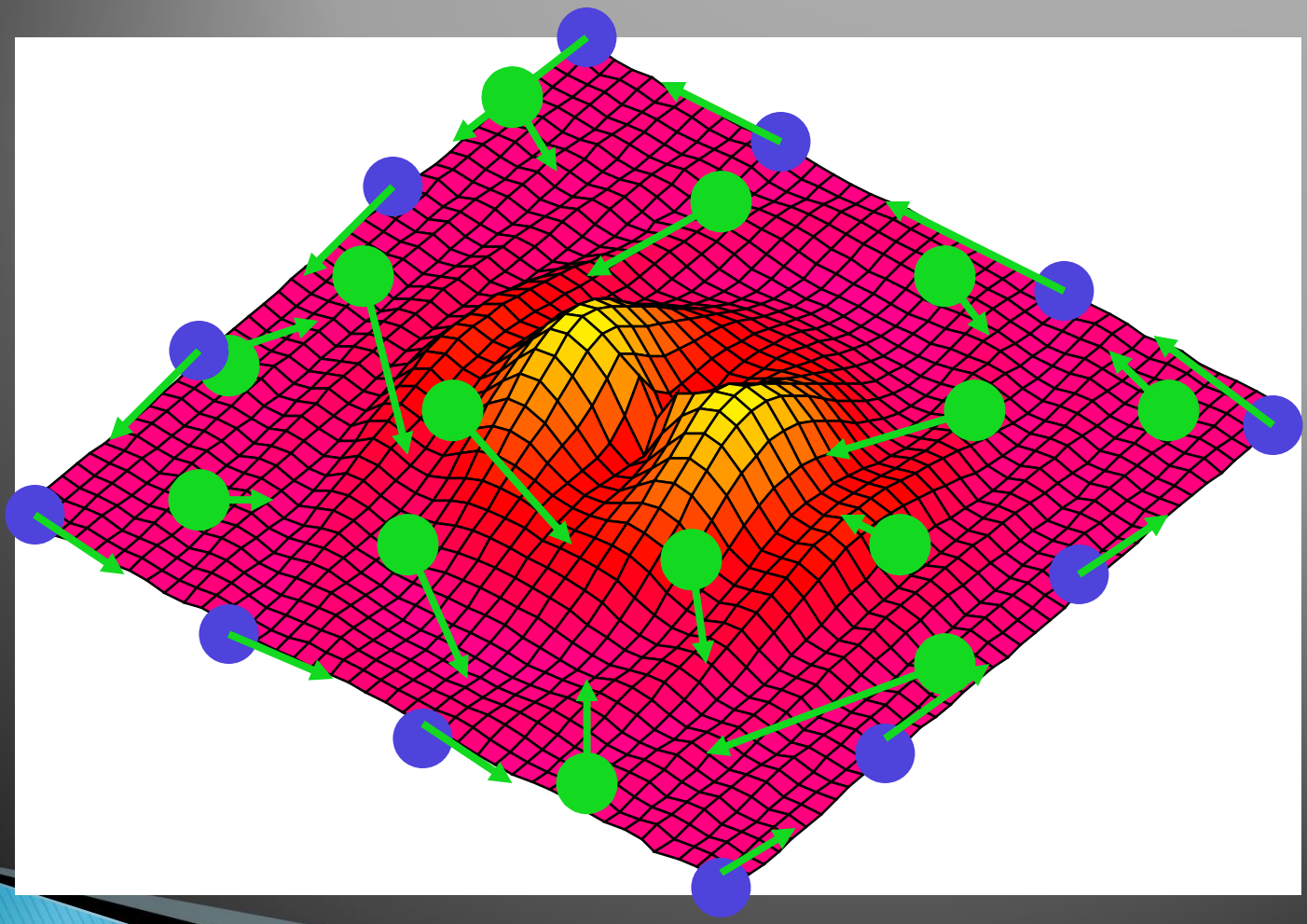
$$GBest = \begin{bmatrix} 0.2227 \\ 0.2057 \end{bmatrix} \quad fitness = -0.25$$

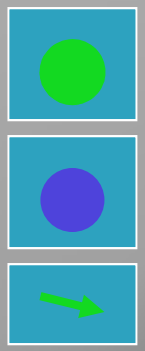
Animated illustration



al
num

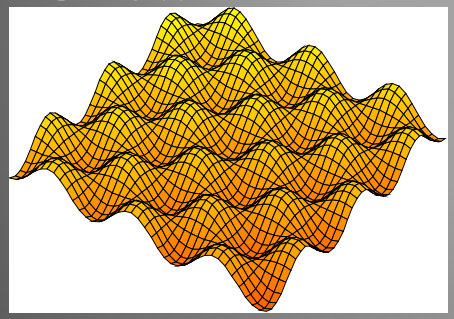
Initialization. Positions and velocities



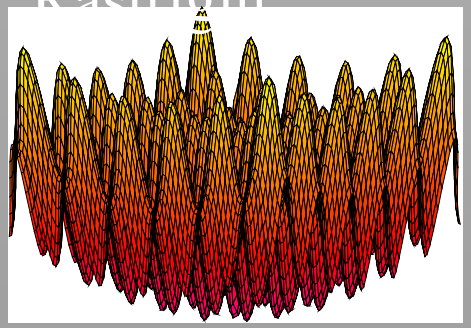


Some functions ...

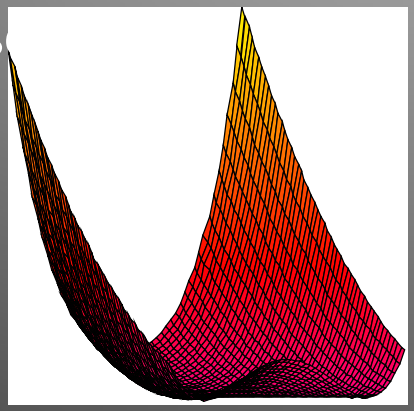
Griewank



Rastrigin



Ros



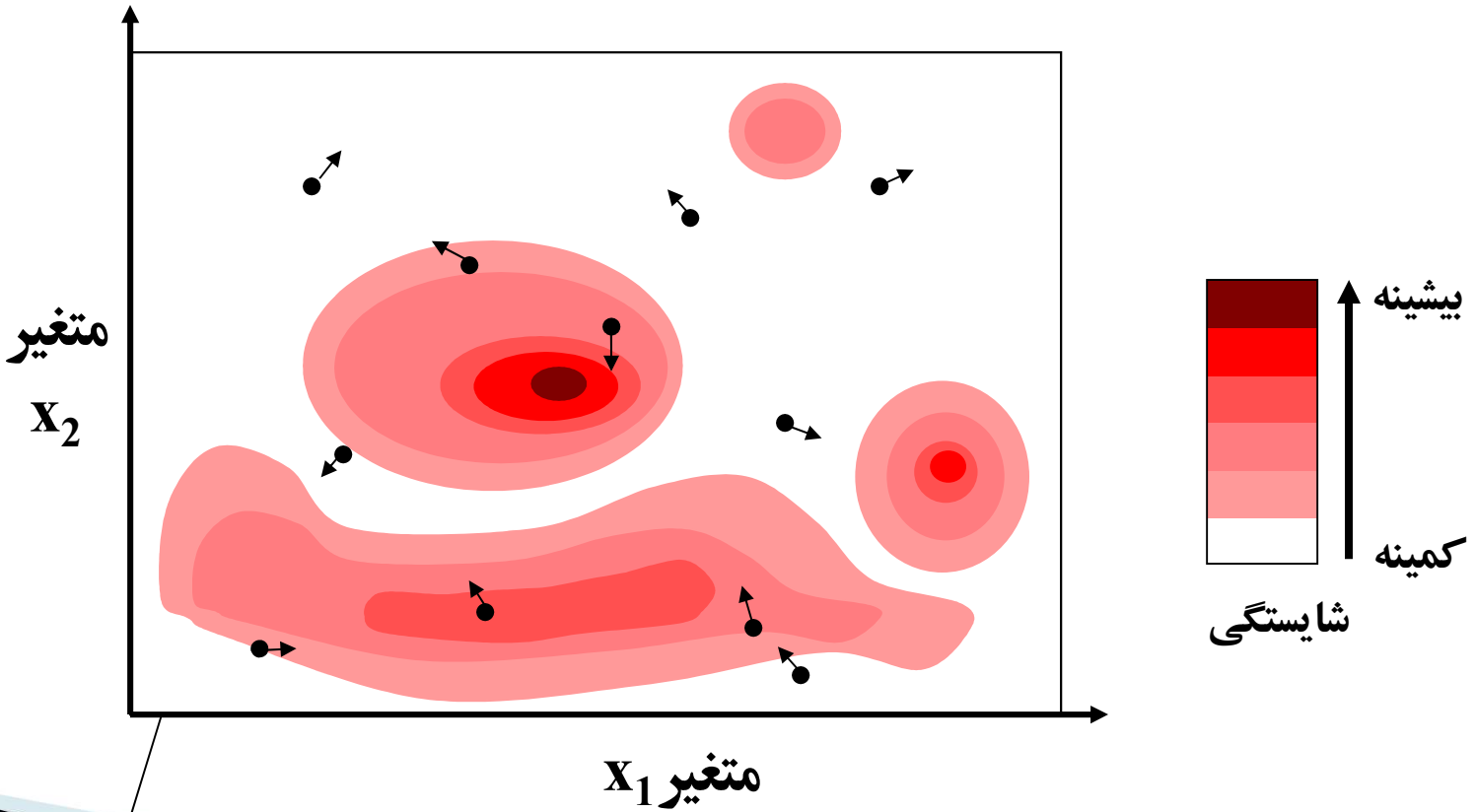
… and some results

Optimum=0, dimension=30

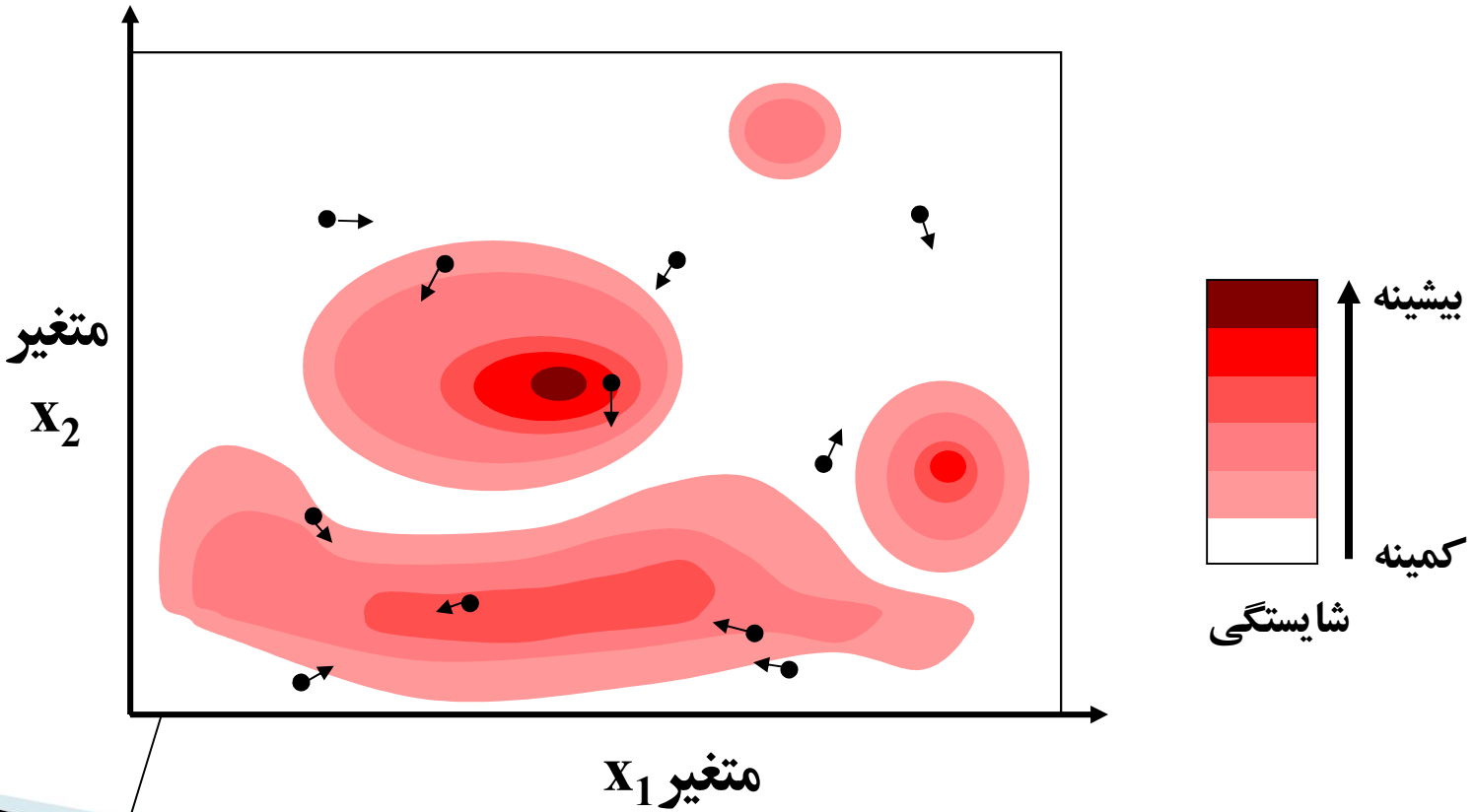
Best result after 40 000 evaluations

30D function	PSO Type 1"	Evolutionary algo.(Angeline 98)
Griewank [±300]	0.003944	0.4033
Rastrigin [±5]	82.95618	46.4689
Rosenbrock [±10]	50.193877	1610.359

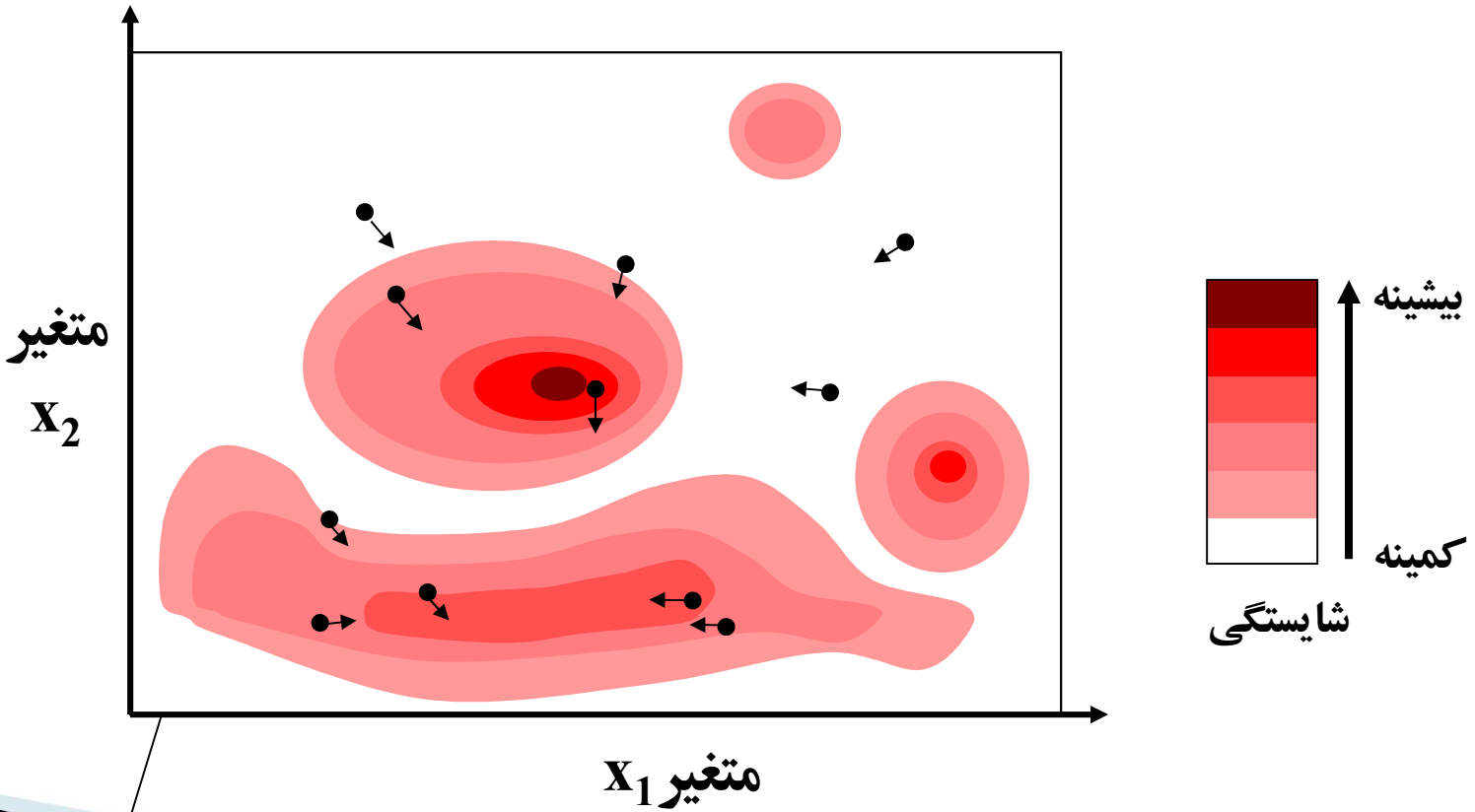
همگرایی ذرات در الگوریتم PSO



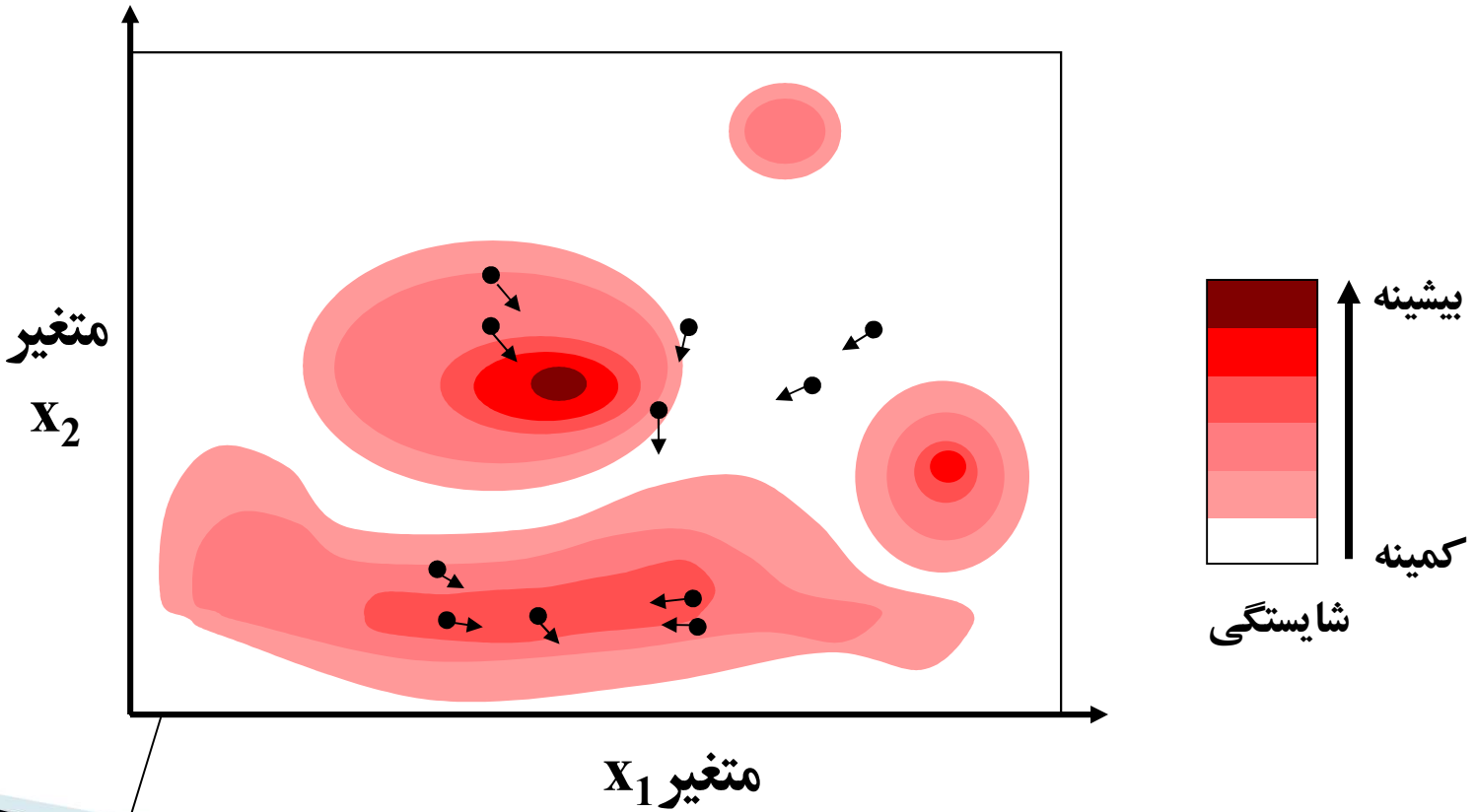
همگرایی ذرات در الگوریتم PSO



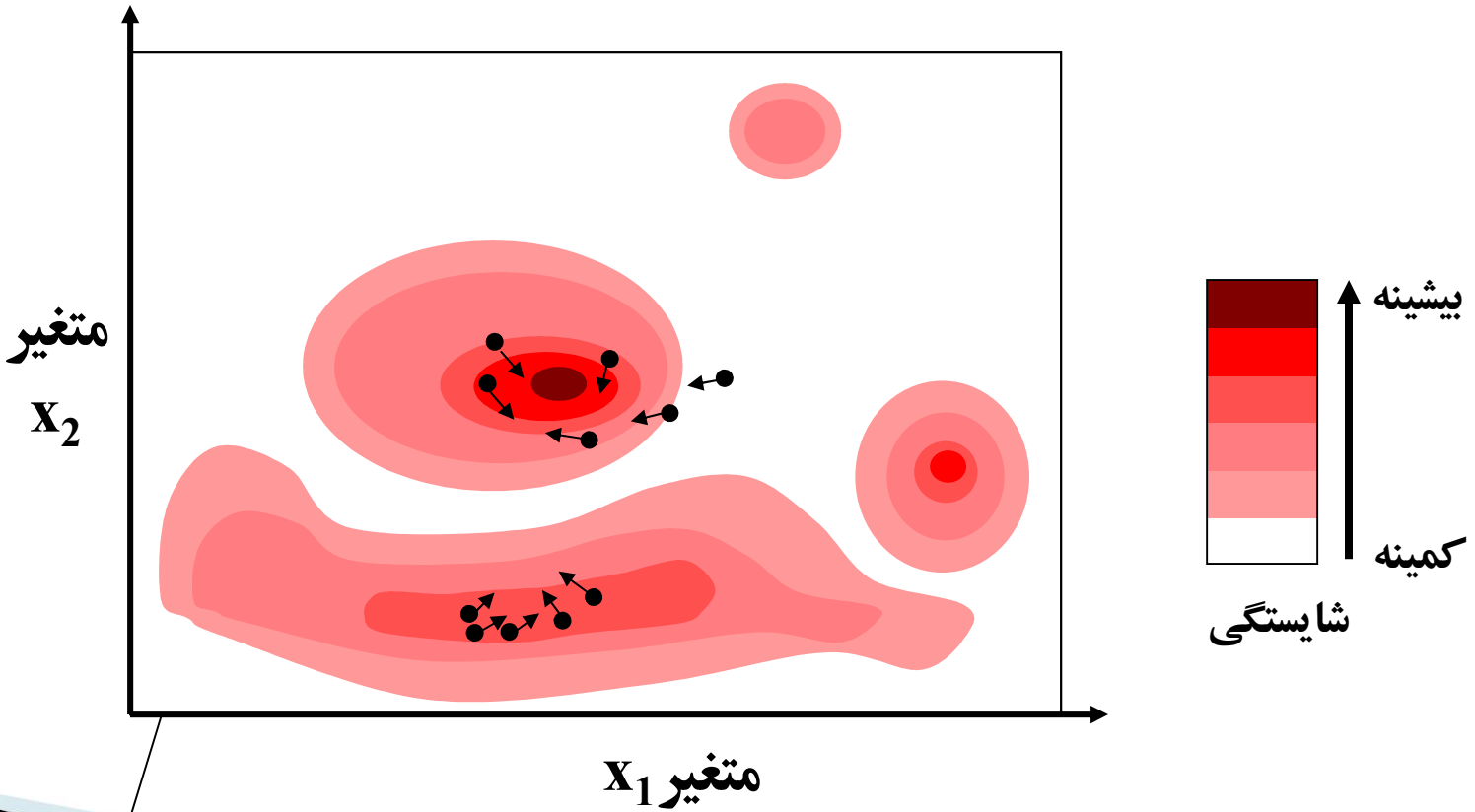
همگرایی ذرات در الگوریتم PSO



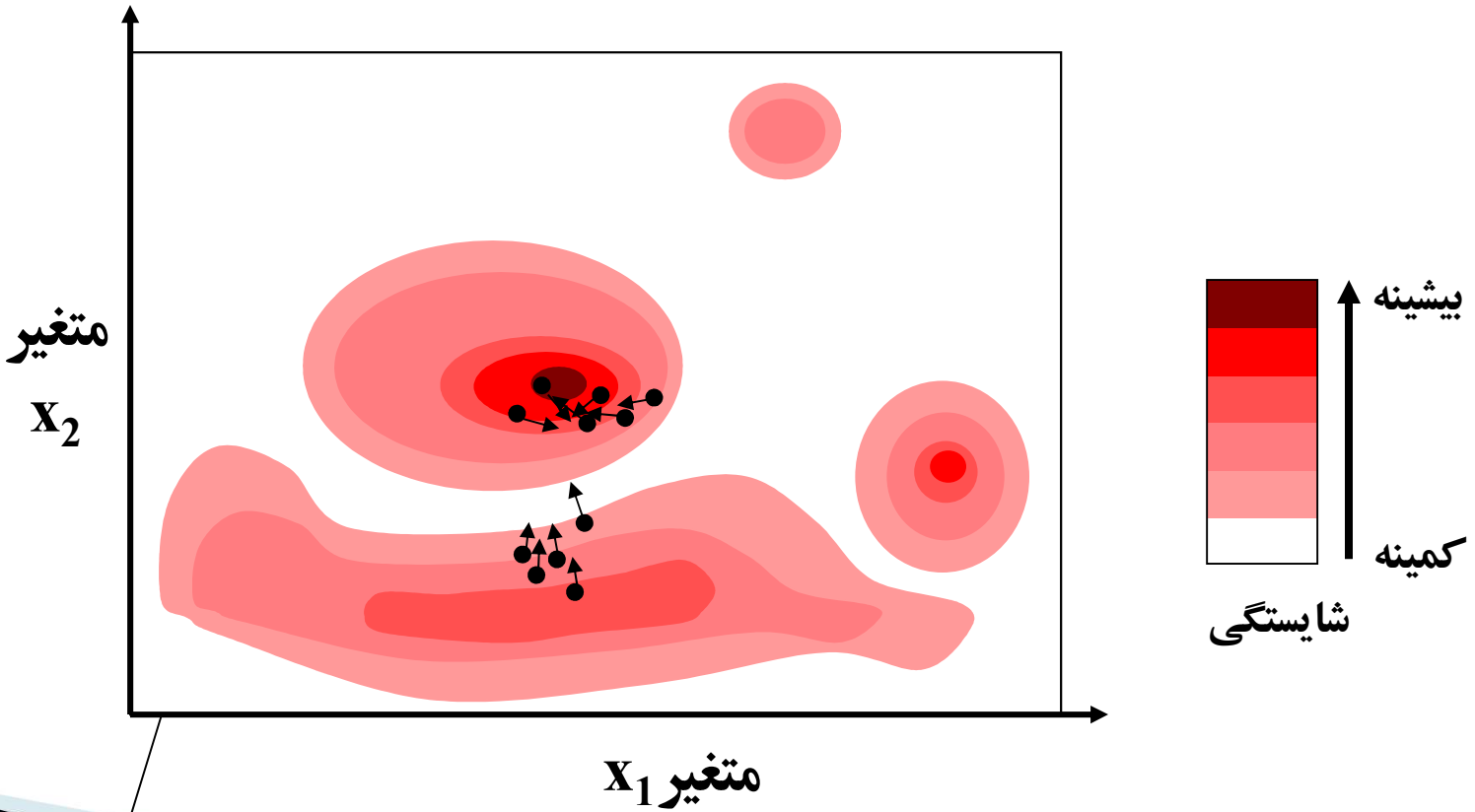
همگرایی ذرات در الگوریتم PSO



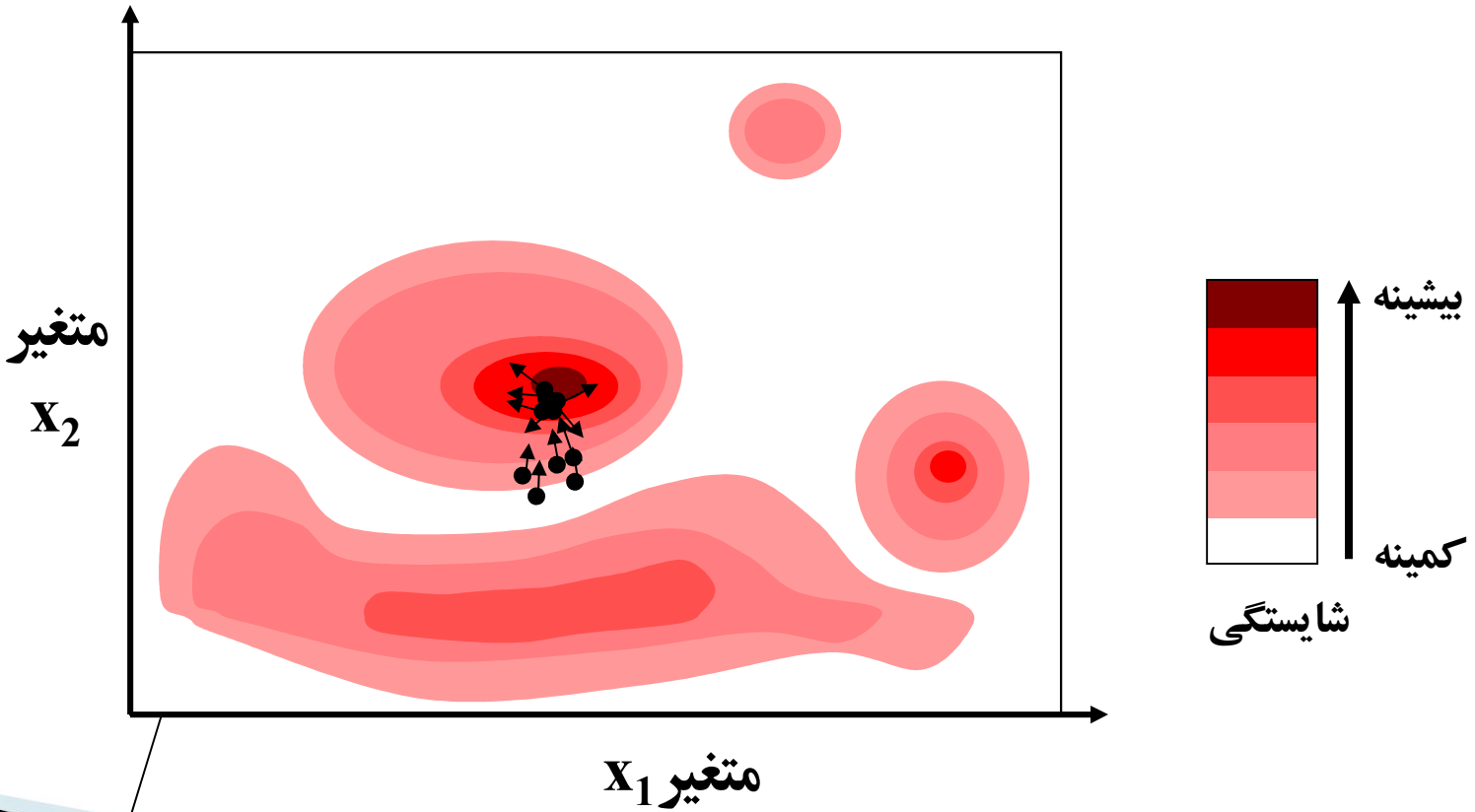
همگرایی ذرات در الگوریتم PSO



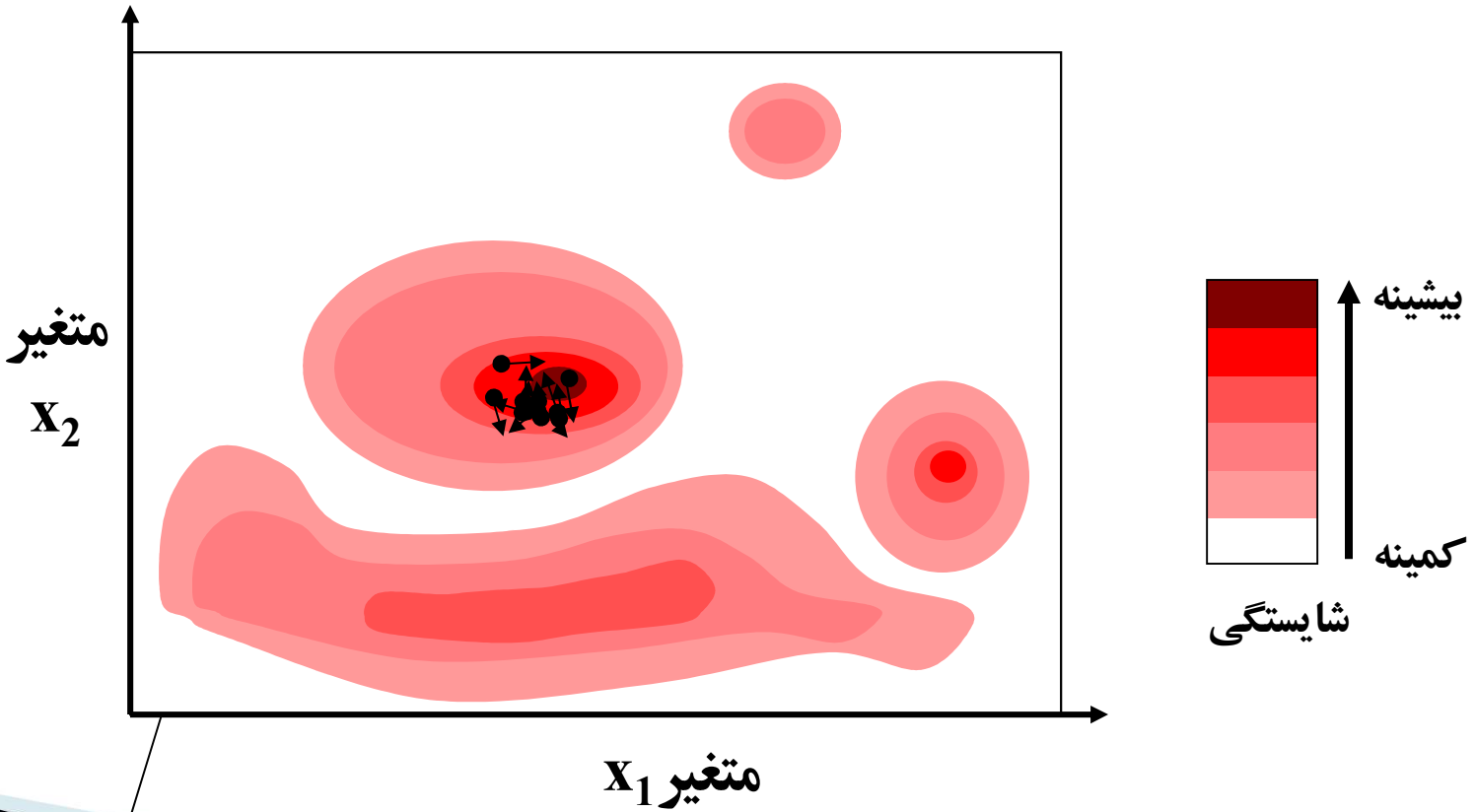
همگرایی ذرات در الگوریتم PSO



همگرایی ذرات در الگوریتم PSO



همگرایی ذرات در الگوریتم PSO



BPSO

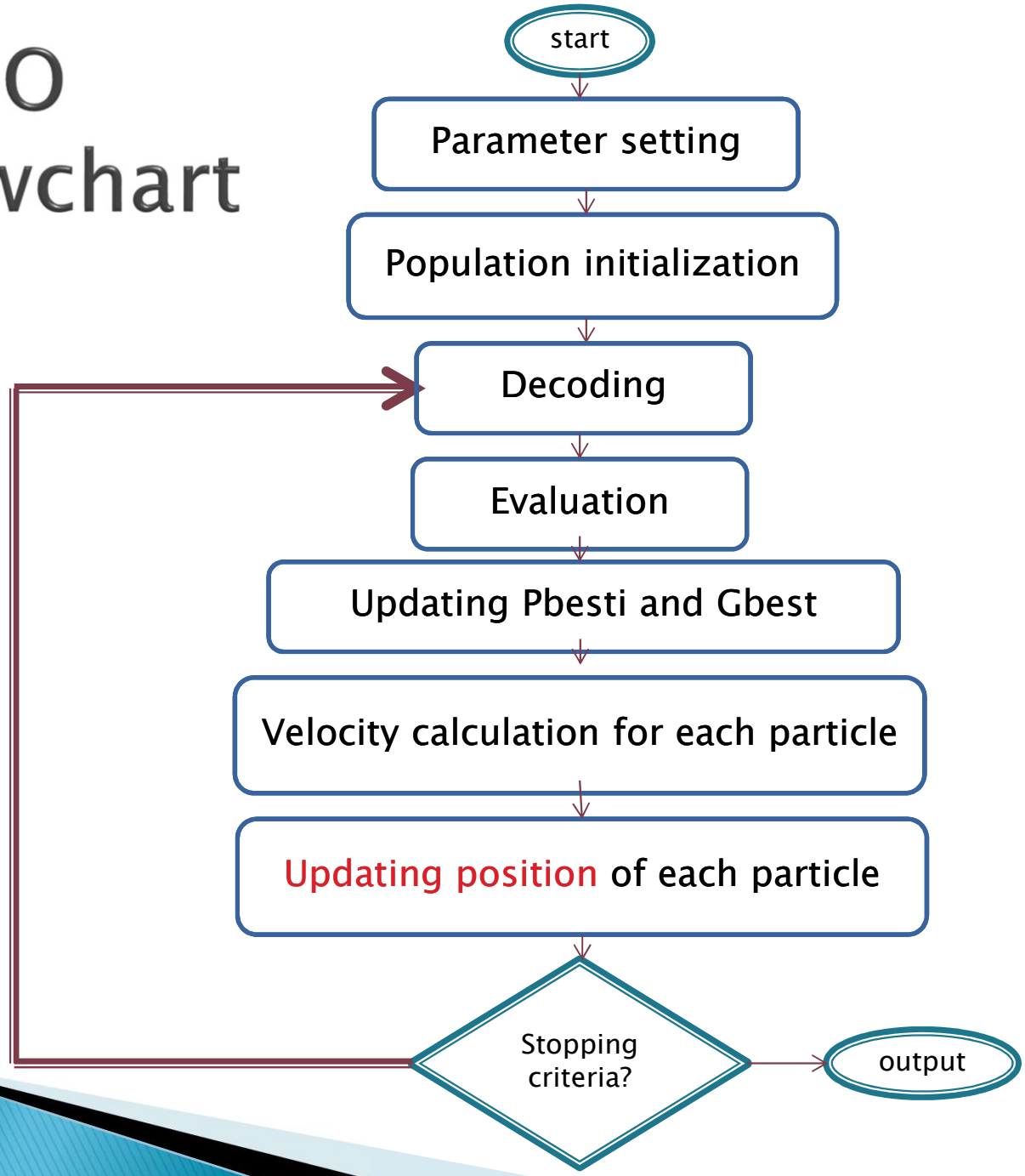
الگوریتم جمعیت ذرات باینری

بسیاری از مسائل بهینه سازی ماهیت گسسته دارند از این رو نیاز به مدل باینری الگوریتم احساس می شود .

در فضای گسسته ذرات جستجوگر در یک فضای صفر و یک حرکت می کنند. یک ابر مکعب را در نظر بگیرید که مختصات هر یک از گوشه های آن با صفر و یک مشخص می شود. هر کدام از گوشه های این ابر مکعب یک جواب مسئله است. برای جستجو لازم است اجرام بین گوشه ها حرکت کنند.

الگوریتم و رابطه محاسبه حرکت مطابق نسخه حقیقی است. اما رابطه به روز رسانی موقعیت تغییر کرده است.

BPSO flowchart



Population in BPSO

- هر ذره یک بردار L بیتی است. با مقادیر ۰ و ۱

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_N \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 \dots\dots\dots x_1^L \\ x_2^1 & x_2^2 \dots\dots\dots x_2^L \\ & \cdot \\ & \cdot \\ & \cdot \\ x_N^1 & x_N^2 \dots\dots\dots x_N^L \end{bmatrix} \qquad X = \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_N \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \dots\dots 1 & 0 \\ 1 & 1 & 0 \dots\dots 0 & 0 \\ & & \cdot & \\ & & \cdot & \\ & & \cdot & \\ 0 & 0 & 1 \dots\dots 1 & 1 \end{bmatrix}$$

- هر ذره یک بردار سرعت L بعدی دارد. با مقادیر حقیقی

$$V = \begin{bmatrix} V_1 \\ V_2 \\ \cdot \\ \cdot \\ \cdot \\ V_N \end{bmatrix} = \begin{bmatrix} v_1^1 & v_1^2 \dots\dots\dots v_1^L \\ v_2^1 & v_2^2 \dots\dots\dots v_2^L \\ & \cdot \\ & \cdot \\ & \cdot \\ v_N^1 & v_N^2 \dots\dots\dots v_N^L \end{bmatrix} \qquad V = \begin{bmatrix} 0.3 & \cdots & -5 \\ \vdots & \ddots & \vdots \\ 3.4 & \cdots & 0.6 \end{bmatrix}$$

تولید جمعیت اولیه

- برای ساختن جمعیت اولیه کافی است که یک ماتریس N*L (شامل Nسطر و L ستون) از صفر و یک بصورت اتفاقی با تابع توزیع یکنواخت ساخته شود.
- در این شرایط هر سطر این ماتریس اطلاعات مربوط به یک ذره را با خود دارد.

- سرعت نیز یک ماتریس N*L است. سرعت اولیه ذرات یا ۰ تنظیم می شود. یا اینکه مقادیر تصادفی حقیقی به آنها نسبت داده می شود.

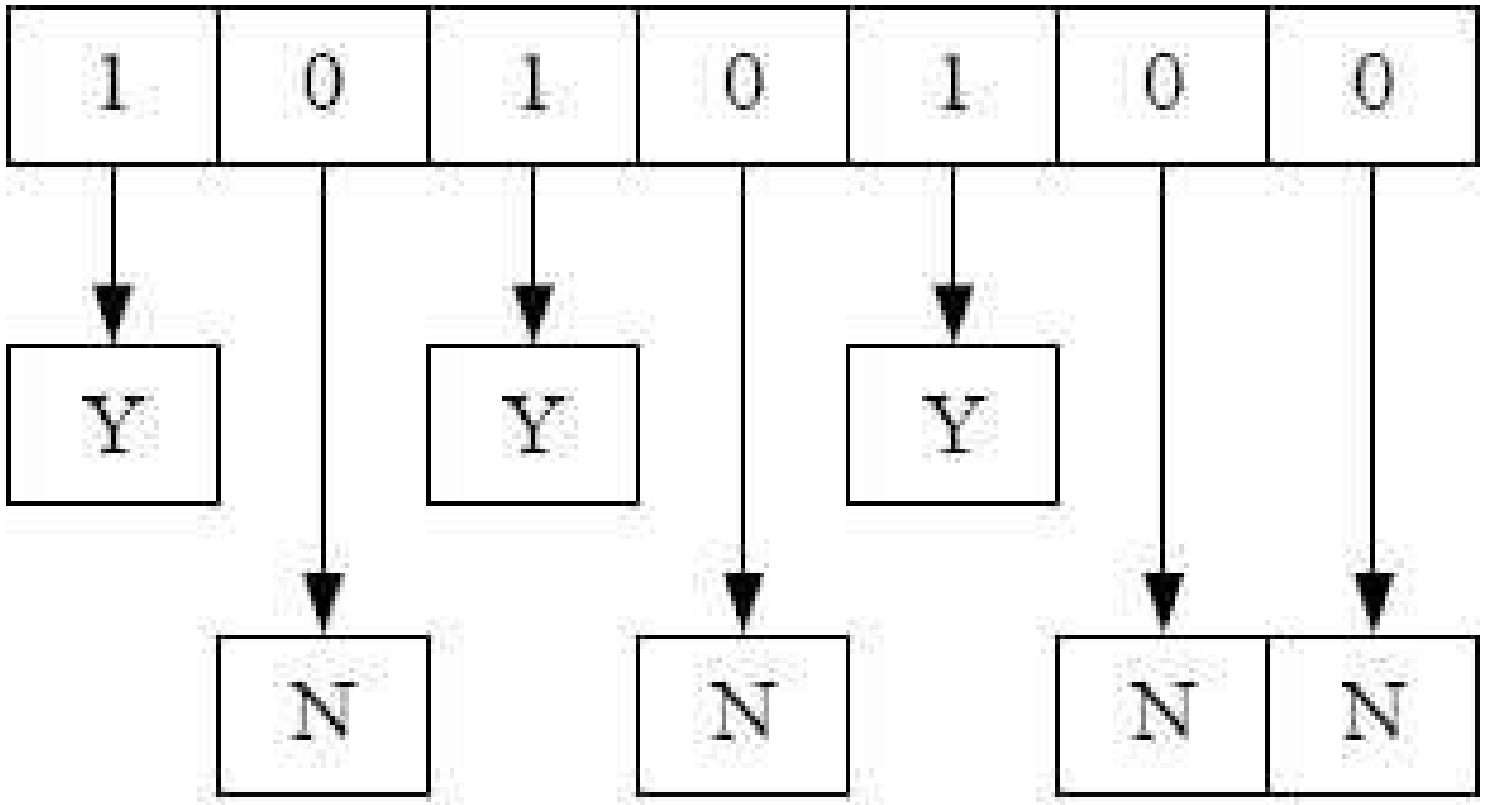
$$X = \begin{bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_N \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 \dots & \dots 1 & 0 \\ 1 & 1 & 0 \dots & \dots 0 & 0 \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ 0 & 0 & 1 \dots & \dots 1 & 1 \end{bmatrix}$$

$$V = \begin{bmatrix} 0.3 & \cdots & -5 \\ \vdots & \ddots & \vdots \\ 3.4 & \cdots & 0.6 \end{bmatrix}$$

رمز گشایی

ابتدا هر عضو جمعیت رمزگشایی می شود.

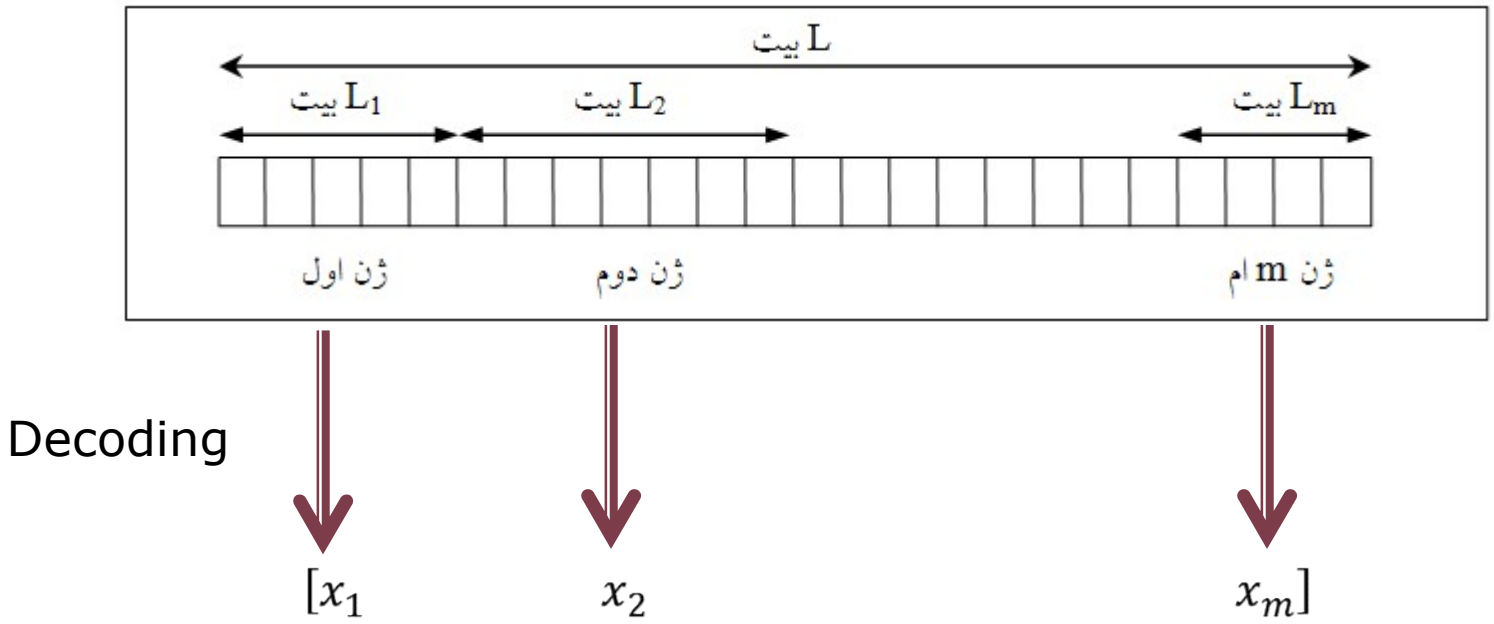
مثال اول:



رمز گشایی

ابتدا هر عضو جمعیت رمزگشایی می شود.

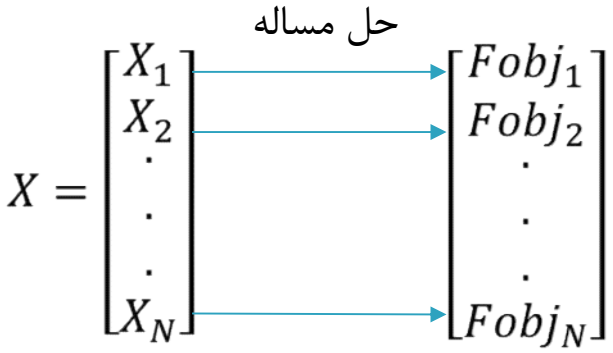
مثال دوم:



Evaluation

- در هر تکرار، به ازای هر عضو یکبار مساله حل می شود.

با استفاده از اعضای رمزگشایی شده



Update Gbest,Pbest

- به روز رسانی:
- بهترین موقعیتی که جمعیت تا لحظه t به آن رسیده است. Gbest(t)
- بهترین موقعیتی که ذره i تا لحظه t به آن رسیده است. $Pbest_i(t)$

$$Gbest(t) = [g^1\ g^2\ \ldots\ldots g^L]$$

$$pbest_i(t) = [p_i^1\ \ p_i^2\ \ \ldots\ldots p_i^L]$$

$$\text{pbest}(t) = \begin{bmatrix} pbest_1(t) \\ pbest_2(t) \\ \cdot \\ \cdot \\ \cdot \\ pbest_N(t) \end{bmatrix}$$

BPSO: Velocity calculation

- به روز رسانی سرعت

$$v_i^d(t+1) = w(t)v_i^d(t) + c_1 r_1(t)[pbest_i^d - x_i^d(t)] + c_2 r_2(t)[gbest^d - x_i^d(t)]$$

$$V = \begin{bmatrix} V_1 \\ V_2 \\ \cdot \\ \cdot \\ \cdot \\ V_N \end{bmatrix} = \begin{bmatrix} v_1^1 & v_1^2 \dots\dots\dots v_1^L \\ v_2^1 & v_2^2 \dots\dots\dots v_2^L \\ & \cdot \\ & \cdot \\ & \cdot \\ v_N^1 & v_N^2 \dots\dots\dots v_N^L \end{bmatrix}$$

BPSO

حرکت ذره در هر بعد به معنای تغییر مقدار آن از صفر به یک و بالعکس خواهد بود. در این روش اینگونه عمل می کنیم که سرعت ذره در هر بعد بصورت یک تابع احتمال در نظر گرفته می شود و بر مبنای آن، ذره با یک احتمال در آن بعد تغییر موقعیت می دهد، در واقع در نسخه باینری v_i^d به جای آنکه بیانگر جابجایی ذره باشد، احتمال صفر یا یک بودن x_i^d را نشان می دهد.

BPSO

جهت تعریف تابع انتقال برای نگاشت سرعت به احتمال جابجایی ذره، چند مفهوم مهم را متذکر می شویم:

مقدار بزرگ سرعت بیان کننده اینست که ذره تمایل بیشتری به جابجایی دارد.
مقدار کوچک سرعت بیانگر اینست که ذره تمایل کمتری به حرکت و تمایل بیشتری به سکون دارد.

براساس مفاهیم فوق:

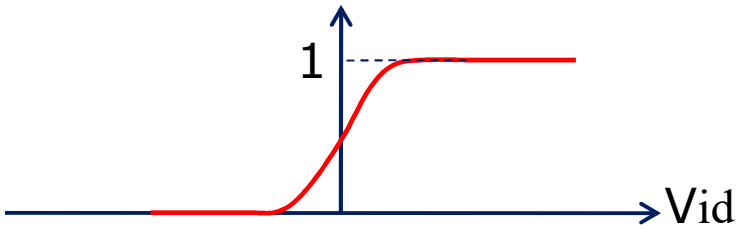
مقدار بزرگ سرعت، باید یک احتمال بزرگ از تغییرات مکانی ذره با توجه به موقعیت قبلی ایجاد کند (از صفر به یک و بر عکس).
مقدار کوچک سرعت، بایستی یک احتمال کوچک از تغییرات مکان ذره را ایجاد کند.
به عبارتی مقدار صفر سرعت، نشان دهنده تمایل به عدم تغییر است.

تابع احتمال حرکت

تابع احتمال پیشنهادی طوری تعریف شده است که برای سرعت صفر احتمال نیم، برای سرعت منفی احتمال حرکت نزدیک به صفر و برای سرعت بزرگ، احتمال حرکت نزدیک به یک است.

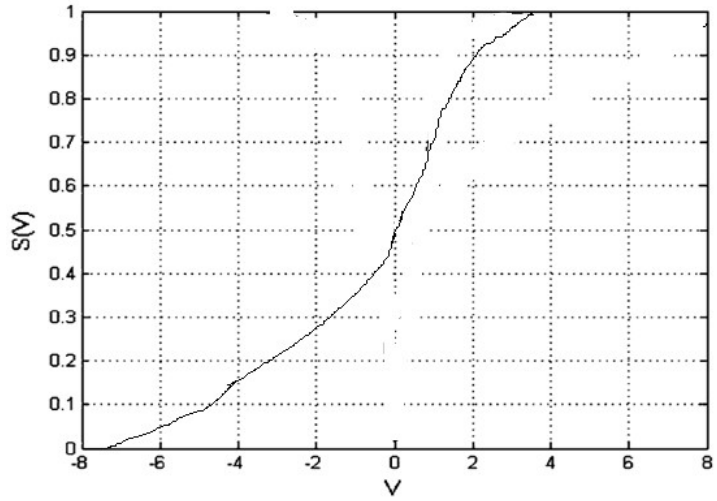
تابع $S(v_i^d)$ را برای تبدیل v_i^d به تابع احتمال تعریف می کنیم. این تابع باید در محدوده [۱-۰] قرار گیرد . رابطه زیر نحوه محاسبه S را نشان می دهد.

$$S(v_i^d) = \frac{1}{1 + e^{-v_i^d}}$$



BPSO

شکل تابع **S(Vid)**



برای همگرایی مناسب الگوریتم، اندازه سرعت باید به یک بازه مناسب محدود شود.

$$|v_i^d| < v_{max}$$

در الگوریتم BPSO مقدار V_{max} برابر ۶ در نظر گرفته می شود.

تابع جابجایی

ذره در هر بعد مطابق رابطه زیر حرکت می کند:

$$if \quad rand < S(v_i^d(t+1)) \quad then$$

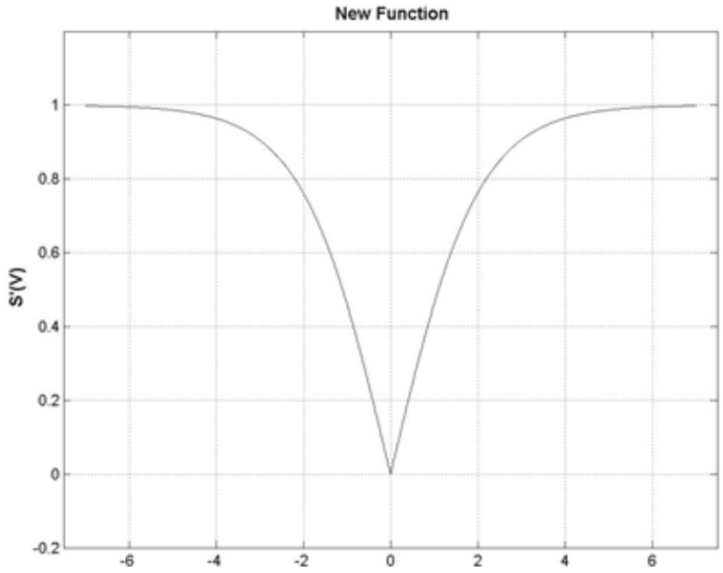
$$x_i^d(t+1) = 1$$

$$else \quad x_i^d(t+1) = 0$$

طبق این رابطه ذره با یک احتمال تغییر موقعیت می دهد که هرچه سرعت ذره در یک بعد بیشتر باشد، احتمال یک بودن در آن بعد بیشتر می شود.

Rand یک عدد تصادفی با توزیع یکنواخت در بازه [۱ و۰] است.

New BPSO



$$if \quad rand() < S'(v_{id}(t+1)) \quad then \quad x_{id}(t+1) = exchange(x_{id}(t))$$

$$else \quad x_{id}(t+1) = x_{id}(t)$$

H. Nezamabadi-pour, M. Rostami-shahrbabaki, M.M. Farsangi, "Binary Particle Swarm Optimization: challenges and New Solutions", The Journal of Computer Society of Iran (CSI) On Computer Science and Engineering (JCSE), vol. 6, no. (1-A), pp. 21-32, 2008.