INESCPORTO
INSTITUTO DE ENGENHARIA DE SISTEMAS
E COMPUTADORES DO PORTO
LABORATÓRIO ASSOCIADO

EPSO - Evolutionary Particle Swarm Optimization, a new meta-heuristic variant with application in Power Systems

POSI/EEA-ESE/60980/2004

Estudo de variantes de Optimização por Enxames Evolucionários de Partículas (EPSO) e o seu comportamento num problema real de previsão de potência de um parque eólico

Cristina Andreia Araújo Cerqueira

Relatório

Outubro de 2005

União Europeia

República Portuguesa

INESCPORTO
INSTITUTO DE ENGENHARIA DE SISTEMAS
E COMPUTADORES DO PORTO
LABORATÓRIO ASSOCIADO

Nota

Este relatório, numa versão ligeiramente distinta, foi objecto de submissão à Faculdade de Ciências como parte do trabalho de estágio da autora no INESC Porto [1], integrada no projecto EPSO.

Índice

1 Introdução

1.1 Motivação e objectivos

Pretendeu-se através do presente trabalho, melhorar a performance do Algoritmo de Optimização por Enxames Evolucionários de Partículas (EPSO, *Evolutionary Particle Swarm Optimization*), desenvolvido em 2002 no INESC Porto [2–4], assim como confirmar propriedades de eficácia, eficiência e robustez do algoritmo, na resolução de problemas reais complexos.

O facto de o EPSO ser um algoritmo com características que o tornam superior a abordagens semelhantes [2, 5], estimulou o estudo mais aprofundado do seu funcionamento e desempenho. Em particular, novas ideias para o esquema adaptativo, regras de comunicação e métodos de exploração do espaço de pesquisa irão ser testados neste estudo (capítulo 4). Outra motivação sempre presente em Investigação Operacional é a constante descoberta de algoritmos mais eficazes que os existentes para a resolução de problemas muito difíceis (por exemplo, a determinação do mínimo global da função Rosenbrock através de métodos computacionais).

Recentemente, surgiu um novo critério para treino paramétrico, supervisionado ou não, de sistemas adaptativos não lineares, com maiores potencialidades que o habitual critério de minimização do Erro Médio Quadrático (EMQ) [16–18]. Este critério de *Information-Theoretic Learning* (ITL) baseia-se na minimização da entropia dos erros de treino do sistema, o que se traduz num aproveitamento máximo da informação da distribuição destes erros, resultando depois num melhor sistema de classificação/previsão.

Até agora, o critério de ITL foi apenas testado pelos seus autores em sistemas de Redes Neuronais Artificiais (MLP's e TDNN's [16]), e neste trabalho, será aplicado a um Sistema de Inferência Difusa do tipo Takagi-Sugeno (SID-TS) de ordem 0, que pela sua organização também pode ser considerado como um Sistema Neuro-Difuso [19].

Como se sabe, o algoritmo de treino mais utilizado neste tipo de sistemas adaptativos, é o algoritmo de Retro-Propagação, cuja função de custo é a minimização do EMQ. Este algoritmo clássico baseado no gradiente tem o problema de poder ficar "preso" em óptimos

locais. Portanto, é sempre interessante avaliar o desempenho de uma alternativa do tipo meta-heurístico como o EPSO, com o objectivo de se conseguirem, de modo eficiente, melhores soluções para o conjunto de pesos que definem o sistema.

Por isso, após uma tentativa de melhoramento das características do algoritmo EPSO, testou-se a sua eficácia no treino de um Sistema de Inferência Difuso do tipo Takagi-Sugeno (ver capítulo 5) , numa aplicação a um problema real de previsão da produção em potência de um parque eólico. O EPSO foi usado para optimizar duas funções de custo: a minimização do EMQ e a minimização da entropia segundo o critério ITL. Pretendeu-se com este estudo, avaliar a real a superioridade do segundo critério para a obtenção de uma melhor previsão e, simultaneamente, mostrar como uma meta-heurística do tipo EPSO determina os pesos de um modelo segundo o paradigma do treino supervisado, em alternativa à clássica retropropagação.

1.2 Metodologia

Para a realização do trabalho proposto, foram utilizadas as tecnologias Java® (JDK versão 1.5), IDE Eclipse (versão 3.0), MATLAB® (versão 7.0) e Excel (Microsoft Office XP). A metodologia adoptada foi a seguinte:

Fase 1: Estudo Bibliográfico de Computação Evolucionária e do EPSO, assim como da tecnologia Java (Linguagem de Programação Orientada a Objectos).
Fase 2: Teste do algoritmo EPSO com funções teóricas conhecidas, com o objectivo de familiarização com o funcionamento do algoritmo.
Fase 3: Implementação de Variantes do algoritmo EPSO, utilizando Java.
Fase 4: Estudo Bibliográfico de conceitos de ITL e Sistemas de Inferência Difusa do tipo Takagi-Sugeno; Implementação de um SID-TS de ordem 0, em MATLAB e Java, cujo algoritmo de treino é um EPSO, com funções de custo os critérios de EMQ e de ITL.
Fase 5: Aplicação prática: Previsão de Produção de Energia de um Parque Eólico.
Fase 6: Preparação do Artigo: V. Miranda, C. Cerqueira, C. Monteiro, *Entropy and Fuzzy Inference Systems – and application to Wind Power Prediction*, submetido a IEEE Transactions on Power Systems.

2 Fundamentos Teóricos

2.1 Optimização

O conceito de optimização está bem identificado como um mecanismo de análise de decisões complexas, envolvendo selecção de valores para variáveis. A intenção é encontrar a melhor solução, respeitando, se necessário, restrições de viabilidade imposta aos parâmetros do problema. Tem-se então o seguinte problema global de optimização:

$$\begin{aligned} &minimizar\ \ f(x) \\ &sujeito\ a\ \ x \in D \end{aligned}$$

onde $x = (x_1, x_2, ..., x_n) \in R^n$, $f : R^n \to R$ é a função objectivo e $D = \{x \mid a_i \le x_i \le b_i\}$, para $i = 1,2,...,n$, é o espaço de procura.

Quando se trata de problemas complexos (não lineares, não convexos ou com inúmeros óptimos locais fora da região do ponto óptimo), os métodos de procura tradicionais, como o método do gradiente ou o método *downhill simplex*, possuem imensa dificuldade em convergir para a zona da melhor solução, pois, uma vez chegados a um óptimo local, não possuem mecanismos que os permitem sair de lá. A dificuldade é ainda maior, quando *f(x)* não tem derivadas definidas, como acontece na maioria das aplicações reais.

Para tentar ultrapassar estes problemas, foram surgindo durante as últimas décadas numerosos métodos heurísticos, designados genericamente como "meta-heurísticas", tendo muitos (embora não todos) uma convergência de natureza estocástica. Abordagens orientadas a populações, inspiradas em selecção natural e comportamento social, mostraram-se eficientes e eficazes, quando comparadas com métodos de optimização matemática ou algorítmica clássicos [2, 8, 9]. Entre estes novos métodos, classificados como pertencendo à nova corrente da "inteligência computacional", por serem métodos que exibem em geral capacidade de aprendizagem, citam-se o "simulated annealing", a pesquisa tabu, a computação evolucionária nas suas diversas variantes, a optimização por colónia de formigas ou a optimização por

enxame de partículas. Nas secções seguintes, serão apresentados alguns desses novos paradigmas.

2.2 Computação Evolucionária

Na natureza, existe um processo de selecção dos seres vivos, onde os indivíduos mais preparados para a competição dominam os mais fracos e sobrevivem. Isto acontece porque esses seres possuem alguma característica que os distingue dos restantes elementos da sua espécie – indivíduos mutados. Por herança, essa característica provavelmente passará para os seus descendentes, e, assim, estes terão também hipóteses de se saírem vencedores acabando a característica por se impor na população.

A partir desta teoria de evolução de Darwin, surgem no início dos anos 60, a Programação Evolucionária (Lawrence J. Fogel), as Estratégias de Evolução (I. Rechenberg e H. P. Schwefel) e os Algoritmos Genéticos (John Holland).

Estas concepções visam encontrar a solução óptima de um problema, qualquer que seja a natureza das suas variáveis. Estas são representadas por indivíduos de uma população (possíveis soluções). Após serem avaliados por uma função a optimizar, são seleccionados os melhores para a criação de novos indivíduos. Os indivíduos da nova geração são, por sua vez, avaliados para eliminar os de pior desempenho e segue-se nova fase de reprodução, originando uma geração subsequente. Este processo repete-se geração após geração, e a população deverá, em princípio, ir melhorando, ou seja, ir-se enriquecendo de indivíduos com melhor avaliação, até que um certo critério de paragem fica satisfeito. O melhor indivíduo encontrado no processo é tomado, então, como a solução do problema de optimização em causa [7].

Mas estes métodos também têm diferenças importantes. Os Algoritmos Genéticos enfatizam operadores genéticos tal como se observam na natureza: mutação e recombinação (*crossover*), e as soluções são codificadas numa sequência, designada por *cromossoma* – método de genótipo. Por sua vez, as Estratégias de Evolução (ES, *Evolutionary Strategies*) e a Programação Evolucionária (EP, *Evolutionary Programming*) são métodos de fenótipo, isto é, a representação das soluções de um problema baseia-se apenas nas variáveis do problema,

sem passar por qualquer algoritmo intermédio de codificação/descodificação, sendo também aplicados operadores de mutação e recombinação.

Na subsecção seguinte irá ser descrito apenas o modelo auto-adaptativo σSA-ES (*self-adaptive Evolution Strategies*) das Estratégias de Evolução, pois foi a partir deste que surgiram ideias para a construção do algoritmo em estudo neste relatório, o EPSO.

2.2.1 Modelo σSA(1,λ)-ES

Nas ES, um indivíduo é composto por parâmetros estratégicos e parâmetros objecto. Os primeiros referem-se a desvios padrão σ para as distribuições que descrevem as probabilidades de mutação e estão também eles sujeitos a evolução, e os segundos correspondem às variáveis do problema.

A estratégia do modelo σSA(1,λ)-ES consiste em: se um indivíduo é seleccionado para uma geração seguinte, os seus parâmetros estratégicos sobrevivem com ele. Estes parâmetros estratégicos, se foram óptimos, devem conduzir o processo num regime de taxa de progressão óptima, ou seja, com o máximo valor esperado de progressão por geração [7].

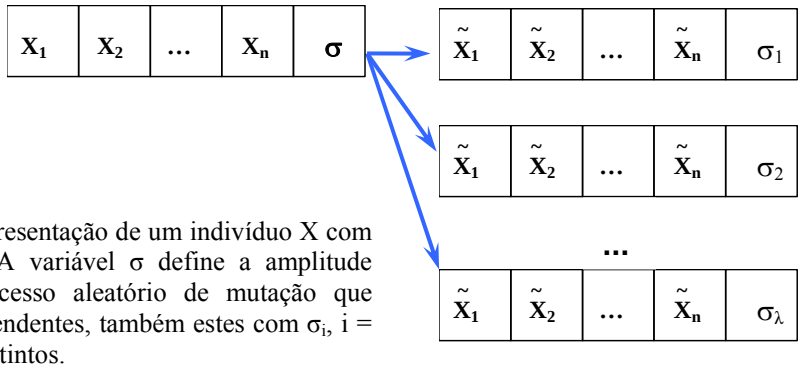


Figura 2.2.1 – Representação de um indivíduo X com n variáveis reais. A variável σ define a amplitude (variência) do processo aleatório de mutação que originará os λ descendentes, também estes com σ_i, i = 1,..,λ mutados e distintos.

O procedimento de selecção pode ser efectuado por elitismo (o melhor de cada geração é conservado para a geração seguinte) ou por torneio estocástico [2]. No torneio estocástico mais simples, o melhor indivíduo é seleccionado com uma dada probabilidade, em geral elevada mas diferente de 1 [7]. Nos métodos implementados no presente trabalho foi definido o parâmetro *sorte*, correspondendo à probabilidade do pior indivíduo sobreviver.

2.3 Optimização por Enxame de Partículas (PSO)

A optimização por enxame de partículas (PSO – *Particle Swarm Optimization*) foi originalmente desenvolvida por James Kennedy (Psicólogo Social) e Russel Eberhart (Engenheiro Electrotécnico) em 1995, e tem sido comparada com algoritmos genéticos [8, 10, 11] pela eficiência em problemas de optimização. O principal conceito deste algoritmo é a partilha de informação e colaboração entre indivíduos (conjuntos de soluções que evoluem no espaço de alternativas – *partículas*) através de simulação do seu comportamento social.

Cada partícula *i* é considerada como uma potencial solução para um dado problema de optimização, num espaço D-dimensional, e é representada por:

- Vector de posição $X_i = (x_{i1}, x_{i2}, ..., x_{iD})$
- Vector de velocidade $V_i = (v_{i1}, v_{i2}, ..., v_{iD})$
- Vector para a melhor posição ocupada pela partícula até ao momento, $b_i = (b_{i1}, b_{i2}, ..., b_{iD})$

A melhor posição ocupada pelo conjunto total de partículas, *enxame*, é também memorizada no vector $b_G = (b_{g1}, b_{g2}, ..., b_{gD})$. Quando uma partícula se desloca no hiperespaço, as suas velocidades (V_i) e posições (X_i) variam de acordo com as seguintes equações (apresenta-se um dos modelos derivados do modelo original):

$$V_i^{novo} = Dec(t) \cdot W_{i0} \cdot V_i + rand_1 \cdot W_{i1} \cdot (b_i - X_i) + rand_2 \cdot W_{i2} \cdot (b_G - X_i)$$
$$X_i^{novo} = X_i + V_i^{novo}$$

Como se pode verificar, a nova velocidade resulta da soma de três termos. O primeiro simboliza a inércia, e conduz a partícula para a direcção que esta vinha seguindo; o segundo simboliza a memória, atraindo a partícula para o melhor ponto encontrado durante a sua trajectória; e por fim, o último termo simboliza a cooperação, que conduz as partículas para o melhor ponto até então encontrado pelo colectivo do enxame.

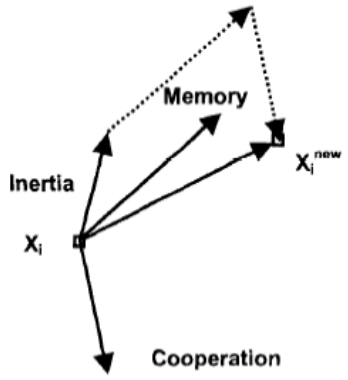


Figura 2.3.1 – Ilustração do movimento de uma partícula *i*, influenciado pelos três termos de inércia, memória e cooperação.

Os parâmetros W_{i1} e W_{i2} são duas constantes positivas, geralmente com valor igual a 2, $rand_1$ e $rand_2$ são valores aleatórios uniformes entre 0 e 1, e $Dec(t)$ é uma função decrescente ao longo do tempo, reduzindo progressivamente a importância da inércia, parâmetro W_{i0}, que toma um valor inicial de 1.4 [12, 13]. Estes valores para os parâmetros foram determinados empiricamente.

Ao contrário das Algoritmos Evolucionários, o algoritmo PSO não efectua a operação de selecção [7, 10], e todas as partículas são mantidas como membros da população ao longo do curso do algoritmo. No entanto, utiliza o conceito de *fitness* como todos os paradigmas evolucionários. O ajustamento das velocidades através de informação de outras posições é comparável com a operação de recombinação existente nos Algoritmos Genéticos, e ocorre também, de uma forma implícita, a operação de mutação, pois os factores de memória e cooperação são ajustados aleatoriamente em cada iteração.

Resumindo, é possível olhar para a PSO através de uma perspectiva evolutiva [7], e neste sentido, pode-se falar em função de adaptação, de população (enxame) e indivíduos (partículas), e de criação de descendentes de geração em geração, através do movimento de partículas de iteração para iteração.

2.4 Príncipio de *Information Theoretic Learning* (ITL)

O Erro Médio Quadrático (EMQ) tem sido utilizado de forma generalizada como critério de treino em todos os sistemas adaptativos, incluindo filtros lineares, redes neuronais

artificiais e sistemas de inferência difusa. Existem duas razões para que isso aconteça: a simplicidade analítica deste critério, e a admissão de que quase todos os fenómenos da natureza podem explicados ou aproximados por uma distribuição Gaussiana. A função densidade de probabilidade (fdp) desta distribuição é caracterizada apenas pelos momentos de primeira e segunda ordem. Daqui, e com a suposição de linearidade e gaussianidade, o EMQ, que se concentra nos momentos de segunda ordem, será capaz de extrair toda a informação possível de um conjunto de dados, cujas estatísticas são definidas apenas pela média e variância [14].

No entanto, como se sabe, a maior parte dos problemas da vida real são governados por equações não lineares, e os fenómenos aleatórios, mesmo que à entrada possam ter comportamento gaussiano, estão muito longe de serem explicados por uma distribuição normal após qualquer transformação não linear. Consequentemente, para o treino de sistemas adaptativos não lineares, é necessário um critério que tenha em conta tanto os momentos de segunda ordem, como os momentos de ordens superiores.

O princípio de *Information Theoretic Learning* (ITL) é uma abordagem muito recente de modelação de sistemas lineares ou não lineares [15], e propõe um novo critério de optimização, mais robusto que o EMQ: a entropia. Por definição, a entropia é uma quantidade escalar que descreve uma medida para a informação contida numa dada função de densidade de probabilidade. Consequentemente, quando a entropia é minimizada, todos os momentos da fdp em questão são considerados [16].

Nos sistemas de emulação da realidade submetidos ao que se designa como "treino supervisado", existe um Output associado a um Input, mas a forma analítica que descreve a relação é desconhecida. O objectivo é então, a partir de uma amostra de dados, descobrir um conjunto de parâmetros (pesos W) que constroem uma função Input-Output adequada, por forma a optimizar um critério de desempenho associado ao erro ou diferença entre o Output-Alvo desejado e o Output realizado pelo sistema.

A ideia base da abordagem ITL é a seguinte: descobertos os pesos W, tais que a fdp dos erros ε = Target – Output é descrita por uma função Delta Dirac (significando que todos os erros são iguais), fica encontrado um sistema que reproduz um Output exactamente igual aos dados alvo ou resultados pretendidos, bastando para isso aplicar um *bias* (que corresponde ao valor médio da fdp dos erros, isto é, o desvio em relação à origem) ao resultado [15].

Como um valor mínimo de entropia é atingido quando a fdp dos erros é uma função Delta Dirac [14], o critério a utilizar será a minimização desta entropia.

2.4.1 Entropia de Renyi

Shannon, em 1948, definiu a entropia de uma distribuição de probabilidade P = (p_1,p_2,…,p_n) como:

$$H_S(P) = \sum_{k=1}^{N} p_k \log\left(\frac{1}{p_k}\right) \text{ com } \sum_{k=1}^{N} p_k = 1 \text{ e } p_k \geq 0$$

Apesar de esta definição possuir todas as propriedades para descrever uma medida de informação, outras definições de entropia existem como a entropia de Renyi, deduzida de princípios mais gerais da teoria da informação [18]:

$$H_{R\alpha} = \frac{1}{1-\alpha} \log \sum_{k=1}^{N} p_k^{\alpha} \quad \text{com } \alpha > 0, \alpha \neq 1$$

De facto, a entropia de Renyi é uma família de funções $H_{R\alpha}$ dependendo do parâmetro α. Existe uma relação entre a entropia de Shannon e a de Renyi, dada por:

$$H_{R\alpha} \geq H_S \geq H_{R\beta} \text{ se } \beta > 1 > \alpha > 0$$
$$\lim_{\alpha \to 1} H_{R\alpha} = H_S$$

A definição de entropia pode ser alargada para o caso de uma variável aleatória contínua Y com fdp $f_Y(z)$, e, no caso da entropia de Renyi quando $\alpha = 2$, tem-se a entropia quadrática:

$$H_{R2} = -\log \int_{-\infty}^{+\infty} f_Y^2(z) dz$$

Como se pode verificar, a entropia de Renyi é muito mais "simpática" de implementar computacionamente do que a entropia de Shannon: ao invés de se ter uma soma de logaritmos pesados probabilisticamente, tem-se agora o logaritmo de uma soma de probabilidades [15].

2.4.2 Método das Janelas de Parzen

O método das Janelas de Parzen é um modo eficiente para aproximação da fdp de uma amostra de valores discretos $y_i \in R^M$, i = 1, …, N num espaço M-dimensional. Tem-se então a estimação para a fdp de uma variável aleatória *Y*, definida como:

$$\hat{f}_Y(z,y) = \frac{1}{N}\sum_{i=1}^{N} k(z - y_i)$$

onde N é o número de pontos da amostra e *k* é uma função kernel. A escolha mais comum para kernel é a função simétrica Gaussiana. Como esta função é contínua e diferenciável, a fdp estimada será também ela, contínua e diferenciável [16]. A descrição de *k*(.) é dada por:

$$k(z) = G(z,\sigma^2 I) = \frac{1}{(2\pi\sigma^2)^{\frac{M}{2}}} e^{-\frac{z^T z}{2\sigma^2}}$$

É fácil de perceber que o tamanho da janela, definido pelo valor de σ, é importante para a obtenção de uma função de estimação da f_Y mais suave (valores de σ grandes) ou mais irregular [15].

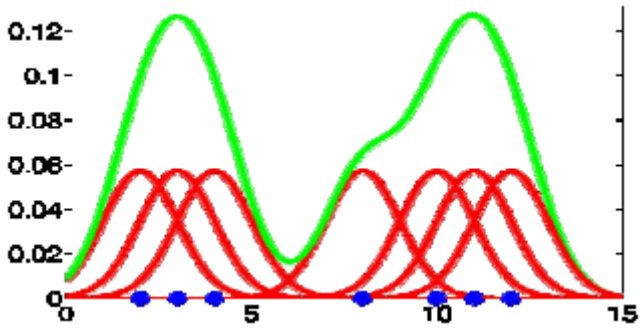


Figura 2.4.1 – Representação gráfica do método de Parzen para a estimação da fdp (a verde) de um conjunto de 7 pontos discretos definidos pelo conjunto D = {2, 3, 4, 8, 10, 11, 12}, onde σ = 1.

2.4.3 Critério ITL

Principe *et al.* demonstraram que, utilizando o conceito de entropia quadrática de Renyi, combinado com a estimação não-paramétrica da fdp através do método de Parzen, é

possível obter um método computacional tratável e suficientemente prático para um critério de optimização baseado em entropia e incidindo directamente nas amostras de erro. Tem-se então o seguinte estimador para a entropia de um conjunto discreto de pontos {y}:

$$H_{R2}(\{y\}) = -\log \int_{-\infty}^{+\infty} \hat{f}_Y^2(z)dz = -\log V(\{y\}) \text{ , onde}$$

$$V(\{y\}) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\int_{-\infty}^{+\infty} G(z-y_i,\sigma^2 I)G(z-y_j,\sigma^2 I)dz$$

Pelas propriedades de convolução da função Gaussiana, vem:

$$V(\{y\}) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} G(y_i - y_j, 2\sigma^2 I)$$

Portanto, para a estimação da entropia $H_{R2}(\{y\})$ não é necessário o cálculo de integrais, bastando apenas calcular o somatório das Gaussianas das distâncias entre pares de amostras. Note que o EMQ é calculado apenas com um conjunto de N amostras, enquanto que a Entropia é estimada com $\binom{N}{2}$ pares diferentes. Logo, está a ser extraída mais informação dos dados, do que com o critério do EMQ.

A grande desvantagem da aplicação do método de ITL é a sua complexidade computacional, quadrática com o número de pontos a utilizar para a estimação da fdp pelo método de Parzen ($O(N^2)$). Isto torna-se um problema, quando o conjunto de dados é elevado e/ou é utilizada uma técnica iterativa para treino [17].

2.5 Sistemas de Inferência Difusa do tipo Takagi-Sugeno

Um Sistema de Inferência Difusa do tipo Takagi-Sugeno (SID-TS) pode ser definido como um sistema que estabelece uma correspondência entre um conjunto de valores difusos e um outro conjunto de valores rígidos (reais, determinísticos).

O seu núcleo consiste numa base de regras difusas e num motor de inferência, o qual calcula em que medida é que cada regra é activada (disparada) para um dado padrão de entrada, e qual a consequência agregada da activação das regras. O grau de pertença do valor

de uma variável a um conjunto difuso é definido pela função de pertença associada a esse mesmo conjunto. Neste texto, adoptaremos uma descrição Gaussiana para as funções de pertença, definidas genericamente da seguinte forma:

$$m = f(x, \mu, \sigma) = e^{-\left(\frac{\mu - x}{\sigma}\right)^2}$$

onde *m* é o grau de pertença, *x* é o valor (real) de instanciação da variável difusa, μ é o desvio em relação à origem e σ é a descrição da amplitude (dispersão) da função Gaussiana. A sua utilização (em vez de funções de pertença triangulares, como é prática comum na construção de conjuntos difusos) deve-se ao facto conveniente de a função Gaussiana poder ser derivável em todo o domínio [18].

Um SID-TS simples pode ser apresentado numa organização "neuronal", como na Figura D.1, cujas regras típicas são da forma:

Se A é MUITO **e** B é MÉDIO **então** C é (função das entradas)

↓ Antecedente ↓ Consequente

Vê-se que neste tipo de regra se tem o antecedente difuso mas o consequente "rígido" ou determinístico – é uma regra "semi-difusa". A activação do antecedente é feita por instanciação de valores concretos (Ex.: a e b) os quais, no processo de difusão, ficam associados a valores de pertença m_i dos conjuntos difusos definindo os conceitos de "muito" ou "médio".

Podendo adoptar-se uma de várias t-normas para definir a operação de conjunção "E", escolheremos a t-norma produto sendo desta forma o valor de pertença do antecedente de cada regra igual ao produto dos valores de pertença m_i de cada um dos termos i. Assim, para a regra k com dois termos i e j no antecedente, a sua "intensidade de activação" (que corresponde ao valor de pertença do antecedente, tomado globalmente) é dada por:

$$g_k = m_i \cdot m_j$$

O resultado de cada regra k é um valor real dado por uma função f_k que assume habitualmente o aspecto:

$$f_k = \sum_{i=1}^{N} w_{ki} x_i + w_k$$

onde N é o número de variáveis de entrada, os x_i são os valores reais instanciados de cada termo do antecedente da regra k e os w_{ki} e w_k são coeficientes (pesos) que definem cada função. Estes pesos diferentes de zero definem um sistema de TS de primeira ordem; se todos os $w_{ki} = 0$ e só os w_k forem não nulos, então fica um SID-TS de ordem zero, e tem-se:

$$f_k = w_k$$

A saída y de um modelo de TS é definida geralmente como:

$$y = \frac{\sum_{j=1}^{NF} g_j f_j}{\sum_{j=1}^{NF} g_j}$$

ou seja, como uma soma do resultado de cada regra pesada pela respectiva intensidade de activação, sendo NF o número de regras (ou o número de regras activadas em cada caso).

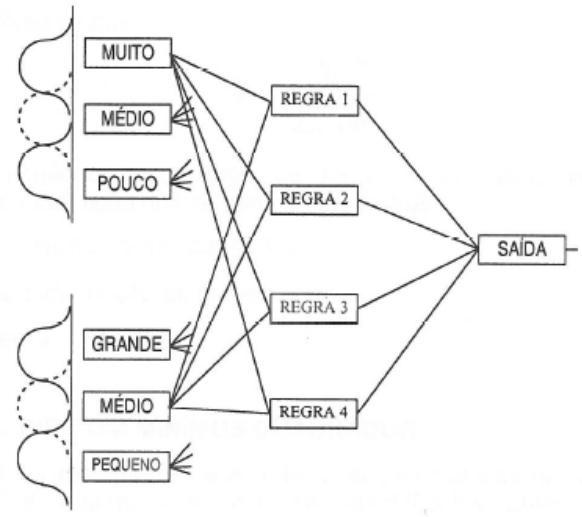


Figura D.1 – Arranjo neuronal de um SID. Representam-se 2 variáveis de entrada, cada uma com 3 valores linguísticos. O antecedente de cada regra é formado pela conjunção "E" das condições. A saída é conseguida por uma operação do tipo "OU" sobre os resultados de cada regra.

3 Optimização por Enxames Evolucionários de Partículas (EPSO)

Como já foi dito atrás, este trabalho centra-se no estudo do algoritmo EPSO – Optimização por Enxames Evolucionários de Partículas, que surge como um híbrido entre as Estratégias Evolutivas e a Optimização por Enxames de Partículas.

Nos modelos σSA-ES, não são só os indivíduos que evoluem para a solução óptima mas também os seus parâmetros estratégicos responsáveis pela operação de mutação, resultando num processo de auto aprendizagem do melhor modo de progressão para o óptimo. Já na PSO não existe competição entre as partículas ou auto-adaptação das suas características (pesos). De facto, se não existisse o factor de *cooperação*, cada partícula evoluiria independentemente das restantes [3]. É a equação de movimento, e este factor de cooperação em particular, que garante as propriedades de convergência e o funcionamento do algoritmo.

O EPSO junta o melhor destes dois paradigmas. É um algoritmo de Optimização por Enxames de Partículas, pois existe troca de informação entre as partículas, enquanto estas se movimentam no espaço de pesquisa; e é um método de Computação Evolucionária, porque as características (pesos) das soluções são mutadas e passadas para as gerações seguintes, através do mecanismo de selecção [3, 4].

No entanto, os autores preferem interpretar o algoritmo sob um ponto de vista totalmente evolucionário. Sob este ponto de vista, será um algoritmo auto-adaptativo evolucionário, onde a clássica operação de recombinação intermediária é refinada, numa forma emprestada da PSO a que os autores chamam de *movimento de partículas* [2].

3.1 Descrição do Algoritmo EPSO

Numa iteração k, tomemos um conjunto de indivíduos, ou partículas, constituídos por um conjunto de parâmetros objecto e parâmetros estratégicos [X, w]. Estes indivíduos irão evoluir ao longo de um dado número de gerações, de acordo com o seguinte esquema geral:

Replicação – cada partícula i é replicada r vezes, resultando num total de r + 1 partículas no espaço de pesquisa. Na versão original do EPSO, foi sempre utilizado um r = 1.

Mutação – cada réplica sofre mutação nos seus parâmetros estratégicos *w*, da seguinte forma:

$$w_{i,j}^{k,novo} = w_{i,j}^{k} + \tau \cdot N(0,1)$$

onde τ é o parâmetro de aprendizagem, fixado externamente, e N(0,1) é um número aleatório com distribuição Gaussiana de média 0 e variância 1. O índice j (j = 1,...,4) refere-se ao peso de inércia, memória, cooperação, e desvio do óptimo. Estes pesos *w* no início do algoritmo tomam valores aleatórios uniformes entre 0 e 1.

Reprodução – cada partícula gera um descendente de acordo com a equação de movimento, semelhante à equação do algoritmo PSO:

$$b_G^{novo} = b_G + w_{i,desvOptimo}^{novo} \cdot N(0,1)$$
$$V_i^{k,novo} = w_{i,inercia}^{k} \cdot V_i^{k} + w_{i,mem}^{k} \cdot (b_i - X_i^{k}) + w_{i,coop}^{k} \cdot (b_G^{novo} - X_i^{k})$$
$$X_i^{k,novo} = X_i^{k} + V_i^{k,novo}$$

Tal como na PSO, também existe um vector que guarda a melhor posição de cada partícula, b_i, e outro que guarda a melhor posição até ao momento encontrada pelo enxame, b_G. No entanto, este último recebe um tratamento diferente. É aplicada uma mutação, com o objectivo de controlar a "dimensão" de uma zona difusa em volta do óptimo corrente encontrado pelo enxame, resultando no vector b_G^{novo}. Com este procedimento, o processo auto-adaptativo pode focar mais ou menos a orientação de um enxame e permite que este continue a ser "agitado" mesmo quando as partículas já convergiram todas para a mesma região do espaço e estão muito próximas [7].

Avaliação – cada descendente tem a sua adaptação avaliada, de acordo com a posição que ocupa no espaço.

Selecção – por Torneio Estocástico, a melhor partícula em cada grupo de r + 1 descendentes de cada indivíduo da geração anterior, é seleccionada com probabilidade (1 – *sorte*), para formar uma nova geração. Por outras palavras, o descendente da partícula original

entra em competição com os r descendentes das partículas replicadas. O parâmetro *sorte* geralmente é um número muito reduzido.

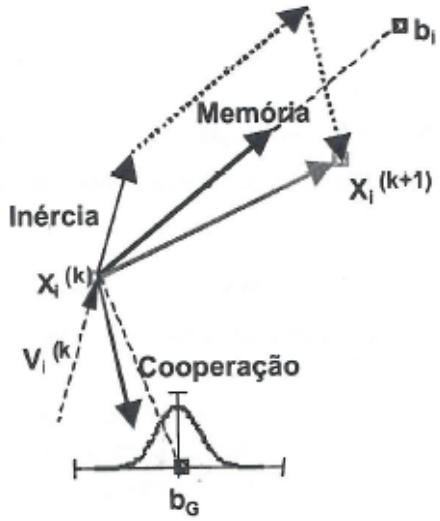


Figura 3.1.1 – Ilustração da reprodução de uma partícula no EPSO. Uma partícula i localizada em *X*, numa iteração (k), origina um descendente na iteração (k+1), sob a influência dos três termos de inércia, memória e cooperação. Relativamente a este último termo, a atracção efectua-se para uma vizinhança definida por uma distribuição Gaussiana do óptimo corrente b_G.

Portanto, dado um enxame de p partículas, o processo pode definir-se como p×σSA(1,r+1)-ES: cada partícula gera r + 1 descendentes e destes só 1 sobrevive. O processo segue em paralelo para todas as p partículas mas mantém-se um acoplamento entre elas por via do termo de cooperação que condiciona a formação de novas partículas [7].

3.2 Alguns Resultados

Na Investigação Operacional há muito que se estabeleceram funções teste para pôr à prova os diversos algoritmos. Optou-se por apresentar algumas dessas funções na secção 4.1, embora sejam também aqui utilizadas para mostrar alguns resultados do EPSO, que servirão de comparação para os resultados das variantes do algoritmo propostas.

Apresentam-se na tabela seguinte, os resultados de 20 simulações do EPSO, para as funções Rosenbrock, Griewank, Parábola e Alpine (todas com óptimo global $f(x) = 0$). O critério de paragem estabelecido foi um número máximo de avaliações à função objectivo,

fixado a 200000. Foram utilizadas 20 partículas, replicadas r = 1 vez, um parâmetro de aprendizagem τ = 0.1 e um parâmetro *sorte* para o Torneio Estocástico igual a 0.01. Podemos também observar na Figura 3.2.2, o comportamento típico do EPSO para estas quatro funções apresentadas.

	Rosenbrock [0;30]^30	Griewank [-300;300]^30	Parábola [-50;50]^30	Alpine [-10;10]^10
Valor médio óptimo	56,92947709	1,97E-02	3,99E-04	0,06362517
Desvio padrão	39,32835988	0,017512552	0,00015763	0,07220618

Tabela 3.2.1 – Resultados de 20 simulações para o ponto óptimo encontrado pelo EPSO, para as funções Rosenbrock, Griewank, Parábola e Alpine, com espaço de procura [0;30], [-300;300], [-50;50] e [-10;10], respectivamente, e dimensão 30, excepto a função Alpine que tem dimensão 10.

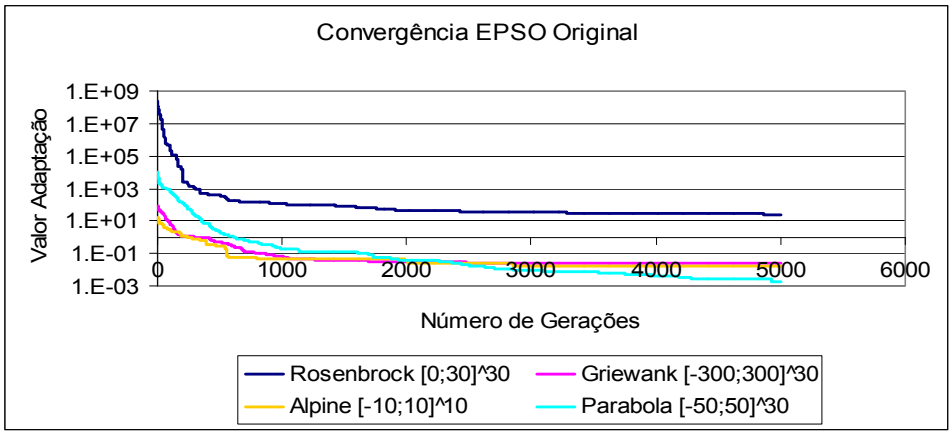


Figura 3.2.2 – Gráfico da convergência típica do EPSO, para as funções teste apresentadas na Tabela 3.2.1.

3.3 EPSO versus PSO

O EPSO ao combinar os mecanismos de mutação/recombinação, que traduzem a equação de movimento das partículas, e a operação de selecção, torna-se mais competitivo e com melhor desempenho que outras abordagens como as ES e o algoritmo PSO, onde apenas um dos operadores contribui para a progressão em direcção ao óptimo [7]. Particularmente, em [5] e [6] está mostrada num conjunto de casos a superioridade do EPSO sobre os Algoritmos Genéticos a a Optimização por Enxames de Partículas.

Comparativamente ao algoritmo PSO, o EPSO possui características auto-adaptativas que o tornam razoavelmente independente de definições externas para os pesos, tornando-o dentro de certos limites insensível à inicialização aleatória dos mesmos [2]. Além disso, testes

efectuados em trabalhos anteriores, mostram que o EPSO é mais robusto (variância dos resultados muito mais pequena do que o PSO Clássica), mais exacto (média dos resultados do EPSO superior à media do PSO) e mais rápido a convergir [7].

A robustez do algoritmo, nesta classe de meta-heurísticas, tem a ver com a garantia (probabilística) de que, independentemente da inicialização, o algoritmo irá convergir para o óptimo ou sua vizinhança. Ela pode avaliar-se, por exemplo, em casos de teste, determinando experimentalmente a distribuição dos resultados numa amostra de casos de um mesmo problema com inicializações aleatórias. Importará, nesta avaliação, determinar não só quão próximo fica o resultado médio do óptimo real como também qual a variância do resultado. É evidente que métodos que resultem num resultado de qualidade e com desvio médio muito pequeno são preferíveis a outros que apresentem grande desvio ou variância, mesmo que o melhor resultado conseguido até seja superior. Isto porque, numa aplicação real, não se espera que o algoritmo seja corrido cem ou mil vezes sobre o mesmo problema – espera-se efectuar apenas uma corrida e confiar no resultado.

De seguida mostram-se alguns resultados obtidos aquando da apresentação do EPSO em 2002, que ilustram a clara superioridade deste algoritmo relativamente ao PSO Clássico.

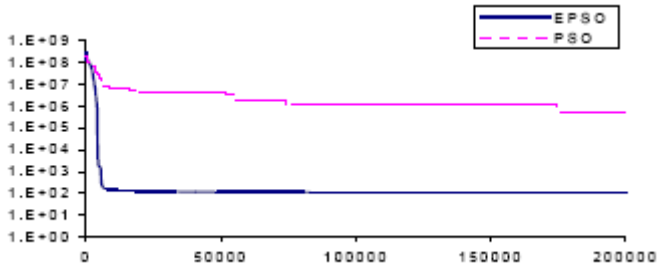


Figura 3.3.1 – Convergências típicas para os algoritmos EPSO e PSO, na função Rosenbrock. No eixo do xx estão representadas o número de avaliações à função, o no eixo do yy está o valor de *fitness* (óptimo) correspondente.

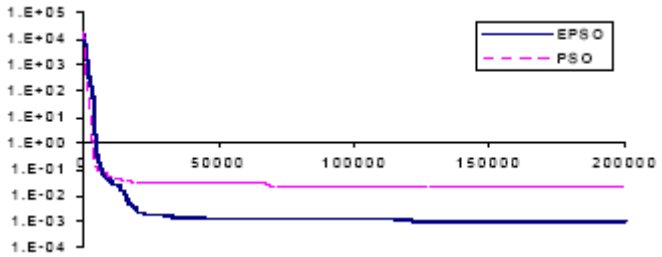


Figura 3.3.2 – Convergências típicas para os algoritmos EPSO e PSO, na função Parábola. No eixo do xx estão representadas o número de avaliações à função, o no eixo do yy está o valor de *fitness* (óptimo) correspondente.

4 Melhorando o Desempenho do EPSO

Nesta secção será descrita a primeira parte do trabalho efectuado neste estágio, que consiste na optimização do algoritmo EPSO. Serão descritas as funções utilizadas para testar as variantes propostas para o algoritmo, e também as variantes propriamente ditas. Optou-se por apresentar os resultados das variantes à medida que estas forem descritas, para melhor compreensão das ideias de optimização para o EPSO.

4.1 Funções Teste

Não há algoritmo de optimização que resolva satisfatoriamente todo o tipo de problema, devido à grande variedade dos mesmos. Por isso, é importante testá-lo com várias funções de características muito diversas como continuidade, descontinuidade, inúmeros pontos locais ou apenas uma solução global. Como as soluções destas funções teste são conhecidas, é possível estão testar a eficiência (rapidez) e eficácia (convergência para a solução global) do algoritmo em questão. Uma vez bem comportado, assume-se que o algoritmo terá performance semelhante em problemas reais, onde não é possível definir analiticamente a função objectivo.

4.1.1 Função Rosenbrock

$$f(X) = \sum_{d=1}^{D-1} (1 - x_d)^2 + 100 \times (x_d^2 - x_{d+1})^2$$

Esta função tem um único mínimo no ponto $X = (1,...,1)_D$, com valor $f(x) = 0$, que se situa num vale longo e com forma parabólica "achatada". Encontrar o vale é trivial, no entanto, a convergência para o óptimo global é muito difícil (ver pela figura 4.1.1). Das funções que serão apresentadas, esta é a mais difícil de optimizar, como aliás se poderá perceber pelos resultados encontrados.

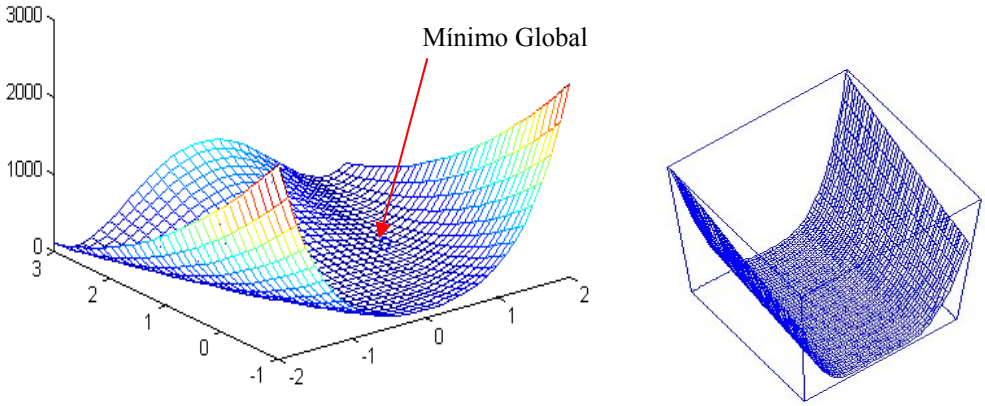


Figura 4.1.1 – Gráfico a 3D, para a função Rosenbrock. À esquerda, esta está representada com mais pormenor, e o ponto de mínimo global está assinalado.

4.1.2 Função Griewank

$$f(X) = \frac{\sum_{d=1}^{D}(x_d - 100)^2}{4000} - \prod_{d=1}^{D}\cos\left(\frac{x_d - 100}{\sqrt{d}}\right) + 1$$

O mínimo global 0.0 encontra-se em $X = (100,...,100)_D$. Esta é uma função com um número elevado de mínimos locais, que vão aumentando com a dimensão. É muito difícil de optimizar, e também de encontrar um método que consiga detectar o mínimo global em menos de 10,000 avaliações à função [20].

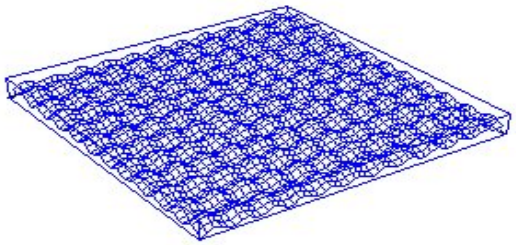


Figura 4.1.2 – Gráfico a 3D, para a função Griewank.

4.1.3 Função Alpine

$$f(X) = \left| x_d \sin(x_d) + 0.1 x_d \right|$$

Uma função nada simétrica, e com um número muito grande de mínimos locais, como se pode ver pela figura em baixo. Um algoritmo como o de Retro-Propagação rapidamente ficaria "encravado" numa destas soluções. O mínimo global fica no ponto $X = (0,\ldots,0)_D$.

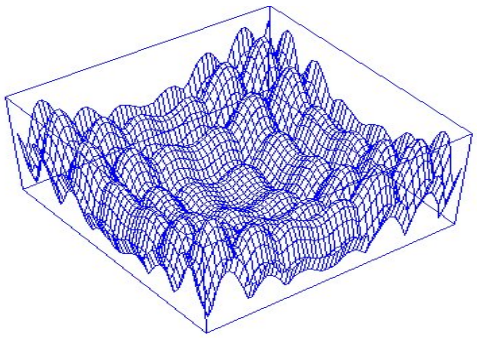


Figura 4.1.3 – Gráfico a 3D, para a função Alpine.

4.1.4 Função Parábola

$$f(X) = \sum_{d=1}^{D} x_d^2$$

Muito conhecida e com apenas um ponto mínimo em $X = (0,\ldots,0)_D$. É muito fácil de ajustar a sua dificuldade, aumentando ou diminuindo a dimensão D. Com esta função, algoritmos baseados no gradiente geralmente funcionam bem, por isso, é um desafio para métodos estocásticos como o EPSO ou o PSO testarem o seu comportamento [21].

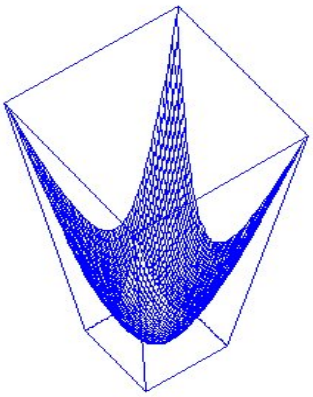


Figura 4.1.5 – Gráfico a 3D, para a função Parábola.

4.2 Implementação de Variantes e Resultados

De seguida, serão apresentadas as variantes propostas para o algoritmo EPSO e os resultados mais significativos obtidos neste estudo. O espaço de pesquisa e dimensão das funções teste utilizadas (tabela seguinte), foram os propostos em [21]. Aqui, encontra-se aberto o desafio de encontrar um algoritmo capaz de optimizar algumas funções teste, dado um valor mínimo para o óptimo global a encontrar.

4.2.1 Implementação de um maior Número de Descendentes

Para melhorar a performance do EPSO, nesta primeira fase tentou-se perceber a influência de um maior número de descendentes na velocidade e qualidade de convergência. Por outras palavras, irá verificar-se se, quando o número de réplicas r é superior a 1, o algoritmo encontra mais rapidamente a zona do ponto óptimo.

Para pôr à prova o algoritmo utilizaram-se as seguintes funções teste, com parâmetros de dimensão D e espaço de procura descritos na seguinte tabela:

Função	D	Domínio
Rosenbrock	30	$[0 ; 30]^D$
Alpine	10	$[-10 ; 10]^D$
Griewank	30	$[-300 ; 300]^D$

Utilizou-se como critério de paragem um número máximo de avaliações à função objectivo igual a 200000, e o número de partículas foi sempre igual a 20 em todas as situações, assim como o parâmetro sorte = 0.01 para o Torneio Estocástico e o parâmetro de aprendizagem τ = 0.1 para a mutação Gaussiana (nas próximas secções também vão ser utilizados estes valores, excepto para a secção 4.2.3, onde é descrito outro tipo de mutação).

De notar que, em regra, o esforço computacional é avaliado com base no número de avaliações à função objectivo, pois é o cálculo da adaptação de um indivíduo que constitui a parte fundamental do esforço computacional dos algoritmos evolucionários ou de enxames [7].

Na seguinte tabela encontram-se os resultados médios e de desvio padrão, do melhor ponto encontrado pelo EPSO em 20 corridas, para algumas das funções atrás descritas.

		r = 1	r = 2	r = 3	r = 4	r = 5	r = 6
Rosenbrock [0;30]^30							
(20 simulações)	media:	55,253326	74,958536	62,170395	87,971531	51,244448	68,054932
	desv padrão:	43,361223	74,176881	63,406294	75,92712	48,356885	57,253469
Alpine [-10;10]^10							
(20 simulações)	media:	0,0812066	0,353969	0,3194998	0,4061612	0,2800334	0,1373007
	desv padrão:	0,0859857	0,4871481	0,4332532	0,4646967	0,3861034	0,1793682
Griewank [-300;300]^30							
(20 simulações)	media:	0,0093572	0,0202653	0,0255501	0,0442176	0,0335398	0,0294582
	desv padrão:	0,0122694	0,0195837	0,0485939	0,0571279	0,0253754	0,0228004
Número de Gerações:		5000	3333	2500	2000	1667	1429

Tabela 4.2.1 – Resultados obtidos após 20 corridas para o ponto óptimo encontrado pelo EPSO, para as funções Rosenbrock, Alpine e Griewank, com um número de réplicas igual a 1 até 6.

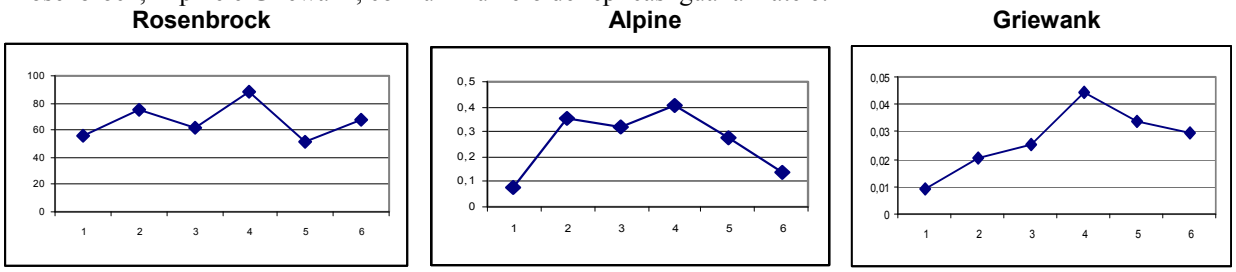


Figura 4.2.2 – Evolução do ponto médio global encontrado pelo EPSO para r = 1 a 6, para cada uma das 3 funções.

Como se pode ver pela Tabela 4.2.1, à medida que r aumenta (o que significa um aumento do número de partículas a serem avaliadas), o número de gerações (iterações) diminui. Dado que o número de avaliações à f.o. é fixo, a relação entre estas variáveis é a seguinte:

$$Iterações = \frac{N^{o}\ Avaliações}{N^{o}\ Partículas \times (r+1)}$$

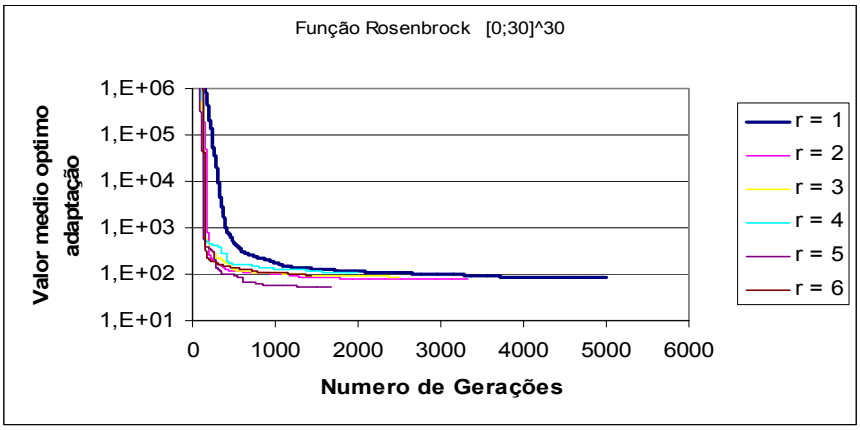


Figura 4.2.3 – Corrida típica para a função Rosenbrock, onde o número de réplicas varia entre 1 (evidenciado) a 6. Nesta simulação, um r = 5 conduziu a uma melhor solução.

Observando novamente a mesma tabela e a Figura 4.2.2, verifica-se que o aumento de partículas não beneficia a procura do melhor ponto. No entanto, pela Figura 4.2.3, constata-se que com esse aumento, a deslocação das partículas para a zona do óptimo global é mais rápida. Por outras palavras, em menos gerações a solução passa de um valor na ordem de 10^6, para um valor de 10^2.

Excepto algumas excepções, a regra geral parece ser que o aumento de r conduz a soluções piores. Isto pode ser explicado, pelo facto de um número de gerações menor não ser suficiente para o algoritmo poder encontrar uma solução mais próxima da real.

4.2.2 Implementação de uma Restrição na Comunicação entre Partículas

Um aspecto importante a ter em conta neste algoritmo, é o modo como as partículas se comunicam entre si. Como já vimos anteriormente, a equação de movimento das mesmas é influenciada pela melhor posição até então encontrada pelo enxame. Ora, nas primeiras iterações, o algoritmo raramente se encontra na vizinhança do óptimo real. Portanto, as partículas serão conduzidas desde início para direcções de "óptimos falsos", e, com grande probabilidade, uma grande parte do espaço de pesquisa não será explorado (esta afirmação será melhor explicada com outros resultados adicionais, na secção 4.2.4).

Pensou-se então em restringir a comunicação entre as partículas, isto é, apenas com uma dada probabilidade, as partículas terão conhecimento da melhor posição até então encontrada. Mais especificamente, apenas 20% das dimensões de cada partícula terão

conhecimento do valor na dimensão correspondente na melhor posição do enxame. Os resultados encontram-se na tabela abaixo.

		r = 1	r = 2	r = 3	r = 4
Rosenbrock [0;30]^30	media	27,097839	27,503186	27,066542	27,472829
	desv padrão	1,4124137	1,2287159	1,0148134	0,9395853
Alpine [-10;10]^10	media	0	0	0	0
	desv padrão	0	0	0	0
Griewank [-300;300]^30	media	1,9697428	1,6740725	0,7353222	0,9974794
	desv padrão	2,2802871	1,6478415	1,038584	1,543596
Número de Gerações:		5000	3333	2500	2000

Tabela 4.2.3 – Resultados de 20 simulações do algoritmo EPSO para as 3 funções referidas. Foi utilizada uma restrição na comunicação e aumento do número de descendentes (réplicas).

Como se pode ver, os resultados melhoraram significativamente para as funções Rosenbrock e Alpine, ao contrário da função Griewank, onde esta variante do EPSO teve pior desempenho que o algoritmo original. Independentemente do número de réplicas escolhido, no caso da função Alpine o óptimo foi encontrado, enquanto que para a função Rosenbrock, esta variante do EPSO retorna uma solução muito robusta (desvio padrão muito menor) e mais próxima do mínimo global.

Concluindo, verifica-se que dependendo do problema que está a ser optimizado, restringir a comunicação entre as partículas, pode ou não ser vantajoso.

4.2.3 Implementação de uma Mutação Lognormal

Uma mutação também muito utilizada em Estratégias de Evolução é a mutação Lognormal. Em [7] obtiveram-se resultados muito satisfatórios, quando se aplicou esta mutação a um problema prático de optimização do Despacho Económico (Sistemas de Potência). Nesta variante, os pesos de cada partícula do algoritmo EPSO são mutados de acordo com:

$$w_i^{novo} = w_i[\log N(0,1)]^{\tau}$$

onde τ é o parâmetro de aprendizagem fixado externamente. Outra mudança efectuada no algoritmo, mas não tão significativa, é a alteração da equação de desvio em relação ao óptimo global, que passa a ser da forma:

$$b_G^{novo} = b_G[1 + w_{i,desvio} N(0,1)]$$

Nesta forma multiplicativa de mutação, a probabilidade de se obter um novo valor multiplicado por m é a mesma que a de ter um valor multiplicado por 1/m. Além disso, os novos valores de mutação e desvio de óptimo global, são sensíveis aos valores anteriores [7].

De seguida mostram-se resultados de 20 simulações, obtidos para a função Rosenbrock aquando do estudo do melhor parâmetro de aprendizagem, Tau (τ). Note-se que não existe restrição da comunicação entre as partículas e o número de réplicas foi sempre igual a 1.

Tau	0.001	0.01	0.05	0.1	0.15	0.2
Media	28.90802921	28.9240841	28.93044851	28.91914369	28.90325701	28.93505598
St Dev	0.169670225	0.143469566	0.104638177	0.11909821	0.222945983	0.127737288

0.25	0.5	1	1.5	2	10	100
28.95885124	28.88238891	28.83871392	28.92277123	28.95013177	28.7933135	28.99120349
0.12981331	0.25318838	0.327507654	0.164080172	0.083981403	0.30862587	0.022361966

Tabela 4.2.4 – Resultados obtidos para a Função Rosenbrock, para diferentes valores de parâmetro de aprendizagem τ, ao fim de 20 simulações para cada.

A partir da Tabela 4.2.4, verifica-se que não existe muita diferença entre os valores médios para o óptimo quando se faz variar τ (o desvio padrão entre os 13 óptimos é de 0.051262), mesmo sendo uma variação muito grande na ordem de grandeza.

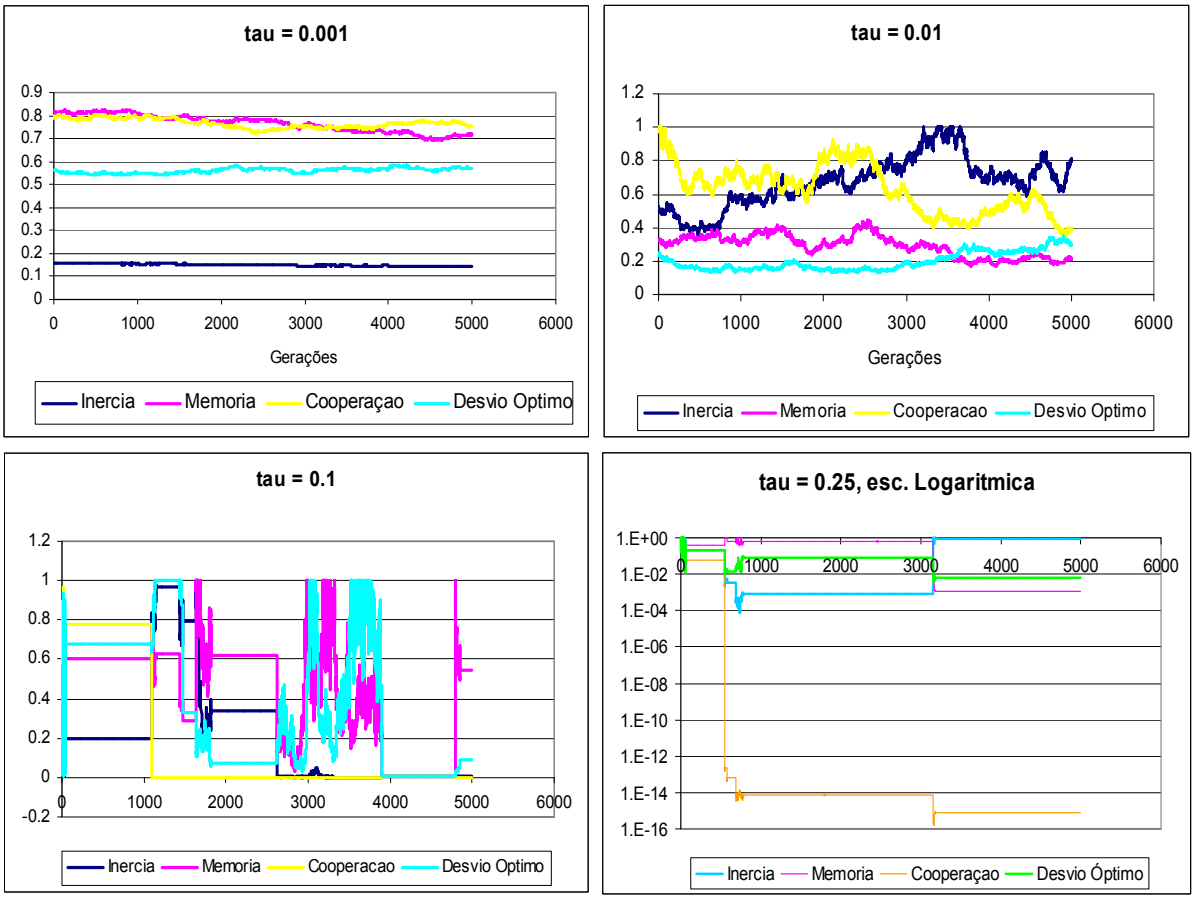


Figura 4.2.5 – Gráficos da evolução dos pesos associados à melhor partícula em cada geração, obtidos a partir de uma convergência típica da função Rosenbrock, para valores de parâmetro de aprendizagem tau = 0.001, 0.01, 0.1 e 0.25.

Na Figura 4.2.5 estão representados os gráficos da evolução dos pesos associados à melhor partícula em cada geração, obtidos a partir de uma corrida típica. Optou-se por apenas colocar neste relatório os gráficos relativos a um τ igual a 0.001, 0.01, 0.1 e 0.25, mas serão descritos resultados de outros valores para este parâmetro de mutação.

Quando τ = 100, temos um algoritmo mais robusto (desvio padrão = 0.022361966), mas, quando se observa a evolução dos pesos, verifica-se que este valor é muito alto, tornando os pesos nulos ao fim de poucas iterações. Esta situação acontece quando τ toma valores a partir de 0.2 (quanto maior é τ, mais depressa e mais perto de zero ficam os pesos de mutação). Com τ = 0.001, os pesos não se conseguem auto-adaptar para um valor óptimo, pois o valor de mutação praticamente não varia. Quando aumentamos τ para 0.01 os pesos já variam, mas não o suficiente (nem da forma esperada) para encontrar da melhor forma o óptimo global.

Por forma esperada de convergência dos pesos, entende-se por exemplo, uma progressiva redução do peso que afecta a perturbação na localização do óptimo global corrente b_G, denotando um ajuste mais fino do algoritmo à medida que a busca se processa numa zona mais limitada do espaço. Outro parâmetro que também se vai reduzindo ao longo das iterações é o de cooperação, pois à medida que as partículas se vão juntando todas na vizinhança da melhor solução, na equação de movimento a posição do melhor ponto já não terá tanto peso.

Podemos ver que, com $\tau = 0.05$, $\tau = 0.1$ e $\tau = 0.15$, a evolução dos parâmetros estratégicos é mais favorável para a procura do óptimo, o que nos leva a concluir que o valor óptimo para o parâmetro de aprendizagem possa estar no intervalo [0.05 ; 0.15].

Outro aspecto a salientar da aplicação desta mutação, é o modo como o algoritmo se comporta ao longo do tempo, na procura pelo óptimo. Ao observar a Figura 4.2.6, constata-se que as partículas se dirigem quase que "imediatamente" para a zona do óptimo global, e, a partir daí, a evolução da procura é praticamente nula.

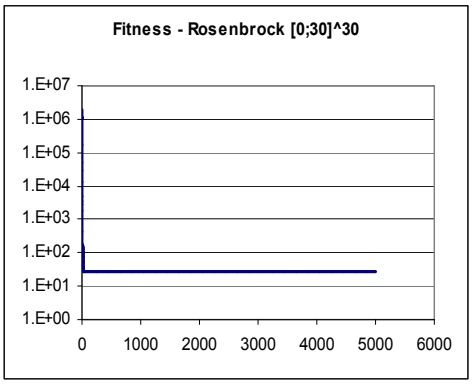


Figura 4.2.6 – Convergência típica da função Rosenbrock, quando a mutação é do tipo Lognormal, com $\tau = 0.1$.

		Restrição Comunicação	
		Não	Sim
Rosenbrock	Média	28,7241	26,114459
	Desv Padrão	0,10435	0,958282
Alpine / Parábola	Média	0	
	Desv Padrão	0	
Griewank	Média	45,4819	3,74709
	Desv Padrão	8,729	2,88382

Tabela 4.2.7 – Resultados de 20 simulações obtidos por aplicação de uma mutação Lognormal, com e sem restrição na comunicação. O valor de τ é igual a 0.1.

Pela Tabela 4.2.7, pode-se verificar que para as funções Alpine e Parábola, quando se aplica uma mutação Lognormal, o EPSO consegue descobrir o óptimo global, sem que haja restrição na comunicação. No caso da função Rosenbrock, existe uma melhoria significativa (tendo em conta que esta é uma função muito difícil de optimizar) quando existe essa restrição, pois o algoritmo consegue atingir um valor próximo de 26. Já para a função

Griewank, mais uma vez esta alteração em relação ao EPSO original resulta em piores resultados.

4.2.4 Análise da Performance do EPSO para a Função Rosenbrock

Este estudo teve como objectivo tentar descobrir porque é que o algoritmo EPSO, com uma mutação Lognormal e sem restrição na comunicação entre as partículas, não consegue evoluir na procura pelo óptimo, ficando "atracado" na zona do valor 28, logo a partir das primeiras gerações (Figura 4.2.6).

Analisou-se então uma simulação típica, cujo valor óptimo encontrado pelo EPSO é igual a 28.73861, e o valor de τ utilizado é 0.1.

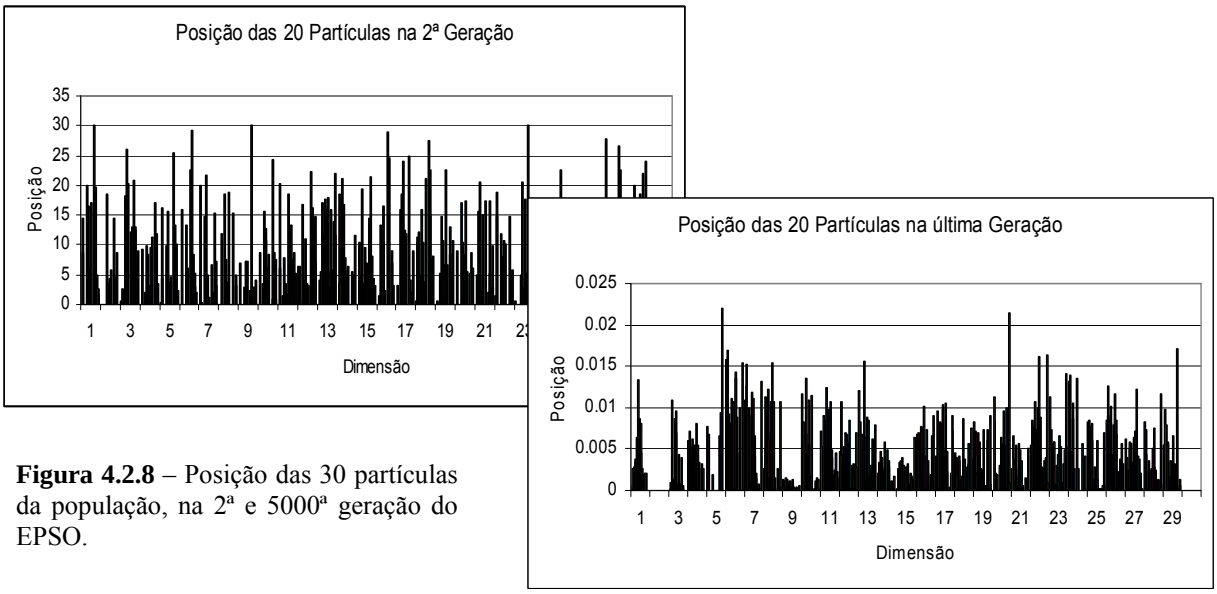


Figura 4.2.8 – Posição das 30 partículas da população, na 2ª e 5000ª geração do EPSO.

Como se pode verificar pelos dois gráficos da Figura 4.2.8, no final das 5000 gerações (o que equivale a 200000 avaliações à f.o.), todas as partículas do Enxame encontram-se muito próximas do ponto x = [0,...,0] (notar a diferença na escala de *yy* entre os dois gráficos). Com este algoritmo, a procura converge quase imediatamente para a zona do óptimo, pois já na 3ª geração o valor de *f(x)* para a melhor partícula é de 40.56 (ver tabela abaixo).

Geração	Melhoria Óptimo > 0.01
1	838425.2388
2	102540.5306
3	40.56497946
4	29
10	28.84079082
21	28.80963962
22	28.75632345
951	28.73861185
5000	*28.73861185*

Tabela 4.2.9 – Evolução da procura pelo óptimo, para uma simulação do EPSO cujo óptimo encontrado é 28.73 (f. Rosenbrock).

Nesta tabela, encontram-se as gerações em que houve uma melhoria na procura do óptimo, onde o valor *threshold* é de 0.01. Como se pode verificar, a partir da 4ª geração o algoritmo não consegue convergir para o óptimo 0.0, estagnando no valor 28.7386 encontrado na 951ª geração.

Na figura 4.2.10, está representada a distância entre as partículas e o melhor ponto encontrado, para a 2ª, 3ª, 10ª e 21ª gerações. Como se pode verificar, a partir da 21ª geração, o enxame já se encontra todo na zona do "óptimo" actual (a distância para a melhor partícula é praticamente nula).

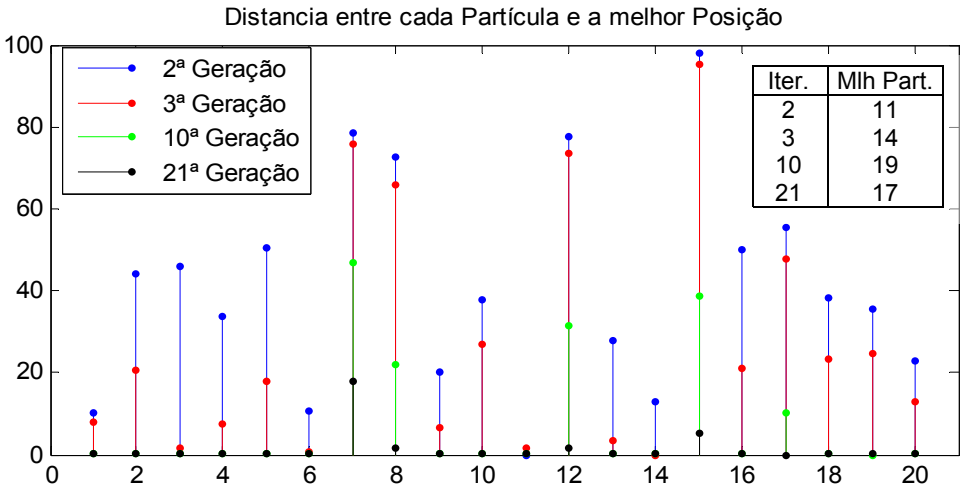


Figura 4.2.10 – Distância entre cada partícula e a melhor posição, para a 2ª, 3ª, 10ª e 21ª geração.

Entre a 4ª e a 22ª geração, aparece com muita frequência o valor de *Fitness* para as partículas igual a 29.0. Como na geração anterior (3ª), a partícula óptima com valor *f(x)* =

40.56 tem todas as posições nulas, excepto a 13ª que ocupa a posição 0.330508, o algoritmo encara a região do óptimo como a vizinhança da posição [0,…,0].

Nota: Quando $x_d = [0,\ldots,0]$, $f(x) = \sum_{d=1}^{D-1} 1$, e neste caso, como D = 30, *f(x)* = 29.0

Outra razão para que isso aconteça, é o facto de que uma partícula com posições de valor um pouco mais acima de 1, tem um valor *f(x)* muito elevado, levando a que o algoritmo não considere que a região vizinha da posição [1,…,1] como óptima.

Na Figura 4.2.11, estão alguns exemplos de partículas que na 3ª geração têm algumas posições próximas do valor 1 (e as restantes nulas), mas com valores *f(x)* elevados. Por exemplo, a terceira partícula, tem a 7ª e a 22ª dimensão com valores próximos de 1, e as restantes nulas, e o seu valor de *f(x)* é 447.99.

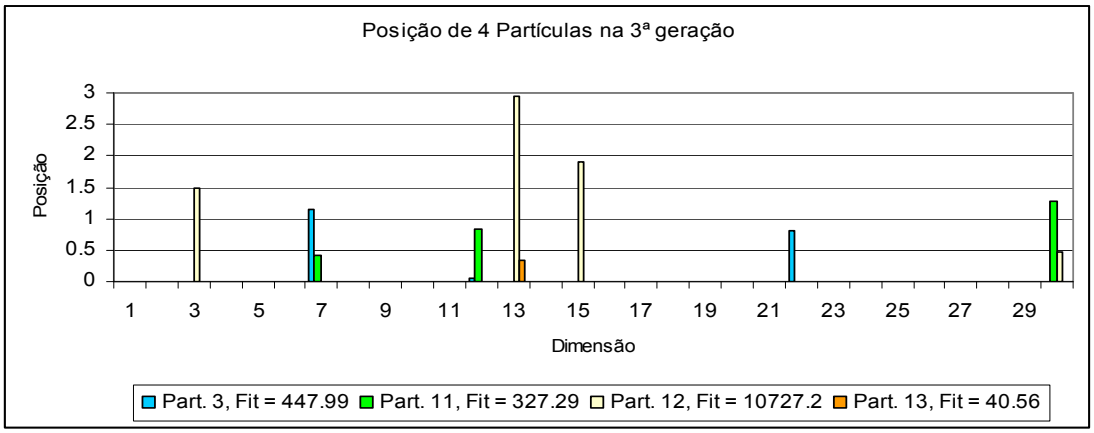


Figura 4.2.11 – Posição de 4 partículas pertencentes à população, na 3ª geração do EPSO.

Note-se que a componente de maior peso na função Rosenbrock é $100(x_d^2 - x_{d+1})^2$, que envolve a diferença entre o quadrado de uma coordenada e a coordenada consecutiva. Para que a função atinja o seu mínimo global, todas as coordenadas têm de ter valor 1. Basta que apenas algumas coordenadas assumam o valor 0, para o valor da f.o. aumentar drasticamente.

Comparando com outras simulações, pode-se dizer que existe progresso na procura do óptimo, apenas nas primeiras gerações. Na tabela abaixo, encontram-se resultados de mais 4 simulações.

Gerações em que houve Melhoria	Óptimo
2, 3, 4, 12, 13	28.78972
2, 3	28.9901
2, 3, 4, 10, 11, 12	28.77252
2, 3, 4, 7, 31	28.75733

Tabela 4.2.12 – Resultados da evolução da procura pelo óptimo, para 4 simulações do EPSO (f. Rosenbrock).

O peso de Cooperação das partículas (ver gráfico 3 da Figura 4.2.5), torna-se nulo mais ou menos ao fim de 1000 iterações, mas, até lá, já as partículas se encontram todas na mesma zona. Experimentou-se restringir a comunicação do óptimo global, onde cada dimensão de cada partícula tem probabilidade 0.2 de obter informação da melhor posição, para que seja possível uma melhor pesquisa do espaço $[0; 30]^{30}$.

De seguida apresentam-se os resultados de uma simulação com restrição na comunicação, onde o óptimo encontrado é igual a 26.94507.

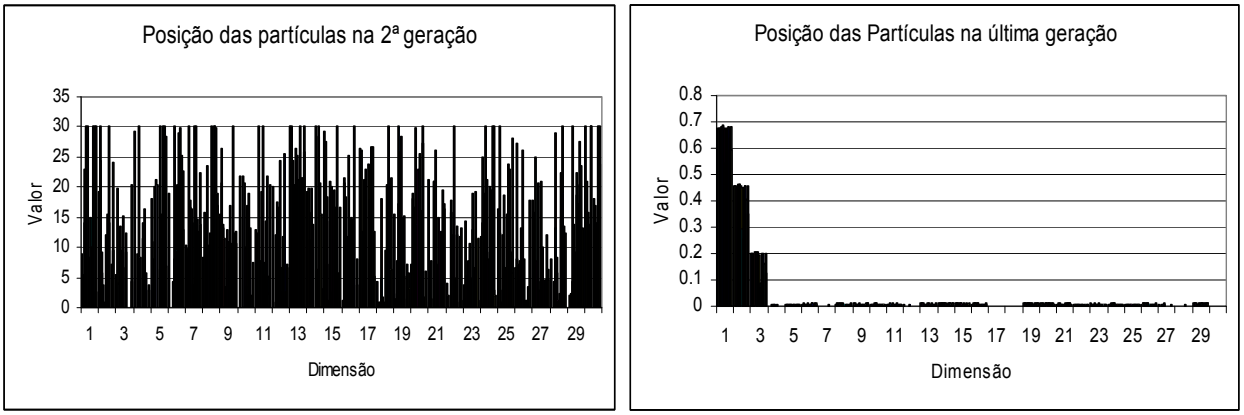


Figura 4.2.13 – Posição das 20 partículas da população, na 2ª e 5000ª geração do EPSO, com uma mutação Lognormal, $\tau = 0.1$ e restrição na comunicação.

Observando o gráfico da última posição das partículas, verifica-se que a 2ª dimensão das mesmas, com valor ≈ 0.45, é um valor aproximado do quadrado da 1ª dimensão (≈ 0.67), e assim sucessivamente. Pela equação da função, para o segundo termo $(x_d^2 - x_{d+1})^2$ se anular, $x_{d+1} = x_d^2$, para todos os d. Idealmente, para o termo se anular, basta que $x_1 = 0.67$, $x_2 = (0.67)^2$, $x_3 = (0.67)^4$, ..., $x_{30} = (0.67)^{60}$. Note-se que isto também acontece quando $x_d = 0$ e $x_d = 1$, $d = 1,...,30$.

Apenas no caso em que $x_d = 1$, $d = 1,...,30$, é que o primeiro termo da função, $(1-x_d)^2$, também se anula. Por isso, quanto mais próximo de 1 for o valor da primeira dimensão da melhor partícula, melhor será o resultado para o óptimo, pois ambos os termos da função serão mais próximos de zero.

Concluindo, podemos constatar que o algoritmo se auto-adapta de forma a conseguir anular o segundo termo da função, o que é muito bom! Note-se que foi necessário implementar uma restrição na comunicação, para que o algoritmo pudesse "estudar" melhor o espaço, de forma a encontrar a sua melhor solução.

Infelizmente, o algoritmo não consegue perceber que a posição óptima é $x_d = 1$, $d = 1,...,30$. Isto porque, seguindo o raciocínio anterior, se a primeira dimensão tem valor 1.00001, *f(x)* toma um valor astronómico! Se $x_1 = 1.0000001$, o resultado final será 2.07017E+23.

Resta dizer que, agora a evolução dos pesos da melhor partícula em cada geração, tem um comportamento mais oscilatório, e a pesquisa do espaço é um pouco mais exaustiva, pois, ao longo do tempo existe sempre uma melhoria do melhor ponto (ao contrário do caso em que não existiu restrição, onde o melhor ponto foi encontrado na 951ª geração).

4.2.5 EPSO Versão A

Os resultados obtidos com o estudo anterior, inspiram uma nova redefinição adaptativa para o espaço, em princípio deduzida a partir da exploração que cada partícula faz da sua própria vizinhança. Surge então esta versão A do EPSO. Aqui, existe elitismo limitado, isto é, a melhor partícula não deixa a melhor posição encontrada até que outra posição ainda melhor seja encontrada. Quer dizer: existe sempre uma partícula na melhor posição, e que vai gerando filhos; mas se nenhum dos filhos for melhor que ela, morrem e volta-se de novo à partícula original. Descreve-se de seguida, para melhor compreensão, o pseudo-código desta nova versão:

Procedimento EPSO versão A:
{
Gerar população de **n** partículas;
Definir *Melhor Partícula*;
Criar Enxame com **d** descendentes da *Melhor Partícula*;

Replicar **r** vezes a população;
Enquanto (nº avaliações ≤ MaxAvaliações) **{**

Mover partículas originais (incluindo *Melhor Partícula*)
Avaliar Fitness partículas originais

Se *Melhor Partícula* movida tem pior Fitness, voltar à posição inicial de *Melhor Partícula*; senão, actualizar *Melhor Partícula* e seus descendentes;

Mutar genes das partículas replicadas
Mover partículas replicadas
Avaliar Fitness partículas replicadas

Mutar genes dos descendentes da Melhor Partícula
Mover descendentes da Melhor Partícula
Avaliar Fitness dos descendentes da Melhor Partícula

Selecção da Melhor Partícula ou seus descendentes.

Selecção por elitismo, das melhores partículas, de entre a originais e as replicadas;

Actualização da *Melhor Partícula*: Verificar se de entre as partículas escolhidas para formar a população da próxima geração, existe uma que é melhor do que a *Melhor Partícula*; Actualização dos descendentes.
}
}

Observando o pseudo-código, verifica-se que existe a opção de a *Melhor Partícula* gerar, além dos descendentes obtidos quando se replica **r** vezes a população, mais **d** descendentes. Além disso, estes descendentes têm uma taxa de mutação mais pequena (τ = 0.005), permitindo assim uma "micro-exploração" da vizinhança da *Melhor Partícula*, isto é, uma exploração mais exaustiva e minuciosa.

Outra diferença é a selecção por elitismo. Aqui as partículas escolhidas para passar para a geração seguinte, são as melhores entre si. Como temos mais o factor **d**, que influencia o número total de partículas utilizadas no final da corrida, o número de gerações é agora dados por:

$$Iterações = \frac{Nº\ Avaliações}{Nº\ Partículas \times (r+1) + Nº\ Descendentes\_d}$$

Na seguinte tabela estão apresentados os resultados desta nova variante do EPSO. Foram efectuadas 10 simulações do algoritmo, com uma mutação Lognormal de parâmetro τ =0.01 e τ para os descendentes da *Melhor Partícula* igual a 0.005.

		r = 1		
Sem Restr. Comunicação		**d = 4**	**d = 8**	**d = 12**
Rosenbrock	Média	26,226696	26,470885	25,93391098
	Desv. Padrão	0,5763617	0,9107334	0,54200101
Griewank	Média	11,72288	11,262248	11,46042221
	Desv. Padrão	4,5767038	4,3229721	3,954158583
		r = 1		
Com Restr. Comunicação		**d = 4**	**d = 8**	**d = 12**
Rosenbrock	Média	17,529253	22,091427	20,7376767
	Desv. Padrão	9,0554841	1,5129403	0,995170352
Griewank	Média	2,569337	2,6903511	3,000049859
	Desv. Padrão	2,6678276	1,7767441	3,053072308

Figura 4.2.14 – Resultados de 10 simulações, para as funções Rosenbrock e Griewank, onde o número de réplicas é igual a 1, e o número de descendentes da Melhor Partículas variam entre 4, 8 e 12.

Os resultados que mais se destacam são os da função Rosenbrock, quando é aplicada uma restricão na comunicação e o número de descendentes da *Melhor Partícula* é igual a 4. Também no caso da Griewank, os melhores resultados são obtidos nestas condições (não esquecer claro que neste caso, o EPSO original tem melhor performance).

Pode-se mesmo afirmar que o valor 17.52 (com desvio padrão de 9.05) obtido pelo EPSO para o ponto óptimo da Rosenbrock é um valor excelente, o que comprova a melhor eficácia desta nova variante para o EPSO.

Concluindo, foram aplicadas com sucesso algumas modificações ao algoritmo original do EPSO, tais como uma mutação Lognormal e uma restrição na comunicação entre as partículas. Verificou-se também que dependendo do problema, umas variantes funcionam melhor que outras, o que é natural, pois é necessário adaptar o algoritmo às características muito próprias do problema em questão.

Foi melhorada a performance para a função Rosenbrock, um problema muito difícil de optimizar, alcançando-se um valor de 17.05 (valor médio de 10 simulações), quando a

dimensão é igual a 30 e o espaço de pesquisa é [0; 30]. Foi também estudado o comportamento do EPSO nesta função, verificando-se a forma "engenhosa" do algoritmo descobrir uma solução.

Excepto para a função Griewank, um algoritmo EPSO com restrição na comunicação proporciona melhores soluções, quer para o problema da função Alpine, com inúmeros óptimos locais, quer para o problema da função Parábola (apenas uma solução global), onde em ambas o óptimo 0.0 foi sempre encontrado. Também para a função Rosenbrock esta variante traz melhorias muito significativas, pois o algoritmo torna-se muito mais robusto que a versão original, com o desvio padrão a descer consideravelmente do valor 39.32 (Tabela 3.2.1) para o valor 1.41 (Tabela 4.2.3).

5 Previsão de Potência de um Parque Eólico

Cada vez mais em Portugal, existe a preocupação em utilizar de maneira racional as principais matérias-primas energéticas, assim como reduzir a elevada dependência energética de recursos não renováveis exteriores. Aliada a estes factos, existe ainda uma crescente consciencialização e preocupação com o meio ambiente e em reduzir as emissões de Gases com Efeito Estufa (GEE), de forma a preservar o planeta. Por isso, assumir os compromissos da Directiva Europeia de Produção de Electricidade de Origem Renovável (que obriga a que 39% dos consumos eléctricos sejam, em 2010, de origem renovável) e do Protocolo de Quioto (entre 2008 e 2012 as emissões de GEE não podem ultrapassar +27% relativamente a 1990) é de extrema importância nos dias que correm.

A energia eólica é uma eficiente fonte de produção de electricidade tendo ainda como vantagem os factos de estar livre de perigos, de ser limpa e de ser abundante. Apesar disso, e da impressionante taxa de crescimento, tirar todo o partido da produção desta energia é ainda muito complicado. Como se sabe, os parques eólicos são estruturas geograficamente distribuídas, sujeitas por vezes ao condicionamento de uma orografia complicada, a intensidade do vento pode variar de gerador para gerador, e o seu rendimento depende do respectivo alinhamento em relação ao vento – para além de que a disposição dos geradores no terreno pode fazer aparecer efeitos de "esteira" que diminuam o aproveitamento energético de geradores colocados atrás de outros (o que varia com a direcção do vento) [15]. Estas dificuldades, acrescidas do facto de as séries de vento serem altamente imprevisíveis, tornam a previsão de produção desta energia num processo muito complexo e não linear. Num futuro muito próximo, com a abertura de mercados de electricidade, é fundamental que haja previsões de curto e médio prazo até 48 horas, a fim de permitir um adequado pré-despacho de centrais convencionais (em particular, as térmicas) e também para permitir que um mercado de energia eléctrica funcione de forma ajustada e sem distorsões no preço da energia, que poderiam resultar de uma má previsão da contribuição eólica disponível (com consequente sobre- ou subavaliação da potência transaccionada e consequente impacto no preço de fecho do mercado). [22].

Nesta secção será descrita a segunda parte do trabalho proposto, que consiste na aplicação dos conceitos de ITL a um problema prático de previsão da produção de energia de um parque eólico, utilizando dados de velocidade e direcção de vento. A previsão será processada através de um Sistema de Inferência Difusa do tipo Takagi-Sugeno (SID-TS) de ordem 0, cujo algoritmo de treino utilizado é o EPSO, ao invés do método tradicional de Retro-Propagação. O sistema terá como funções de custo, a minimização da entropia da distribuição dos erros (diferença entre o output desejado e o do sistema), e o critério habitual do Erro Mínimo Quadrado (EMQ).

Importa realçar que o objectivo deste estudo não é apresentar um modelo de previsão de energia eólica com melhor performance que os já existentes, mas mostrar que a minimização da Entropia, tal como é descrita na abordagem de ITL, é um critério mais robusto e com mais potencialidades para desenvolver um sistema de previsão mais exacto do que o critério de minimização da variância [15, 16].

5.1 Modelo de Previsão

Os dados (reais) são compostos por três séries temporais: velocidade de vento (metros/segundo), direcção de vento (graus) e potência de saída de um parque eólico situado na zona Norte de Portugal. Foram recolhidos de hora a hora, desde o dia 1 de Janeiro de 2004 até 20 de Fevereiro de 2005, através do sistema SCADA (Supervised Control and Data Acquisition System). Por razões de confidencialidade, os valores de potência realmente produzida no parquet foram transformados numa percentagem relativa à máxima capacidade do parque, o qual tem um número importante de geradores de potência nominal próxima de 1 MW, distribuídos sobre linhas de cumeada de montanhas e totalizando uma caapacidade instalada de cerca de 40 MW.[15].

Dos 10000 dados recolhidos, foram apenas utilizados os últimos 5000, divididos depois num conjunto de treino (1000 dados) e num conjunto de teste (restantes 4000 dados). Como a função custo de minimização da entropia tem uma complexidade computacional quadrática ($O(N^2)$), não é possível utilizar um conjunto de treino muito extenso (com 1000 dados, são necessárias 1000×1000 operações para apenas uma avaliação à função objectivo, tornando o treino do SID-TS num processo muito lento). Daí o facto de se ter adoptado um

conjunto de treino muito menor que o conjunto de teste.

Note-se que, mesmo tendo em conta a propriedade de simetria da função Gaussiana, que permite calcular apenas as distâncias entre os erros i e j, para $i \neq j$, o algoritmo tem uma complexidade também quadrática, embora menor: $(N \times (N - 1))/2$.

A razão pela qual foram utilizados apenas os últimos 5000 dados tem a ver com o facto de os primeiros registos apresentarem algumas irregularidades. Nomeadamente, existem pontos com velocidade alta e potência nula que podem ser explicados devido a desconexões do parque [15]; e também pontos que parecem descrever uma outra curva de potência, levando a diferentes níveis de saturação do parque (capacidade máxima instalada a 90%, 85% e 70% aproximadamente). Isto aconteceu porque, em meados de 2004, foram instalados mais aerogeradores no parque, o que levou ao aumento da capacidade total de produção eólica e consequente aumento do nível de saturação. Estes dados podem ser observados na Figura 5.1.1 (a), onde está representado o diagrama Velocidade versus Potência, mais conhecido como curva de potência do parque eólico. Por sua vez, na Figura 5.1.1 (b) estão representados os dados a utilizar na previsão.

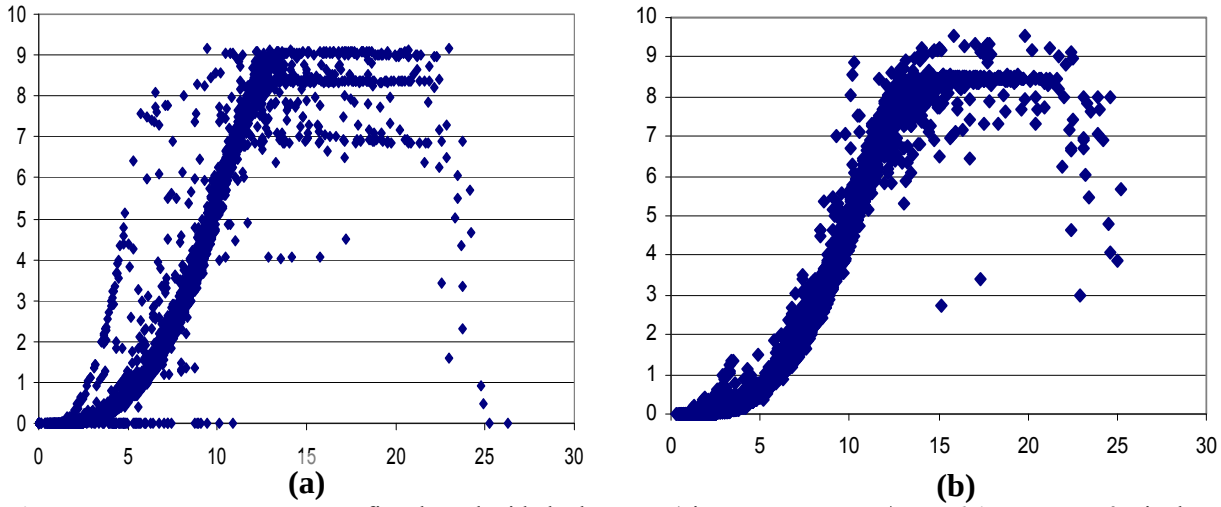
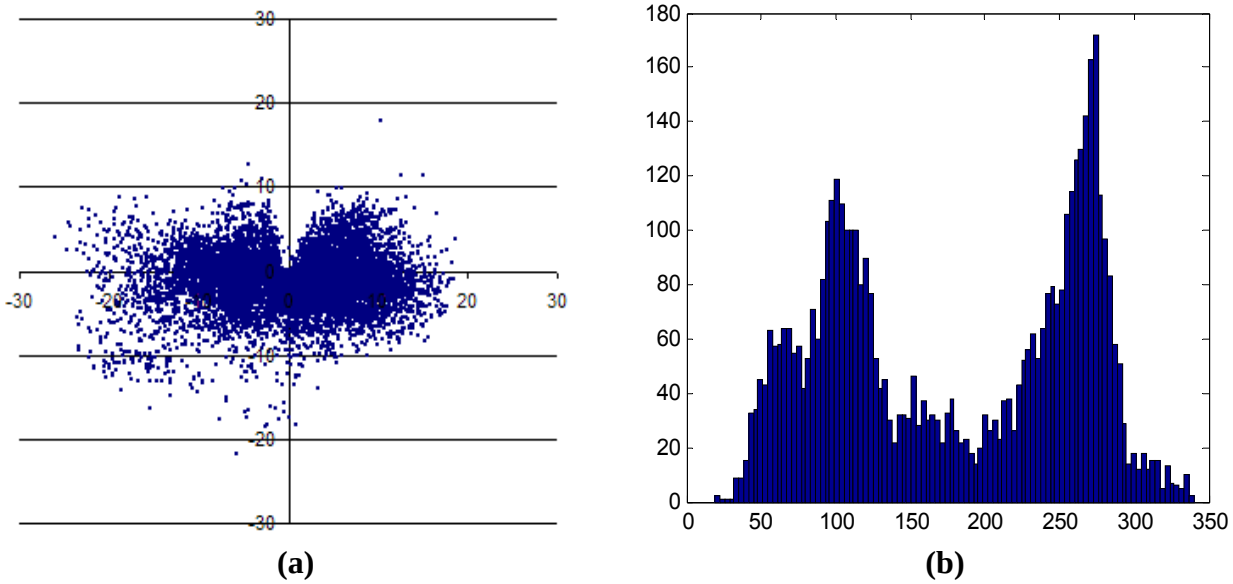


Figura 5.1.1 – Representação gráfica da Velocidade do vento (eixo xx, em metros/segundo) versus Potência do Parque Eólico (eixo yy). A potência produzida no parque é dada em p.u. (por unidade) relativa à capacidade instalada do parque.. **(a)** Primeiros 5000 dados não tratados, **(b)** Últimos 5000 dados a utilizar na previsão.



(a) **(b)**

Figura 5.1.2 – **(a)** Distribuição da velocidade (m/s) e direcção do vento, medidas num ponto próximo do parque eólico. **(b)** Histograma dos 5000 dados de direcção do vento.

O sistema SID-TS de ordem 0 utilizado para testar os critérios de ITL e EMQ, possui as seguintes características:

a) Duas variáveis de input: velocidade do vento V, em metros/segundo, e direcção do vento D, em graus;
b) O intervalo de V entre 0 e 30, e o de D entre 0 e 360;
c) O intervalo do output Potência entre 0 e 1;
d) O universo de discurso[1] de V foi dividido em 5 conjuntos difusos com funções de pertença Gaussianas;
e) O universo de discurso de D foi dividido em 2 conjuntos difusos com funções de pertença Gaussianas;

O objectivo será descobrir apenas os pesos *w* adequados para o consequente das regras (base de conhecimento), mas outros parâmetros poderiam ser ajustados durante o treino do sistema. Como se optou por utilizar funções de pertença Gaussianas, os centros e amplitudes destas têm de ser também optimizados. No entanto, isto não é conveniente em muitos casos,

[1] Tal como os conjuntos ordinários, os conjuntos difusos também são definidos sobre um domínio (Universo de Discurso), mas diferem daqueles porque não possuem uma fronteira claramente definida. As Funções de Pertença representam numericamente o grau de certeza com que um elemento pertence a um conjunto.

porque os inputs V e D estão associados com expressões linguísticas, e a mudança da forma e localização das funções de pertença pode mudar o significado que estas supostamente representam [15].

Um aspecto muito importante a ter em conta é a escolha do número de partições de cada variável, pois muitas partições resultam em grande quantidade de parâmetros a optimizar, enquanto que um número pequeno conduz a sistemas que não conseguem representar a relação presente no conjunto de dados.

Escolheram-se apenas 2 conjuntos difusos para definir o universo de discurso da direcção, pois como se pode ver pela Figura 5.1.2, os dados estão "concentrados" em dois conjuntos de valores distintos, separados aproximadamente por 180 graus. Os centros das funções de pertença destes conjuntos ficam então definidos com os valores 100 e 280 (ver histograma).

Como a base de conhecimento é estabelecida de forma a "cobrir" todo o espaço de input, utilizando todas as combinações possíveis entre os conjuntos difusos de V e D, optou-se por escolher 5 conjuntos difusos para a velocidade. Assim, o número de parâmetros a optimizar é adequado (5×2 = 10 pesos). Desta vez, os valores para os centros e amplitudes das funções de pertença da velocidade, foram determinados através do sistema ANFIS – *Adaptive Neuro-Fuzzy Inference System*, disponível na toolbox de Lógica Difusa do MATLAB.

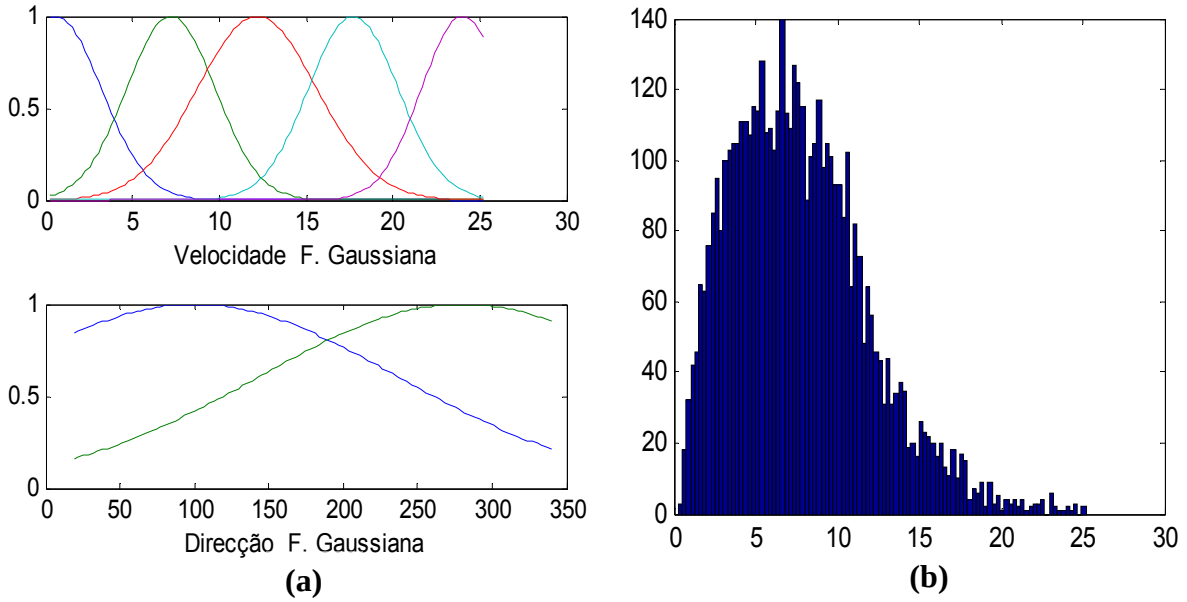


Figura 5.1.3 – **(a)** Gráficos das Funções de Pertença Gaussianas, para os Inputs de Velocidade e Direcção, com 5 e 2 MF's respectivamente. **(b)** Histograma dos 5000 dados de velocidade de vento. Como se pode verificar, a

função de pertença da Velocidade centrada em 12.5 (aprox.) tem uma amplitude muito grande, pois pelo histograma, o número de dados com valores entre 10 m/s e 15 m/s varia muito.

O algoritmo de treino EPSO utilizado tem como parâmetros exteriores fixos: 20 partículas, replicadas uma vez, e parâmetro de aprendizagem $\tau = 0.5$ para a mutação Gaussiana. Efectua selecção por elitismo, restrição na comunicação com factor probabilístico de 0.2 e o critério de paragem atribuído é um número máximo de 200 gerações, para ambas as funções objectivo de minimização da entropia e minimização do EMQ.

Importa dizer que estes parâmetros foram escolhidos após várias simulações, onde se verificou por exemplo, que com uma mutação Lognormal o algoritmo tinha pior performance. Outra razão para o facto de se ter utilizado uma versão mais simples do que a versão A, por exemplo, é o tempo computacional de uma avaliação à função objectivo. Tentou-se por isso, estabelecer um EPSO que utilize um número de descendentes baixo.

5.2 Função Objectivo

5.2.1 Critério Erro Médio Quadrático (EMQ)

Nos sistemas de treino adaptativos, a equação de erro mais utilizada é a raiz do EMQ (média da diferença quadrática entre o output do sistema SID-TS e o output dos dados de treino), definida por:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y_i - T_i)^2}{N}}$$

onde N é o número de dados utilizados, y_i o output obtido pelo sistema difuso e T_i o output desejado.

5.2.2 Critério Entropia

Um resultado importante a mencionar quando se minimiza a entropia, é que esta é independente da média do erro do sistema. Devido a esta propriedade, o algoritmo irá convergir, com probabilidade de 1, para um conjunto de pesos óptimos que originam uma

distribuição dos erros que não tem média zero [14]. No entanto, como já foi dito atrás, isto pode ser corrigido acrescentando-se no final do treino um *bias* adequado. Outro facto a realçar é o de que a entropia é uma função custo com muitos mínimos locais [14]; portanto, poderá pensar-se que um algoritmo de convergência estocástica como o EPSO terá mais facilidade em encontrar os melhores pesos do que um algoritmo baseado no gradiente, como o de Retro-Propagação.

Pela abordagem ITL, a entropia dos erros é estimada da seguinte forma:

$$V(\{e\}) = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=i}^{N} G(e_i - e_j, 2\sigma^2)$$

$$= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \frac{1}{\sqrt{2\pi}\sqrt{2\sigma^2}} \exp\left(\frac{-(e_i - e_j)^2}{2(2\sigma^2)}\right)$$

$$= \frac{1}{2\sigma\sqrt{\pi}N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \exp\left(-\frac{1}{4\sigma^2}(e_i - e_j)^2\right)$$

$$entropia = -\log_2 V(\{e\})$$

Na Teoria da Informação é utilizado o logaritmo de base 2 para o cálculo da entropia (medição da quantidade de informação em bits), pelo que aqui não foi excepção.

Quanto à questão da largura da janela σ do método de Parzen, esta deve de ser criteriosamente escolhida, pois a adaptação depende da interação entre as amostras (distância) e da janela da função kernel. O estudo efectuado levou à conclusão de que um valor de σ = 0.01 é suficiente para definir a forma da fdp dos erros, de modo a conduzir a um conjunto de pesos que realmente possuem menor entropia. É fácil de perceber que uma fdp estimada com um σ muito maior, terá uma forma mais suave e portanto, os pesos encontrados podem conduzir a um sistema com pior performance que o EMQ, pelo simples facto de a fdp não corresponder à forma correcta.

Investigações demonstraram que adaptar a janela durante o treino pode conduzir a melhores resultados [17]. No entanto, mais uma vez há que ter em conta que uma mudança de janela significa uma mudança de função objectivo, pois a forma e o óptimo global da superfície serão alterados. O impacto de uma função objectivo dinâmica pode ser desastroso quando o algoritmo de procura assume que a superfície possui uma forma estática,

conduzindo rapidamente a divergência e instabilidade [17]. No caso do EPSO, como as partículas são constantemente reavaliadas na função objectivo, este vai-se adaptando às diversas formas que a fdp pode tomar. Os resultados obtivos indicaram que o algoritmo funcionou melhor com uma janela fixa adequada, embora as diferenças não sejam muito significativas, relativamente a uma variação iterativa do σ de 1 até 0.01. Porém, como não foi feita uma investigação sistemática sobre o procedimento de redução progressiva da dimensão das janelas de Parzen, devem interpretar-se estes resultados com a prudência adequada.

5.3 Resultados

Nesta secção serão mostrados os resultados obtidos pelo sistema SID-TS treinado com o EPSO, com o objectivo de comparar o critério de ITL com o de EMQ. Para a minimização da entropia dos erros, optou-se apenas por apresentar os resultados relativos a uma janela de Parzen fixa igual a 0.01. As curvas de potência dos dados de treino e teste utilizados estão representadas na figura seguinte.

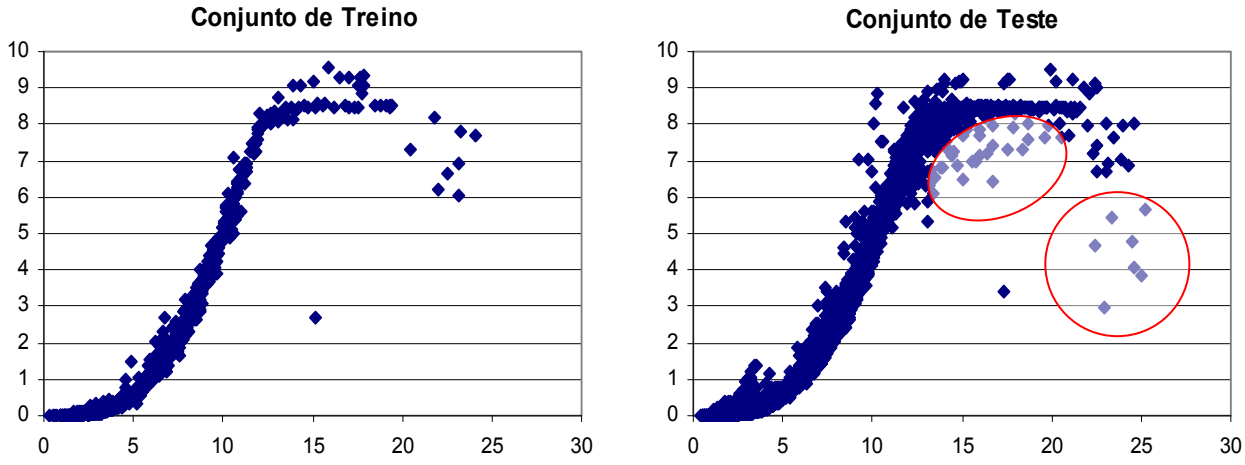


Figura 5.3.1 – Representação gráfica da Velocidade do vento (eixo xx, em m/s) versus Potência do Parque Eólico (eixo yy), para os conjuntos de Treino e Teste.

Como se pode ver, o conjunto de treino não está a representar todos os valores de (velocidade, potência) que podem ocorrer no parque eólico em questão (assinalados a vermelho). Consequentemente, o erro no conjunto de teste será maior do que no conjunto de treino, não porque ocorreu *overfitting*, mas porque os dados de treino não são suficientes para

a construção de um sistema de previsão adequado. Como o objectivo deste estudo é apenas mostrar que com o critério ITL se obtêm distribuições de erro mais estreitas (próximas de uma função Delta Dirac, significando uma maior quantidade de erros com valor nulo), do que com o critério EMQ, optou-se por utilizar apenas 1000 dados (número suficiente) para o conjunto de treino. De notar que, como já foi dito atrás, esta é uma função com uma complexidade computacional quadrática, o que torna o algoritmo muito lento (de cada vez que uma partícula é avaliada, são efectuadas 1000^2 operações de soma de valores de distâncias entre erros).

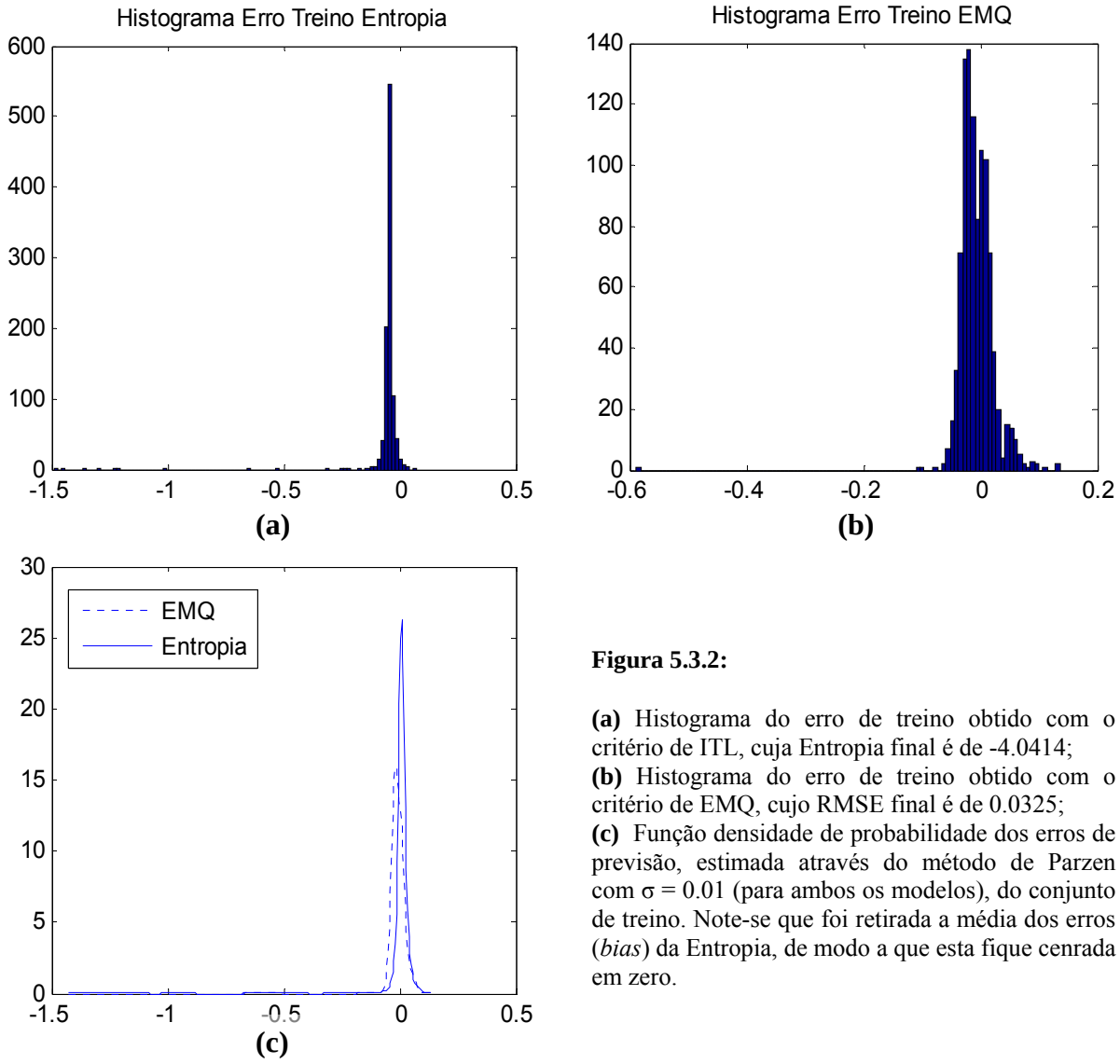


Figura 5.3.2:

(a) Histograma do erro de treino obtido com o critério de ITL, cuja Entropia final é de -4.0414;
(b) Histograma do erro de treino obtido com o critério de EMQ, cujo RMSE final é de 0.0325;
(c) Função densidade de probabilidade dos erros de previsão, estimada através do método de Parzen com $\sigma = 0.01$ (para ambos os modelos), do conjunto de treino. Note-se que foi retirada a média dos erros (*bias*) da Entropia, de modo a que esta fique cenrada em zero.

A Figura 5.3.2 mostra claramente que a distribuição dos erros de previsão obtidos quando se treina um EPSO cuja função objectivo é a minimização da entropia, tem um

número muito maior de valores próximos de zero, quando comparada com a distribuição obtida pelo critério EMQ. Este é um resultado esperado, pois com a minimização da entropia a fdp dos erros fica parecida com a fdp Delta Dirac (centrada em zero, pois foi adicionado o bias necessário).

De modo a apreciar o impacto dos resultados no domínio do tempo, tem-se a seguinte figura que representa uma sequência de valores do conjunto de teste para a previsão da potência através dos modelos de ITL e EMQ, incluindo a medição do sistema SCADA.

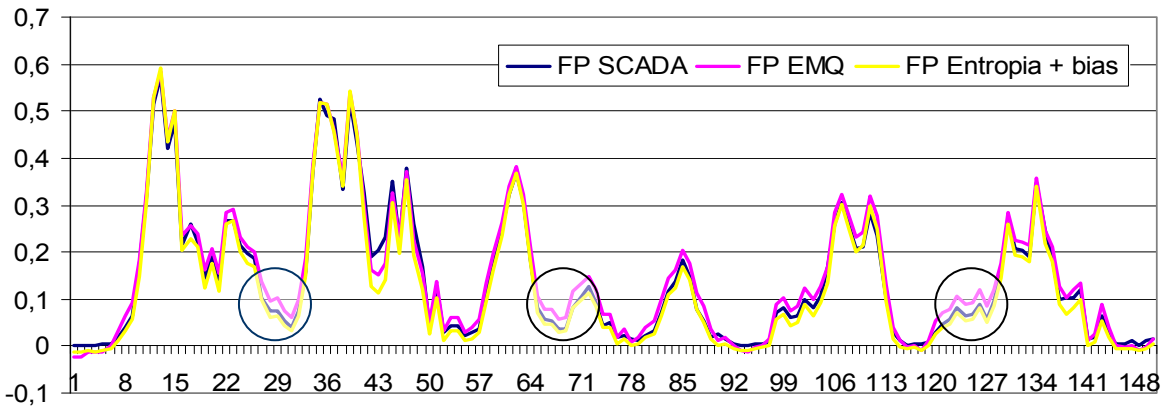


Figura 5.3.3 – Comparação, nos primeiros 150 dados de previsão do conjunto de treino, da performance dos dois modelos. Os círculos identificam zonas onde é clara a superioridade do critério de ITL sobre o do EMQ.

Claro que observando todos os 5000 pontos, encontram-se zonas onde o modelo EMQ produz um erro menor, mas como vemos pela Figura 5.3.2 (c), os erros obtidos com o modelo ITL são maioritariamente próximos do valor zero.

Pelos histogramas da Figura 5.3.3, pode-se verificar a diferença básica entre a minimização da entropia e da variância. Enquanto que com o primeiro critério a maior parte dos erros está concentrada em zero, existindo depois alguns erros com valores muito altos, o segundo prefere que estes tenham uma distribuição bem comportada dentro de um intervalo o mais pequeno possível. De facto, isto pode ser deduzido pela seguinte razão: suponhamos que é possível obter várias distribuições para os erros, com a mesma variância. Como a distribuição Gaussiana tem entropia máxima, entre fdp's com variância fixa, esta seria a distribuição menos desejável para o critério ITL. Outra distribuição indesejável seria a Uniforme. A entropia prefere distribuições com picos agudos, ou seja, um número de picos do tipo Dirac com variância semelhante. [16]

Nas figures seguintes estão representados os resultados relativos ao conjunto de teste, e como se pode verificar, a conclusão esperada é a mesma: o critério ITL tem mais potencialidades de gerar um modelo de previsão mais robusto (próximo dos valores reais) do que o critério do EMQ.

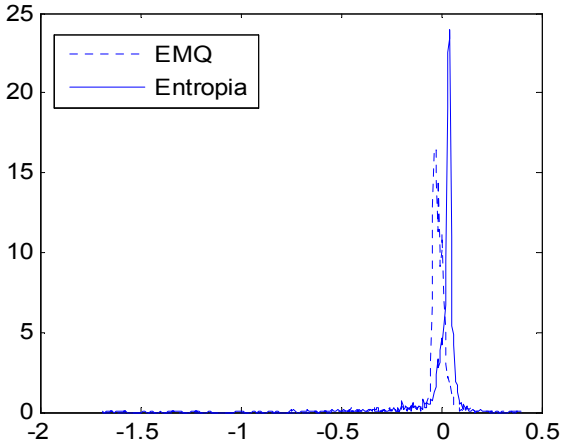


Figura 5.3.4:

(c) Função densidade de probabilidade dos erros de previsão, estimada através do método de Parzen com $\sigma = 0.001$ (para ambos os modelos), do conjunto de treino. Note-se que foi retirada a média dos erros (*bias*) da Entropia, de modo a que esta fique cenrada em zero.
A entropia final do modelo ITL é de -3.2743; o RMSE final do modelo EMQ é de 0.0785.

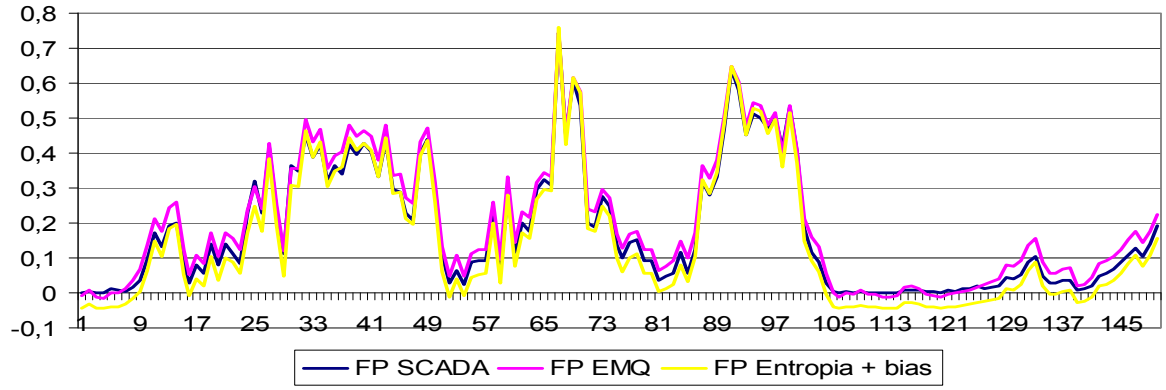


Figura 5.3.5 – Comparação, em 150 dados de previsão do conjunto de teste, da performance dos dois modelos. Como se pode verificar, existem zonas onde ambos os critérios não tiveram uma performance satisfatória. Isto é explicado pelo facto de não terem sido utilizados dados suficientes para treinar o sistema.

Estes resultados são muito importantes pois questionam a aplicação constante, por parte dos investigadores, do EMQ como medida de desempenho de sistemas adaptativos de Inferência Difusa, Redes Neuronais e outros que envolvem ajustamento de parâmetros através de treino. Na práctica, as distribuições de erro não são bem comportadas nem simétricas, e assumir uma distribuição Gaussiana não obedece à realidade. Como já foi dito, são necessários momentos de ordens superiores à média e à variância, para descrever estas

distribuições, e tal só é possível quando é utilizado um critério que tem em conta toda a informação disponível na fdp dos erros.

6 Conclusões

O trabalho desenvolvido neste estágio é dividido em duas partes fundamentais: a apresentação de novas variantes para o algoritmo EPSO, e a aplicação deste para o estudo de um novo critério de desempenho de sistemas adaptativos de Inferência Difusa do tipo Takagi-Sugeno, onde são optimizados parâmetros através de treino iterativo.

Relativamente aos novos modelos de EPSO estudados, foram analisadas alterações na comunicação entre partículas, tipo de mutação e exploração do espaço de pesquisa, que para a maior parte dos problemas analisados, se mostraram superiores ao esquema original do algoritmo. Particularmente, aplicar uma restrição da comunicação entre as partículas torna-se muito vantajoso, pois como se viu, para os problemas mais simples analisados (funções Parábola e Alpine) o EPSO conseguiu em 100% das vezes alcançar o óptimo global.

Infelizmente por falta de tempo, não foi efectuado um estudo mais aprofundado para o problema de minimização da função Griewank. Apesar de as soluções encontradas para o óptimo 0.0 pela versão original do EPSO serem muito satisfatórias (óptimo à volta do valor 1×10^{-2}), ficou por se perceber a razão pela qual as novas variantes do algoritmo tinham, só neste caso, pior desempenho.

No capítulo 5 foi apresentado um estimador não paramétrico para a entropia quadrática de Renyi, que pode ser aplicado directamente nas amostras de erro recolhidas durante o treino de sistemas adaptativos, sem que seja preciso alguma suposição acerca da função densidade probabilidade. Este critério ITL – *Information-Theoretic Learning*, é mais robusto que o critério de minimização da variância, mas por outro lado, é mais pesado computacionalmente, sendo uma grande desvantagem quando o número de amostras de erro é muito elevado. Enquanto que para o EMQ é calculado o quadrado dos erros, para a entropia é calculada a diferença (gaussiana) entre estes. No entanto, para sistemas *off-line* este esforço extra é compensador, se no final é gerado um sistema que produz melhores previsões.

Referências e Bibliografia

[1] Manual de Acolhimento do INESC Porto.

[2] V. Miranda and N. Fonseca, *EPSO – Evolutionary Particle Swarm Optimization, a New Algorithm with Applications in Power Systems*, Proceedings of the IEEE Transmission and Distribution Asia-Pacific Conference 2002, Yokohama, Japan, October 2002.

[3] V. Miranda and N. Fonseca, *EPSO – Best of Two Worlds Meta-Heuristic Applied To Power System Problems*, Proceedings of WCCI 2002 – World Congress on Computational Intelligence – CEC – Conference on Evolutionary Computing, Honolulu, Hawaii, U.S.A., May, 2002.

[4] V. Miranda and N. Fonseca, *New Evolutionary Particle Swarm Algorithm (EPSO) Applied to Voltage/VAR Control*, PSCC, Sevilla, 24-28 June 2002.

[5] H. Mori and Y. Komatsu, *A Hybrid Method of Optimal Data Mining and Artificial Neural Network for Voltage Stability Assessment*, Proceedings of IEEE St. Petersburg PowerTech Conference, Russia, June 2005.

[6] N. W. Oo and V. Miranda, *Multi-energy Retail Market Simulation with Intelligent Agents*, Proceedings of IEEE St. Petersburg PowerTech Conference, Russia, June 2005.

[7] Vladimiro Miranda, *Computação Evolucionária Fenotípica*, Versão 1.0 – Outubro 2004.

[8] J. Kennedy and R. C. Eberhart (1995), *Particle Swarm Optimization*, Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, pp.1942-1948.

[9] D. B. Fogel (1994), *An Introduction to Simulated Evolutionary Optimization*, IEEE Transactions on Neural Networks, Vol. 5, No. 1, January 1994.

[10] R. C. Eberhart, Y. Shi (1998), *Comparison between Genetic Algorithms and Particle Swarm Optimization*, Proceedings of the 7th Annual Conference on Evolutionary Programming, San Diego, California.

[11] Kennedy, J., Spears, W. (1998), *Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and some Genetic Algorithms on the Multimodal Problem Generator*, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA.

[12] R. C. Eberhart, Y. Shi (1998), *Parameter Selection in Particle Swarm Optimization*, Proceedings of the 7th Annual Conference on Evolutionary Programming.

[13] R. C. Eberhart, Y. Shi (1998), *A Modified Particle Swarm Optimizer*, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, 1998.

[14] Jose C. Principe, Deniz Erdogmus, *From Adaptive Linear to Information Filtering*, IEEE 2000.

[15] V. Miranda, C. Cerqueira, C. Monteiro, *Entropy and Fuzzy Inference Systems – and application to Wind Power Prediction*, submitted to IEEE Transactions on Power Systems, August 2005.

[16] D. Erdogmus and J. C. Principe (2002), *An Error-Entropy Minimization Algorithm for Supervised Training of Nonlinear Adaptive Systems*, IEEE Transactions on Signal Processing, Vol. 50, No. 7, July 2002.

[17] R. A. Morejon and J. C. Principe (2004), *Advanced Search Algorithms for Information-Theoretic Learning With Kernel-Based Estimators*, IEEE Transactions on Neural Networks, Vol. 15, No. 4, July 2004.

[18] Jose C. Principe, Dongxin Xu, *Information-Theoretic Learning Using Renyi's Quadratic Entropy*, Computational NeuroEngineering Laboratory, University of Florida, USA.

[19] Vladimiro Miranda, *Sistemas de Inferência Difusa do Tipo Takagi-Sugeno*, Junho 2000

[20] M. Locatelli and G. R. Wood, *Objective functions, stochastic algorithms and convergence rates*.

[21] http://clerc.maurice.free.fr/pso/Semi-continuous_challenge/Semi-continuos_challenge.htm

[22] Wind Energy: Short-term Prediction – An Overview, John Wiley & Sons, 2004.

Outras referências consultadas:

[23] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*, Morgan Kaufmann Publishers, 2001

[24] R. C. Eberhart, Y. Shi (1999*), Empirical Study of Particle Swarm Optimization.*

[25] Jose C. Principe, John. W. Fisher III, DongXin Xu, *Information-Theoretic Learning*, Computational NeuroEngineering Laboratory, University of Florida, May 1998

[26] G. Castellano, A. M. Fanelli, C. Mencar, *A neuro-fuzzy network to generate human-understandable knowledge from data*, Cognitive Systems Research 3 (2002), 125-144.

[27] *Fuzzy Logic Toolbox, for use with Matlab: user's guide*, version 2, The MathWorks, 2001.

[28] Wind Energy: Hourly Wind Power Variations in the Nordic Countries, 2004 John Wiley & Sons, 2004.