1. Otimização por Enxame de Partículas, Introdução

Otimização por Enxame de Partículas, conhecido como PSO (*Particle Swarm Optimization*) é uma técnica de otimização estocástica de funções não-linear baseada em populações desenvolvida pelo Psicólogo Social Kennedy e pelo Engenheiro Elétrico Eberhart em 1995, com a inspiração biológica na simulação do comportamento social de um bando de pássaros [1]. O objetivo inicial era simular graficamente o comportamento dos pássaros na sincronização do seu vôo, onde em determinadas situações descobriram que o enxame inteiro convergia para um determinado ponto, sendo o ponto escolhido ótimo para aquela situação. A partir desse comportamento que utilizam regras simples produzir um resultado coletivo complexo, surgiria aí a mais nova família de algoritmos baseados em Enxames.

Desde sua criação, o PSO se tornou um dos algoritmos mais estudados no campo da Computação Inteligente nos últimos anos, tendo inúmeros artigos sendo divulgados, várias variantes sendo estudadas e aprimoradas, e diversas topologias sendo aplicadas a problemas específicos. Essas melhorias só foram possíveis porque o algoritmo foi concebido a partir de um conceito muito básico como a utilização de regras simples e fáceis de implementar computacionalmente com pouquíssimas linhas de código, requer apenas alguns operadores matemáticos primitivos e poucos recursos de memória e processamento computacionais [1]. Em contrapartida, o algoritmo traz um grande poder de otimização, tendo sido utilizado com boa aceitação na maioria dos problemas que podem ser resolvidos pelos Algoritmos Genéticos.

No PSO, os indivíduos são chamados de partículas, pois cada partícula representa uma potencial solução para o problema [2]. Tais partículas "voam" no espaço de busca em forma de "bando" em um objetivo em comum: busca. Tal busca pode ser tratada por otimização de funções ou resolver problemas de permutação ou até mesmo treinar uma rede neural MLP [5]. A mudança de posição de cada partícula no espaço de busca é baseada na tendência de cada indivíduo em voltar à sua melhor posição (componente cognitivo) e da tendência em ir para a melhor posição da sua vizinhança (componente social).

2. Onde utilizar

A recomendação do seu uso é geralmente em problemas de otimização de funções não lineares que sejam de espaços contínuos, ambientes estáticos e com uma ou várias dimensões (multidimensional). O seu uso é acompanhado de uma *fitness* que tem o objetivo de avaliar a solução proposta informando ao algoritmo o quão perto do objetivo está essa solução atual. Em resumo, estamos buscando que entradas serão fornecidas ao problema original para encontrar um objetivo, seja maximizar ou minimizar a função *fitness* de forma eficiente.

Essa característica de busca nos leva a pensar que a função de avaliação *fitness* seja realmente uma função matemática, porém tem muitos problemas que não será possível desenvolver uma função para avaliar a solução, e aí é onde entra a simulação. A simulação de uma solução proposta é computada e avaliada como sendo o delta entre o valor simulado e o valor objetivo. Nessa situação vários problemas podem ser resolvidos com PSO, dentre os citados estão o treinamento de uma rede neural, por exemplo, onde o PSO irá buscar no espaço de busca uma solução com o número de dimensões sendo o número de conexões sinápticas da rede, e como função de avaliação a simulação dos pesos da rede com os padrões de treinamento.

Como citado por Kennedy e Eberhart [1] em seus testes com o treinamento de uma rede neural MLP para a aproximação da função que representa um eletro encefalograma gráfico, com o algoritmo de *backpropagation* padrão a rede neural acertava em 89% dos casos, já com o treinamento com PSO a rede acertava em 92% dos casos corretamente [1].

3. Resumo da técnica

No PSO, um enxame de partículas é distribuído pelas diversas dimensões do espaço de busca do problema a ser resolvido e através da comunicação entre essas partículas as soluções ótimas para o problema são encontradas. Como já foi dito, essas partículas são dotadas de inteligência bastante reduzida e as interações que existem entre elas são também bastante simples. A capacidade deste algoritmo de resolver problemas complexos surge, então, da inteligência que emerge do enxame (conjunto de partículas e interações entre elas).

Cada partícula está sempre buscando a melhor solução dentro do espaço de busca do problema, para isso ela faz uso de seu conhecimento próprio (componente cognitivo), ou seja, ela guarda consigo a localização do melhor ponto que ela já encontrou, e do conhecimento obtido das partículas que estão em sua vizinhança (componente social), representado pelo melhor ponto que elas já encontraram. A velocidade com a qual as partículas se movimentam é a parte central do algoritmo e pode ser calculada utilizando-se diversas abordagens diferentes conforme será apresentado mais adiante [5].

3.1. Partes básicas do algoritmo

O funcionamento básico do PSO é composto por partículas que são dispostas de forma uniforme e aleatória em todo espaço de busca, tais partículas são dotadas de uma memória da melhor posição alcançada, de um vetor velocidade que indica a direção e velocidade da partícula e da memória da melhor posição alcançada por alguma partícula que faz parte da sua vizinhança. Cada parte do algoritmo tem uma função característica genérica que é mapeada entre os algoritmos de busca, são eles o mecanismo de convergência, mecanismo de diversidade e a avaliação de sucesso.

3.1.1. Inicialização

A posição inicial das partículas é feita de maneira a cobrir uniformemente todo o espaço de busca, pois o PSO é sensível a diversidade inicial do enxame. Caso alguma região do espaço de busca não foi coberta pelo enxame inicial, o PSO pode ter dificuldades em descobrir algum ótimo caso esteja justamente nessa região não coberta. Existe uma possibilidade das partículas descobrirem essa região, se caso alguma partícula que esteja por perto seja atraída e caia bem nessa região.

As velocidades das partículas geralmente são inicializadas com o valor zero, pois isso evitaria já na primeira iteração ter partículas saindo do espaço de busca. De fato, se pensarmos no algoritmo em si não faria sentido inicializar as velocidades com valores aleatórios, visto que não sabemos para qual lado deveríamos ir, fazendo com que o algoritmo levasse até mesmo mais tempo para corrigir as iterações iniciais.

Nessa mesma idéia é também inicializada a melhor posição da partícula: após a escolha aleatória da posição da partícula no espaço de busca, é feita uma avaliação da função *fitness* da posição inicial e essa é também atribuída a melhor posição da partícula [5].

3.1.1. Mecanismo de convergência

Pode-se dizer que a convergência do algoritmo está centrada em torno das componentes cognitivas e sociais que fazem parte do vetor velocidade, permitindo assim que as partículas se movam vagarosamente para um único ponto do espaço de busca, garantindo assim a convergência. Sem esses componentes as partículas se moveriam sem destino até atingirem a borda do espaço de busca sem haver assim uma convergência [2].

3.1.2. Mecanismo de diversidade

Já a capacidade de exploração do PSO é garantida pela boa inicialização das partículas pelo espaço de busca e pelas variáveis aleatórias que são multiplicadas pelos componentes cognitivos e sociais. Essas variáveis garantem uma capacidade de exploração do enxame a outras áreas que não foram exploradas anteriormente. Por exemplo, uma determinada partícula está longe do máximo global, ela possui uma força maior de atração para esse máximo, mas se a sua variável aleatória for baixa, a atração não será tão forte assim, fazendo com que a partícula se mova menos do que realmente ela deveria. Nesse momento ocorre a busca por exploração, pois a partícula começa a se mover com uma direção previsível, mas com distâncias imprevisíveis.

Vale lembrar que tanto o mecanismo de convergência como o de diversidade possuem duas constantes que são fundamentais no peso de cada uma das partes que compõem o vetor velocidade. A constante cognitiva controla o efeito "nostalgia", onde a partícula sempre se lembra de forma mais acentuada ou menos acentuada o seu melhor lugar já visitado. Já a constante social determina o comportamento da partícula com relação ao enxame como todo, controlando a intensidade com que as partículas trocam informações entre si.

3.1.3. Avaliação de sucesso

A avaliação de sucesso no PSO é baseada em uma função de avaliação chamada *fitness*. Essa função deve prover a capacidade de identificar boas regiões de busca das más, baseando-se para isso da posição em que as partículas se encontram no espaço de busca.

Para descrever o ambiente de busca a função de avaliação pode ser representada por:

1. Uma função matemática que defina o problema ou que se aproxime do mesmo, caso seja conhecida;
2. Uma heurística que indica uma possível direção da solução do problema, sendo um guia para a solução;
3. Ou, pela simulação da solução proposta utilizando dos valores do vetor em simulador ou da aplicação direta no problema real, avaliando assim seu comportamento.

3.2 Topologias

No PSO, a direção na qual as partículas se movimentam no espaço de busca é, geralmente, determinada pela resultante de duas forças de atração principais: (i) a melhor posição já encontrada pela própria partícula; e (ii) a melhor posição já encontrada por todas as partículas de sua vizinhança [4]. A forma como a vizinhança da partícula é estabelecida, portanto, é de fundamental importância para determinar o comportamento geral do algoritmo. Quanto maior a vizinhança de uma partícula, mais rapidamente o algoritmo irá convergir. No entanto, se o algoritmo convergir muito rápido significa

que ele irá explorar pouco o espaço de busca do problema e, conseqüentemente, estará mais suscetível a cair em mínimos ou máximos locais.

Mesmo com diversos estudos tem sido feito sobre as topologias, não existe uma topologia que seja a melhor para todos os tipos de problemas. O que esses estudos indicam é, por exemplo, que a topologia *gbest* é mais indicada para problemas unimodais. Já problemas multimodais, a recomendação é para o uso de topologias menos conectadas, tais como a topologia anel ou quatro-ninchos *(four-clusters*).

A seguir serão apresentadas algumas das topologias utilizadas pelo PSO.

3.2.1 Estrela (*Star*)

Foi a primeira topologia utilizada pelo PSO [6]. Ela forma uma rede social em que cada partícula comunica-se com todas as demais (Figura 3.1), ou seja, a vizinhança de cada partícula é formada por todo o enxame. Desta forma, a melhor solução já encontrada pelo enxame é rapidamente compartilhada com todas as partículas. A implementação do PSO que utiliza esta topologia é conhecido por *Global Best PSO* ou *gbest* e seu algoritmo será explicado em detalhes mais a diante.

Essa topologia apresenta uma rápida convergência, o que é interessante quando o problema que se está tentando resolver é unimodal, ou seja, apresenta apenas um ponto de máximo ou mínimo. No entanto, para problemas multimodais, que representam a grande maioria dos problemas reais, esta topologia aumenta muito a probabilidade de o algoritmo ficar preso em mínimos ou máximos locais [5].

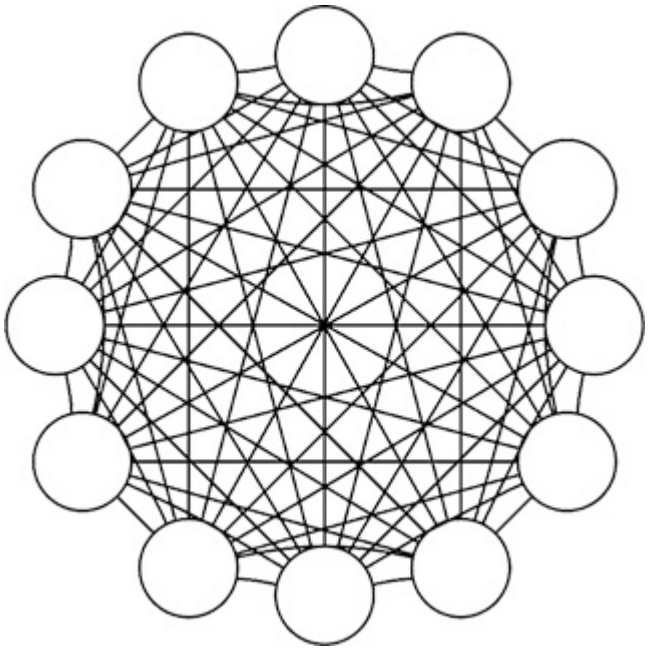

Figura 3.1 - Topologia Estrela

3.2.2 Anel (*Ring*)

Nesta topologia a vizinhança da partícula é limitada por seus *n* vizinhos mais próximos. Em uma topologia de vizinhança *n* igual a 2, por exemplo, se torna uma topologia conhecida como anel onde uma partícula *i* irá se comunicar com as partículas *i-1* e *i+1* (Figura 3.2). A informação da melhor posição já encontrada pelo enxame ainda flui por todas as partículas porque existem interseções entre

as vizinhanças, ou seja, uma mesma partícula pode fazer parte de várias vizinhanças diferentes. No entanto, como uma partícula não pode se comunicar diretamente com todas as demais do enxame, a velocidade com a qual a informação da melhor posição já encontrada se espalha é bem menor que na topologia em estrela. Esse tipo de topologia é comumente chamado de *Local Best PSO* ou *lbest*, pois como foi dito a informação da melhor posição é sempre relativa às melhores posições locais da partícula.

Dessa forma, este algoritmo converge mais lentamente, mas explora melhor o espaço de busca e, por isso, está menos sujeito a ficar preso em mínimos ou máximos locais, sendo uma boa opção para resolver problemas multimodais.

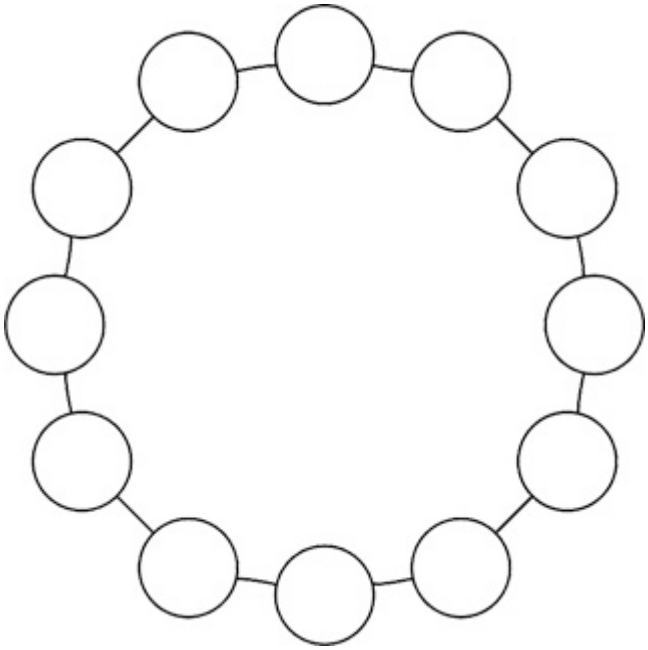

Figura 3.2 - Topologia Anel

3.2.3 Roda (*Wheel*)

As partículas dentro de uma vizinhança ficam isoladas umas das outras e toda a comunicação entre elas é feita através de uma única partícula eleita como a partícula focal (Figura 3.3). Esta partícula concentra as informações sobre as melhores posições encontradas por todas as demais do enxame. A cada iteração do algoritmo, ela irá mover-se na direção do vizinho que possui a melhor posição. Caso a nova posição da partícula focal seja melhor que a sua anterior, ela dissemina essa informação para todas as partículas de sua vizinhança. Desta forma, a topologia roda retarda a propagação das melhores soluções através do enxame [5].

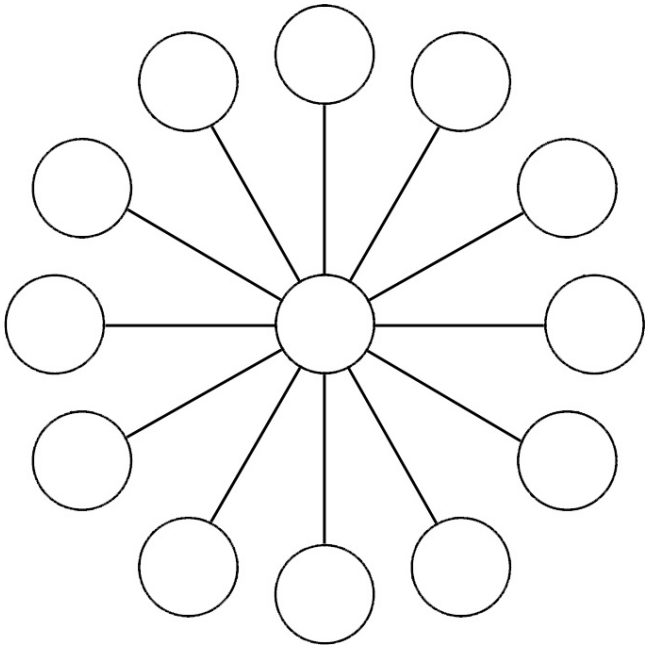

Figura 3.3 - Topologia Roda (*Wheel*)

3.2.4. Outras topologias

Diversas outras diversas topologias que podem ser utilizadas para a comunicação entre os indivíduos do PSO. Algumas topologias específicas foram criadas para solucionar problemas específicos, por exemplo, a topologia quatro-ninchos (*four-clusters)* (Figura 3.5), por exemplo, faz com que quatro sub-enxames procurem no mesmo espaço de busca em diferentes pontos simultaneamente, melhorando na busca local. O problema dessa abordagem é que as partículas dos sub-enxames são fixas e, portanto não representam o melhor ponto do sub-enxame para os outros sub-enxames.

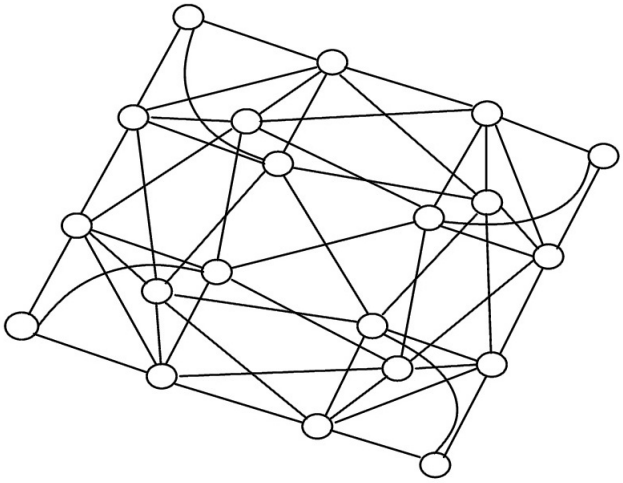

Figura 3.4 - Topologia Pirâmide

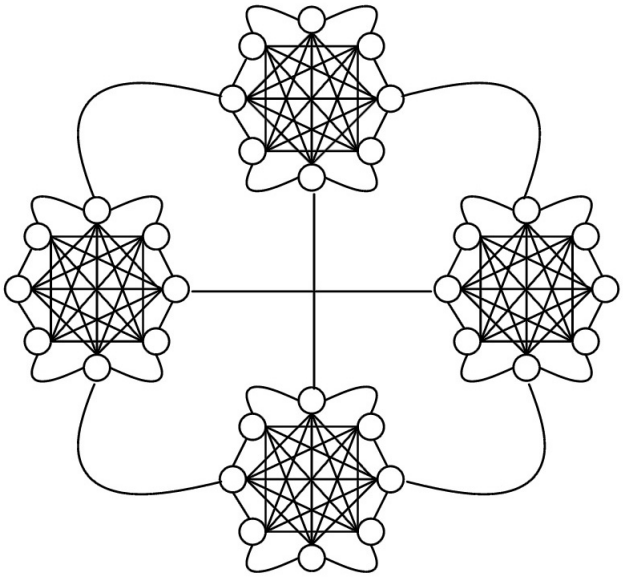

Figura 3.5 - Topologia quatro-ninchos (*four-clusters*)

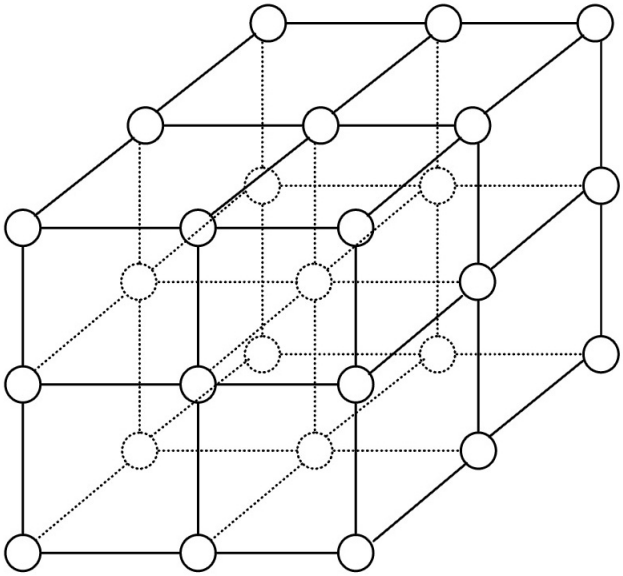

Figura 3.6 - Topologia Von Neumann

3.4. Parâmetros Básicos

O PSO possui diversos parâmetros que influenciam seu comportamento, sua velocidade de convergência e a qualidade da solução final encontrada por ele. Saber escolher bem esses parâmetros e combiná-los bem entre si é de fundamental importância para se obter boas soluções para os problemas a serem resolvidos. A seguir, serão apresentados alguns de seus principais parâmetros, mostrando-se qual o seu papel e como encontrar bons valores para eles.

3.4.1. Tamanho do Enxame

Esse parâmetro corresponde ao número de partículas que farão parte do enxame. Ao utilizar um exame com um grande número de partículas e iniciá-lo de modo uniformemente aleatório, se ganha uma grande diversidade inicial e, conseqüentemente, melhora-se o resultado final do algoritmo. Ao mesmo tempo, com mais partículas no enxame é possível explorar porções maiores do espaço de busca a cada iteração, aumentando-se as chances de o algoritmo convergir mais rapidamente.

No entanto, quanto mais partículas o enxame possuir, maior será a complexidade computacional (custo de processamento) que deverá ser tratada a cada iteração do algoritmo. Além disso, já foi provado que é possível encontrar boas soluções com enxames pequenos de 10 a 30 partículas [5]. O importante, portanto, é observar que o tamanho do enxame a ser utilizado irá depender do problema a ser resolvido e o melhor é encontrar-se o tamanho ideal através de métodos de validação cruzada. Embora, como já foi dito, pequenos enxames são suficientes.

3.4.2. Tamanho da Vizinhança

Quanto maior o tamanho da vizinhança, maior será o número de interações de cada partícula com as demais partículas do enxame e mais rapidamente o algoritmo irá convergir. No entanto, uma rápida convergência, em geral, implica em pouca exploração do espaço de busca e, conseqüentemente, aumenta-se as chances de se ficar preso em pontos de máximo ou mínimo locais. Por outro lado, pequenas vizinhanças diminuem a velocidade com a qual informações espalham-se pelo enxame e, em geral, a velocidade de convergência do algoritmo. Ao mesmo tempo, o algoritmo tende a explorar melhor o espaço de busca e diminuem-se as chances de ele ficar preso em mínimos ou máximos locais.

O tamanho da vizinhança, portanto, irá variar de acordo com o problema a ser resolvido. De modo geral, grandes vizinhanças são boas para resolver problemas unimodais, ao passo que vizinhanças menores são melhores para problemas multimodais.

3.4.3. Quantidade de Iterações

A quantidade de iterações utilizadas pelo PSO para achar boas soluções depende do problema a ser resolvido. Valores baixos podem fazer com que a busca termine de maneira precipitada; em contrapartida, valores altos podem levar a um uso desnecessário de recursos computacionais incluindo chamadas desnecessárias a função de *fitness*.

3.4.4. Coeficientes de Aceleração

Conforme explicado anteriormente, a velocidade de cada partícula pode ser decomposta em duas componentes principais: (i) a componente cognitiva; e (ii) a social. A influência de cada uma dessas componentes sobre a velocidade da partícula é determinada pelos coeficientes de aceleração c_1 e c_2, respectivamente. Estas duas constantes são também chamadas de parâmetros de confiança, onde c_1 expressa o quão confiante a partícula está de se mesmo e c_2 o quanto a partícula confia em sua vizinhança [5]. Em geral, c_1 e c_2 possuem valores estáticos e próximos entre si. Para problemas unimodais, a componente social exerce maior influência na solução e, por isso, c_2 pode ser maior que c_1. Já em problemas multimodais, é interessante que a partícula sofra menos influência de seus

vizinhos e explore mais o ambiente com base em sua própria experiência e, portanto, c_1 pode ser maior que c_2.

3.5. Variações básicas

Desde a criação do PSO várias variações foram propostas a fim de resolver alguns problemas que a técnica original apresenta. Todos os casos que serão apresentados abaixo abordam alguns dos seguintes problemas:

- Controlar melhor a capacidade de exploração em largura e a exploração em profundidade;
- Limitar a velocidade, evitando assim que as partículas "saltem" das boas regiões de busca;
- Controlar a aceleração das partículas, evitando que elas saiam do espaço de busca.

3.5.1. Limitador de Velocidade

Na aplicação da versão original do PSO, as aplicações mais recentes apresentaram problemas na velocidade de algumas partículas: ela aumentava rapidamente, fazendo com que elas saltassem de possíveis locais bons e inclusive acabassem saindo do espaço de busca, especialmente para partículas que estavam longe do seu *gbest* (melhor posição da vizinhança) e *pbest* (melhor posição da partícula). A esse comportamento foi atribuído o nome de explosão.

A fim de resolver esse problema, foi criado o recurso de limitador de velocidade, impedindo que a velocidade da partícula ultrapasse um valor previamente determinado, ou seja, caso a partícula tenha uma velocidade inferior a velocidade máxima definida, a sua velocidade atual será permanecida. Caso contrário, a velocidade da partícula máxima será definida como a velocidade limite máxima. Então, a velocidade da partícula será ajustada de acordo com a seguinte fórmula:

$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & \text{if } v'_{ij}(t+1) < V_{max,j} \\ V_{max,j} & \text{if } v'_{ij}(t+1) \geq V_{max,j} \end{cases}$$

Figura 3.7 - Cálculo do limitador de velocidade

Deve ser definido um limitador de velocidade para cada dimensão. Valores muito grandes acabam não evitando a explosão da busca, ou seja, as partículas acabam "pulando", os valores ótimos, já valores pequenos demais acabam comprometendo a performance do algoritmo e até mesmo permitir que as partículas fiquem presas em mínimos locais [5].

Para o valor máximo da velocidade, devem ser definidos valores menores que o espaço de busca (x_{max} - x_{min}), caso contrário as partículas poderão ir além do espaço de busca. Lembrando que o limitador de velocidade não impede que a partícula saia do espaço de busca, ele apenas controla os tamanhos dos passos da partícula evitando assim a explosão.

A figura 3.8 mostra como funciona o limitador de velocidade. Na dimensão x_1, a velocidade foi limitada de $v_2(t + 1)$ para $v'_2(t + 1)$, que por conseqüência faz com que a posição final da partícula

seja $x'_i(t + 1)$ ao invés de $x_i(t + 1)$.

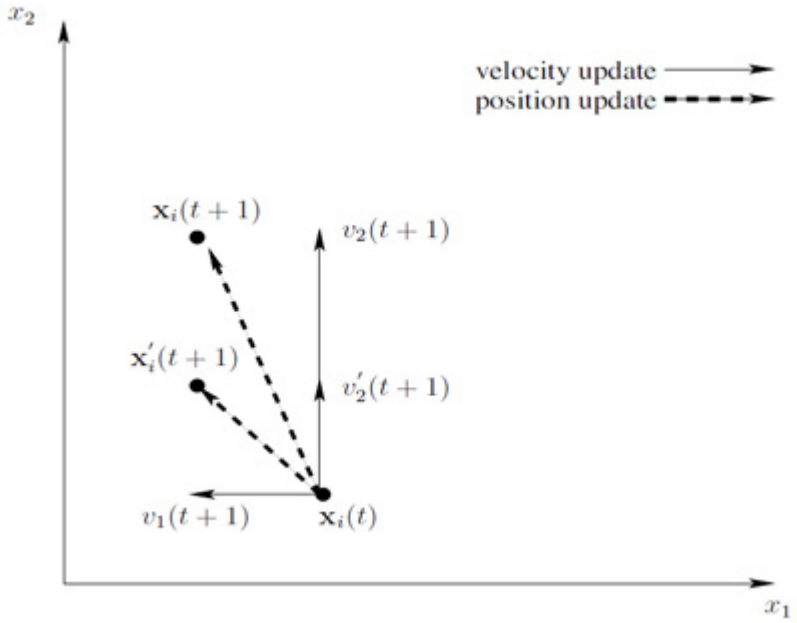

Figura 3.8 - Limitador de velocidade em ação

3.5.2. Peso Inercial

Esta variante do algoritmo original foi proposta por Eberhart e Shi e tem como objetivo controlar a capacidade da busca em largura e profundidade do enxame, assim como eliminar a necessidade de limitar a velocidade das partículas através do limitador de velocidade máxima. Esta variação conseguiu alcançar o seu primeiro objetivo, porém não conseguiu eliminar completamente a necessidade de se utilizar um limitador de velocidade. Um valor w é inserido na equação original utilizada para o calculo da velocidade da partícula, ficando:

$$v_i^{k+1} = wv_i^k + c_1 r_1 \left(p_i^k - x_i^k\right) + c_2 r_2 \left(p_g^k - x_i^k\right)$$

Figura 3.9 - Fórmula de atualização da velocidade com peso inercial

O valor de w é extremamente importante para garantir o comportamento de convergência, e de modo mais eficiente o *tradeoff* da busca em profundidade e largura. Valores altos para w resulta em trajetórias de partículas relativamente simples, com saltos significantes (*overshoot*) resultando em uma boa busca global melhorando a exploração. Valores pequenos resultam em trajetórias irregulares com uma redução nos saltos obtendo propriedades desejáveis para uma busca local refinada melhorando a busca em profundidade. Implementações iniciais desta variação utilizavam um valor *w* constante. Atualmente existem técnicas que permitem que o valor do peso seja atualizado dinamicamente. Nestas técnicas o peso é iniciado com valores altos e com o tempo este será reduzido.

Esta versão produz melhor resultados que o algoritmo original do PSO e ajuda a aumentar a taxa de convergência [7].

Existe também uma variação que torna o peso inercial dinâmico. Essa variação tenta eliminar alguns dos problemas encontrados na abordagem constante, pois ela impõe uma redução linear no peso de inércia w durante a busca, que em geral o *w* irá variar entre 0,8 e 0,4. Este mecanismo faz com que o algoritmo comece realizando busca em largura inicialmente e à medida que a busca evolui ele passe a realizar mais busca em profundidade.

3.5.3. Coeficiente de Restrição

Neste método é adicionado um coeficiente de restrição K na equação de velocidade cujo efeito é de reduzir a velocidade das partículas assim como a progresso da busca, com isso contraindo o diâmetro total do enxame resultando progressivamente em um domínio menor sendo pesquisado. O valor do coeficiente de restrição é calculado como uma função dos parâmetros cognitivo (c_1) e social (c_2).

$$v_i^{k+1} = K * \left[v_i^k + c_1 r_1 \left(p_i^k - x_i^k \right) + c_2 r_2 \left(p_g^k - x_i^k \right) \right]$$

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

Onde $\varphi = c_1 + c_2$, $\varphi > 4$.

3.5.4. Atualização Assíncrona (Paralelismo)

Na atualização síncrona, utilizado no *lbest* e *gbest*, a melhor solução encontrada é atualizada após uma iteração completa, ou seja, após todas as partículas se moverem. Na atualização assíncrona imediatamente após o movimento de uma partícula o valor da melhor posição encontrada é atualizado. Deste modo ocorre uma resposta imediata ao ser encontrado uma melhor solução no espaço de busca.

Carlisle e Dozier entendem que a atualização assíncrona é mais importante no algoritmo de *lbest* onde uma resposta mais rápida é mais benéfica em enxames conectados de forma livre, enquanto atualizações síncronas são mais adequadas ao algoritmo *gbest*. Essa estrutura permite ao algoritmo que seja possível a execução paralela do cálculo das posições das partículas, aumentando assim o poder computacional do algoritmo.

3.5.5. Modelos de Velocidade

Modelo somente cognitivo

No modelo somente cognitivo a componente social da equação da velocidade não existe. Nesse modelo, as partículas tendem a voltar ao melhor ponto que passaram. Esse tipo de efeito é chamado de efeito "nostalgia". Uma forma prática de implementar o modelo somente cognitivo seria com a definição de $c_2 = 0$, isso faria com que a componente cognitiva não influenciasse na velocidade da partícula.

Em um estudo empírico, Kennedy reportou que esse modelo é mais vulnerável a falhas que o modelo totalmente conectado. Ele possui a tendência de fazer buscas locais dos pontos onde

as partículas são inicializadas [5]. Porém, esse comportamento é benéfico no caso de buscas multimodais, quando é necessário obter mais de uma solução possível (mínimos locais) do espaço de busca.

Modelo somente social

No modelo somente social a componente cognitiva da equação da velocidade não existe. Nesse modelo, as partículas não possuem possui nenhuma tendência para retornar a sua melhor posição. Todas as partículas são atraídas para a melhor partícula do enxame (*gbest*). Uma forma de implementar o modelo somente social seria com a definição de c1 = 0, isso faria que a componente social não influenciasse na velocidade da partícula.

Em estudos empíricos, Kennedy mostrou que o modelo somente social é mais rápido e eficiente que o modelo original [5]. Porém, esse comportamento pode impedir que o espaço de busca não seja totalmente explorado, evitando assim que pontos melhores possam ser encontrados.

3.6. Critérios de parada

O critério de parada no PSO deve levar em consideração dois fatores importantes:

- Não convergir à busca prematuramente para regiões sub-ótimas;
- Proteger do cálculo desnecessário de chamadas freqüentes a função *fitness*, aumentando assim a complexidade computacional [5].

Sendo assim, citamos algumas das condições de parada:

- **Finalizar quando um número máximo de iterações for atingido:** A utilização desse método pode comprometer a convergência do algoritmo, visto que se o número máximo de iterações for pequeno, o algoritmo pode parar antes de achar uma boa solução, permanecendo em uma região sub-ótima;
- **Finalizar quando uma solução aceitável for encontrada:** Esse método é bastante difícil de utilizar, pois nem sempre sabemos o quão aceitável pode ser a solução para o seu problema e nem se a mesma poderá ser alcançada caso os parâmetros de aceitação sejam demasiadamente altos;
- **Finalizar quando não for observada nenhuma alteração em um número de iterações:** Esse método pode ser utilizado caso o custo computacional da função de *fitness* seja irrelevante, pois ele cria novos dois parâmetros para o critério de parada, a quantidade de iterações sem nenhuma alteração e o valor mínimo aceitável para considerar que houve uma variação. Caso o numero de iterações seja alto demais ou o valor mínimo seja baixo demais, ou até mesmo ambos, farão chamadas desnecessárias a função de *fitness*;
- **Finalizar quando o raio do enxame de partículas ficarem próximo de zero:** Nesse método é calculada a distância euclidiana entre as partículas do enxame. Se elas forem menores que uma determinada variável, a condição de parada seria atingida. Mais uma vez,

esse método também presume que você conheça um pouco sobre o problema, pois a má definição da variável mínima comprometerá a finalização do algoritmo.

- **Finalizar quando a função encosta for aproximadamente zero:** Nesse método a função é calculada como sendo a variação da partícula em torno da sua melhor posição encontrada. Caso não sejam encontrados mais máximos globais (por exemplo), e essas partículas fiquem presas em mínimos locais, o algoritmo finaliza prematuramente, sendo um potencial problema.

4. Algoritmo

Nesta seção serão apresentados os dois primeiros e principais algoritmos do PSO: o *Global Best PSO* e o *Local Best PSO*. [1][4] Em ambos, cada partícula do enxame é representada por três vetores [4]:

- $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$, que representa sua posição no espaço de busca D-dimensional;
- $y_i = (y_{i1}, y_{i2}, ..., y_{iD})$, que representa a melhor posição encontrada pela própria partícula;
- $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$, que representa a velocidade atual da partícula;

As posições e velocidades iniciais das partículas do enxame são, geralmente, inicializadas de modo uniformemente aleatório através do espaço de busca do problema. Além disso, a posição atual das partículas é atualizada de acordo com a seguinte equação:

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

O que irá mudar entre os algoritmos, portanto, será a forma como a vizinhança da partícula é estabelecida - qual topologia é utilizada - e o modo como a velocidade de cada partícula é atualizada.

De modo genérico, o pseudocódigo do PSO pode ser escrito como segue:

```
Cria e inicializa um enxame n_x-dimensional;
repete
    para cada partícula do enxame faça
        // estabelece a melhor posição da partícula
        se f(x_i) < f(y_i) então
            y_i = x_i;
        fim
        // estabelece a melhor posição global
        se f(y_i) < f(ŷ) então
            ŷ = y_i;
        fim
    fim
    para cada partícula do enxame faça
        atualiza a velocidade atual da partícula;
        atualiza a posição atual da partícula conforme a equação (1);
    fim
até a condição de parada ser verdadeira;
```

Algoritmo 4.1. - Pseudocódigo básico do algoritmo do PSO

4.1 *Global Best PSO*

Conforme mostrado na seção 2.2.1, este algoritmo utiliza a topologia estrela e, portanto, a vizinhança de cada partícula é formada por todo o enxame. Após sua inicialização, cada partícula terá sua velocidade (v_i) atualizada, a cada iteração, de acordo com a seguinte equação:

- $v_{ij}(t + 1) = v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)]$ (2)

A velocidade da partícula, conforme mostrada em (2), é formada por três termos: (i) a velocidade atual da partícula, $v_{ij}(t)$; (ii) a componente cognitiva da partícula, $y_{ij}(t) - x_{ij}(t)$, que mede o quão boa é a posição atual da partícula com relação à melhor posição já encontrada por ela; e (iii) a componente social da partícula, $\hat{y}_j(t) - x_{ij}(t)$, que mede o quão boa é a posição atual da partícula com base na melhor posição já encontrada por sua vizinhança.

Ainda em (2), $y_{ij}(t)$ é a melhor posição encontrada pela partícula *i* na dimensão *j*; $x_{ij}(t)$ é a posição atual da partícula *i* na dimensão *j*; $\hat{y}_j(t)$ é a melhor posição encontrada pelo enxame na dimensão *j*; c_1 e c_2 são constantes de aceleração utilizadas para regular o grau de contribuição da componente cognitiva e social, respectivamente; e r_1 e r_2 são números aleatórios utilizados para introduzir a componente estocástica do algoritmo.

4.2. *Local Best PSO*

Este algoritmo, por sua vez, utiliza a topologia anel e, conseqüentemente, sua vizinhança é limitada por *n* vizinhos. Nele, a melhor posição encontrada pela vizinhança de uma partícula ($\hat{y}_i$) pode ser definida por:

- $\hat{y}_i(t + 1) \in \{ N_i \mid f(\hat{y}_i(t + 1)) = \min\{ f(x) \}, \forall x \in N_i \}$ (3)

onde N_i é a vizinhança de cada partícula e pode ser definida como:

- $N_i = \{ y_{i-nNi}(t), y_{i-n(Ni+1)}(t), \ldots , y_{i-1}(t), y_i(t), y_{i+1}(t), \ldots , y_{i+nNi}(t) \}$ (4)

sendo n_{Ni} o tamanho da vizinhança.

Com base nessas informações, a velocidade de cada partícula é atualizada de acordo com a seguinte equação:

- $v_{ij}(t + 1) = v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)]$ (5)

A velocidade da partícula, conforme mostrada em (5), é formada por três termos: (i) a velocidade atual da partícula, $v_{ij}(t)$; (ii) a componente cognitiva da partícula, $y_{ij}(t) - x_{ij}(t)$, que mede o quão boa é a posição atual da partícula com relação à melhor posição já encontrada por ela; e (iii) a componente social da partícula, $\hat{y}_{ij}(t) - x_{ij}(t)$, que mede o quão boa é a posição atual da partícula com base na melhor posição já encontrada por sua vizinhança.

Ainda em (5), $y_{ij}(t)$ é a melhor posição encontrada pela partícula *i* na dimensão *j*; $x_{ij}(t)$ é a posição atual da partícula *i* na dimensão *j;* $\hat{y}_{ij}(t)$ é a melhor posição encontrada pela vizinhança da partícula *i* na dimensão *j* e é atualizada conforme a equação (3)*;* c_1 *e* c_2 são constantes de aceleração utilizadas para regular o grau de contribuição da componente cognitiva e social, respectivamente; e r_1 e r_2 são números aleatórios utilizados para introduzir a componente estocástica do algoritmo.

5. Referências

[1] KENNEDY, James; EBERHART, Russell. **Particle Swarm Optimization.** In Proceedings of IEEE International Conference on Neural Networks, 1995. pp. 1942–1948. Perth: nov./dez. 1995.

[2] SHI, Yuhui; EBERHART, Russell. **A Modified Particle Swarm Optimizer**. In: Proceedings of IEEE International Conference on Evolutionary Computation, 1998. pp. 69–73. Alasca: mai. 1998.

[3] KENNEDY, James; EBERHART, Russell C.; SHI, Yuhui. **Swarm Intelligence**, Morgan Kaufmann, 2001. 512 p.

[4] BRATTON, Daniel; KENNEDY, James **Defining a Standard for Particle Swarm Optimization**. In: Swarm Intelligence Symposium, 2007 (SIS 2007). IEEE. pp. 120–127. Honolulu: abr. 2007.

[5] ENGELBRECHT, Andries P. **Computional Intelligence: An Introduction**. 2nd ed. John Wiley & Sons, 2007. 597p.

[6] EBERHART, Russell; KENNEDY, James. **A New Optimizer Using Particle Swarm Optimization.** In Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995

[7] Xiujuan Lei, Zhongke Shi. **The Variations, Combination Strategies Analysis of Particle Swarm Optimization.** In: Proceedings of the Third International Conference on Natural Computation, 2007. pp. 743-750.