

InTELIGÊNCIA ARTIFICIAL

Métodos de Pesquisa (*Search*) – Parte IV

Paulo Moura Oliveira

Departamento de Engenharias

Gabinete F2.15, ECT-1

UTAD

email: oliveira@utad.pt

Enxames (*Swarms*)

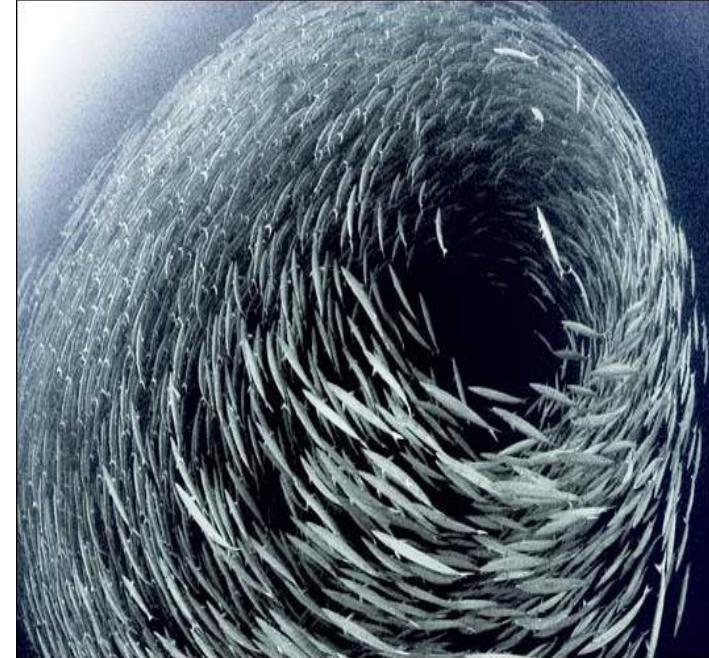
A forma coordenada e sincronizada como animais se movem em bandos despertou a curiosidade desde sempre.



Porque é que os animais se movimentam desta forma?

Há vantagens neste tipo de comportamento?

Há desvantagens neste tipo de comportamento?



As **vantagens da partilha** social da informação claramente superam as **desvantagens da competição**.

BOIDS

Uma aplicação computacional pioneira que se inspirou no comportamento de bandos de pássaros foi desenvolvida por Craig Reynolds:^{*}

BOIDS:

Objetivo: simular graficamente o movimento coordenado de pássaros para gerar animações computerizadas.

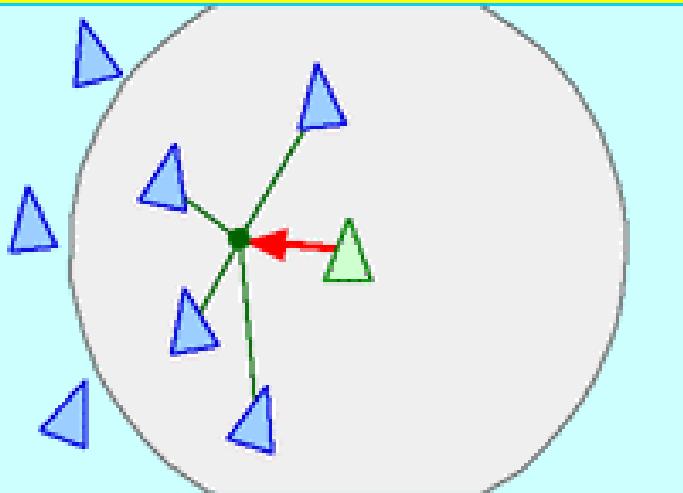
Reynolds propôs (originalmente) três regras fundamentais para simular computacionalmente o movimento de um bando:

- Coesão (*Cohesion*) - C
- Separação (*Separation*) - S
- Alinhamento (*Alignment*) - A

* Reynolds, C. W. (1987) Flocks, herds and schools: A distributed behavioral model. ACM SIGGRAPH Computer Graphics, Vol. .21, No. 4, pp. 25-34

BODS

Regra da Coesão:



- ✓ Esta regra promove o agrupamento dos Boids em torno da média das suas posições.
- ✓ Considerando a posição representada por \mathbf{x} , um enxame com m elementos a **posição média do enxame** pode ser determinada usando:

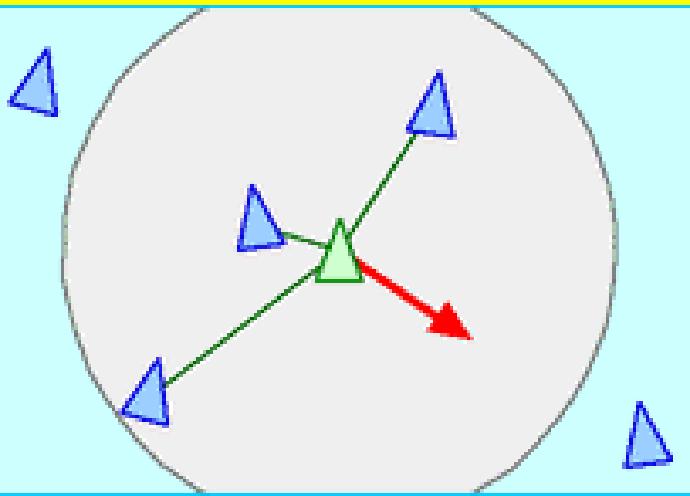
$$x_{ctr}(t) = \frac{1}{m} \sum_{j=1}^m x_j(t)$$

<https://www.red3d.com/cwr/boids/>

Acedido em 28-7-2018

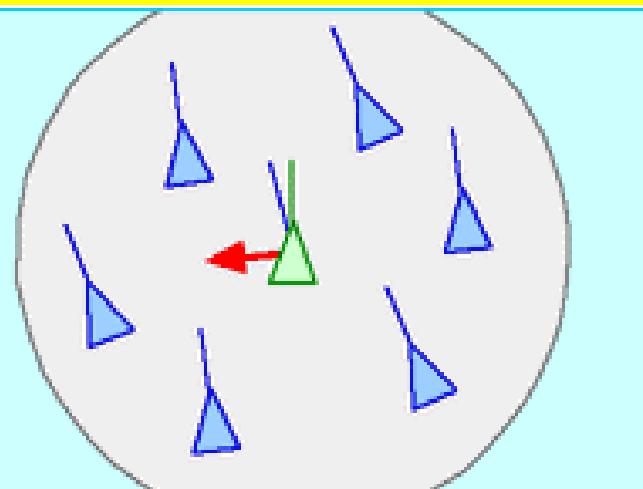
BOIDS

Regra da Separação:



- ✓ Previne o choque e a ocupação do mesmo espaço pelos Boids.
- ✓ Nesta regra é usual considerar uma distância mínima ao redor de cada Boid, penalizando a **velocidade, v**, dos boids que a violem.

Regra do Alinhamento:



- ✓ Promove o alinhamento da velocidade de um BOID com a dos seus vizinhos:
- ✓ A média das velocidade do resto do enxame pode determinada por:

$$v_{ctr}(t) = \frac{1}{m-1} \sum_{(j=1) \wedge (j \neq i)}^m v_j(t)$$

BOIDS

- ✓ Sabendo que a posição de cada agente (Boid), i , pode ser determinada na iteração $t+1$ a partir da posição na iteração anterior, t , usando a seguinte expressão:

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

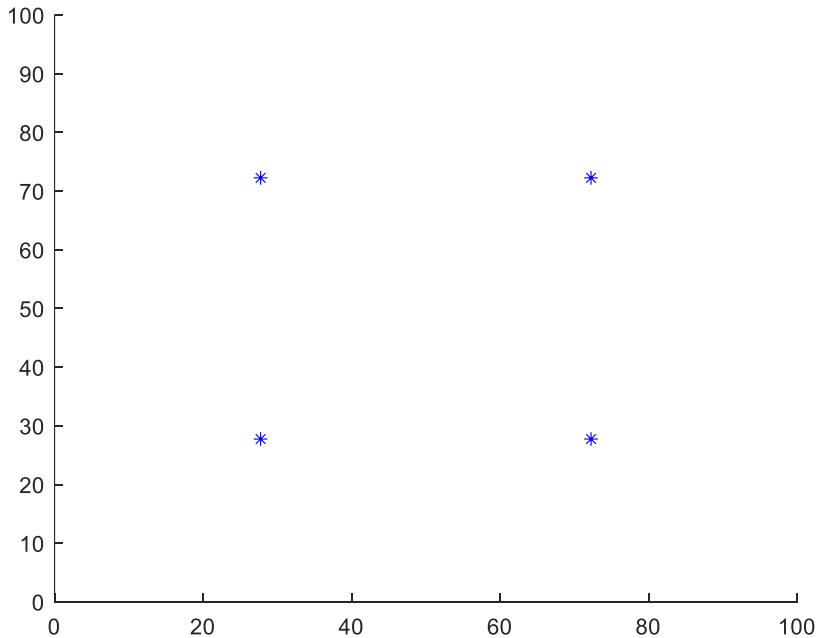
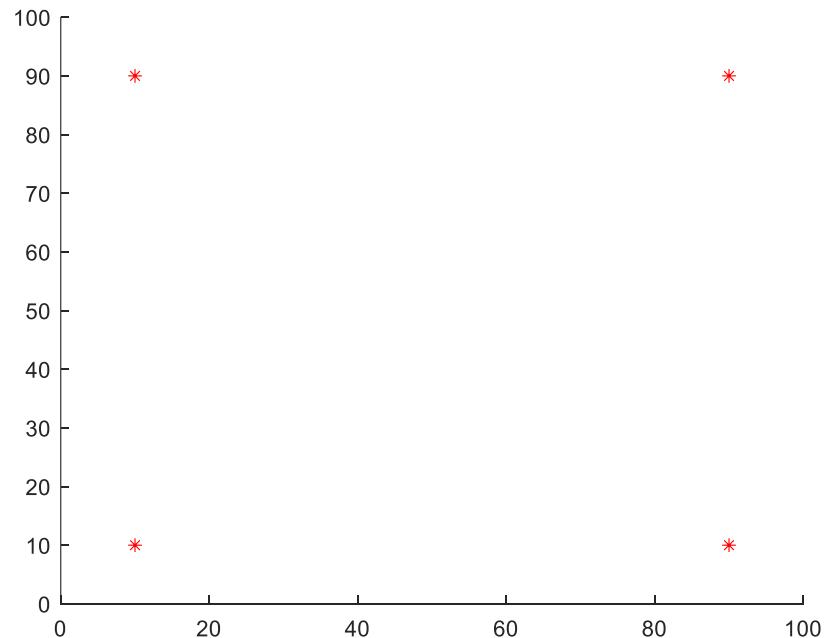
- ✓ A velocidade de cada Boid pode ser calculada usando uma soma ponderada das velocidades da coesão (**vc**) da separação (**vs**) e da agregação (**va**):

$$v_i(t+1) = v_i(t) + \beta_1 vc_i(t) + \beta_2 vs_i(t) + \beta_3 va_i(t)$$

BOIDS

- ✓ Desafio proposto aos alunos:

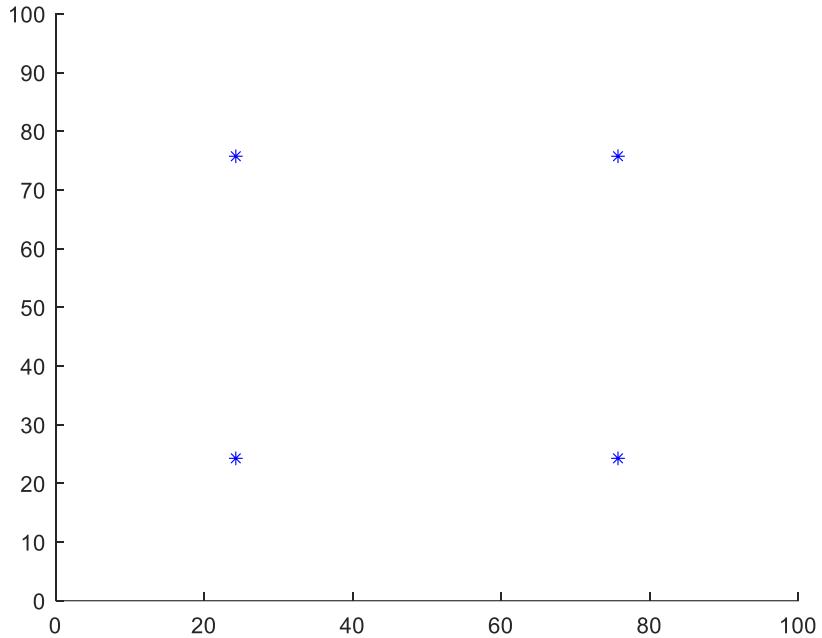
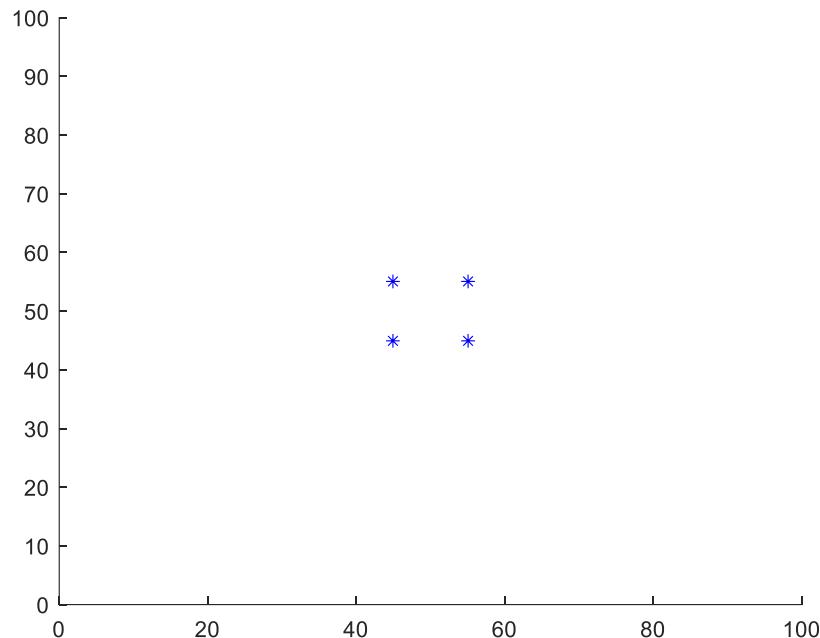
Implementação de um programa para simular um enxame de BOIDS.



BOIDS

- ✓ Desafio proposto aos alunos:

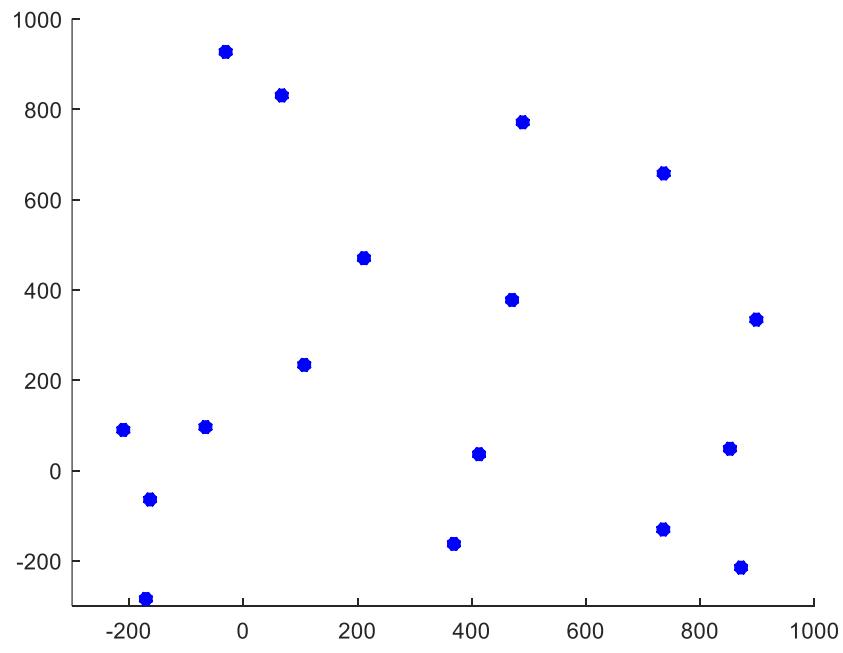
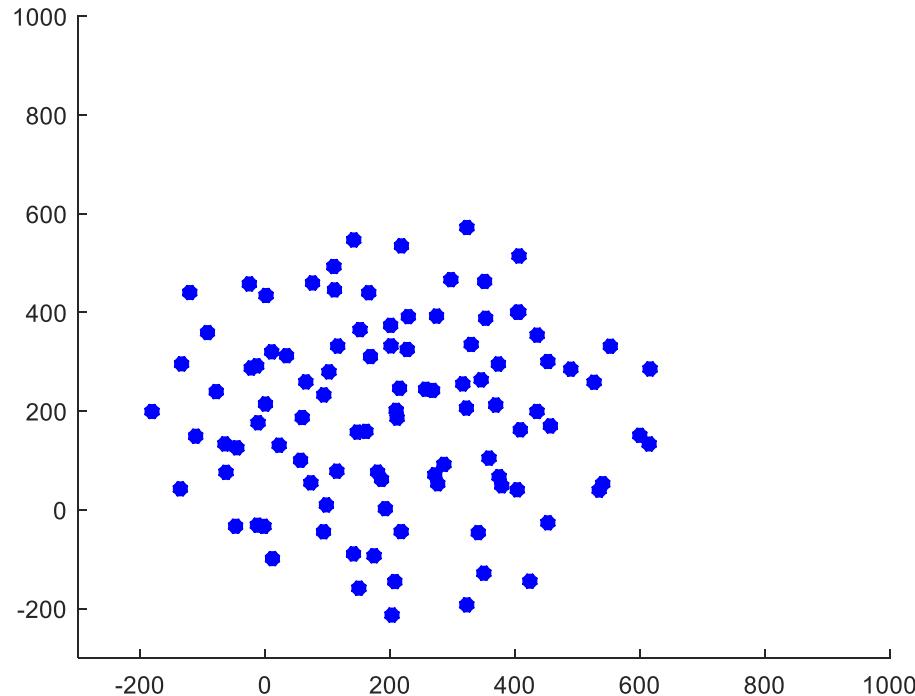
Implementação de um programa para simular um enxame de BOIDS.



BOIDS

- ✓ Desafio proposto aos alunos:

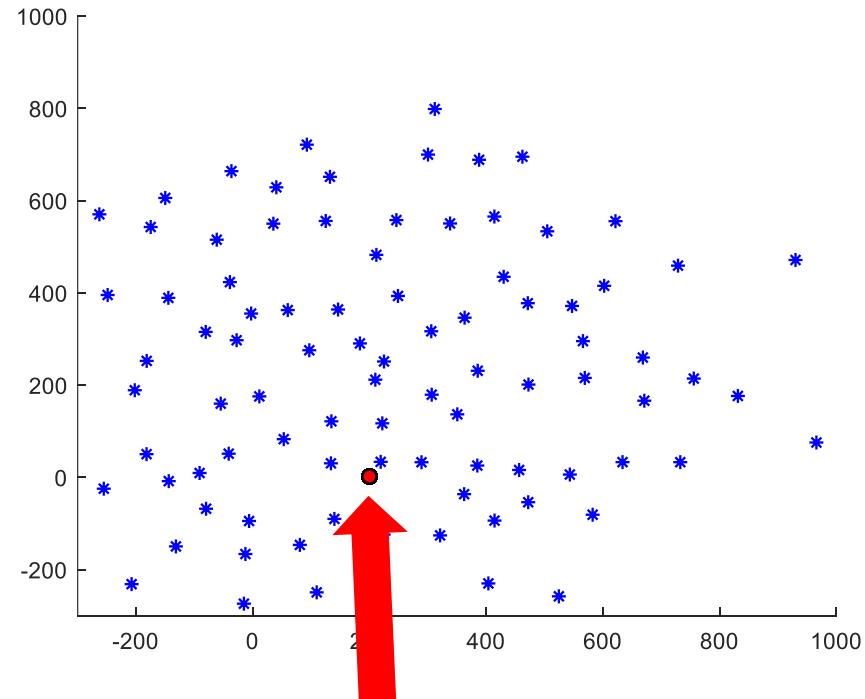
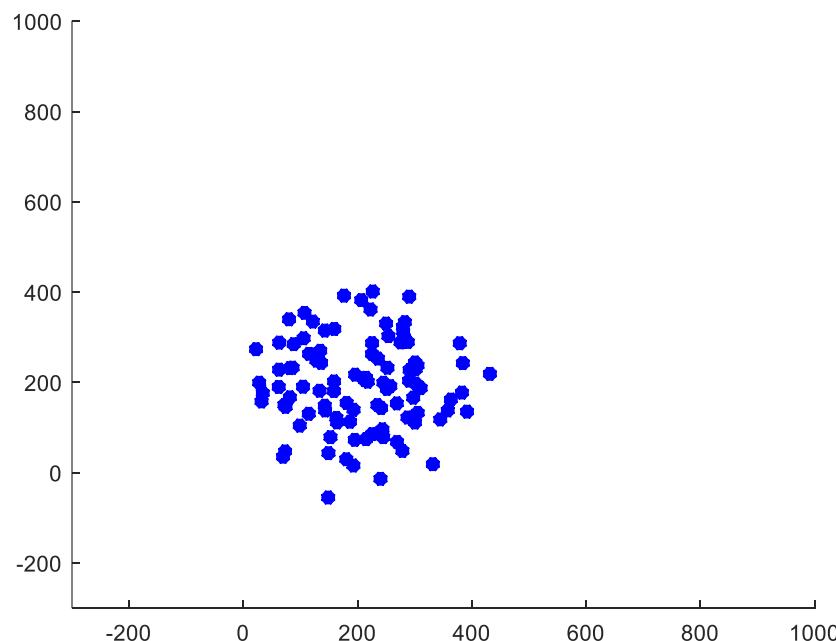
Implementação de um programa para simular um enxame de BOIDS.



BOIDS

✓ Desafio proposto aos alunos:

Implementação de um programa para simular um enxame de BOIDS.

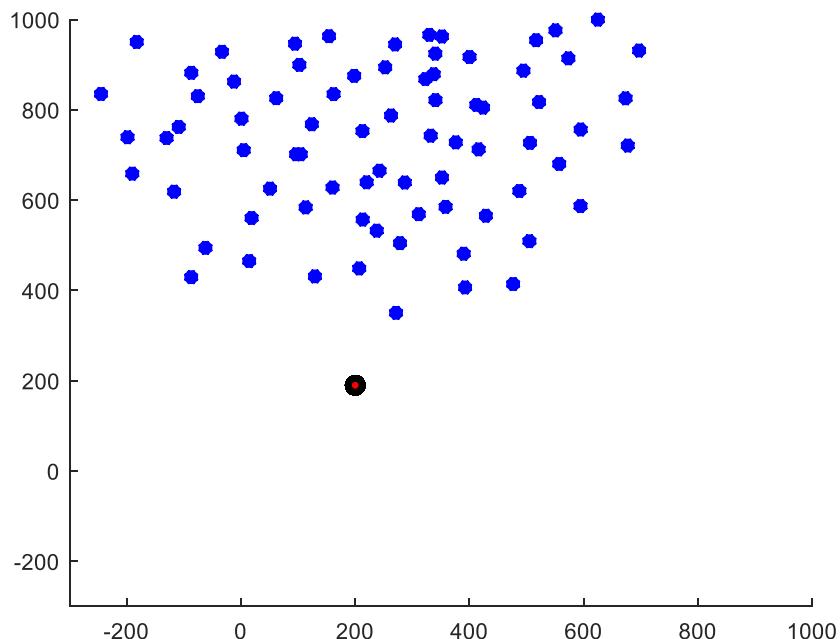
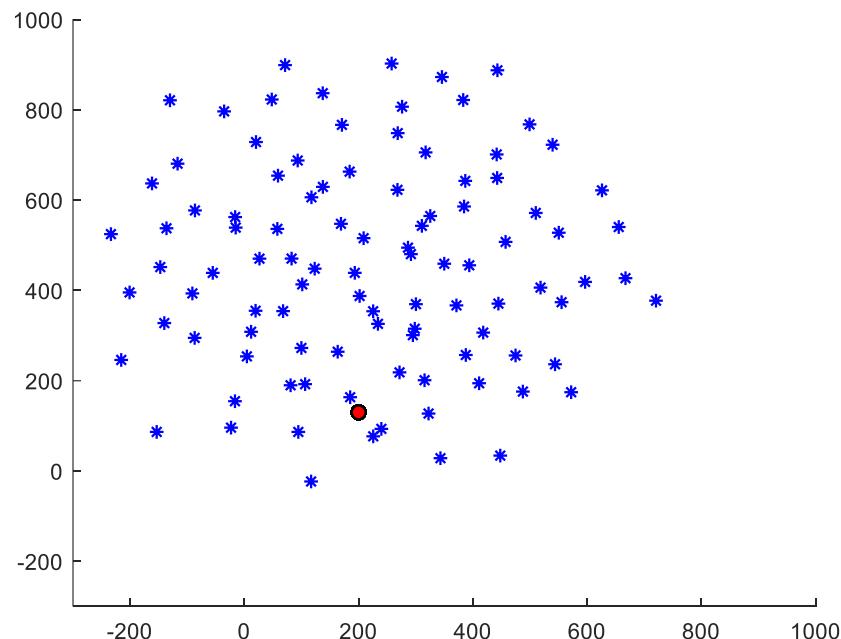


“Predador ou Ameaça”

BOIDS

✓ Desafio proposto aos alunos:

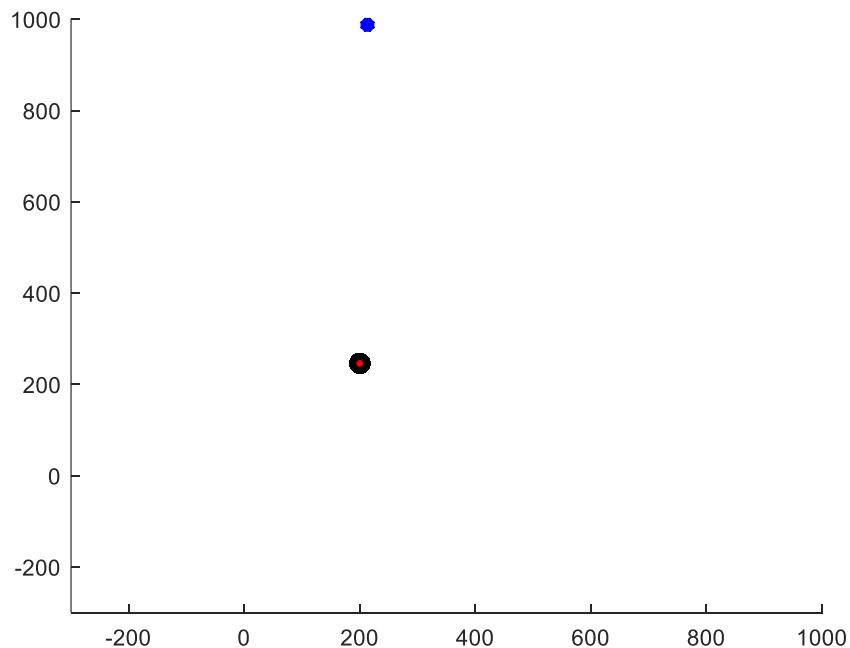
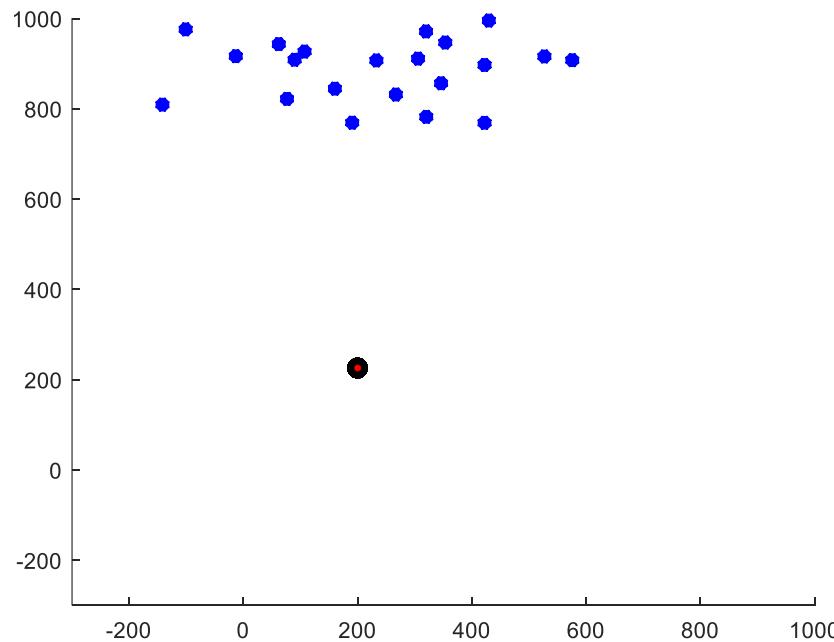
Implementação de um programa para simular um enxame de BOIDS.



BOIDS

✓ Desafio proposto aos alunos:

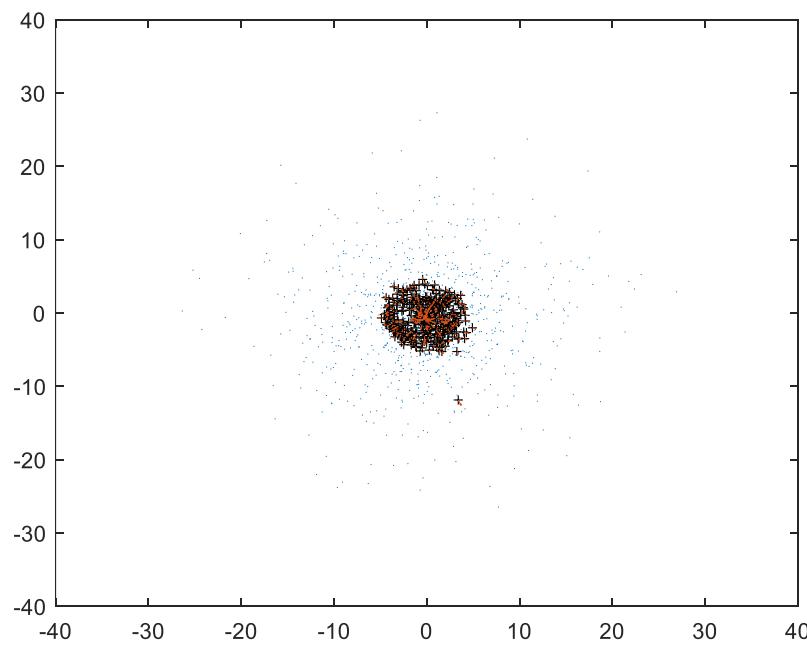
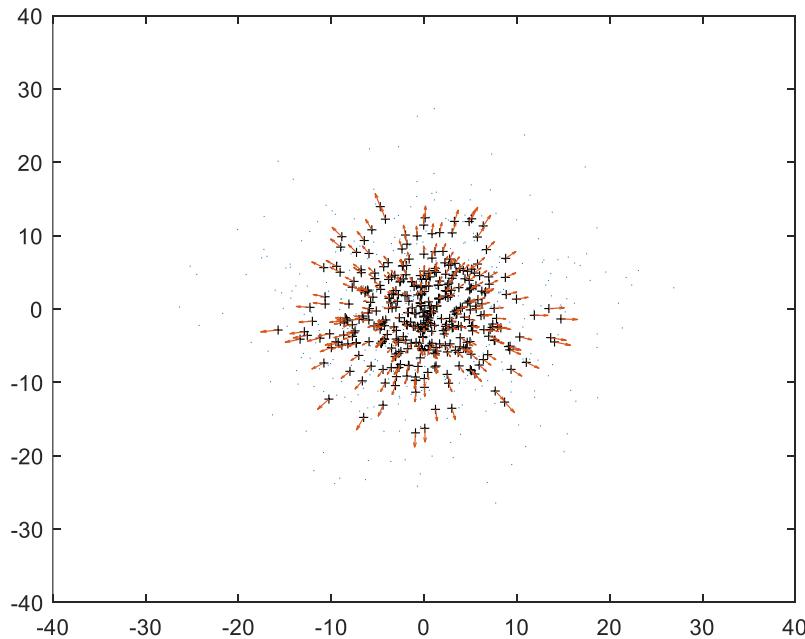
Implementação de um programa para simular um enxame de BOIDS.



BOIDS

✓ Desafio proposto aos alunos:

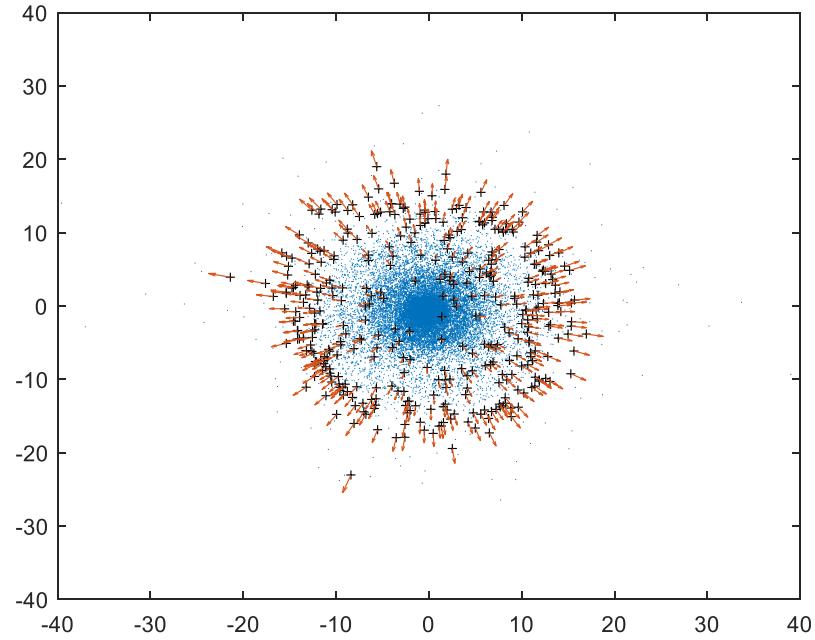
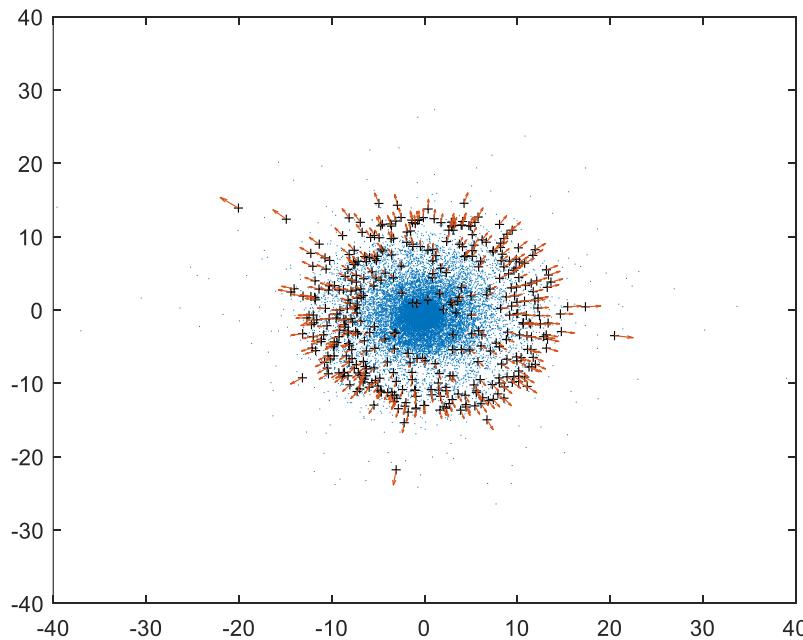
Implementação de um programa para simular um enxame de BOIDS.



BOIDS

✓ Desafio proposto aos alunos:

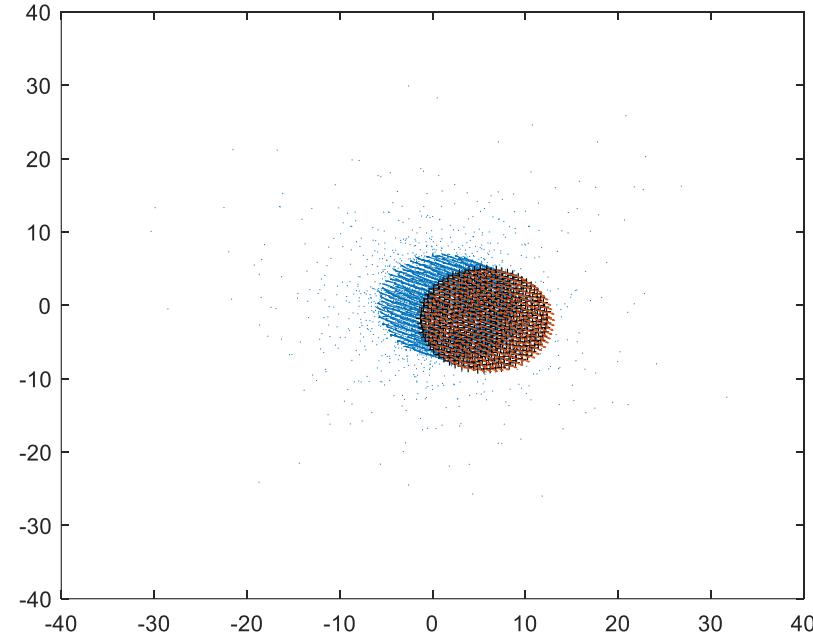
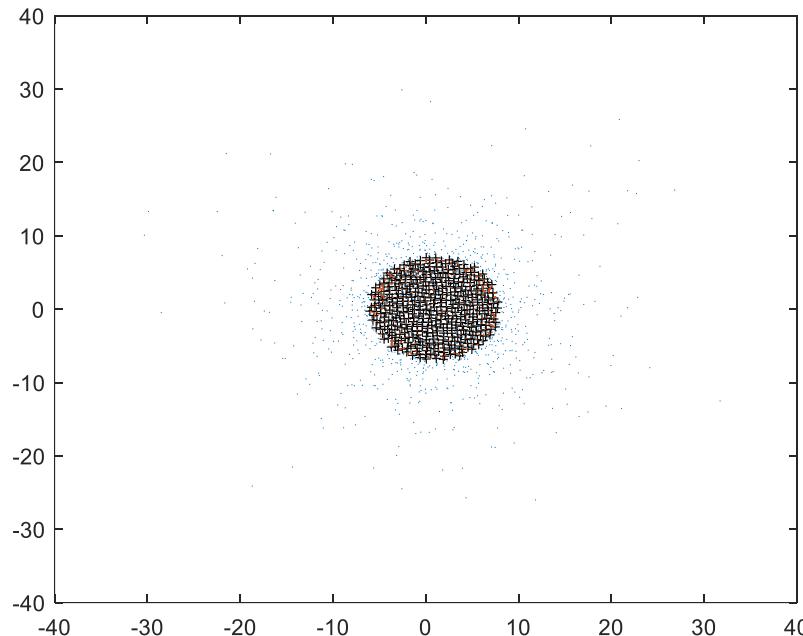
Implementação de um programa para simular um enxame de BOIDS.



BOIDS

✓ Desafio proposto aos alunos:

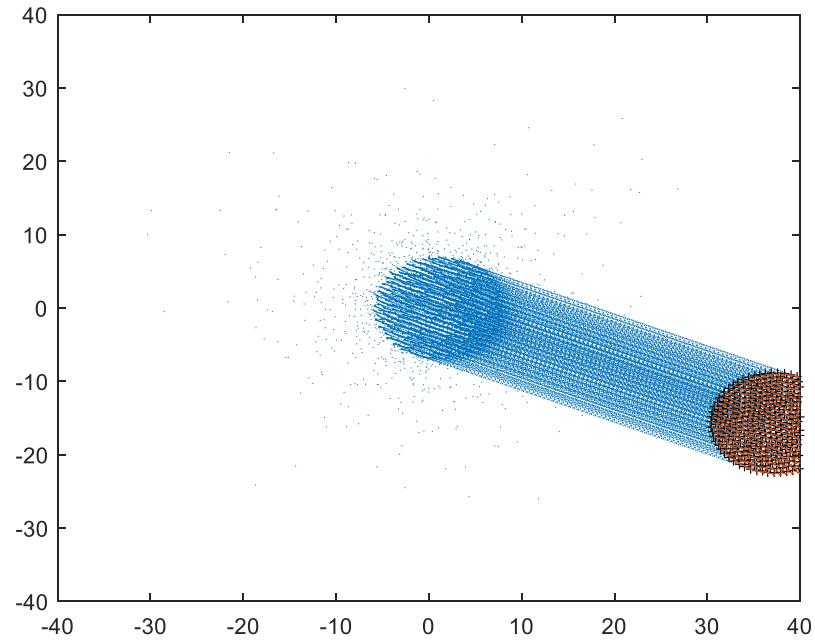
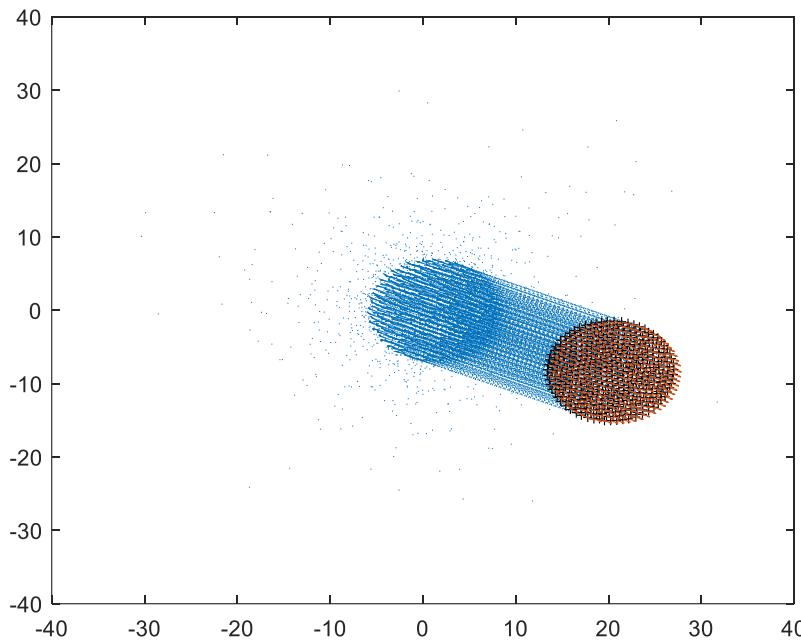
Implementação de um programa para simular um enxame de BOIDS.



BOIDS

✓ Desafio proposto aos alunos:

Implementação de um programa para simular um enxame de BOIDS.



Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

- ✓ O PSO foi proposto originalmente por Kennedy e Eberhart (*).

Como é que cada agente decide para onde se vai mover no espaço?

- ✗ O deslocamento de cada agente é efetuado baseado :
 - ★ na sua própria experiência.
 - ★ na experiência do grupo (bando, enxame....).

* Kennedy J. and Eberhart R. C. (1995) Particle swarm optimization. Proc. IEEE Int'l. Conf. on Neural Networks, IV, Vol. 4, pp. 1942–1948, Piscataway, NJ: IEEE Service Center

Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

- ✓ O PSO foi proposto originalmente por Kennedy e Eberhart (*).

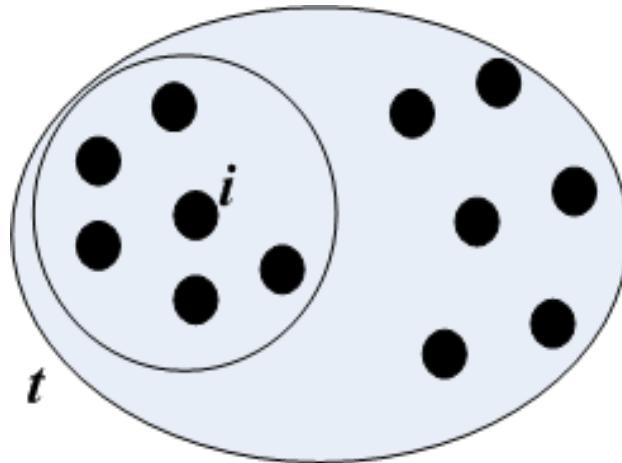
Como é que cada agente decide para onde se vai mover no espaço?

- ✗ O deslocamento de cada agente é efetuado baseado :
 - ★ na sua própria experiência.
 - ★ na experiência do grupo (bando, enxame....).

* Kennedy J. and Eberhart R. C. (1995) Particle swarm optimization. Proc. IEEE Int'l. Conf. on Neural Networks, IV, Vol. 4, pp. 1942–1948, Piscataway, NJ: IEEE Service Center

Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

- ✓ Considere-se a seguinte representação de um enxame de partículas:



- ✓ Cada partícula, i , é caracterizada por duas variáveis com dimensão d :

x_i – *Posição da partícula i*

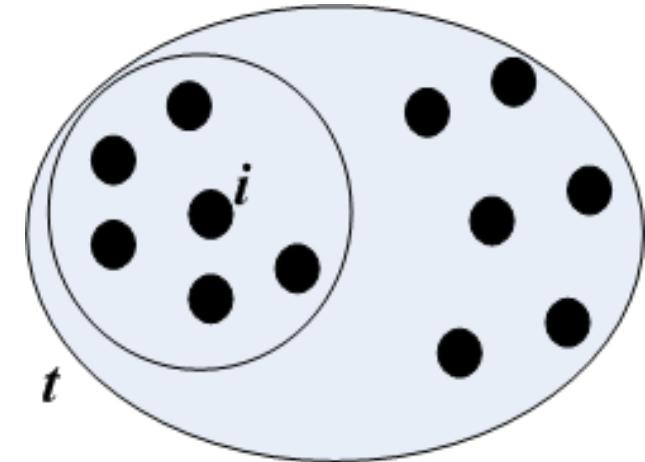
v_i – *Velocidade da partícula i*

Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

- ✓ Em cada iteração, t , guarda-se:

b_i – Melhor posição atingida por cada partícula

g – Melhor posição atingida pelo enxame (ou numa dada vizinhança)



- ✓ A velocidade é atualizada usando:

$$v_i(t+1) = v_i(t) + c_1 \varphi_1 [(b_i(t) - x_i(t))] + c_2 \varphi_2 [(g(t) - x_i(t))]$$

Representa a própria experiência da partícula

- ✓ A posição é atualizada usando:

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

Representa a partilha social com o bando

Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

- ✓ c_1 é chamada a **constante cognitiva** (e.g. $c_1=2$)
- ✓ c_2 é chamada a **constante social** (e.g. $c_2=2$)
- ✓ φ_1 e φ_2 são dois números aleatórios gerados no intervalo [0,1]

$$v_i(t+1) = v_i(t) + c_1 \varphi_1 [(b_i(t) - x_i(t))] + c_2 \varphi_2 \cdot (g(t) - x_i(t))$$

Represents the particle's own experience

Represents the social sharing with the flock

Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

- ✓ Consideraremos agora **somente a parte cognitiva** com $c_1=1$:

$$v_i(t+1) = v_i(t) + c_1 \phi_1 \cdot (b_i(t) - x_i(t))$$

Expressão tem semelhanças com:

$$v_{k+1} = v_k - \alpha f'(v_k)$$

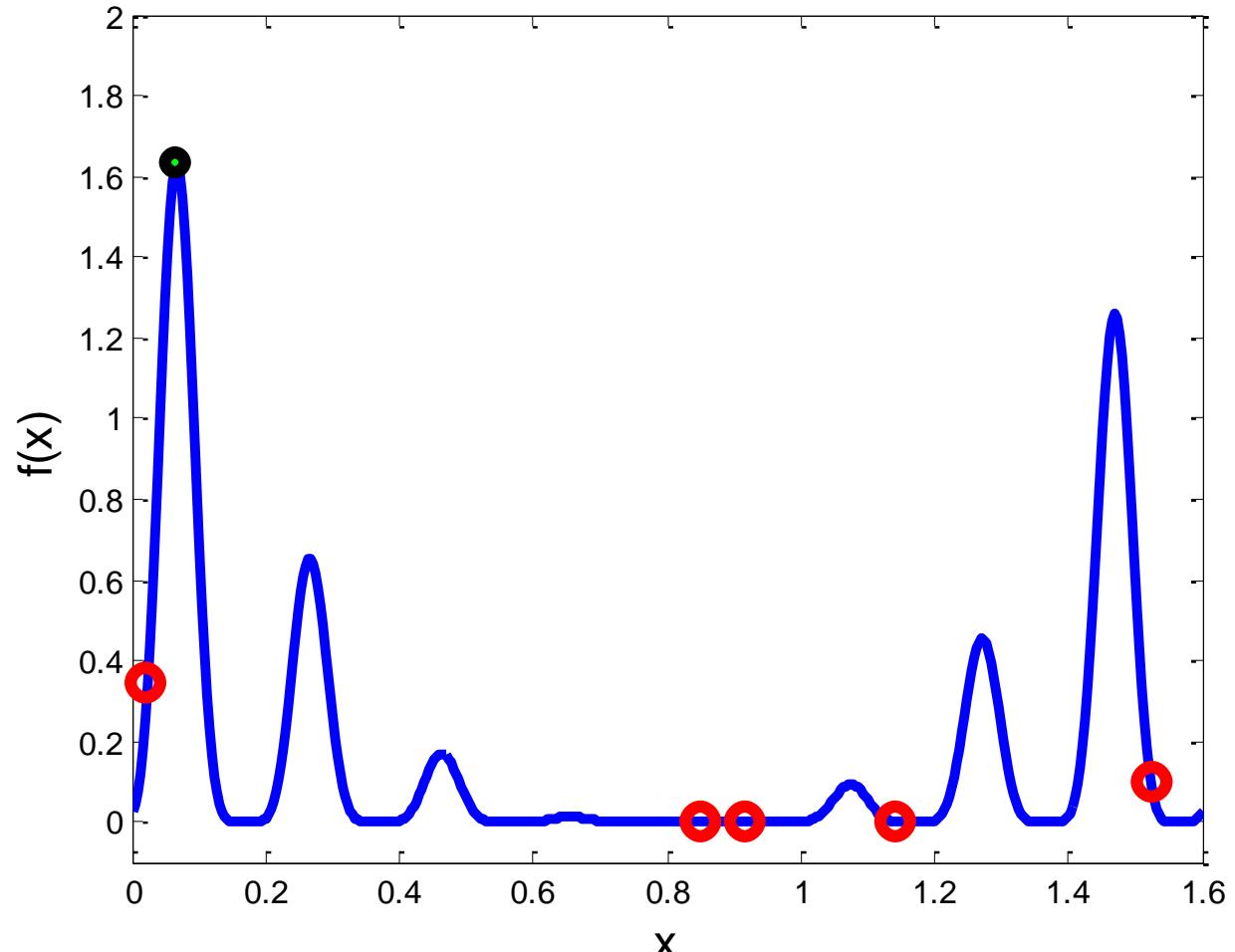
- ✓ Ou seja se executarmos um PSO só com a componente cognitiva de certa forma estamos a executar um algoritmo do gradiente descendente (ou ascendente) em paralelo

Neste caso não há qualquer partilha de informação no enxame e cada partícula só usa a memorização da sua melhor posição até ao momento.

Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 1:

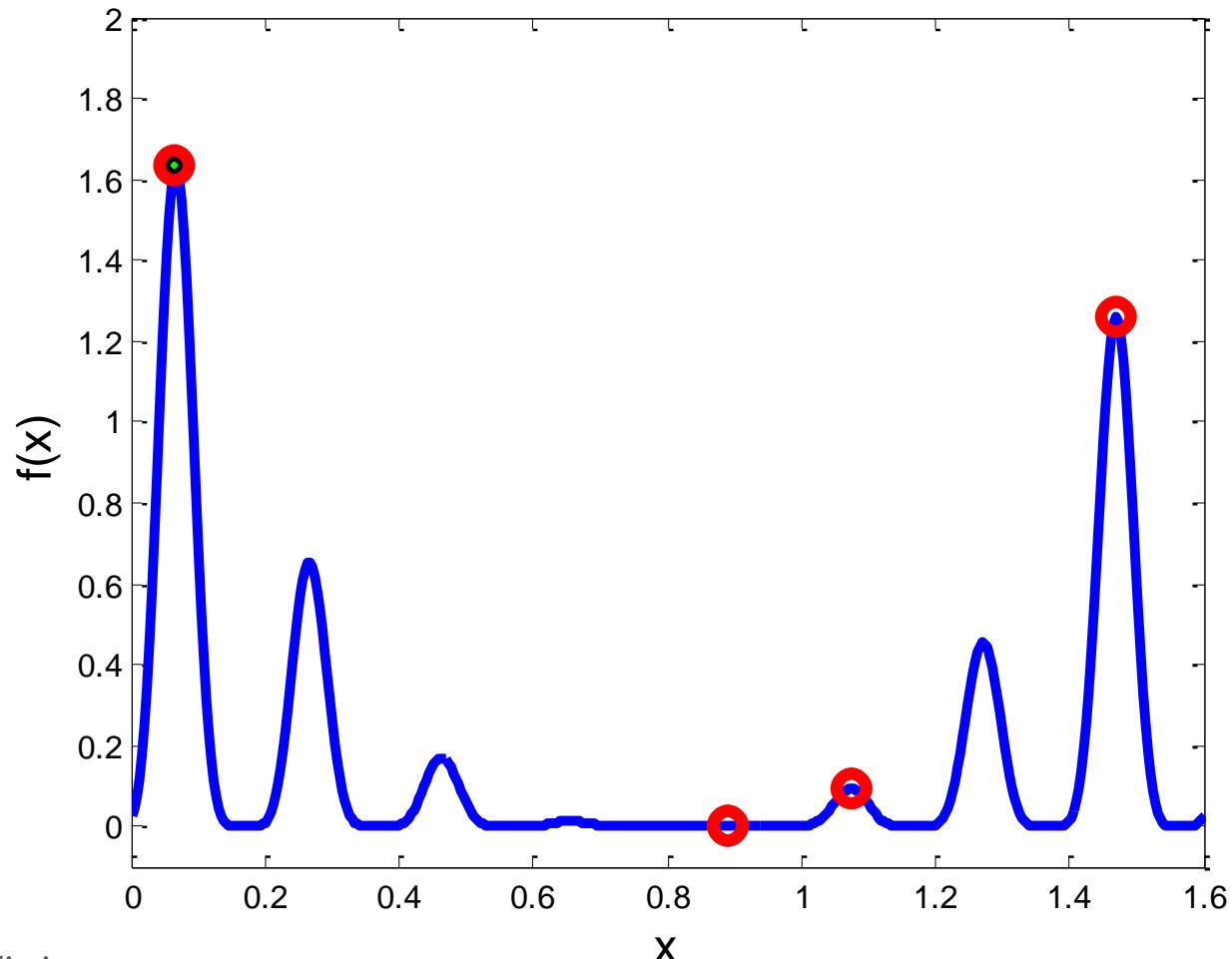
- ✓ Distribuição inicial aleatória:
- ✓ (Mini) Enxame com 5 partículas.
- ✓ Velocidade máxima de cada partícula limitada a 1.6/100.



Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 1:

✓ Distribuição no fim de 100 iterações:

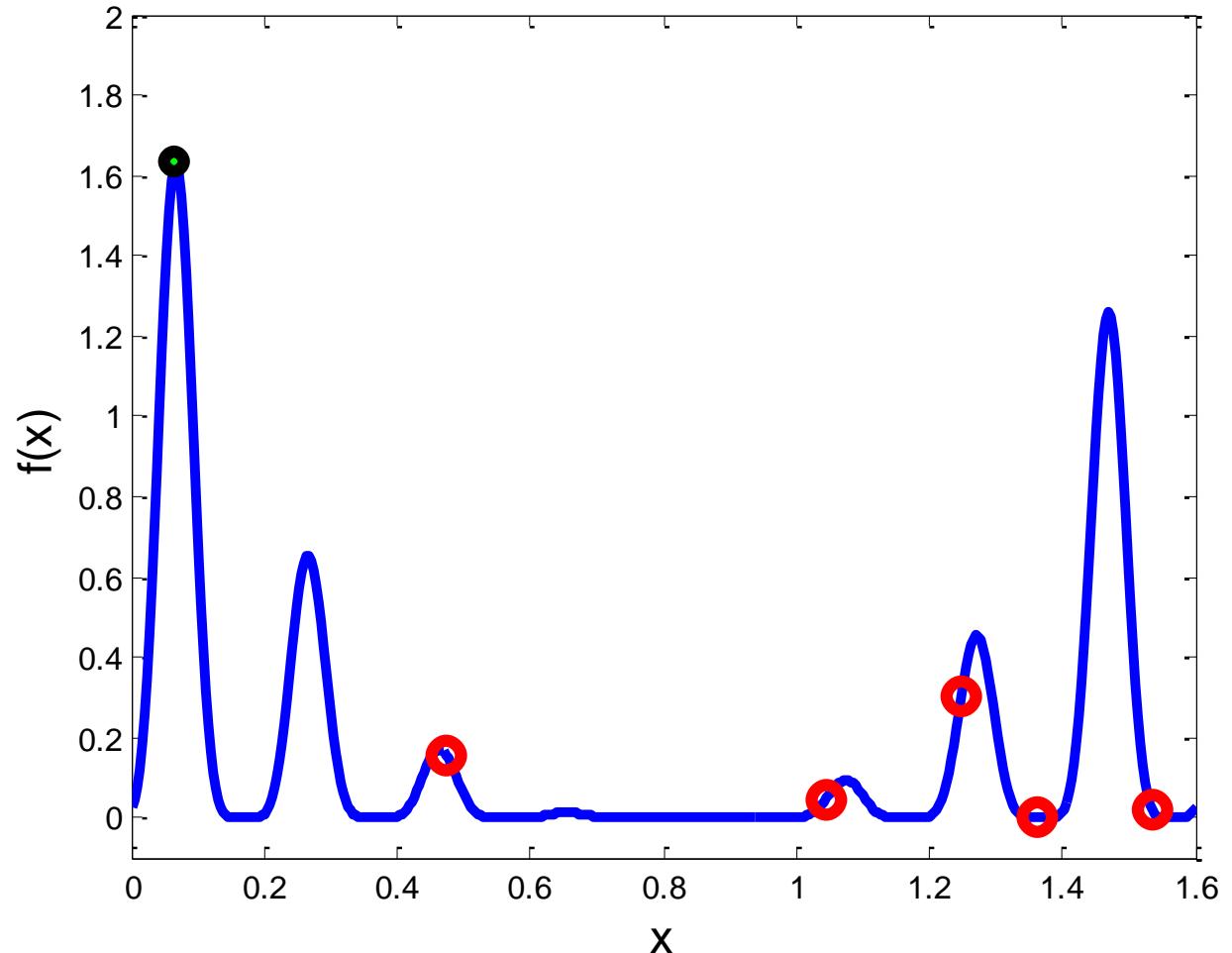


Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 2:

- ✓ Velocidade máxima de cada partícula limitada a 1.6/100.

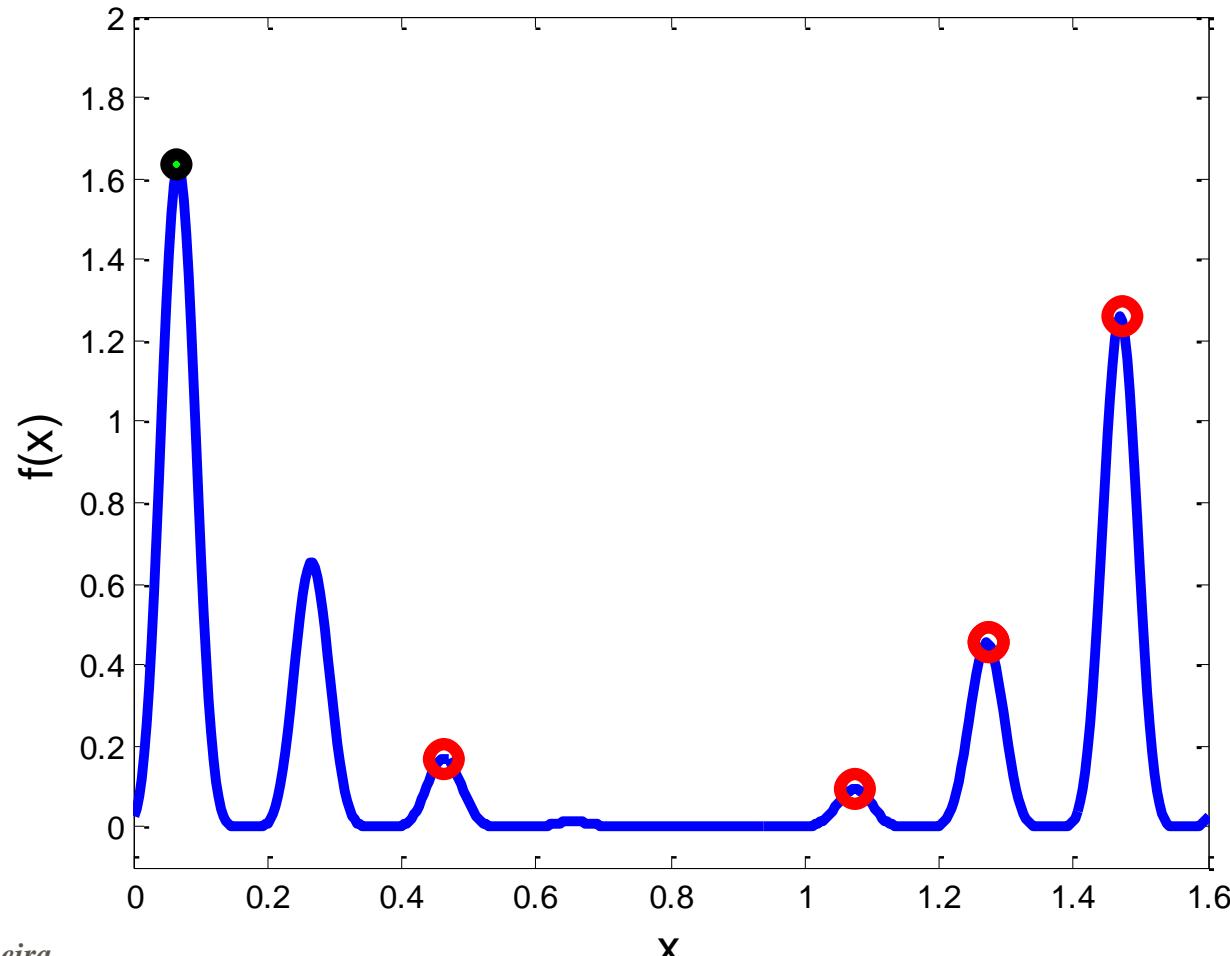
- ✓ Distribuição inicial aleatória:



Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 2:

✓ Distribuição no fim de 100 iterações:

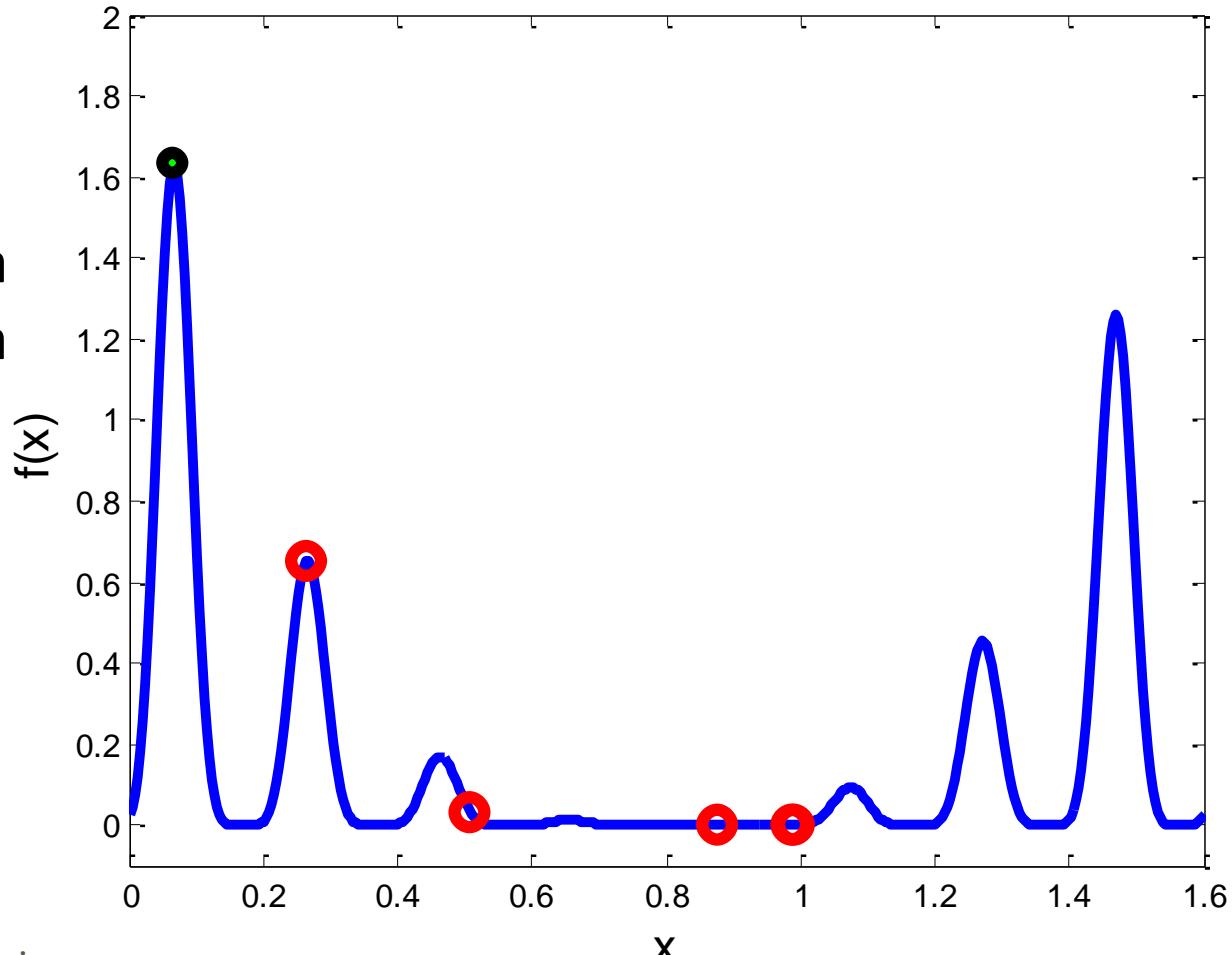


Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 3:

- ✓ Velocidade máxima de cada partícula limitada a **1.6/10**.

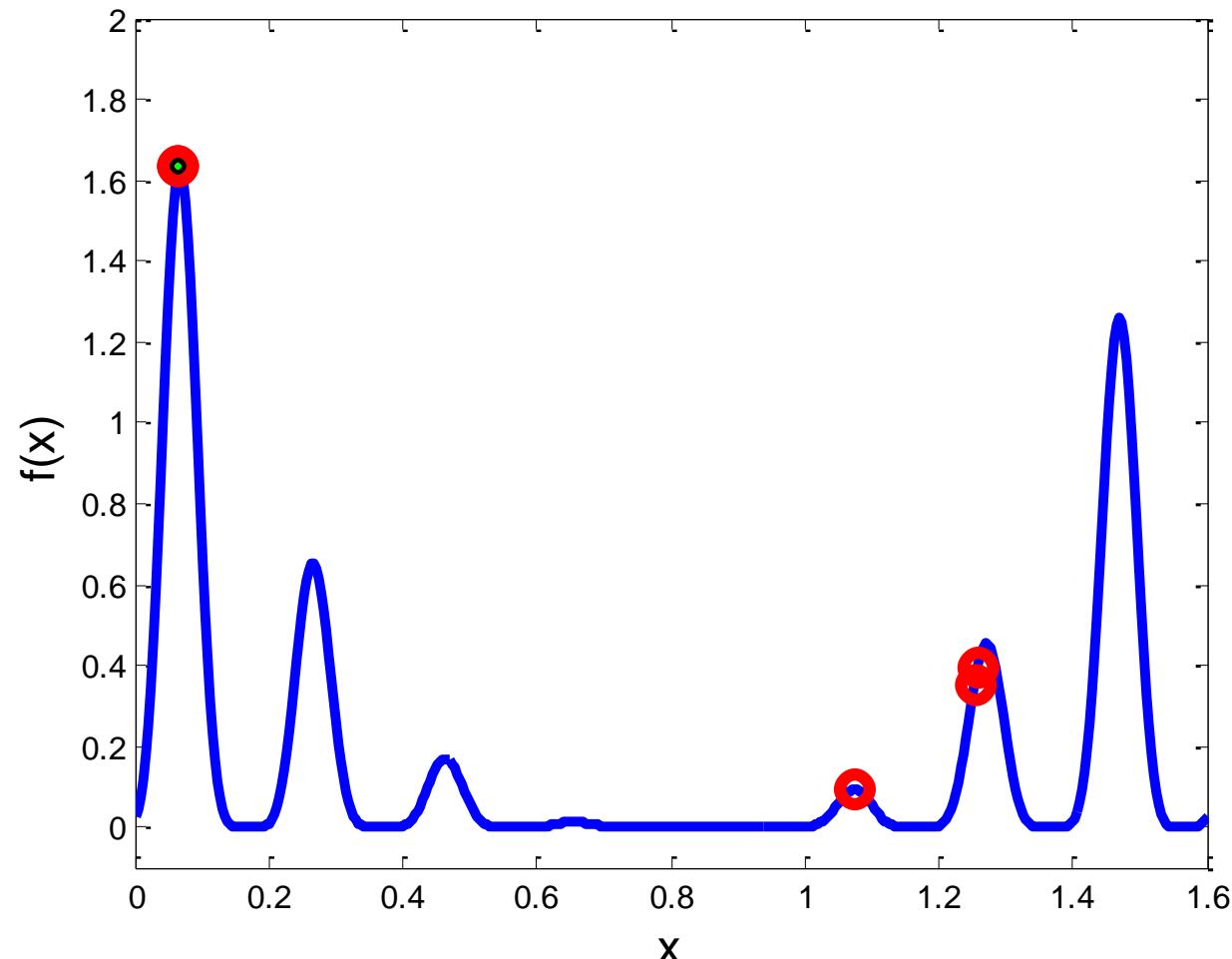
- ✓ Distribuição inicial aleatória:



Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 3:

✓ Distribuição no fim de 100 iterações:

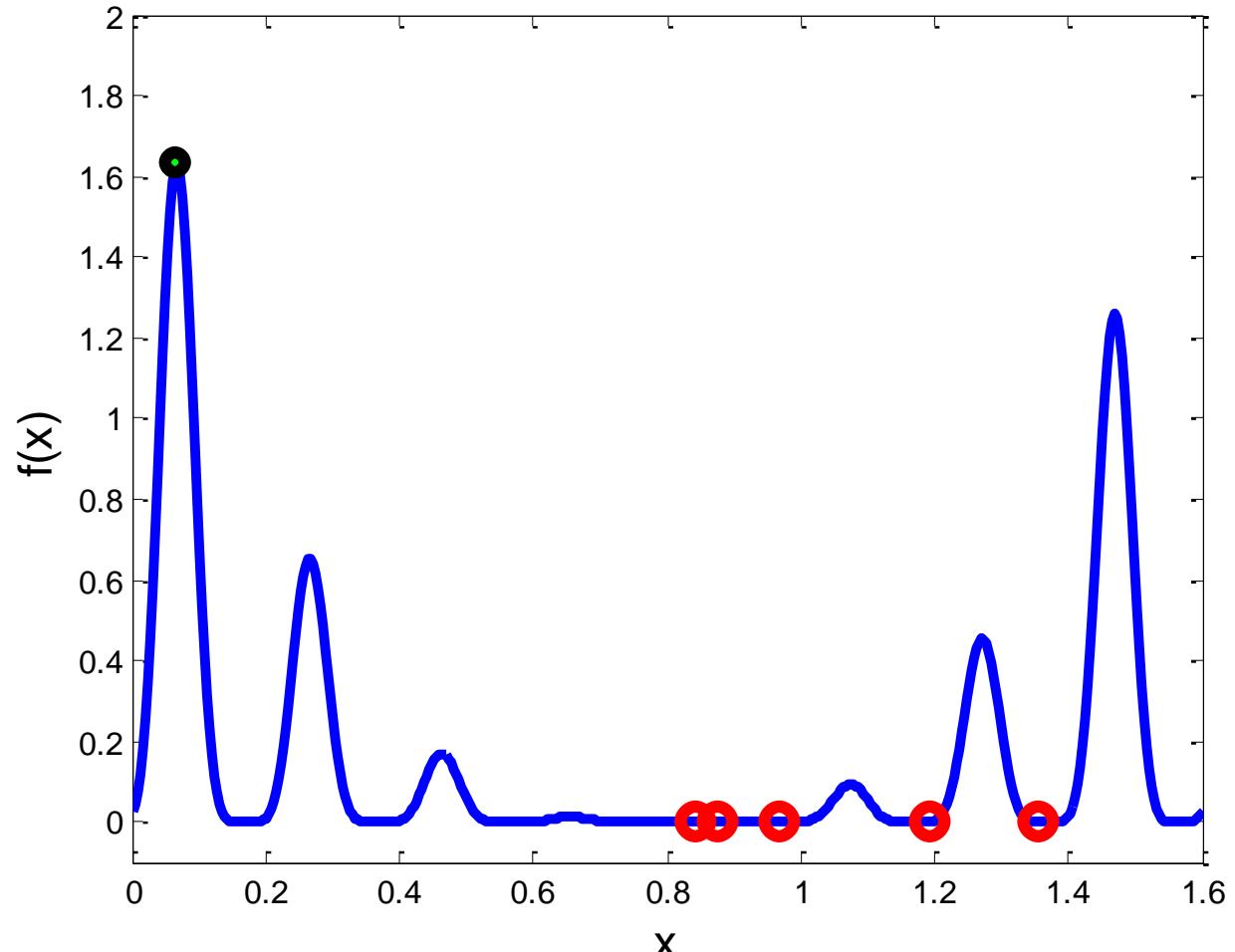


Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 4:

- ✓ Velocidade máxima de cada partícula limitada a **1.6/10**.

✓ Distribuição inicial aleatória:

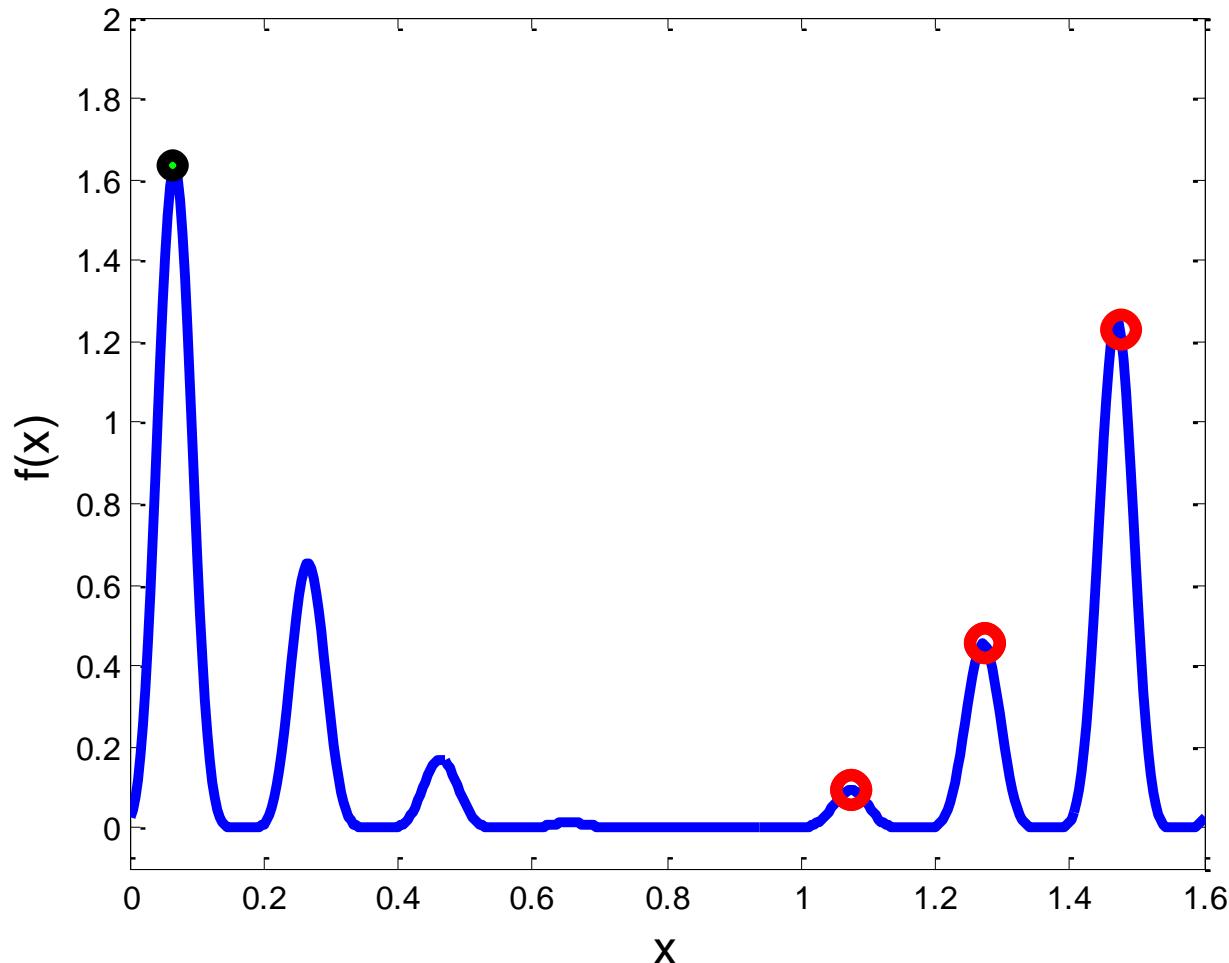


Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 4:

- ✓ E se a velocidade máxima de cada partícula for limitada a : **1.6/2** aumentará o sucesso da pesquisa?

✓ Distribuição no fim de 100 iterações:



Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

- ✓ Consideraremos agora **somente a parte social** com $c_2=1$:

$$v_i(t + 1) = v_i(t) + c_2 \varphi_2(g(t) - x_i(t))$$

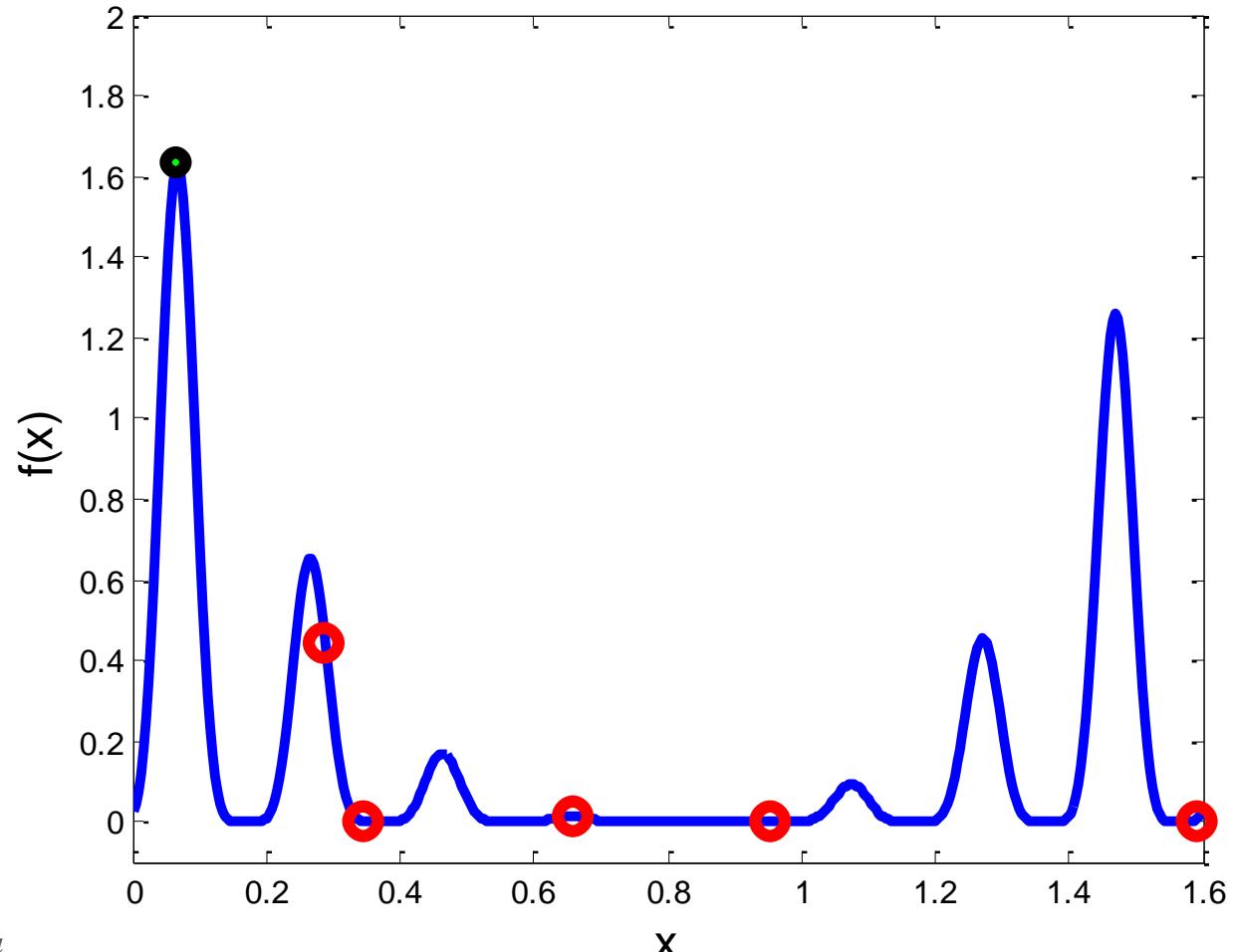
Neste caso qualquer elemento da população vai partilhar a sua posição com o enxame e a melhor posição global vai atrair o enxame.

Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 5:

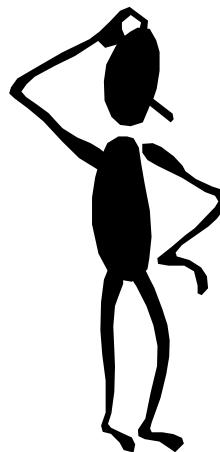
- ✓ Velocidade máxima de cada partícula limitada a **1.6/2**.

- ✓ Distribuição inicial aleatória:

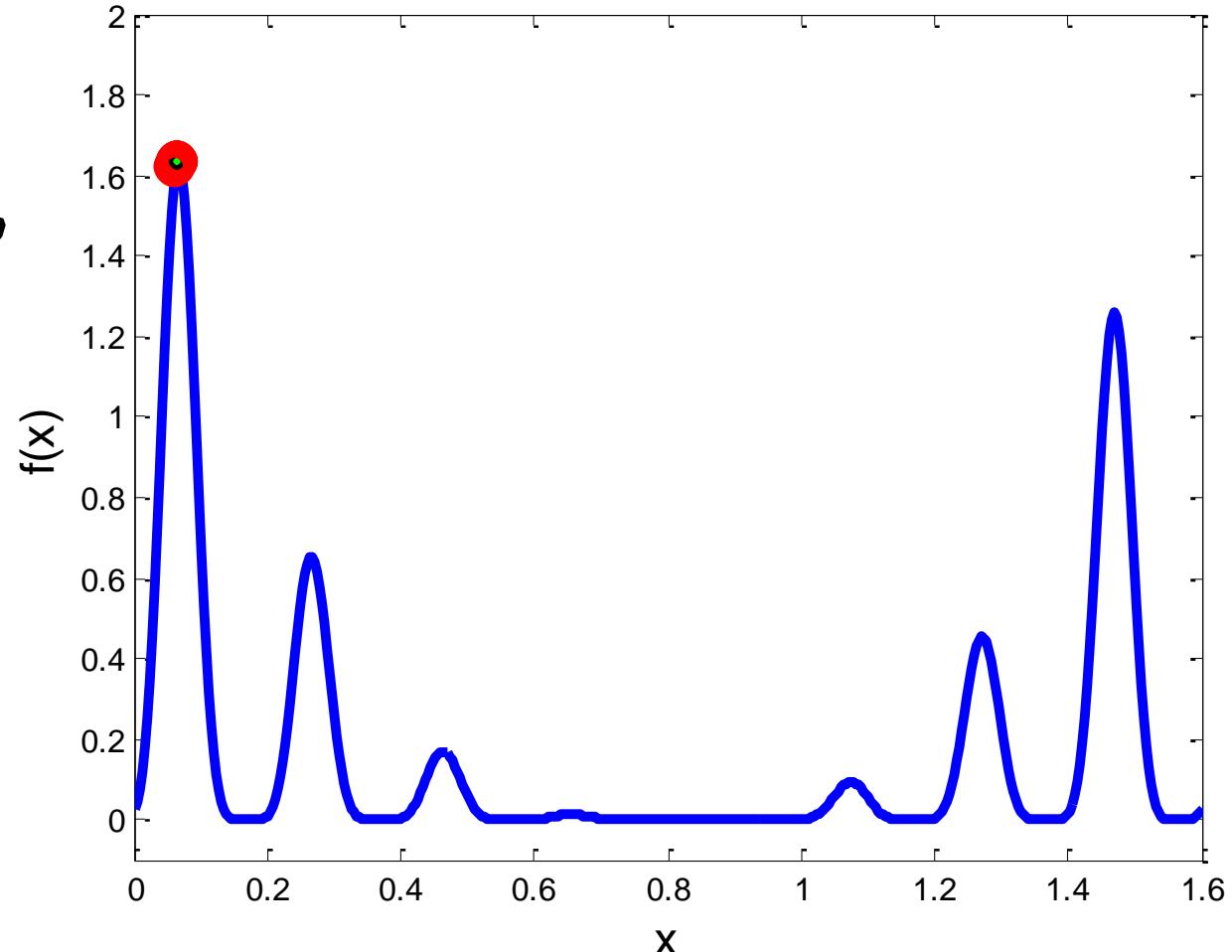


Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 5:



✓ Distribuição no fim de 100 iterações:



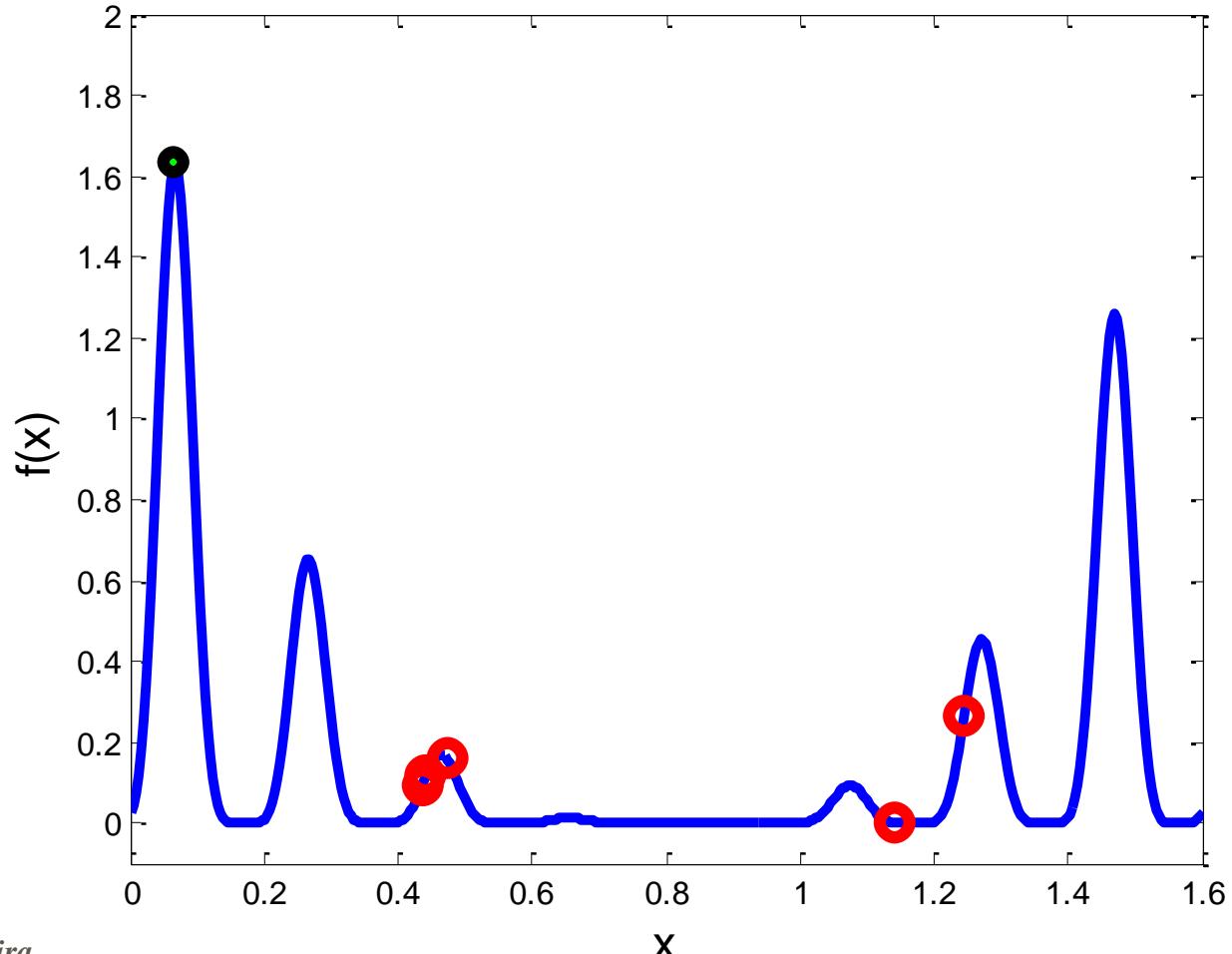
✓ Como se pode observar as 5 partículas convergiram para a zona do máximo global.

Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)



- ✓ Velocidade máxima de cada partícula limitada a **1.6/2**.

✓ Distribuição inicial aleatória:



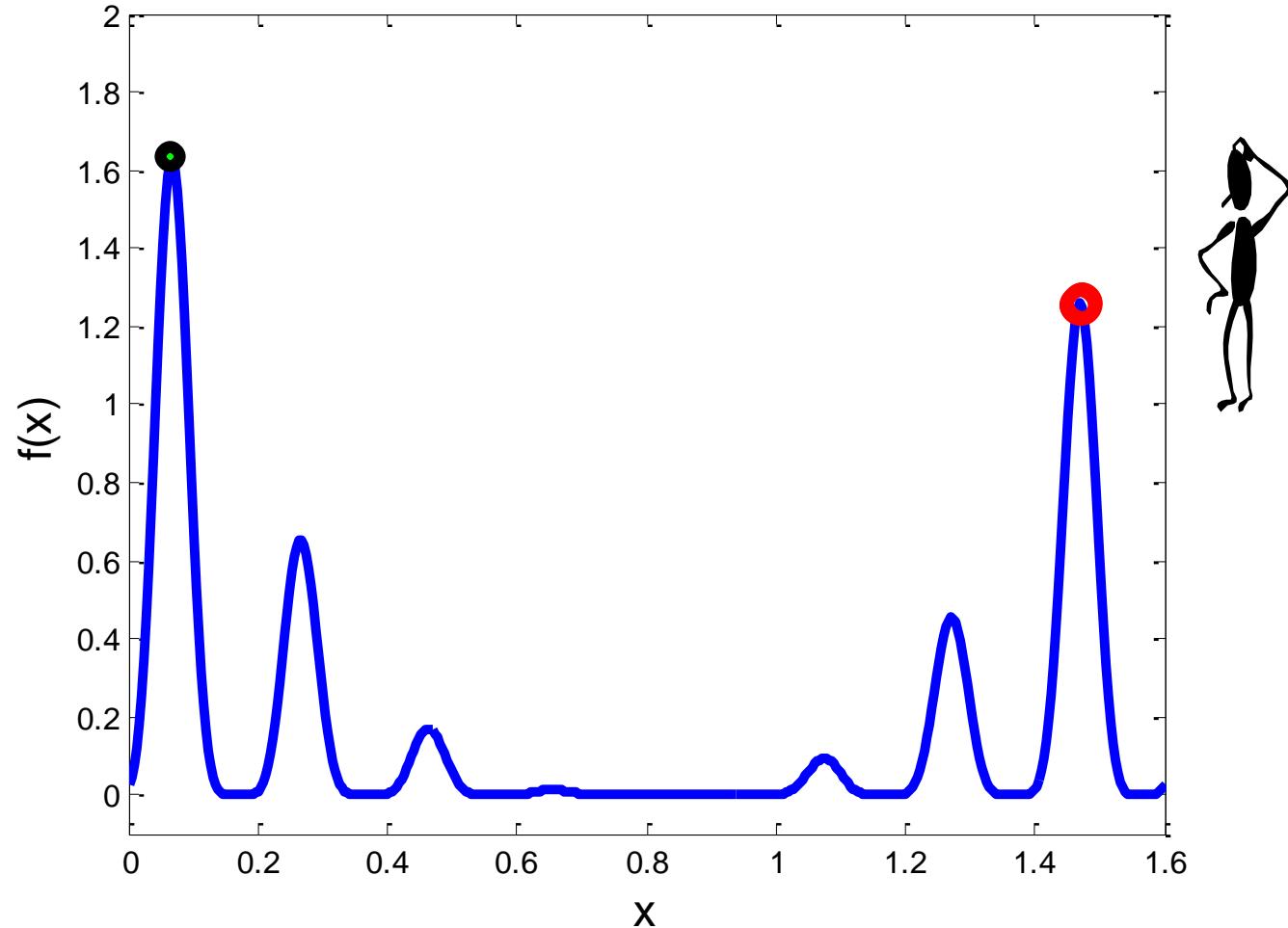
Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)



Exemplo 6:

- ✓ Como se pode observar as 5 partículas convergiram para a zona de um **máximo local**.

✓ Distribuição no fim de 100 iterações:



Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

$t = 0$

inicializar o enxame $X(t)$

avaliar $X(t)$

while(!(critério de paragem))

$t = t + 1$

atualizar os melhores locais e global(ais)

atualizar a velocidade das partículas

atualizar a posição das partículas

determinar $X(t + 1)$

avaliar $X(t + 1)$

end while

Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

- ✓ Uma melhoria significativa no desempenho do PSO foi a inclusão de um fator de inércia para ponderar o valor da velocidade anterior:

$$v_i(t+1) = \omega v_i(t) + c_1 \varphi_1 \cdot (b_i(t) - x_i(t)) + c_2 \varphi_2 \cdot (g(t) - x_i(t))$$

Fator de Inércia

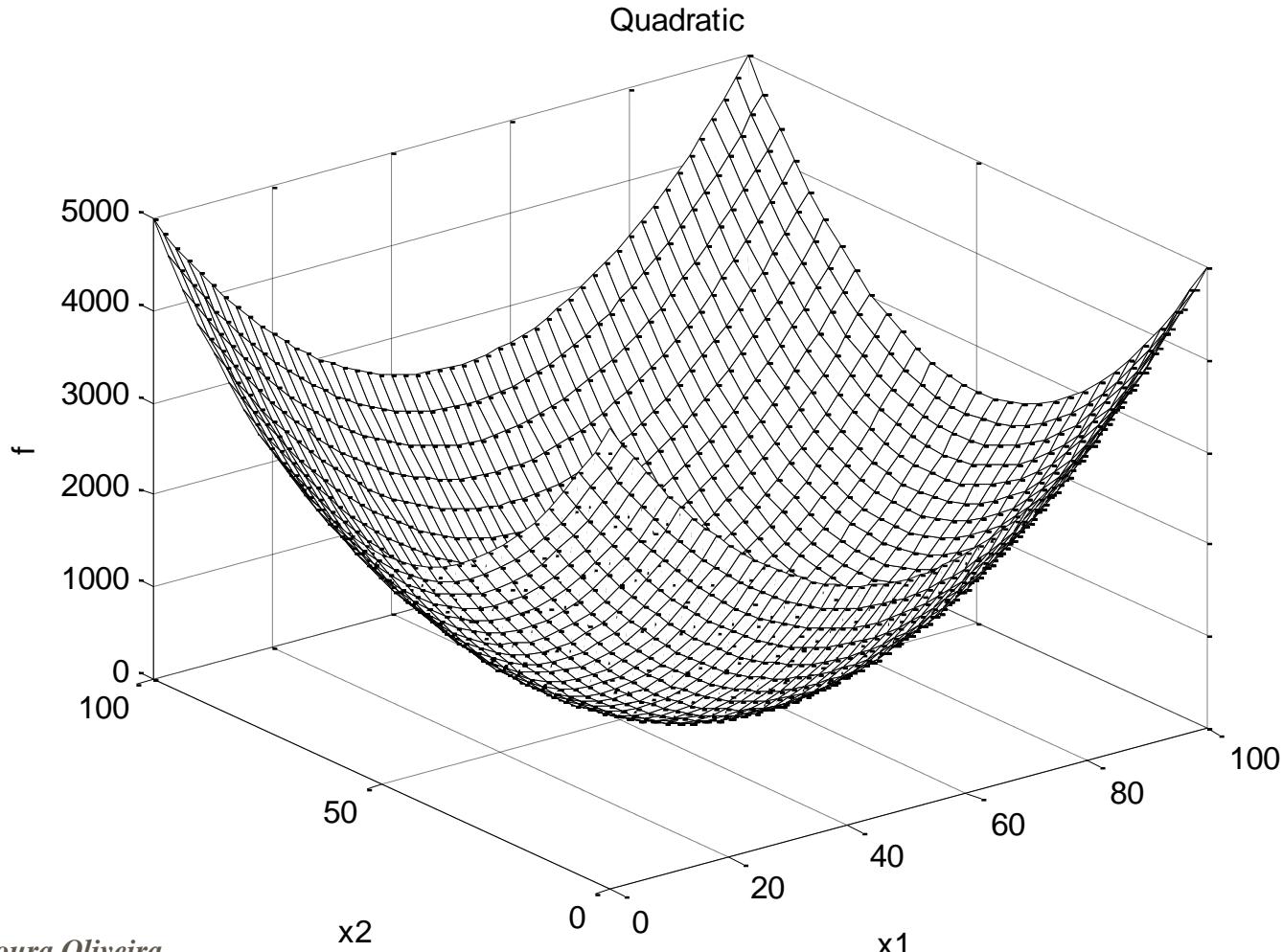
- ✓ É usual atribuir um valor mais elevado a ω no início da pesquisa e ao longo da mesma diminuir este valor até a um valor mínimo. Estabelece-se assim um **compromisso importantíssimo** entre:

Exploração versus Especialização

Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 7:

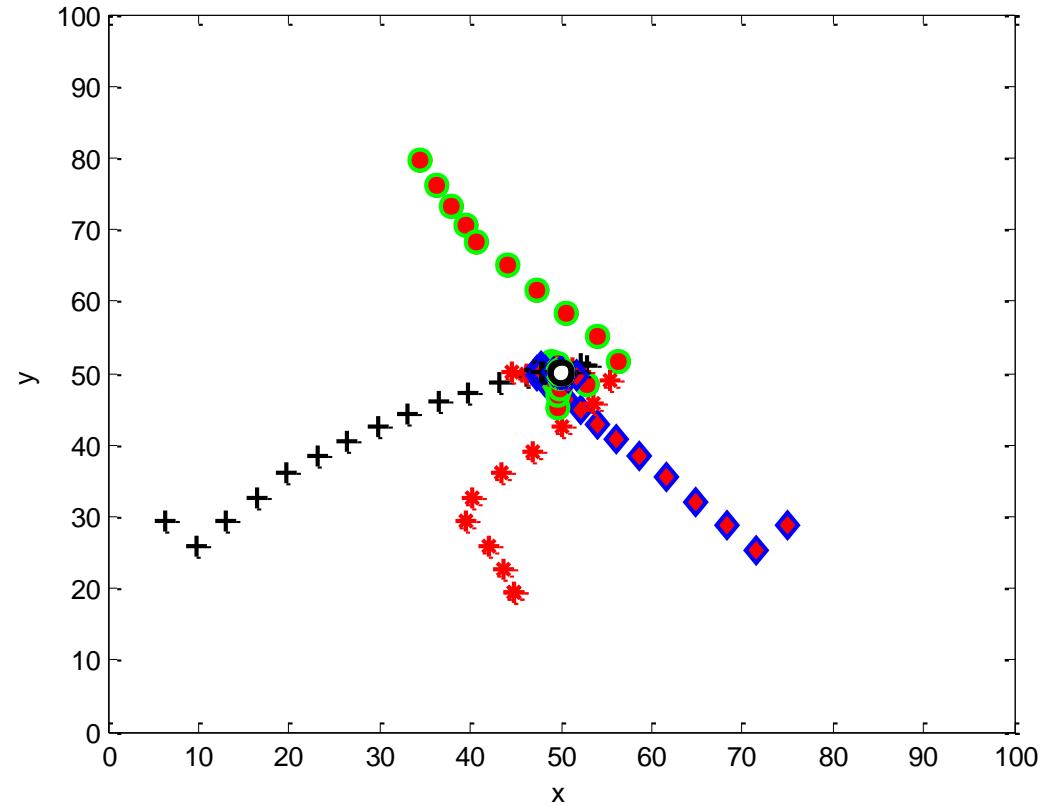
$$f(x, y) = (x - 50)^2 + (y - 50)^2$$



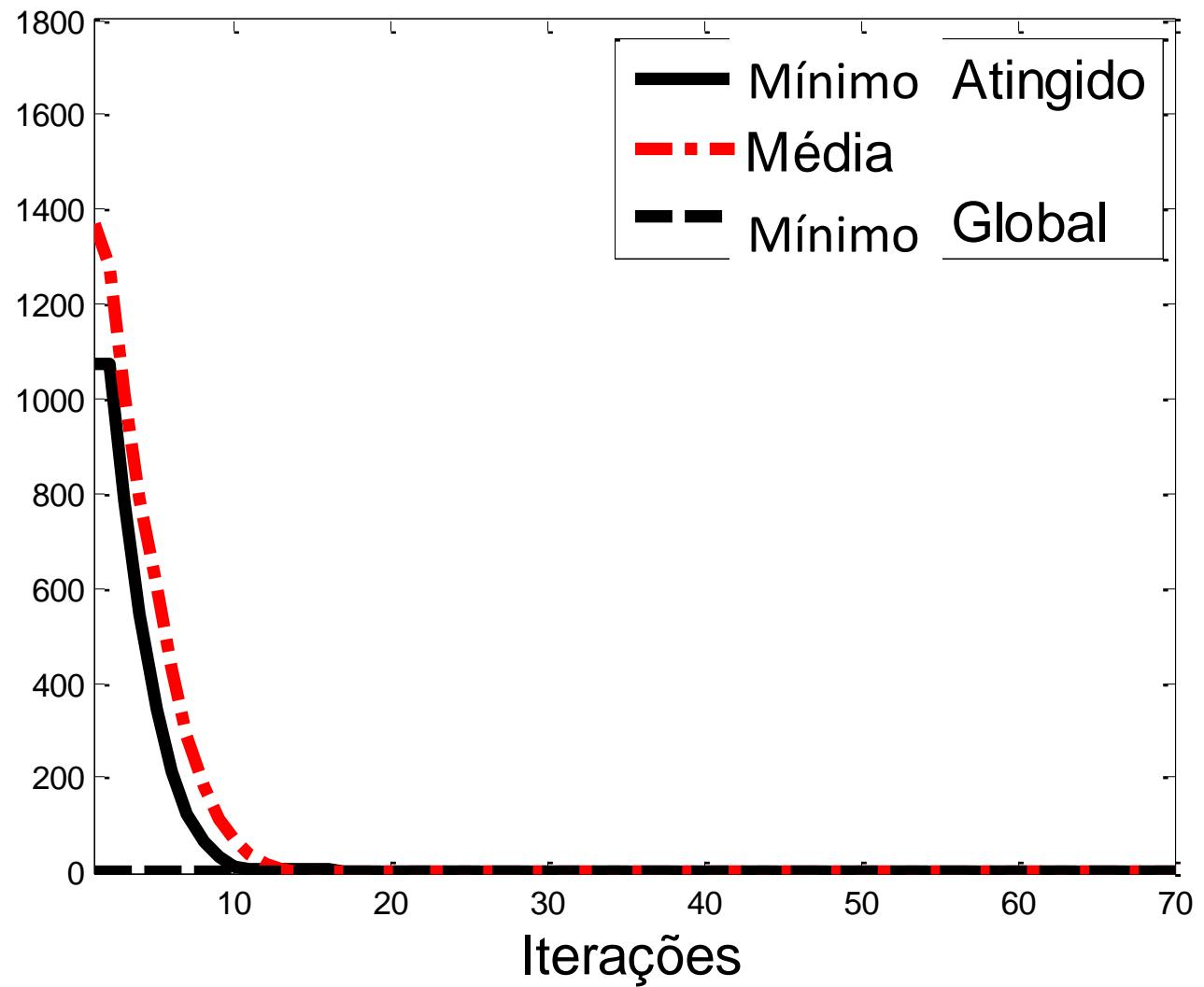
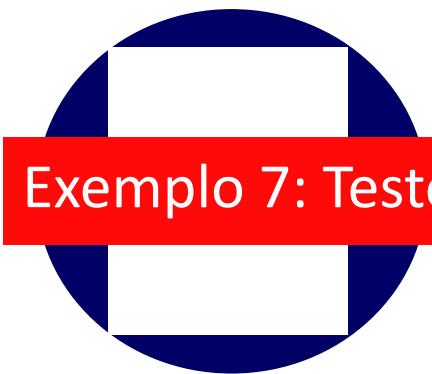
Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Exemplo 7: Teste 1

- 70 iterações;
- 4 partículas;
- Inicialização aleatória no espaço todo $[0,100]$, $[0,100]$
- $v_{\text{inicial}} = 0$



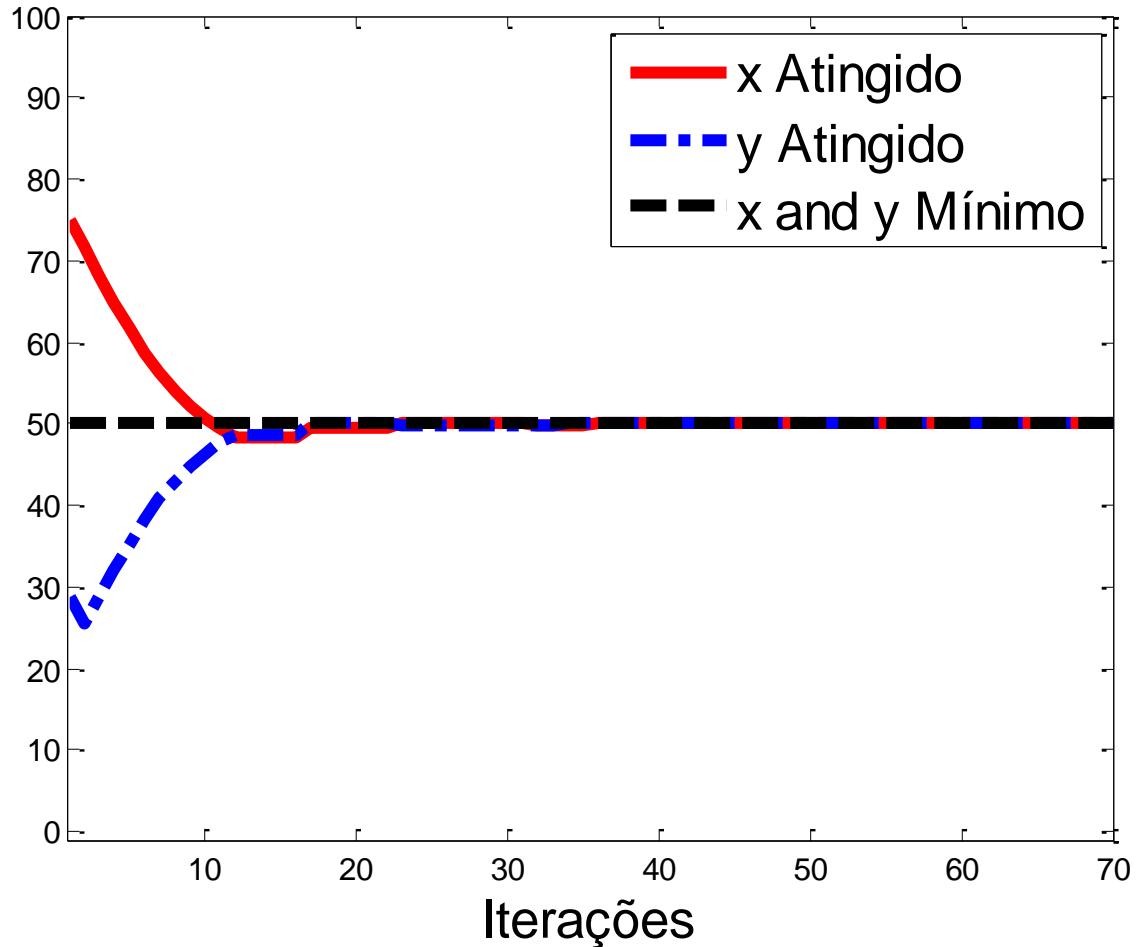
Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)



Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)



- Mais testes para este exemplo disponível no vídeo sobre o PSO disponível em:



Algoritmo de Otimização por Enxame de Partículas (*Particle Swarm Optimization*, PSO)

Tópicos Selecionados

- 1. Como limitar a velocidade?** É usual limitar a velocidade a metade da amplitude do espaço de pesquisa por dimensão
- 2.** Existe uma versão que não necessita da limitação de velocidade usando um fator de constrição (Clerc e Kennedy, 2002)
- 3. Como lidar com partículas que violam os limites do espaço de pesquisa?**
- 4.** Existem centenas de variantes e híbridos do algoritmo PSO.

