

Inteligência de Enxames

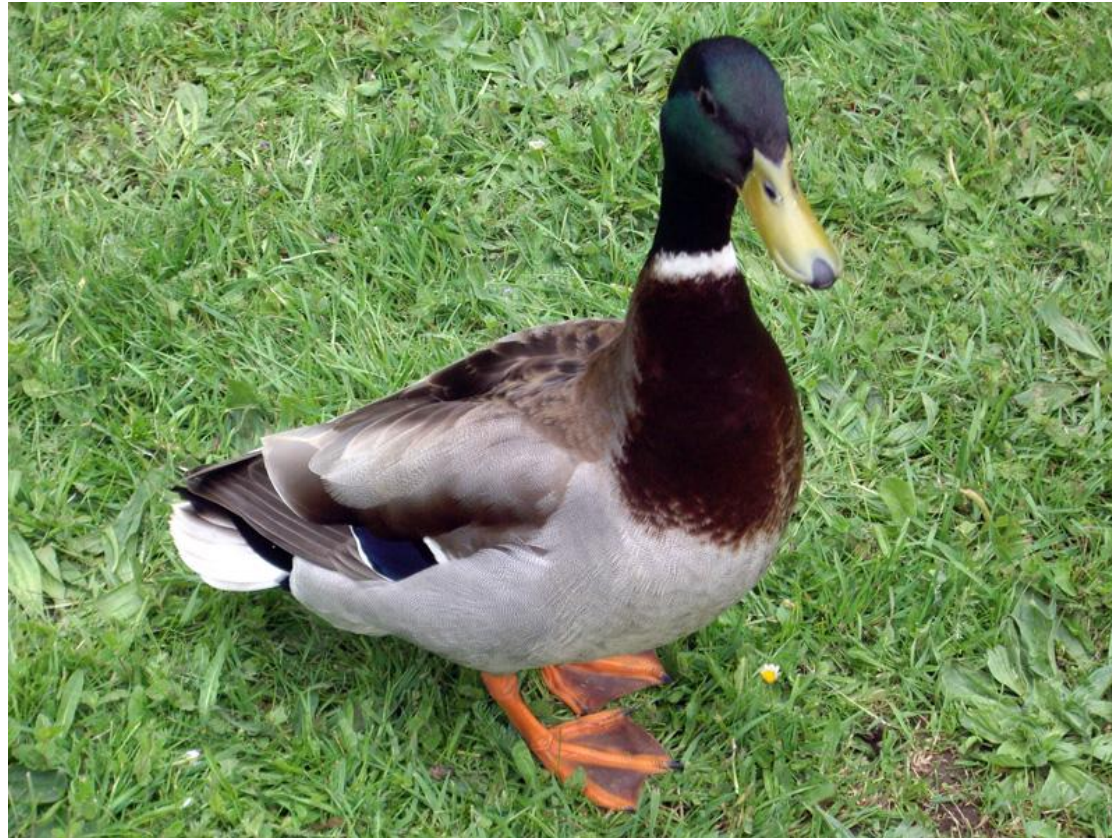


Foto: <http://fotolog.javivicente.com/index.php?showimage=22>

Fabricio Breve - fbreve@gmail.com



PARTICLE SWARM OPTIMIZATION

(OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS)

Particle Swarm Optimization

- Proposta em 1995 por:
 - James Kennedy: psicólogo social
 - Russ Eberhart: engenheiro elétrico
- Surgiu a partir de experimentos que modelavam o comportamento de enxames
 - Visto em várias espécies de pássaros e de peixes
- Objetivo inicial era estudar o comportamento social e cognitivo de animais

Particle Swarm Optimization

- **Motivação:**
 - Criar uma simulação do comportamento social
- **População de indivíduo capazes de interagir**
 - Com o ambiente
 - Uns com os outros
 - Particularmente com um conjunto de vizinhos

Particle Swarm Optimization

- **Princípio de Adaptação Cultural**
 - Cada indivíduo:
 - Tem sua própria experiência e sabe quão boa ela é
 - Tem algum conhecimento de como os indivíduos de sua vizinhança tem se desempenhado
 - Correspondem a:
 - Aprendizado individual
 - Transmissão cultural ou social

Particle Swarm Optimization

- Probabilidade de um indivíduo tomar uma decisão depende de:
 - Quanto sucesso ele teve com essa decisão no passado
 - Influências sociais
- Algoritmo PSO
 - Indivíduos são influenciados pelo sucesso de sua vizinhança social

Particle Swarm Optimization

- Princípios de adaptação cultural:
 - Avaliação
 - Capacidade de um indivíduo “sentir” o ambiente e quantificar um grau de quão bom ele é em relação a algum parâmetro ou tarefa
 - Comparação
 - Pessoas usam outros como padrões para se avaliar, o que serve como motivação para aprender e mudar
 - Imitação
 - Compreende tomar a perspectiva de outra pessoa, não apenas imitando um comportamento, mas também entendendo seus propósitos

Particle Swarm Optimization

- Principio Básico
 - Comportamento é governado por regras semelhantes em todas as sociedades
 - Compartilhamento de informação entre os indivíduos pode oferecer alguma vantagem
- PSO
 - Indivíduos aprendem com sua própria experiência e com a experiência dos outros
 - Se avaliam e se comparam aos seus vizinhos e imitam apenas os vizinhos superiores a eles próprios

Particle Swarm Optimization

- Tem sido aplicado com sucesso a vários problemas de busca e otimização
 - Engenharia
 - Computação
- Procedimento de otimização numérica
 - Busca num espaço l -dimensional de valores reais
 - Reveja slides de computação evolutiva sobre busca no espaço de possíveis soluções

Particle Swarm Optimization

- Vários modelos de movimento de organismos em bandos e cardumes foram investigados
 - Reynolds estudou coreografia dos bandos de pássaros
 - Heppner procurou as regras que possibilitam vôo sincronizado de grandes bandos de pássaros, mesmo com:
 - Pássaros mudando de direção subitamente
 - Pássaros se espalhando e re-agrupando

Bando de Pássaros

- A lição dos gansos
 - Estudo dos motivos que levam os gansos a voarem em forma de “V” levaram a 5 descobertas

Bando de Pássaros

- Descoberta I
 - À medida em que cada pássaro bate suas asas, ele cria sustentação para a ave seguinte
- Voando em formação "V", o grupo consegue voar pelo menos 71% a mais do que se cada pássaro voasse isoladamente

Bando de Pássaros

- Descoberta 2
 - Sempre que um ganso sai fora da formação "V", ele sente uma forte resistência ao tentar voar sozinho
 - Volta rapidamente à formação anterior
 - Por se beneficiar do poder de sustentação do(s) pássaro(s) imediatamente à sua frente

Bando de Pássaros

- Descoberta 3
 - Quando o ganso líder se cansa, ele vai para o fundo do "V"
 - Enquanto um outro ganso líder assume a ponta

Bando de Pássaros

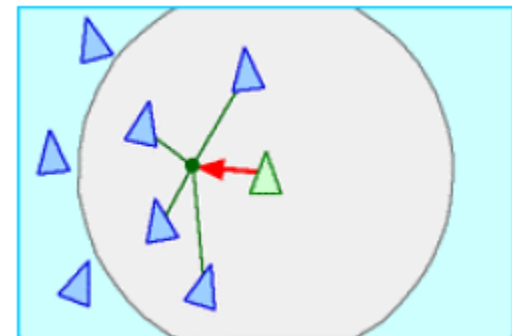
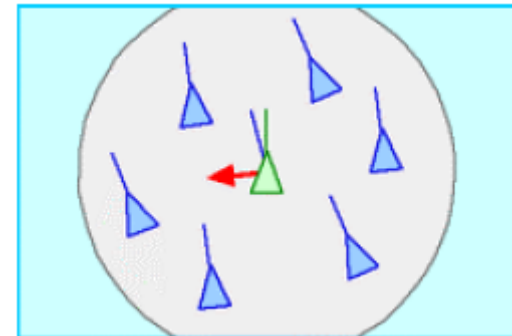
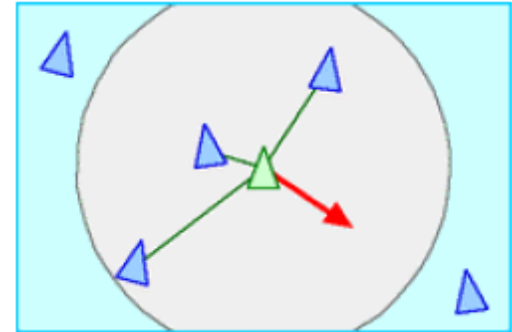
- Descoberta 4
 - Os gansos de trás grasnam para encorajar os da frente e manter o ritmo e a velocidade

Bando de Pássaros

- Descoberta 5
 - Quando um ganso adoece ou se fere e deixa o grupo, dois outros gansos saem da formação e acompanham o ganso doente até que a dificuldade seja superada
 - Mais tarde, os três juntos reiniciam a jornada ou se unem a uma outra formação, até reencontrarem o antigo grupo

Bando de Pássaros

- Cada pássaro segue regras simples
 - Separação
 - Evitar colisão com outros pássaros do bando
 - Alinhamento
 - Voar no mesmo sentido dos demais pássaros do bando
 - Coesão
 - Voar em direção à posição média do bando



Particle Swarm Optimization

- De acordo com E O Wilson (socio-biologo):
 - “Pelo menos em teoria, membros individuais de um cardume podem se beneficiar das descobertas e experiências anteriores de outros membros do cardume na busca por alimentos
 - Essa vantagem pode se tornar decisiva, compensando as desvantagens da competição por alimentos”
- Compartilhamento de informação dentro de uma espécie gera uma vantagem evolutiva
 - Hipótese fundamental para o desenvolvimento de PSO

Particle Swarm Optimization

- Peixes e pássaros ajustam seu movimento físico para:
 - Evitar predadores
 - Procurar por alimento e companheiro(a)
 - Otimizar parâmetros ambientais
 - Temperatura

Particle Swarm Optimization

- Baseado em modelos desenvolvidos pelo biólogo Frank Heppner
 - Exibe os mesmos comportamentos para bandos de pássaros encontrados em outros modelos
- Apresenta uma característica adicional:
 - Os pássaros são atraídos para uma área de abrigo (poleiro)

Particle Swarm Optimization

- Supor um grupo de pássaros procurando por um abrigo
 - Procura ocorre de forma aleatória
 - Existe apenas um abrigo
 - Nenhum pássaro sabe onde está o abrigo
 - Mas todos sabem, a cada instante de tempo, quão longe estão do abrigo
 - Qual a melhor estratégia para encontrar o abrigo?

Particle Swarm Optimization

- Possível estratégia:
 - Seguir o pássaro que está mais próximo do abrigo
 - Estratégia usada pelo PSO

Particle Swarm Optimization

- Nas simulações realizadas
 - Cada solução = um pássaro voando no espaço de busca
 - Os pássaros inicialmente voavam ao acaso
 - Sem um destino definido
 - Espontaneamente, os pássaros formavam bandos
 - Até um dos pássaros sobrevoar um abrigo (ou uma fonte de alimentos)
 - Se o desejo de pousar tivesse um valor maior que o desejo de permanecer voando (valores programados), o pássaro deixava o bando e pousava

Particle Swarm Optimization

- Nas simulações realizadas (cont.)
 - Pássaros usavam regras simples para definir sua direção e velocidade
 - Cada pássaro tentava:
 - Ficar no meio dos pássaros mais próximos
 - Evitar colidir com esses pássaros
 - Como resultado
 - Quando um pássaro saía do bando para ir ao abrigo, pássaros próximos acompanhavam o movimento
 - Cada vez mais pássaros desciam no abrigo
 - Até que o bando inteiro tivesse descido

Particle Swarm Optimization

- Encontrar um abrigo é análogo a encontrar uma solução no universo de possíveis soluções
- Maneira como o pássaro que encontrou o abrigo lidera seus vizinhos em direção a ele, aumenta a chance desses também encontrarem o abrigo
 - Encontra eco na definição sócio-cognitiva de mente
 - Que a mente, e portanto a inteligência, é social

Particle Swarm Optimization

- Partículas podem voar sobre o espaço de soluções e pousar na melhor solução
 - Como evitar que as partículas pousem em qualquer solução?
 - E não apenas na melhor?
 - Princípio importante da visão sócio-cognitiva:
- Indivíduos aprendem a partir do sucesso de seus vizinhos
 - Como implementar este princípio em um algoritmo?

Particle Swarm Optimization

- Não têm inteligência
- Voam através das coordenadas de um espaço n-dimensional
- Quando se movem, mandam suas coordenadas para uma função de aptidão
 - Mede aptidão da partícula
 - Relacionada ao problema investigado

Partículas

- Partículas lembram de:
 - Suas coordenadas atuais
 - Sua velocidade atual
 - Quão rápido está se movendo ao longo das dimensões do espaço de soluções
 - Seu melhor valor de aptidão até então recebido
 - Coordenadas de onde foi calculado seu melhor valor de aptidão

Partículas

- Cada partícula tem acesso a 4 informações:
 - Sua posição atual
 - Sua velocidade atual
 - Melhor posição encontrada até o momento por uma partícula em sua vizinhança
 - No espaço de soluções
 - Sua aptidão

Partículas

- A cada iteração, cada partícula tem seus parâmetros atualizados usando:
 - Melhor solução obtida até o momento pela partícula
 - Melhor solução obtida até o momento por qualquer partícula da sua vizinhança

Vizinhança

- Não está relacionada à proximidade física dos indivíduos no espaço de soluções
 - Vizinhanças são definidas antes das partículas iniciarem a caça
 - Buscas dentro de uma vizinhança podem ocorrer em áreas distintas do espaço de soluções

Vizinhança

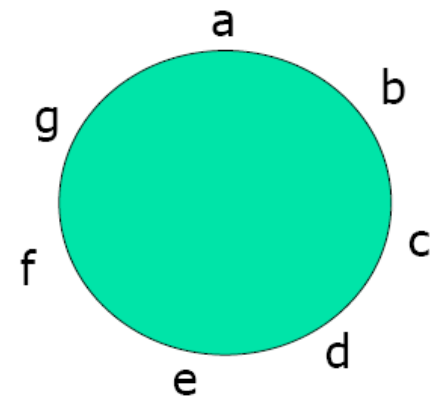
- Quem são são vizinhos de uma partícula?
 - Sucessos dos vizinhos irão influenciar sucesso da partícula

Vizinhança

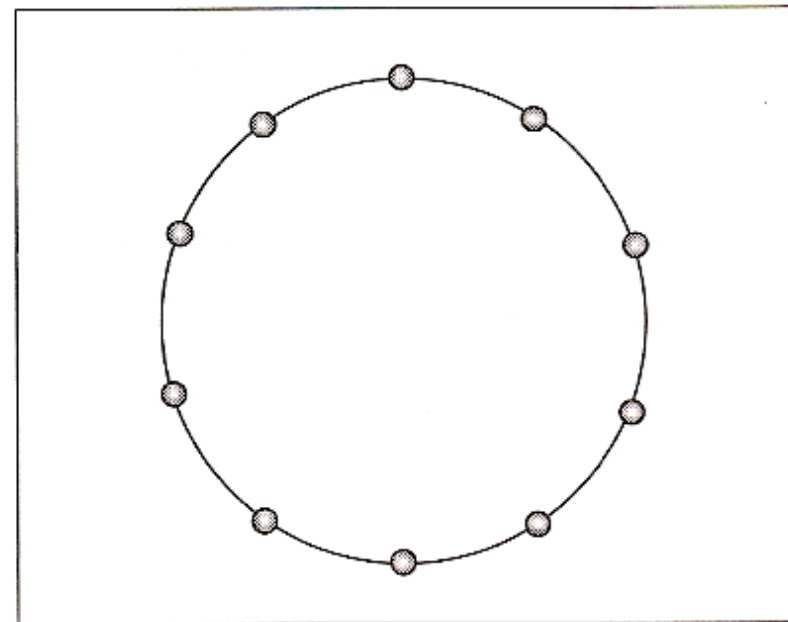
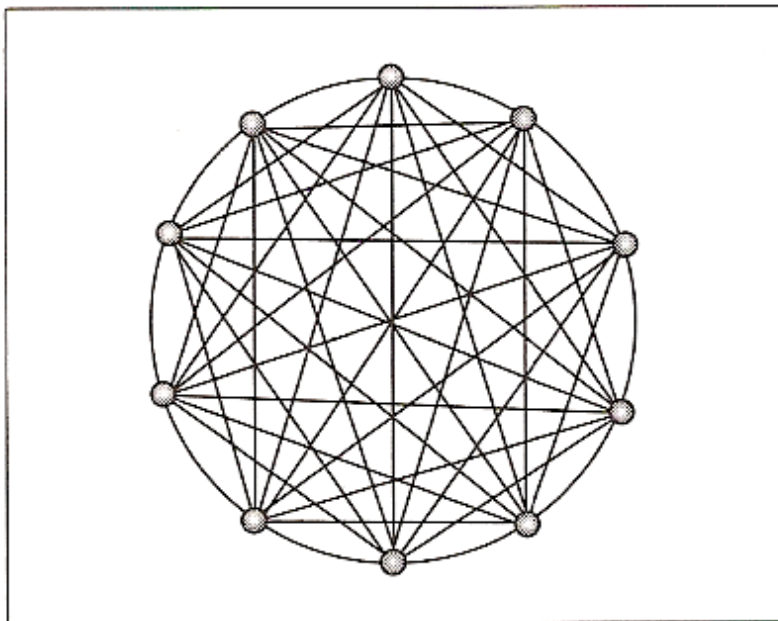
- Opção:
 - Tratar todas as partículas como vizinhos
 - Aparentemente boa
 - Mas reduz chance de convergência para solução ótima em vários problemas
 - Uso de vizinhanças menores, com sobreposição tem sido mais efetivo

Vizinhança

- Supor 7 partículas: a , b , c , d , e , f e g
 - Usar as vizinhanças: gab , abc , bcd , cde , efg e fga
 - Esta topologia atrasa convergência e permite maior exploração do espaço de busca
 - Partícula a é vizinha de g e de b
 - Se b ou g encontrarem uma solução melhor que a , a será influenciada por esta decisão
 - g e b , por sua vez, têm sua própria vizinhança e serão influenciadas por ela



Vizinhança Global X Local



Algoritmo PSO

- Atualização de velocidade

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t))$$

onde

\mathbf{v}_i é a velocidade da partícula i

t é o tempo

\mathbf{p}_i é a melhor posição encontrada até o momento pela partícula i

\mathbf{p}_g é a melhor posição encontrada até o momento pela vizinhança de i

φ_1 é um vetor de números aleatórios em uma distribuição uniforme entre 0 e AC_1

φ_2 é um vetor de números aleatórios em uma distribuição uniforme entre 0 e AC_2

AC_1 e AC_2 são constantes de aceleração. Normalmente $AC_1 = AC_2 = 2$

Kennedy também define :

$AC_1 + AC_2 = 4,1$ (geralmente $AC_1 = AC_2 = 2,05$)

Algoritmo PSO

- Atualização de peso

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

onde

v_i é a velocidade da partícula i (calculada no slide anterior)

t é o tempo

x_i é a posição da partícula i

Treinamento

- Permite a propagação lenta para o ótimo global por meio de todas as vizinhanças
 - Durante a propagação
 - Partículas das vizinhanças ainda não afetadas continuarão sua busca no espaço de soluções
 - Seguindo o caminho que elas julgam melhor
 - Aumenta as chances de encontrar uma nova solução global melhor, próxima a eles ou entre eles
 - Se existir

Aptidão

- Avaliação da função de aptidão
 - Partícula não precisa saber qual é o seu valor de aptidão
 - Precisa apenas saber quão boa é sua aptidão em relação ao ótimo
 - Valor do ótimo é definido pelo projetista
 - Valor de aptidão é usado para comparar aptidão de duas partículas qualquer
 - Para definir melhor posição das partículas e das vizinhanças

Pseudocódigo

- Inicializar partículas
- Repetir
 - Para cada partícula
 - Calcular sua aptidão
 - Escolher partícula com maior aptidão
 - Para cada partícula
 - Atualizar sua velocidade
 - Atualizar sua posição
- Até critério de parada ser atingido

Limitar movimentação das partículas

- Para limitar a mudança de posição das partículas é possível estabelecer um limites:

$$\text{Se } v_{id} > v_{\max} \text{ então } v_{id} = v_{\max}$$

$$\text{Se } v_{id} < v_{\min} \text{ então } v_{id} = v_{\min}$$

Algoritmo

```
procedure [X] = PS(max_it, AC1, AC2, vmax, vmin)
  initialize X //usually  $\mathbf{x}_i$ ,  $\forall i$ , is initialized at random
  initialize  $\Delta\mathbf{x}_i$  //at random,  $\Delta\mathbf{x}_i \in [v_{\min}, v_{\max}]$ 
  t  $\leftarrow$  1
  while t < max_it do,
    for i = 1 to N do, //for each particle
      if  $g(\mathbf{x}_i) > g(\mathbf{p}_i)$ ,
        then  $\mathbf{p}_i = \mathbf{x}_i$ , //best indiv. performance
      end if
      g = i //arbitrary
      //for all neighbors
      for j = indexes of neighbors
        if  $g(\mathbf{p}_j) > g(\mathbf{p}_g)$ ,
          then  $g = j$ , //index of best neighbor
        end if
      end for
       $\Delta\mathbf{x}_i \leftarrow \Delta\mathbf{x}_i + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i)$ 
       $\Delta\mathbf{x}_i \in [v_{\min}, v_{\max}]$ 
       $\mathbf{x}_i \leftarrow \mathbf{x}_i + \Delta\mathbf{x}_i$ 
    end for
    t  $\leftarrow$  t + 1
  end while
end procedure
```

Algorithm 5.4: Standard particle swarm optimization (PS) algorithm.

Algoritmo

- Observações:

$$\Delta \mathbf{x}_i = \mathbf{v}_i$$

- $g(.)$ é a função de aptidão
- g é índice do melhor vizinho
- O ideal seria calcular todas as velocidades em um laço *for* e atualizar os pesos em outro laço *for*
 - Evitando que o cálculo do melhor vizinho misturasse termos de $\mathbf{x}(t)$ e $\mathbf{x}(t+1)$

Parâmetros

- Tamanho do enxame $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$
 - *Quantidade de partículas*
 - Normalmente entre 10 e 50
 - $N \in [10, 50]$
 - Dimensão do espaço de busca L
 - Número de parâmetros livres do problema
 - Escolha de representação
 - Números reais
 - Escolha da função de aptidão
 - Depende do problema

Exemplo

- Problema: encontrar valor inteiro de x que minimiza $f(x) = x^2, x \in [-7, +7]$
 - População inicial \Rightarrow aleatória
 - População de tamanho 4
 - Função de aptidão $\Rightarrow f(x)$
 - Melhor aptidão = posição de x que gera melhor $f(x)$
 - $\varphi_1 = \varphi_2 = 2.05$
 - $v_{\max} = 2 \quad v_{\min} = -2$
 - Vizinhaça:
 - \mathbf{x}_1 e \mathbf{x}_2 e \mathbf{x}_3
 - \mathbf{x}_2 e \mathbf{x}_3 e \mathbf{x}_4
 - \mathbf{x}_3 e \mathbf{x}_4 e \mathbf{x}_1
 - \mathbf{x}_4 e \mathbf{x}_1 e \mathbf{x}_2

Exemplo

$$\mathbf{x}_1 = [2] \quad \mathbf{x}_2 = [3] \quad \mathbf{x}_3 = [-4] \quad \mathbf{x}_4 = [-2]$$

$$\mathbf{v}_1 = [1] \quad \mathbf{v}_2 = [-2] \quad \mathbf{v}_3 = [2] \quad \mathbf{v}_4 = [-1]$$

Aptidão

$$g(\mathbf{x}_1) = 4$$

$$g(\mathbf{x}_2) = 9$$

$$g(\mathbf{x}_3) = 16$$

$$g(\mathbf{x}_4) = 4$$

Melhores

$$P_1 = 2$$

$$P_2 = 3$$

$$P_3 = -4$$

$$P_4 = -2$$

$$P_{\text{GERAL}} = 2$$

Exemplo – Passo I

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t))$$

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

- $\mathbf{v}_1 = 1 + 1.6 \times (2 - 2) + 1.8 \times (2 - 2) = 1$
- $\mathbf{x}_1 = 2 + 1 = 3$
- $\mathbf{v}_2 = -2 + 0.3 \times (3 - 3) + 1.9 \times (2 - 3) = -3.9 = -2.0$
- $\mathbf{x}_2 = 3 - 2 = 1$
- $\mathbf{v}_3 = 2 + 1.3 \times (-4 - (-4)) + 0.2 \times (-2 - (-4)) = 2.4 = 2.0$
- $\mathbf{x}_3 = -4 + 2 = -2$
- $\mathbf{v}_4 = -1 + 0.6 \times (-2 - (-2)) + 1.1 \times (2 - (-2)) = 3.4 = 2.0$
- $\mathbf{x}_4 = -2 + 2 = 0$

Aptidão

$$g(\mathbf{x}_1) = 9$$

$$g(\mathbf{x}_2) = 1$$

$$g(\mathbf{x}_3) = 4$$

$$g(\mathbf{x}_4) = 0$$

Melhores

$$P_1 = 2$$

$$P_2 = 1$$

$$P_3 = -2$$

$$P_4 = 0$$

$$P_{\text{GERAL}} = 0$$

Exemplo – Passo 2

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t))$$

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

- $\mathbf{v}_1 = 1 + 2.0 \times (2-3) + 1.0 \times (0-3) = -4 = -2.0$
- $\mathbf{x}_1 = 3 - 2 = 1$
- $\mathbf{v}_2 = -2 + 1.6 \times (1-1) + 0.3 \times (2-1) = -1.7$
- $\mathbf{x}_2 = 1 - 1.7 = -0.7$
- $\mathbf{v}_3 = 2 + 0.9 \times (-2-(-2)) + 1.9 \times (0-(-2)) = 5.8 = 2.0$
- $\mathbf{x}_3 = -2 + 2 = 0$
- $\mathbf{v}_4 = 2 + 1.6 \times (0-0) + 2.0 \times (0-0) = 2$
- $\mathbf{x}_4 = 0 + 2 = 2$

Aptidão

$$g(\mathbf{x}_1) = 1$$

$$g(\mathbf{x}_2) = 0.5$$

$$g(\mathbf{x}_3) = 0$$

$$g(\mathbf{x}_4) = 4$$

Melhores

$$P_1 = 1$$

$$P_2 = -0.7$$

$$P_3 = 0$$

$$P_4 = 0$$

$$P_{\text{GERAL}} = 0$$

Exemplo – Passo 3

$$\mathbf{v}_i(t+1) = \mathbf{v}_i(t) + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t))$$

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

- $\mathbf{v}_1 = -2 + 1.3 \times (1-1) + 0.1 \times (0-1) = -2.1 = -2.0$
- $\mathbf{x}_1 = 1 - 2 = -1$
- $\mathbf{v}_2 = -1.7 + 1.7 \times (-0.7 - (-0.7)) + 1.9 \times (0 - (-0.7)) = -0.4$
- $\mathbf{x}_2 = -0.7 - 0.4 = 1.1$
- $\mathbf{v}_3 = 2 + 1.4 \times (0-0) + 1.6 \times (0-0) = 2$
- $\mathbf{x}_3 = 0 + 2 = 2$
- $\mathbf{v}_4 = 2 + 1.5 \times (0-2) + 0.8 \times (0-2) = -2.6 = -2.0$
- $\mathbf{x}_4 = 2 - 2 = 0$

Aptidão

$$g(\mathbf{x}_1) = 1$$

$$g(\mathbf{x}_2) = 1.2$$

$$g(\mathbf{x}_3) = 4$$

$$g(\mathbf{x}_4) = 0$$

Melhores

$$\mathbf{P}_1 = 1$$

$$\mathbf{P}_2 = -0.7$$

$$\mathbf{P}_3 = 0$$

$$\mathbf{P}_4 = 0$$

$$\mathbf{P}_{\text{GERAL}} = 0$$

Espaço de Soluções

- Número de dimensões = número de parâmetros livres do problema
 - Variáveis de valores desconhecidos
- Exemplo: $x^2 - 3y + 7$
 - Tem 2 dimensões (x, y)
 - Localização da partícula no espaço de soluções é definido por 2 coordenadas

Espaço de Soluções

- Exemplo: $4x^3 - 3y^2 + 4(w - z)^2$
 - Tem 4 dimensões (x, y, z e w)
 - Localização da partícula no espaço de soluções é definido por 4 coordenadas
- PSO não tem dificuldades para trabalhar com 4 ou mais dimensões
 - Otimização de uma RN com 50 pesos teria 50 dimensões

Exercício

- Seja a função $f(x) = 3x^2 - 5y + 2z^3 - 4$
 - Encontrar valores de x , y e z que fazem com que $f(x,y,z) = 0$
 - Partícula a
 - Coordenadas: 2, 1, 1
 - Partícula b
 - Coordenadas: 3, 7, 2

Exploration X Exploitation

- É necessário um balanço entre exploration e exploitation
 - Exploration (exploração):
 - Procurar por uma boa solução no espaço de busca, visitando pontos desconhecidos
 - Pouca exploration: algoritmo converge para a primeira boa solução encontrada
 - Exploitation (usufruir):
 - Extrair o máximo de informação das soluções encontradas e usá-las para obter as próximas soluções
 - Pouca exploitation: algoritmo nunca converge

Sociabilidade X individualidade

- Balanço exploration X exploitation pode ser visto como de outra forma:
 - Sociabilidade X individualidade
 - Indivíduos devem ter individualidade
 - Pássaros não querem chocar-se com outro(s)
 - Também devem ser sociáveis
 - Saber onde estão as boas soluções encontradas por outros, podendo aprender com eles

Modificações do PSO

- Peso de inércia w

$$\mathbf{v}_i(t+1) = w \cdot \mathbf{v}_i(t) + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t))$$

- w tem um valor inicial que pode ser reduzido durante o processo de adaptação
 - Pode ser considerado uma temperatura
 - Vai esfriando durante o processo
 - Da mesma forma que no Simulated Annealing (Recozimento Simulado)

Modificações do PSO

- Coeficiente de restrição χ

$$\mathbf{v}_i(t+1) = \chi(\mathbf{v}_i(t) + \varphi_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \varphi_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t)))$$

- com $\chi \approx 0.729$
- Modificação conhecida hoje como o PSO padrão

PSO X AGs

- Similaridades
 - Metaheurísticas baseadas em populações
 - Inicializadas com uma população aleatória de indivíduos
 - Utilizam função de aptidão para avaliar cada indivíduo
 - Buscam por ótimo global em várias gerações
 - Critério de parada
 - Não garantem sucesso

PSO

- Vôo no espaço de busca
 - Velocidade interna, memória
- Mais fácil de implementar
- Menos parâmetros para ajustar
- Apenas melhor indivíduo (da vizinhança) transmite informações
- Utiliza valores reais (padrão)

AG

- Operadores genéticos
 - Seleção, Crossover, etc...
- Mais difícil de implementar
- Mais parâmetros para ajustar
- Cromossomos compartilham informações
- Utiliza valores binários (padrão)

PSO X AGs: diferenças

PSO: Resumo

- Baseado em comportamento social
- Algoritmo muito simples
- Poucas linhas de código
- Além de rápido, utiliza pouca memória
- Utiliza operadores matemáticos simples
- Tem obtido bons resultados em várias aplicações
 - Otimização de parâmetros de Redes Neurais

Referências Bibliográficas

- CASTRO, Leandro Nunes. *Fundamentals of Natural Computing: Basic Concepts, Algorithms, And Applications*. CRC Press, 2006.
- CARVALHO, André Ponce de Leon F. de. *Notas de Aula*, 2007.
- HAYKIN, Simon. *Redes Neurais: Princípio e Prática*. Bookman, 2001.
- KOVACS, Zsolt L. *Redes Neurais Artificiais: Fundamentos e Aplicações*. Livraria da Física, 2006.
- MITCHELL, Melanie. *An Introduction to Genetic Algorithms*. MIT Press, 1998.
- BISHOP, Christopher M. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
- <http://www.swarmintelligence.org/>

