REDES NEURAIS GRANULARES EVOLUTIVAS

Daniel F. Leite*, Pyramo Costa Jr.†, Fernando Gomide*

*Faculdade de Engenharia Elétrica e Computação - Universidade Estadual de Campinas (UNICAMP), Campinas, SP, Brasil

†Programa de Pós-Graduação em Engenharia Elétrica - Universidade Católica de Minas Gerais (PUC-MG), Belo Horizonte, MG, Brasil

Emails: danfl7@dca.fee.unicamp.br, pyramo@pucminas.br, gomide@dca.fee.unicamp.br

Abstract— The objective of this study is to introduce the concept of evolving granular neural networks (eGNN) and to develop a framework of information granulation and its role in the online design of neural networks. The suggested eGNN are neural models supported by granule-based learning algorithms whose aim is to tackle classification problems in continuously changing environments. eGNN are constructed from streams of data using fast incremental learning algorithms. Basically, they try to find information occurring in the incoming data using the concept of granules and T-S neurons as basic processing elements. The main characteristics of eGNN models are continuous learning, self-organization, and adaptation to unknown environments. To illustrate the effectiveness of the approach, the paper considers a real power transformer fault detection problem.

Keywords— Granular Computing, Evolving Systems, Pattern Recognition.

Resumo— O objetivo deste estudo é introduzir o conceito de redes neurais granulares evolutivas (eGNN) e desenvolver uma plataforma de granulação de informação e seu papel no projeto *on-line* de redes neurais. As eGNN sugeridas são modelos neurais munidos com algoritmos de aprendizado baseados em grânulos cujo objetivo é lidar com problemas de classificação em ambientes continuamente em mudança. eGNN são construídas a partir de fluxos de dados usando algoritmos incrementais. Basicamente, elas tentam encontrar informações ocorrendo nos dados usando o conceito de grânulos e neurônios T-S como elementos de processamento básicos. As características principais de eGNN são aprendizado contínuo, auto-organização e adaptação a ambientes desconhecidos. Para ilustrar a abordagem, o artigo considera um problema real de detecção de falta em transformador.

Keywords— Computação Granular, Sistemas Evolutivos, Classificação de Padrões.

1 Introdução

Um dos aspectos mais importantes da inteligência de sistemas recai sobre suas habilidades de adaptação a novas situações (Bouchachia et al., 2007). Para alcançar adaptação, estes sistemas devem ser munidos com algoritmos de aprendizado que impliquem um melhoramento contínuo ou, ao menos, a não-degradação do desempenho do sistema em ambientes sempre em mudança. As abordagens sugeridas neste artigo são modelos orientados à classificação que apresentam adaptabilidade incremental. Em particular, pesquisadores em reconhecimento de padrões estão fazendo frente ao desafio de classificar fluxos de dados usando modelos incrementais que adaptam ou evoluem suas estruturas e parâmetros baseado em possíveis novas informações contidas nos dados. Estes modelos *on-line* de fluxos de dados são muitas vezes referidos como classificadores evolutivos.

Um classificador é dito ser evolutivo se ele possui as seguintes características: **i)** habilidade de aprendizado *on-line* a partir de fluxos de dados. Uma vez que os dados são processados não há como recorrer a dados históricos; **ii)** nenhum conhecimento *a priori* sobre a estrutura do sistema é necessário (adaptação estrutural *on-line*); **iii)** nenhum conhecimento prévio sobre as propriedades estatísticas dos dados é requerido (classificação não-paramétrica); **iv)** nenhum conhecimento prévio sobre o número de classes e protótipos por classe é necessário; e **v)** nenhuma inicialização de protótipos é requerida. Além destas características, é desejado que classificadores evolutivos apresentem rápida assimilação de conhecimento, poucos requerimentos de memória e habilidade de separação não-linear de classes de dados.

O projeto de classificadores evolutivos é principalmente compreendido pela construção de mecanismos de aprendizado objetivando induzir a geração de novo conhecimento sem, no entanto, perder o conhecimento "do passado" (esquecimento catastrófico); além disso, classificadores evolutivos tentam refinar o conhecimento existente. O problema pode ser sumarizado em como acomodar novos dados no modelo de forma incremental enquanto manter o sistema em operação.

Processos de classificação algumas vezes requerem a construção e o teste do modelo, de forma simultânea, em ambientes que constantemente evoluem no tempo. Se um classificador estático for usado para lidar com fluxos de dados, a precisão do processo de classificação pode decair quando, por exemplo, exista uma repentina série de exemplos pertencendo a uma classe particular. Classificadores pré-treinados *off-line* podem apresentar bom desempenho em certos contextos, entretanto eles necessitam ser re-projetados para novas circunstâncias.

Exemplos de classificadores evolutivos que têm sido discutidos na literatura incluem Growing Neural Gas (Fritzke, 1995), Evolving Self-Organizing Map (Kasabov, 2007), Evolving

Neuro-Fuzzy Network (Kasabov, 2001), eClass (Angelov et al., 2007), Fuzzy Min-Max Neural Network FMM (Simpson, 1992). FMM é um caso particular da mais geral eGNN sugerida neste artigo no sentido que grânulos/protótipos podem individualmente assumir valores intermediários entre a operação min e max do primeiro modelo. Isto é realizado a partir do conceito de normas nulas e neurônios T-S. Além disto, existem consideráveis diferenças e melhorias no algoritmo de aprendizado de FMM que fazem eGNN ser um modelo mais competitivo e efetivo. Contudo, a idéia intuitiva do paradigma de aprendizado por contração e expansão permanece.

Em comum, os mecanismos de aprendizado mencionados anteriormente geram protótipos quando os dados são suficientemente dissimilares aos protótipos existentes. Caso contrário, um ajuste de alguns protótipos é conduzido. A decisão de associar novos dados a um protótipo existente ou a um novo protótipo é baseada em alguns parâmetros de controle como o tamanho de grânulos/hipercaixas/clusters, limiar de erro ou limiar de dissimilaridade.

Após esta introdução, o artigo procede introduzindo neurônios T-S na Seção II. As Seções III e IV detalham a modelagem eGNN e o procedimento de aprendizado associado. A Seção V apresenta resultados do comportamento de eGNN quando resolvendo um problema real de classificação. O artigo conclui com a Seção VI sumarizando sua contribuição principal e sugerindo direções para investigações futuras.

2 Breve Introdução a Neurônios T-S

Neurônios T-S são implementações neurais de funções de agregação T-S. Neste artigo, enfatizamos normas nulas (Calvo et al., 2001), uma classe especial de operadores T-S que incluem T-normas e T-conormas (S-normas) como casos limite. Normas nulas estendem T-normas e S-normas, pois elas oferecem flexibilidade na escolha do elemento neutro, chamado elemento de absorção, de operações de agregação de conjuntos fuzzy. Normas nulas ligam construções T-normas e S-normas. Como resultado conseguimos comutação entre as propriedades *and* e *or* dos operadores lógicos ocorrendo nestas construções. Neurônios T-S herdam o processamento conectivo lógico de normas T e S como uma conjunção. Isto os tornam flexíveis e logicamente atraentes.

2.1 Normas Nulas

Normas triangulares T e T-conormas S são operadores binários comutativos, associativos e crescentes definidos no quadrado unitário cujas condições limite são $T(a,0) = 0$ e $T(a,1) = a$; e $S(a,0) = a$ e $S(a,1) = 1$, $a \in [0,1]$, respectivamente. O elemento neutro de normas T e S são $e = 1$ e $e = 0$, respectivamente. Considere uma norma triangular contínua T e uma conorma triangular contínua S. Um operador binário F é chamado de função de agregação T-S se ele é crescente e comutativo, e satisfaz as condições limite $F(a,0) = T(F(1,0),a)$ e $F(a,1) = S(F(1,0),a) \forall a \in [0,1]$.

Normas nulas podem ser vistas como uma forma de generalizar normas triangulares. Estas normas são flexíveis com relação à escolha do elemento neutro, admitindo $e \in [0,1]$. Quando $e = 0$, uma norma nula se torna uma T-norma. Por outro lado, quando $e = 1$, a norma nula se torna uma S-norma. Formalmente, uma norma nula é uma relação binária $V : [0,1] \times [0,1] \to [0,1]$ satisfazendo as seguintes propriedades:

Comutatividade: $V(a,b) = V(b,a)$,
Monotonicidade: $V(a,b) \leq V(a,c)$, se $b \leq c$,
Associatividade: $V(a,V(b,c)) = V(V(a,b),c)$,
Elemento de absorção $e \in [0,1]$: $V(a,e) = e$,

e que satisfaz $V(a,0) = a \ \forall \ a \in [0,e]$ e $V(a,1) = a \ \forall \ a \in [e,1]$, onde $a,b,c \in [0,1]$. Limitamos o estudo deste artigo à seguinte família de construtores de normas nulas:

$$V(a,b) = \begin{cases} eS\left(\frac{a}{e}, \frac{b}{e}\right) & , a,b \in [0,e], \\ e + (1-e)\,T\left(\frac{a-e}{1-e}, \frac{b-e}{1-e}\right) & a,b \in [e,1], \\ e & \text{caso contrário.} \end{cases}$$

Em virtude desta restrição, podemos observar na Fig. 1 que a escolha do valor de e define o tamanho da região Ω e oferece flexibilidade à realização de normas nulas. A discriminação entre normas nulas and-dominada e or-dominada refere-se ao valor do elemento de absorção $e \in [0, 0.5[$ e $e \in]0.5, 1]$, respectivamente.

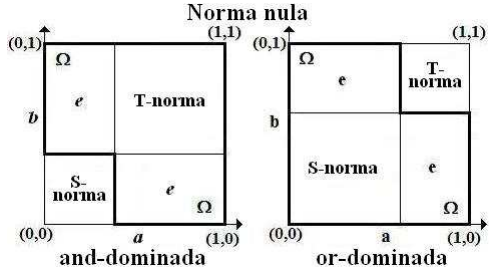


Figura 1: Realizações de normas nulas

Uma motivação para o uso de normas nulas é deduzida do fato que T-normas descrevem situações em que ambas as condições a e b são absolutamente necessárias. Caso uma destas condições não for satisfeita, rejeita-se completamente a alternativa correspondente. Em normas nulas caso uma destas condições não for satisfeita, podemos ainda considerar a alternativa.

2.2 Neurônios T-S

Neurônios T-S são modelos de neurônios artificiais que utilizam normas nulas como operação de agregação. Formalmente, seja $X = \{x_1, x_2, ..., x_n\}$

um vetor de entrada no hipercubo unitário, $X \in [0,1]^n$, e $W = \{w_1, w_2, ..., w_n\}$ seu correspondente vetor de pesos, $W \in [0,1]^n$. A agregação por norma nula das entradas ponderadas produz a saída $o \in [0,1]$ dada por:

$$o = V(x_1 w_1, x_2 w_2, ..., x_n w_n).$$

Esta relação enfatiza a flexibilidade paramétrica residindo nas conexões, necessária para dar suporte ao aprendizado. Esta expressão será denotada por $o = V(x, w)$. A Fig. 2 mostra uma visão do processamento neural. É notável a diversidade de características não-lineares que o mapeamento produzido por neurônios T-S pode assumir dependendo da escolha de W e e.

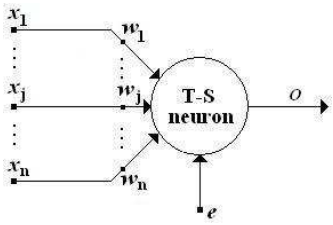


Figura 2: Visão esquemática de um neurônio T-S

Dada esta estrutura, o processamento de cada entrada pode ser feito independentemente e por causa do valor ajustável do elemento de absorção, encontramos uma maneira diferente de processamento de agregação que permite considerações intermediárias entre T-normas e S-normas.

3 Rede Neural Granular Evolutiva

A computação granular (GrC) tem emergido nos últimos dez anos como um novo paradigma de processamento de informação (Pedrycz e Gomide, 2007). O fundamento da GrC é o fato que precisão é um objetivo caro e as vezes desnecessário para a modelagem de sistemas complexos. Em particular, o raciocínio humano não aparenta operar em um nível numérico de precisão, mas em um nível de detalhamento maior e mais abstrato. GrC é uma plataforma para a expressão destas abstrações em processos computacionais.

O conceito de redes neurais granulares (GNN) foi estabelecido em (Pedrycz e Vukovich, 2001). Basicamente, o desenvolvimento de GNN envolve duas fases principais (refira-se a Fig. 3):

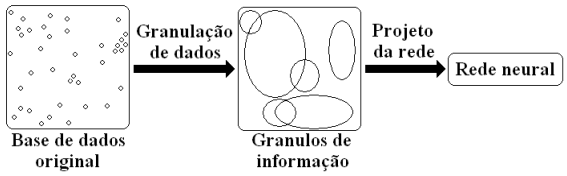


Figura 3: Projeto de redes neurais granulares

i) Granulação de dados numéricos. Neste passo uma coleção de grânulos de informação é formada; **ii)** Construção da rede neural. Deste passo em diante, o aprendizado é baseado nos grânulos de informação ao invés de nos dados originais. Como conseqüência, a rede neural não é exposta aos dados originais muito mais numerosos.

Neste artigo, introduzimos versões evolutivas de GNN. O objetivo de eGNN é classificar dados de processos dinâmicos continuamente em mudança. Modelos eGNN são construídos a partir do processamento de fluxos de dados usando algoritmos de aprendizado incrementais. Basicamente, modelos eGNN tentam encontrar granulações de informação ocorrendo nos dados usando grânulos e neurônios T-S como elementos de processamento. As características principais destes modelos são aprendizado contínuo, auto-organização e adaptação a ambientes desconhecidos. Regras de associação na forma SE-ENTÃO podem ser facilmente extraídas de sua estrutura e parâmetros durante o processo de evolução. Durante a evolução, grânulos são incrementalmente criados, atualizados ou divididos ao longo de uma ou mais dimensões. Em particular, quando a entrada atual é percebida pelos grânulos existentes, três tipos de situações podem acontecer: **i)** dois ou mais grânulos existentes são aptos a se adaptarem a nova entrada. Então, aquele que apresentar o maior grau de pertinência obtido a partir de uma operação de agregação baseada em norma nula é selecionado para acomodar a entrada. A acomodação consiste no ajuste dos parâmetros dos grânulos e dos neurônios T-S no sentido de melhor receberem entradas similares em passos futuros; **ii)** a entrada atual pertence aos limites de apenas um grânulo existente. O grânulo e seu correspondente neurônio T-S são atualizados para acomodar a entrada; e **iii)** nenhum dos grânulos existentes acomoda a entrada. Então, um novo grânulo é criado. O mecanismo de refinamento por divisão de grânulos é aplicado quando o grânulo que acomoda uma entrada é associado a um rótulo de classe diferente. O seguinte mecanismo de decomposição então procede: **i)** particionar o grânulo em dois grânulos menores; **ii)** criar um novo grânulo acomodando a entrada; e **iii)** associar este novo grânulo à classe desejada.

4 Aprendizagem em eGNN

Modelos eGNN aprendem a partir de um fluxo de dados $(x, C)^{[h]}$, $h = 1, 2, ...$, onde o rótulo de classe $C^{[h]}$ é assumido ser conhecido dado o vetor de entrada correspondente $x^{[h]}$. eGNN acomoda o novo conhecimento contido no dado criando um novo grânulo de informação ou adaptando os parâmetros dos grânulos e neurônios T-S existentes. Cada grânulo γ^i de uma coleção finita de grânulos $\gamma = \{\gamma^1, \gamma^2, ..., \gamma^c\}$, definida no espaço de entrada $X \subseteq \Re^n$, é associado a uma classe k da coleção finita de classes $C = \{Class_1, Class_2, ..., Class_m\}$ no espaço de saída $Y \subseteq \mathbb{N}^m$. O grânulo γ^i pode assumir diferentes formas geométricas e tamanhos.

Neste artigo enfatizamos intervalos fuzzy multidimensionais.

A estrutura eGNN é ilustrada na Fig. 4. A rede consiste de 5 camadas. A camada de entrada basicamente distribui vetores $x_j^{[h]}$, $j = 1, ..., n$, na rede. A camada evolutiva consiste de grânulos de informação γ^i, $i = 1, ..., c$, extraídos a partir do fluxo de dados. Grânulos γ^i são definidos por funções de pertinência A_j^i, $j = 1, ..., n$, em diferentes dimensões e universos, cada uma representando um atributo de entrada. Na camada de agregação se encontram os neurônios T-S, TSn^i, $i = 1, ..., c$. Eles processam entradas normalizadas $\tilde{x}_j^{i[h]} \forall j, i$, provenientes do processo de granulação, e agregam dados gerando a saída o^i $\forall i$. A saída pode ser vista como uma medida de compatibilidade entre o dado e os grânulos existentes. A camada de decisão compara os valores da agregação. O grânulo com a maior ativação, o grânulo vencedor, produz um vetor de saída binário $\bar{C}_k^{[h]}$, $k = 1, ..., m$, com 1 na posição correspondente à classe associada ao grânulo vencedor, e 0 nas demais. A camada de saída compara a saída binária da rede $\bar{C}_k^{[h]}$ com o vetor de saída desejado $C_k^{[h]}$, $\forall k$. Esta operação resulta no erro de estimação da rede ϵ_k, $\forall k$. Em modelos eGNN, nenhum grânulo e neurônio T-S existe anteriormente ao aprendizado, eles são criados e adaptados durante o processo evolutivo sempre que novas informações são encontradas no fluxo de dados. Os pesos das conexões $w_j^i \forall j, i$, associam diferentes graus de relevância a diferentes atributos (*wrapper* incremental), enquanto os pesos $\delta^i \forall i$, dependem da quantidade de dados pertencendo a γ^i. Em geral, δ^i é um mecanismo para indicar regiões de dados mais densas ou esparsas.

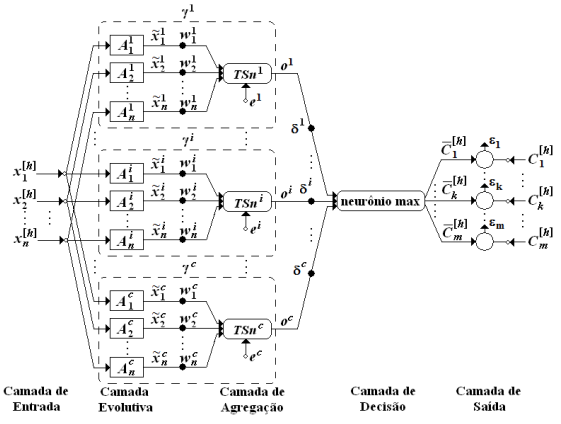
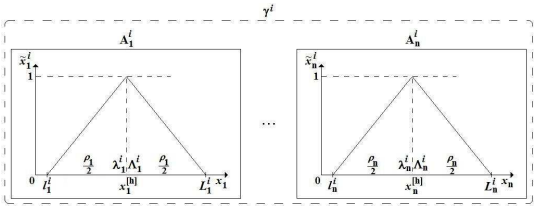


Figura 4: Rede neural granular evolutiva

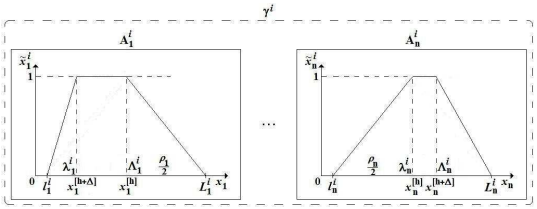
Seja $\rho \in X \subseteq \Re^n$ o tamanho máximo que um grânulo pode assumir; escolhas pertinentes de ρ são muito importantes logo que ele está diretamente relacionado com a transparência e precisão do modelo. Grânulos maiores que ρ podem resultar na perda de regiões desejadas. Geralmente, quanto maior ρ, menos grânulos são criados e, apesar de a interpretação ser mais simples, menor é a habilidade de capturar não-linearidades entre classes. Por outro lado, valores menores de ρ podem levar ao sobre-ajuste do modelo e a perda de interpretabilidade. A idéia é encontrar um compromisso aceitável entre as situações extremas.

4.1 Criando e Atualizando Grânulos

Funções de pertinência $A_j^i \forall j$ associadas a grânulos γ^i são definidas em diferentes domínios. Assumimos funções trapezoidais ou triangulares para A_j^i de γ^i. Logo, as funções são definidas por quatro parâmetros: $j-$ésimo limite inferior, l_j^i; $j-$ésimo limite superior, L_j^i; e valores intermediários da função, λ_j^i e Λ_j^i. Para funções trapezoidais $\lambda_j^i < \Lambda_j^i$, e para funções triangulares $\lambda_j^i = \Lambda_j^i$. Grânulos γ^i são associados às classes $C_k^{[h]} \forall k$.



(a) Criação de γ^i para acomodar a entrada $x^{[h]}$



(b) Adaptação de γ^i para acomodar $x^{[h+\Delta]}$

Figura 5: Criação e adaptação de grânulos

Sempre que um novo dado $x_j^{[h]} \notin [l_j^i, L_j^i] \forall j, i$, cria-se um novo grânulo. O procedimento também é adotado quando, embora $x_j^{[h]} \in [l_j^i, L_j^i] \forall j$, o rótulo de classe $C^{[h]}$ difere daquele associado a γ^i. O novo grânulo γ^{c+1} é construído usando funções triangulares $A_j^{c+1} \forall j$, com centro $x^{[h]} \in \Re^n$. Seus parâmetros assumem $l_j^{c+1} = x_j^{[h]} - \frac{\rho_j}{2}$; $\lambda_j^{c+1} = \Lambda_j^{c+1} = x_j^{[h]}$; e $L_j^{c+1} = x_j^{[h]} + \frac{\rho_j}{2}$; $\forall j$. A Fig. 5(a) ilustra o procedimento. Caso $x^{[h+\Delta]}$, onde Δ é um inteiro positivo, pertença aos limites de γ^i e sua respectiva classe $C^{[h]}$ é a mesma de γ^i, então os parâmetros λ_j^i e $\Lambda_j^i \forall j$ são atualizados acomodando $x_j^{[h+\Delta]} \forall j$. Basicamente, a adaptação consiste em ajustar $\lambda_j^i = x_j^{[h+\Delta]}$, se $x_j^{[h+\Delta]} \in [l_j^i, \lambda_j^i]$; ou $\Lambda_j^i = x_j^{[h+\Delta]}$, se $x_j^{[h+\Delta]} \in [\Lambda_j^i, L_j^i]$. A Fig. 5(b) ilustra o procedimento.

4.2 Ajustando os Pesos da Camada Evolutiva

Os pesos da camada evolutiva $w_j^i \forall j, i$, objetivam ponderar a importância do atributo j em diferen-

ciar classes. Inicialmente, todos w_j^i são 1. Durante o aprendizado, alguns w_j^i podem reduzir seus valores dependendo do fluxo de dados.

Novos dados $(x, C)^{[h]}$ podem causar a revisão de γ^i se este grânulo é o mais compatível com $x^{[h]}$, mas $C^{[h]}$ é diferente da classe associada a γ^i. O seguinte procedimento é usado para comprimir γ^i reduzindo sua compatibilidade com $x^{[h]}$. Duas situações são de interesse: **i)** se A_j^i, $i = 1, ..., c$, para algum j resulta em $\tilde{x}_j^{i[h]} \in]0, 1[$, então faça $l_j^i = x_j^{[h]}$ se $x_j^{[h]} \leq \lambda_j^i$, ou faça $L_j^i = x_j^{[h]}$ se $x_j^{[h]} \geq \Lambda_j^i$. Este procedimento comprime a função de pertinência A_j^i de γ^i; e **ii)** se $A_j^i \forall i$ é tal que $\tilde{x}_j^{i[h]} = 1$ para algum j, então os parâmetros de A_j^i são mantidos e o peso associado w_j^i é ajustado:

$$w_j^i(\text{novo}) = \beta w_j^i(\text{velho}) \quad ,$$

onde $\beta \in]0, 1[$ é uma constante de decaimento. O procedimento de atualização é justificado, pois A_j^i não ajuda satisfatoriamente a diferenciar classes de dados. Após os passos **i)** e **ii)**, um grânulo γ^{c+1} é criado com funções A_j^{c+1} centradas em $x_j^{[h]} \forall j$. γ^{c+1} é então associado a classe $C^{[h]}$. Note que, dando diferentes graus de importância a cada atributo, o procedimento relembra técnicas como análise de componentes principais no sentido de procurar por subconjuntos de atributos relevantes.

4.3 Ajustando os elementos de absorção de neurônios T-S

O modelo eGNN sugerido usa neurônios T-S anddominados para processar dados. O $i-$ésimo neurônio TSn^i processa os n atributos da entrada normalizada $\tilde{x}_j^{i[h]}$, $j = 1, ..., n$, associada ao grânulo γ^i através de $V(\tilde{x}_j^{i[h]}, w_j^i)\ \forall j$. O resultado é um valor de saída único o^i que pode ser visto como a compatibilidade entre $x^{[h]}$ e γ^i. Um procedimento para inicializar e ajustar o elemento de absorção de neurônios T-S é o seguinte: escolher uma T-norma e uma S-norma para os neurônios T-S; quando um grânulo γ^{c+1} é criado, ajustar seu elemento de absorção fazendo $e^{c+1} = 0$. Isto induz uma T-norma. Durante a evolução, alguns e^i, $i = 1, ..., c$, podem aumentar seus valores dependendo dos dados. Por exemplo, caso um ou poucos atributos de $x_j^{[h]} \forall j$ não ative as funções $A_j^i \forall j$ podemos ainda considerar γ^i como candidato para acomodar $(x, C)^{[h]}$ aumentando o elemento de absorção de seu respectivo neurônio T-S como segue

$$e^i(\text{novo}) = e^i(\text{velho}) + \chi(1 - e^i(\text{velho})) \quad ,$$

onde $\chi \in]0, 1[$ é uma constante de crescimento.

Ajustando e^i durante a evolução, baixos valores de pertinência são compensados por valores maiores usando processamento baseado em S-norma. Caso poucos atributos do dado atual não pertençam a γ^i, então o dado pode ainda assim ser compatível com γ^i a um certo grau através do aumento de e^i. Isto geralmente evita a criação de grânulos muito similares e ajuda a manter uma quantidade razoável de grânulos na estrutura do modelo. A Fig. 6 ilustra a idéia do procedimento que pode ser vista como uma forma de ajuste adaptativo do tamanho dos grânulos. Note que, dependendo da T e S-normas assumidas, e.g. Lukasiewicz e Máximo; Mínimo Nilpotente e Soma Limitada, os grânulos e neurônios T-S podem ser vistos no espaço de entrada como assumindo diferentes formas geométricas e tamanhos.

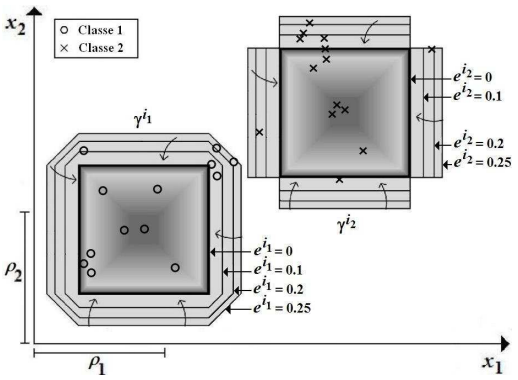


Figura 6: Adaptação de grânulos e neurônios T-S conforme o ajuste de e

4.4 Ajustando os Pesos da Camada de Decisão

As conexões da camada de decisão são ponderadas por δ^i. Ele representa a quantidade de dados pertencendo a $\gamma^i \forall i$. Em geral, os pesos δ^i são uma forma de identificar regiões densas e esparsas de dados. Quanto maior o valor de δ^i, maior a probabilidade de ativação de γ^i em passos subseqüentes.

Inicialmente, ajusta-se $\delta^i = 1, \forall i$. Durante a evolução, os valores de alguns δ^i podem reduzir ou aumentar dependendo de critérios préestabelecidos. Por exemplo, um critério simples é o seguinte: se γ^i não ativa após um certo número de passos, então reduza δ^i fazendo

$$\delta^i(\text{novo}) = \zeta^i \delta^i(\text{velho}) \quad ,$$

onde $\zeta^i \in]0, 1[\ \forall i$. Caso contrário, se γ^i ativa com freqüência, aumente δ^i a partir de

$$\delta^i(\text{novo}) = \delta^i(\text{velho}) + \zeta^i(1 - \delta^i(\text{velho})) \quad .$$

Uma conseqüência deste procedimento é que, em ambientes sempre em mudança, grânulos inativos por um certo número de passos são suprimidos. Isto significa que o processo atual mudou e a exclusão de grânulos significa $\delta^i \to 0^+$. Isto é justificado para manter uma quantidade razoável de informação atualizada disponível. Caso a aplicação requeira a memorização de eventos raros, então o procedimento de exclusão é proibitivo.

4.5 O Papel do Neurônio Max

O neurônio max do modelo eGNN sugerido computa sua entrada c-dimensional, que significa

graus de compatibilidade entre $\gamma^i \forall i$ e $x^{[h]}$, e determina a classe associada ao grânulo que apresentou a maior compatibilidade a entrada. O neurônio max implementa a abordagem *winner-takes-all* apresentando na saída um vetor m-dimensional com 1 na posição vencedora e 0 nas demais.

Seja $\bar{C}_k^{[h]}$, $k = 1, ..., m$, o vetor binário m-dimensional estimado pela rede para $x^{[h]}$, então $\epsilon_k = C_k^{[h]} - \bar{C}_k^{[h]}$ é o erro entre $\bar{C}_k^{[h]}$ e a classe desejada $C_k^{[h]}$ $\forall k$. Caso $\epsilon = \vec{0}$, refina-se os parâmetros da rede. Caso $\epsilon \neq \vec{0}$, os mecanismos de compressão de grânulos e ajuste de pesos procedem.

5 Detecção de Falta em Transformadores

∗ *Descrição básica:* Dados de análises químicas de gases dissolvidos no óleo de um transformador foram obtidos a partir da Companhia de Energia de Minas Gerais, CEMIG. Eles contém 10 classes que representam 9 condições de falta e uma condição de operação saudável. Nenhum par de classes é linearmente separável.

∗ *Características da base de dados:* Número de exemplos: 3500; Número de atributos: 8 entradas quantitativas nomeadas H_2, CH_4, CO, C_2H_2, C_2H_4 e C_2H_6, gás combustível e temperatura do óleo, e uma saída representando 10 condições de operação; Dados de treino/teste: 60/100 %. Para comparação de performance, consideramos os modelos: redes Perceptron Multi-Camadas (MLP) e Elman, Fuzzy C-Means (FCM), extended e evolving Takagi-Sugeno (xTS) (eTS), e eGNN.

∗ *Resultados:* Para avaliar o efeito de diferentes parametrizações, eGNN adota três conjuntos de parâmetros: eGNN-1 usa $\rho_1 = 0.7$, $\beta_1 = \zeta_1 = 0.9$, $\chi_1 = 0.1$; eGNN-2 adota $\rho_2 = 0.4$, $\beta_2 = \zeta_2 = 0.95$, $\chi_2 = 0.05$; e eGNN-3 emprega $\rho_3 = 0.2$, $\beta_3 = \zeta_3 = 0.9$, $\chi_3 = 0.1$. Cada experimento é processado cinco vezes com os mesmos parâmetros. Em cada realização, os dados são misturados para induzir diferentes ordens do fluxo, e são apresentados seqüencialmente apenas uma vez a eGNN. eGNN começa a aprender de uma base de regras vazia e sem pré-treinamento. A Tabela I mostra a performance de eGNN e dos demais modelos.

Tabela 1: Detecção de falta em transformadores

Modelo	No. protótipos	% Precisão^{+}	% Precisão^{++}
MLP	79	73,17	65,99
Elman	79	71,37	65,23
FCM	90	89,14	83,46
eTS*	indisponível	90,03	88,09
xTS*	indisponível	91,06	89,83
eGNN-1*	48	96,57	94,83
eGNN-2*	56	98,34	96,93
eGNN-3*	79	99,46	98,40

*** Modelos *on-line*; $^{+}$ Melhor; $^{++}$ Média.**

Os resultados mostram que os modelos eGNN obtiveram as melhores performances. Além disso, eles podem operar em tempo real usando estruturas compactas. Os maiores fatores influenciando este estudo comparativo são as peculiaridades de cada algoritmo de aprendizado e as diferentes considerações estruturais. No caso de eGNN foram o maior grau de flexibilidade dos neurônios T-S, sua estrutura adaptativa e a abordagem baseada em grânulos. Modelos eGNN mostraram um forte potencial para a resolução de problemas de classificação que demandam adaptabilidade incremental.

6 Conclusão

O conceito de modelos neurais granulares evolutivos foi introduzido neste trabalho. Uma rede neural granular evolutiva eGNN foi sugerida como um mecanismo de desenvolvimento de modelos adaptativos de sistemas. A abordagem eGNN evolui modelos a partir de grânulos de informação e neurônios T-S. Isto provê à modelos eGNN a habilidade de desenvolver estruturas altamente flexíveis e mecanismos robustos para acompanhar a evolução. Experimentos em um problema de detecção de falta em transformador mostrou que eGNN é competitiva quando comparada à técnicas não-lineares alternativas. Trabalhos futuros abordarão aplicações em predição e controle.

Agradecimento

O primeiro autor agradece a CAPES pelo apoio financeiro. O segundo autor manifesta apreço a CEMIG pelo suporte P&D178. O último autor é grato ao CNPq pelo suporte 304857/2006-8.

Referências

Angelov, P.; et al. (2007). "Architectures for evolving fuzzy rule-based classifiers". *IEEE Conf. on Systems, Man and Cybernetics*, pp: 2050-2055.

Bouchachia, A.; Gabrys, B.; Sahel, Z. (2007). "Overview of some incremental learning algorithms". *IEEE Fuzzy Systems Conf.*, Vol. 1, 6p.

Calvo, T.; De Baets, J.; Fodor, J. (2001). "The functional equations of Frank and Alsina for uninorms and nullnorms". *Fuzzy Sets and Systems*, Vol. 120, pp: 385-394.

Fritzke, B. (1995). "A Growing Neural Gas Network Learns Topologies". *Advances in Neural Information Processing Systems*, pp: 625-632.

Kasabov, N. (2007). Evolving Connectionist Systems. Springer, 2^a edição, 451p.

Kasabov, N. (2001). "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning". *IEEE Transactions on SMC - Parte B*, Vol. 31-6, pp: 902-918.

Pedrycz, W.; Gomide, F. (2007). Fuzzy Systems Engineering: Toward Human-Centric Computing. Wiley-IEEE Press, 1^a edição, 526p.

Pedrycz, W.; Vukovich, W. (2001). "Granular Neural Networks". *Neurocomputing*, Vol. 36, pp: 205-224.

Simpson, P. (1992). "Fuzzy min-max neural networks. Part I: Classification". *IEEE Trans. on Neural Networks*, Vol. 3-5, pp: 776-786.

SISTEMAS CONEXIONISTAS EVOLUTIVOS

Daniel F. Leite*, Pyramo Costa Jr.†, Fernando Gomide*

*Faculdade de Engenharia Elétrica e Computação - Universidade Estadual de Campinas (UNICAMP), Campinas, SP, Brasil

†Programa de Pós-Graduação em Engenharia Elétrica - Universidade Católica de Minas Gerais (PUC-MG), Belo Horizonte, MG, Brasil

Emails: danfl7@dca.fee.unicamp.br, pyramo@pucminas.br, gomide@dca.fee.unicamp.br

Abstract— The aim of this study is to introduce the concept of evolving connectionist systems (ECOS) and to suggest a framework for adaptive processes modeling. The outlined approaches are online neural models constructed from data streams. Basically, ECOS accommodate the possible new knowledge contained in the data in their structures and parameters using one-pass non-recursive incremental learning algorithms. ECOS applications extend to classification, regression, prediction and control problems in continuously changing environments. Their main characteristics include continuous learning, self-organization and adaptation to unknown environments. To illustrate the effectiveness of the modeling approach, the paper considers two types of ECOS, namely evolving Multi-Layer Perceptron (eMLP) and evolving Self-Organizing Map (eSOM), for solving a real electrical machines fault diagnosis problem.

Keywords— Evolving Systems, Neural Networks, Pattern Recognition.

Resumo— O objetivo deste estudo é introduzir o conceito de sistemas conexionistas evolutivos (ECOS) e sugerir abordagens para a modelagem de processos que demandam adaptabilidade incremental. ECOS são modelos neurais *on-line* construídos a partir de fluxos de dados. Basicamente, eles acomodam o possivel novo conhecimento contido nos dados em suas estruturas e parâmetros usando algoritmos de aprendizado incrementais, não-recursivos e de um passo. As aplicações de ECOS se estendem a problemas de classificação, regressão, predição e controle em ambientes sempre em mudança. Suas características principais incluem aprendizado contínuo, auto-organização e adaptação a ambientes desconhecidos. Para ilustrar a efetividade da abordagem, o artigo considera dois tipos de ECOS, Perceptron Multi-Camadas evolutivo (eMLP) e Mapa Auto-Organizável evolutivo (eSOM), para resolução de um problema real de diagnóstico de falta em máquinas elétricas.

Keywords— Sistemas Evolutivos, Redes Neurais, Classificação de Padrões.

1 Introdução

A revolução digital tem facilitado a captura e o armazenamento de dados (Fayyad e Uthurusamy, 1996) (Inmon, 1996). Com o rápido desenvolvimento de hardwares e softwares, grandes quantidades de dados têm sido coletadas e armazenadas em bases de dados. A taxa com que estes dados têm sido armazenados tem crescido rapidamente. Como resultado, algumas técnicas estatísticas e de inteligência computacional para aprendizado de máquina e construção de modelos se tornaram inadequadas ou pouco eficientes (Mitra et al., 2002). Dados brutos são raramente úteis de forma direta. O seu real valor depende da habilidade de extração de informações úteis contidas nos dados para suporte a decisões ou exploração e entendimento do fenômeno que governa a fonte de dados.

Em certos contextos, a análise dos dados é um processo manual. Um especialista pode ser altamente capaz de lidar com certos tipos de dados provenientes de um mesmo fenômeno assim provendo sumários e/ou relatórios a respeito do processo. Nestes casos, o próprio especialista pode ser visto como um processador sofisticado ou um modelo do processo. Entretanto, é evidente que a medida que a quantidade e dimensão dos dados aumentam, esta forma de análise colapsa. Especialistas recorrem então a tecnologias computacionais para automatizar estes processos que lidam com grandes quantidades de dados.

Tem emergido, ao longo de aproximadamente uma década, um novo paradigma no campo da inteligência computacional baseado na construção de modelos a partir de algoritmos incrementais e fluxos de dados. Estes modelos, ditos evolutivos, não somente minimizam o problema do armazenamento de grandes quantidades de dados, mas também oferecem características importantes para a modelagem de processos não-lineares adaptativos. As principais características de modelos evolutivos são aprendizado contínuo, auto-organização e adaptação a ambientes desconhecidos. Geralmente, regras de associação podem ser facilmente extraídas a partir de suas estruturas e parâmetros em qualquer passo durante o processo evolutivo.

Um dos mais importantes aspectos da inteligência de sistemas recai sobre a habilidade de adaptação de modelos a situações inéditas (Bouchachia et al., 2007). Para alcançar adaptação, estes modelos devem ter suporte de algoritmos de aprendizado que impliquem um melhoramento contínuo ou ao menos a não-degradação do desempenho do sistema em ambientes sempre em mudança. As abordagens sugeridas neste artigo são modelos conexionistas orientados a problemas de classificação, regressão, predição e controle que apresentam adaptabilidade incremental *on-line*. Em particular, pesquisadores estão fazendo frente ao desafio de modelar fluxos de dados usando algoritmos incrementais que adaptam a estrutura e os parâmetros dos modelos baseados nas

possíveis novas informações contidas nos dados.

Um processo evolutivo pode ser definido como um processo que desenvolve, modificando-se de forma contínua ao longo do tempo. Este processo interage com outros processos do ambiente sendo que, a princípio, não é possível determinar o resultado destas interações. Processos evolutivos são geralmente difíceis de se modelar, pois alguns de seus parâmetros podem não ser conhecidos *a priori*, e perturbações ou mudanças inesperadas podem acontecer em qualquer momento durante o desenvolvimento.

Por outro lado, um modelo é dito ser evolutivo se ele possui as seguintes características:

∗ Habilidade de aprendizado *on-line* a partir de fluxos de dados. Uma vez que os dados são processados não há como recorrer a dados históricos;
∗ Nenhum conhecimento anterior sobre a estrutura do sistema é necessário (adaptação estrutural *on-line*);
∗ Nenhum conhecimento prévio sobre as propriedades estatísticas dos dados é requerido (modelo não-paramétrico);
∗ Nenhuma inicialização de protótipos é requerida.

Além das características mencionadas acima, é muitas vezes desejado que modelos evolutivos possuam rápida assimilação de conhecimento, poucos requerimentos de memória e habilidade não-linear de aproximação de funções ou separação de classes.

O projeto de modelos evolutivos é principalmente compreendido pela construção de mecanismos de aprendizado com o intuito de induzir a geração de novo conhecimento sem, no entanto, perder o conhecimento "do passado" (esquecimento catastrófico); além disto, modelos evolutivos tentam refinar o conhecimento existente. O problema pode ser sumarizado em como acomodar novos dados no modelo de forma incremental enquanto manter o sistema em operação.

Processos de modelagem de dados muitas vezes requerem a construção e o teste do modelo de forma simultânea em um ambiente que constantemente evolui no tempo. Se um modelo estático for usado para evolução de fluxos de dados, sua precisão pode decair repentinamente quando, por exemplo, uma característica do ambiente mudar. Modelos pré-treinados *off-line* podem apresentar bom desempenho em vários contextos, entretanto eles necessitam ser re-projetados para novas circunstâncias.

Após esta introdução, o artigo prossegue na Seção II caracterizando os modelos evolutivos abordados. A Seção III mostra o comportamento dos modelos quando resolvendo um problema real de diagnóstico de falta em máquinas elétricas. O artigo conclui com a Seção IV sumarizando sua contribuição principal e sugerindo tópicos para investigações futuras.

2 Sistemas Conexionistas Evolutivos

Um sistema conexionista evolutivo é uma rede neural ou uma coleção de redes que opera continuamente no tempo e adapta sua estrutura e funcionalidades através de interações contínuas com o ambiente e com outros sistemas. Em termos gerais, um sistema conexionista $\{E, P, W, F, R, FO\}$, que é definido pela sua estrutura E, seu conjunto de parâmetros P, os pesos de suas conexões W, sua função F, um procedimento de aprendizado R, e sua função objetivo FO, aprende se o sistema otimizar ao menos parte de sua estrutura E e função F após a observação dos eventos ψ_1, ψ_2, ..., de um problema no espaço Ψ. Através do processo de aprendizado, o sistema melhora sua reação aos eventos observados e captura informações que podem posteriormente representar conhecimento.

ECOS constituem um pequeno subconjunto de todos os modelos conexionistas no contexto da inteligência computacional. Eles evoluem tanto no espaço de dados do problema, como no próprio espaço do sistema. Eles aprendem em qualquer modo de aprendizado: supervisionado, não-supervisionado, por reforço ou por uma combinação destes. Basicamente, o aprendizado em ECOS consiste nas seguintes etapas:

∗ Aquisição de dados;
∗ Pré-processamento e avaliação de características;
∗ Modelagem conexionista;
∗ Aquisição de conhecimento.

Em geral, suas principais características incluem: ECOS

∗ Evoluem em domínios abertos;
∗ Aprendem continuamente durante toda a sua existência;
∗ Usam aprendizado construtivo e possuem estruturas evolutivas;
∗ Dividem o espaço do problema permitindo uma rápida adaptação.

Nas próximas seções são apresentados fundamentos e propostas de algoritmos de aprendizado dentro deste contexto.

2.1 Medição de Distância

A medição de distância é um passo fundamental em métodos de clusterização, quantização e prototipação (Kasabov, 2007). A distância entre dois pontos no espaço Euclidiano $\Re^n$ é geralmente dada pela distância Euclidiana (norma-2), que é uma generalização do teorema de Pitágoras para mais de duas coordenadas e também a idéia intuitiva de distância. Outras distâncias, baseadas em outras normas são algumas vezes consideradas.

Formalmente, para um ponto (x_1, ..., x_n) e um ponto (z_1, ..., z_n), a distância de Minkowski D de ordem p (norma-p) é definida:

$$D = (\sum_{i=1}^{n} |x_i - z_i|^p)^{\frac{1}{p}},\ p \geq 1.$$

No caso especial em que $p = \infty$ (norma-infinito) tem-se:

$$\begin{aligned} D &= lim_{p\to\infty}(\sum_{i=1}^{n} |x_i - z_i|^p)^{\frac{1}{p}} = \\ &= max(|x_i - z_i|, ..., |x_n - z_n|). \end{aligned}$$

Além da norma-2, a norma-1 (também conhecida como distância de Manhattan) e a norma-infinito (distância de Chebyshev) são as mais usadas, sendo as demais raramente consideradas. Note que p não precisa ser necessariamente um número inteiro, podendo assumir valores reais, desde que não sejam menores que 1 de acordo com o teorema da inequação do triângulo para todos os espaços Euclidianos (Deza e Deza, 2006).

2.2 Perceptron Multi-Camadas Evolutivo

O modelo eMLP foi originalmente proposto em (Kasabov, 2003). Neste artigo sugerimos uma variação do algoritmo de aprendizado eMLP que resulta uma maior eficiência da abordagem em termos de velocidade de processamento e requerimentos de memória. Além disso, o algoritmo de aprendizado sugerido apresenta melhorias que acreditamos tornar eMLP uma abordagem mais efetiva e competitiva.

eMLP é uma rede neural evolutiva que consiste de três camadas de neurônios i, j e k denominadas nesta ordem: camada de entrada, com funções lineares ou não-lineares; camada evolutiva; e camada de saída, com funções de ativação lineares saturadas. A camada evolutiva é a camada que se adapta aos dados e é também a camada a qual o aprendizado está mais relacionado. Admita um fluxo de pares de entrada/saída de dados $(x, y)^{[h]}$, $h = 1, 2, ...$, dirigido a um modelo eMLP. O nível de ativação de um neurônio ϕ^ν da coleção finita de neurônios existentes na camada evolutiva $\Phi = \{\phi^1, ..., \phi^\nu, ..., \phi^c\}$ considerando função de ativação linear é obtido a partir de:

$$A^\nu = 1 - D^\nu,$$

onde D^ν é a distância normalizada entre o vetor de entrada atual $x^{[h]} \in \Re^n$ e o vetor de pesos das conexões da entrada do neurônio ϕ^ν, $W_{ij}^\nu \in \Re^n$. A medida D^ν pode ser calculada, por exemplo, a partir da distância Euclidiana entre $x^{[h]}$ e $W_{ij}^\nu \forall \nu$.

O aprendizado eMLP é baseado na acomodação de novos exemplos de treinamento $x^{[h]}$ nos parâmetros e estrutura do modelo através do ajuste dos pesos sinápticos $W_{ij}^\nu\ \forall \nu$, e/ou da adição de novos neurônios ϕ^{c+1}, ϕ^{c+2}, ..., estendendo a coleção atual de neurônios Φ. Note que $\Phi = \emptyset$ anteriormente ao aprendizado. Em particular, o modelo eMLP gera protótipos quando o dado é suficientemente dissimilar aos protótipos existentes. Caso contrário, um refinamento de alguns protótipos existentes é conduzido. A decisão de associar novos dados a protótipos existentes ou a novos protótipos é baseada no valor limiar ρ^ν que determina a área de atuação de um neurônio ϕ^ν. Por exemplo, caso $D^\nu \leq \rho^\nu$ para algum ν, então ajusta-se os parâmetros W_{ij}^ν correspondentes; caso $D^\nu > \rho^\nu$, então cria-se ϕ^{c+1}.

Quando um neurônio ϕ^{c+1} é criado, seu respectivo vetor de pesos W_{ij}^{c+1} é ajustado exatamente para o vetor de entrada $x^{[h]}$, enquanto que seu vetor de pesos de saída W_{jk}^{c+1} é ajustado exatamente para o vetor de saída desejado $y^{[h]}$.

Um neurônio ϕ^ν é dito vencedor quando ele apresenta a maior ativação A^ν para a entrada atual $x^{[h]}$. Os pesos W_{ij}^ν do neurônio vencedor são ajustados de acordo com:

$$W_{ij}^\nu(\textbf{novo}) = W_{ij}^\nu + \eta_1(x^{[h]} - W_{ij}^\nu),$$

onde η_1 é a taxa de aprendizado 1. O vetor de pesos da saída do neurônio vencedor é ajustado de acordo com:

$$W_{jk}^\nu(\textbf{novo}) = W_{jk}^\nu + \eta_2(A^\nu E^\nu),$$

onde η_2 é a taxa de aprendizado 2 e E^ν é um vetor de erro obtido a partir de:

$$E^\nu = y^{[h]} - \bar{y}^{[h]},$$

onde $y^{[h]} \in \Re^m$ é a saída desejada e $\bar{y}^{[h]}$ é a saída estimada pela rede.

O procedimento de aprendizagem $eMLP$ é sumarizado abaixo:

INÍCIO
Definir ρ, η_1, η_2, $c = 0$;
Fazer (1)
  {Propagar $x^{[h]}$, $h = 1, 2, ...$, na rede;
   Se $h = 1$ //Primeiro par de dados
     Adicionar um neurônio ϕ^{c+1} associado a $y^{[h]}$; $c = c + 1$;
   Caso contrário
     {Se ($D^\nu > \rho^\nu$, $\nu = 1, ..., c$, ou o neurônio vencedor ϕ^ν
      não é associado a $y^{[h]}$, $h = 1, 2, ...$)
        Criar ϕ^{c+1} e associá-lo a $y^{[h]}$; $c = c + 1$;
      Caso contrário
        Atualizar W_{ij}^ν e W_{jk}^ν de ϕ^ν; } }
FIM

2.3 O Operador Agregação

A agregação de neurônios da camada evolutiva de redes eMLP é um mecanismo sugerido para manter controle sobre a quantidade de neurônios se desenvolvendo durante o processo de aprendizado. Basicamente, este operador agrega dois neurônios se desenvolvendo de forma similar em um único neurônio. O operador de agregação pode ser aplicado em todas as iterações do algoritmo de aprendizado ou após um certo número de iterações. Em geral, este operador melhora a capacidade de generalização da rede e possibilita manter uma quantidade razoável de informação ativa na memória.

Isto pode facilitar a interpretação da rede por parte de especialistas. O procedimento de agregação é sumarizado como segue:

INÍCIO
Fazer para ϕ^{ν}, $\nu = 1, ..., c$;
{Encontrar subconjuntos de neurônios $R \in N^m$ tais que as distâncias entre dois neurônios ϕ^{r1}, $\phi^{r2} \in R$, $D(W_{ij}^{r1}, W_{ij}^{r2})$ e $D(W_{jk}^{r1}, W_{jk}^{r2})$, sejam inferiores a um limiar W_{th};
Mesclar neurônios de R em ϕ^{c+1} com parâmetros:

$$W_{ij}^{c+1} = \frac{1}{m-1} \sum_{i=1}^{m-1} D(W_{ij}^{r_i}, W_{ij}^{r_{i+1}}),$$

$$W_{jk}^{c+1} = \frac{1}{m-1} \sum_{i=1}^{m-1} D(W_{jk}^{r_i}, W_{ik}^{r_{i+1}});$$

Excluir todos os neurônios em R; }
FIM

A agregação de neurônios é uma regularização importante que pode trazer melhorias de desempenho de uma rede eMLP em certas aplicações.

2.4 Método de Clusterização Evolutiva - ECM

ECM é um algoritmo incremental de um passo usado para clusterização dinâmica de dados onde não há um número pré-definido de clusters (Kasabov e Song, 2002). ECM se baseia na distância entre novos exemplos e os centros dos clusters existentes para evoluir o modelo do processo.

Inicialmente, ECM possui uma coleção de clusters vazia. O aprendizado se inicia após a chegada do primeiro dado de um fluxo. Vetores $x^{[h]}$, $h = 1, 2, ...$, são determinados mais próximos do centro de um cluster C^{ν} da coleção atual de clusters $C = [C^1, ..., C^{\nu}, ...C^c]$ a partir de alguma medida de distância D^{ν}. Caso esta distância seja maior que um limiar ρ^{ν} pré-definido, cria-se um novo cluster C^{c+1} aumentando a coleção de clusters existentes. O centro de C^{c+1} é denominado Cc^{c+1} e seu raio Rd^{c+1} é definido inicialmente como sendo zero. Por outro lado, caso D^{ν} seja menor que ρ^{ν} para algum $\nu \in [1, ..., c]$, então o centro Cc^{ν} e o raio Rd^{ν} são ajustados no sentido de deslocar o cluster no domínio de entrada e/ou aumentar seu raio de alcance. Qual e quanto um cluster será modificado, depende do exemplo atual e dos clusters existentes. O raio de um cluster Rd^{ν} não é mais atualizado quando ele atinge o limiar máximo de expansão ρ^{ν}.

A Fig. 1 ilustra um exemplo de clusterização ECM onde são modelados o fluxo de dados $x^{[h]}$, $h = 1, ..., 9$. Note que:

* $x^{[1]}$ causa a criação de C^1,
* $x^{[2]}$ atualiza C^1,
* $x^{[3]}$ produz C^2,
* $x^{[4]}$ é descartado,
* $x^{[5]}$ atualiza C^1,
* $x^{[6]}$ é descartado,
* $x^{[7]}$ causa o ajuste de C^2,
* $x^{[8]}$ gera C^3,
* $x^{[9]}$ causa o ajuste de C^1.

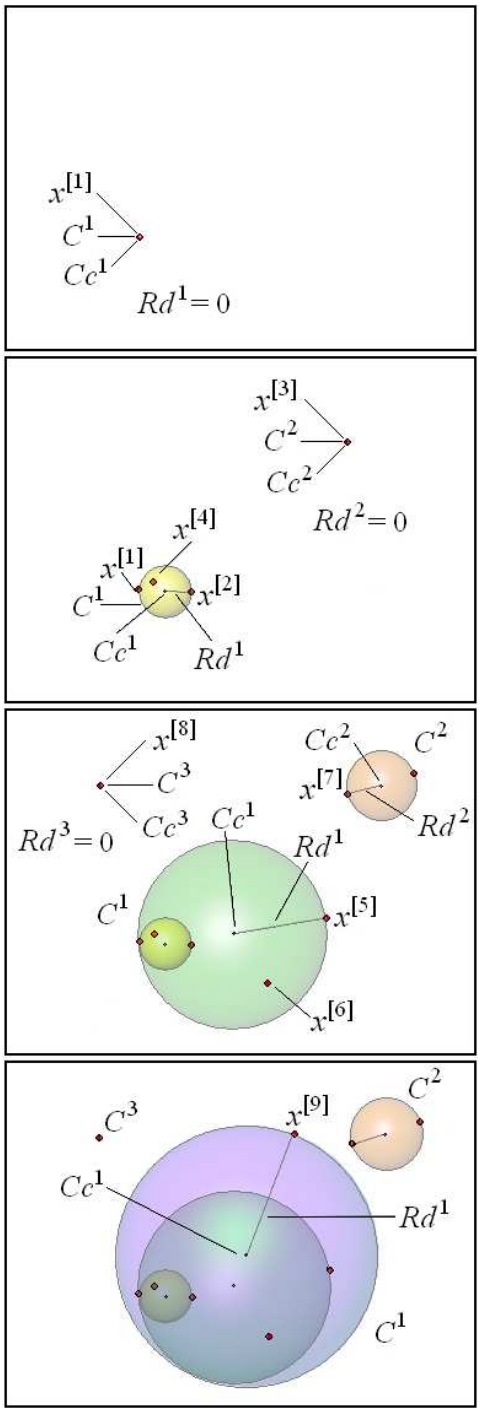


Figura 1: Processo de clusterização evolutiva

Em ECOS, os centros de clusters ECM são representados pelos pesos das conexões dos neurônios, enquanto os raios dos clusters são representados pelos limiares ρ^{ν} $\forall\nu$. O ECM é o princípio que rege o funcionamento de eSOM. Em eSOM, clusters são representados por neurônios no espaço de saída. Em contraste, neurônios eSOM possuem relação de vizinhança. Isto os oferecem parâmetros adicionais e flexibilidade ao aprendizado.

2.5 Mapa Auto-organizável Evolutivo

O modelo eSOM foi originalmente proposto em (Da Deng e Kasabov, 2000). Sugerimos neste artigo uma variação de algoritmo de aprendizado eSOM mais objetiva e efetiva em termos de tempo de processamento e aplicabilidade em tempo real.

eSOM desenvolve os princípios de mapeamento de SOM, porém nenhuma restrição topológica é estabelecida *a priori* ao aprendizado. eSOM consiste de duas camadas i (entrada) e j (saída), sendo a última a camada que evolui de acordo com o fluxo de dados. eSOM permite a evolução dos neurônios no espaço de dados original e ao mesmo tempo desenvolve e mantém uma representação topológica. Em contraste ao SOM, a vizinhança dos neurônios evoluídos não é pré-definida. Ela é definida durante o aprendizado de acordo com a distância entre os neurônios existentes.

Inicialmente admite-se um mapa nulo. Gradualmente, neurônios são gerados a partir de novas informações contidas no fluxo de dados. Caso um novo exemplo seja dissimilar a um certo grau dos neurônios existentes, um novo neurônio é criado. Por outro lado, caso o exemplo seja relacionado com um neurônio existente, então refina-se os parâmetros associados. Formalmente, a partir de um fluxo de dados $x^{[h]}$, $h = 1, 2, ...$, a ativação do neurônio ϕ^ν da coleção de neurônios existentes $\Phi = \{\phi^1, ..., \phi^\nu, ..., \phi^c\}$ é dada por:

$$A^\nu = e^{(\frac{-D(x^{[h]} - W_{ij}^\nu)^p}{\epsilon^p})},$$

onde ϵ é um raio em torno de W_{ij}^ν e p é a ordem da medida de distância D. A seguinte função para minimização do erro de aproximação estocástico *on-line* é admitida:

$$\zeta^\nu = \sum_{i=1}^{c} A^\nu D(x^{[h]} - W_{ij}^\nu),$$

onde c é o número de neurônios do $eSOM$ antes da chegada de $x^{[h]}$. Para minimizar a função acima, os vetores $W_{ij}^\nu \forall \nu$, são atualizados pelo método do gradiente descendente. Assumindo $p = 2$ temos:

$$\frac{\partial \zeta^\nu}{\partial W_{ij}^\nu} = A^\nu (W_{ij}^\nu - x^{[h]}) + D(x^{[h]} - W_{ij}^\nu)^2 \frac{\partial A^\nu}{\partial W_{ij}^\nu}.$$

Por praticidade, assumimos que a variação da ativação será menor a cada vez que os pesos forem ajustados. Assim, A^ν, $\nu = 1, ..., c$, são constantes e a seguinte relação para ajuste de pesos surge:

$$\Delta W_{ij}^\nu = \eta A^\nu (x^{[h]} - W_{ij}^\nu),$$

onde η é a taxa de aprendizado. A probabilidade de que um exemplo $x^{[h]}$ seja associado ao ν-ésimo neurônio W_{ij}^ν é dada por:

$$P_i(x^{[h]}, W_{ij}^\nu) = A^\nu (\sum_{\nu=1}^{c} A^\nu)^{-1}.$$

Durante o aprendizado, caso neurônios se tornem inativos por um longo tempo, estes são removidos do mapa. O procedimento de modelagem $eSOM$ é sumarizado como segue:

INÍCIO
Definir ρ, η, Δ, $c = 0$;
Fazer (1)
{Propagar $x^{[h]}$, $h = 1, 2, ...$, na rede;
Encontrar um conjunto de neurônios S que são próximos a $x^{[h]}$ a um grau pré-determinado ρ;
Se S é nulo
Criar um novo neurônio ϕ^{c+1} e conectá-lo aos dois vizinhos mais próximos, formando um conjunto S;
Calcular a ativação A^ν e ajustar W_{ij}^ν dos neurônios em S;
Recalcular as conexões entre neurônios a partir de alguma medida de distância;
Após $h + \Delta$ iterações, excluir as conexões mais fracas e/ou neurônios isolados; }
FIM

---

3 Detecção de Falta em Máquinas Elétricas

3.1 Descrição Básica do Problema

Atualmente, máquinas elétricas estão entre os mais importantes equipamentos industriais. Geralmente, elas são componentes críticos em processos de automação industrial requerendo certos níveis de confiabilidade, segurança, eficiência e performance. Por estas razões, questões relacionadas à proteção destas máquinas contra faltas têm recebido considerável atenção por engenheiros de manutenção e especialistas.

Sistemas de detecção de falta têm sido desenvolvidos e aperfeiçoados para o propósito de monitoramento das condições de operação de máquinas elétricas (Leite et al., 2007) (Nandi e Toliyat, 2005). Estes sistemas podem trazer diversos benefícios às industrias relativos à redução de períodos de parada e de desconexões desnecessárias da planta, agendamento da disponibilidade de pessoal para manutenção, redução de custos de reparo e de peças em estoque, minimização de perdas em produção, entre outros.

Em particular, um dos tipos de faltas mais comuns em motores de indução é o curto-circuito entre espiras dos enrolamentos do estator. Esta falta primária acontece após o rompimento do isolamento entre espiras. Após a falta, o processo de degradação da máquina se intensifica e falhas mais danosas podem surgir, resultando em danos irreversíveis ao motor. Entretanto, se a falta de espira para espira for detectada por sistemas de diagnóstico em estagio incipiente, por exemplo, o enrolamento da fase faltosa pode ser substituído, reduzindo significantemente perdas financeiras e aumentando a segurança operacional.

3.2 Características da base de dados

∗ Número de exemplos (observações): 3500;
∗ Número de atributos: 13 variáveis de entrada quantitativas denominadas tensões abc, correntes

Tabela 1: Detecções corretas de falta em máquinas elétricas

Modelo	No. protótipos	Épocas de treinamento	Tempo requerido (s)	% Precisão (Melhor)	% Precisão (Média)
MLP	[13-35-35-3]	10000	821,7	94,8	92,3
Elman	[13-35-35-3]	10000	1280,5	94,8	93,6
SOM	[13-70]	5000	350,0	86,1	81,9
FCM	80	56	16,8	75,0	72,7
$eMLP_1$	54	1	0,36	98,5	96,3
$eMLP_2$	69	1	0,50	99,1	96,6
$eSOM_1$	43	1	0,20	97,8	95,1
$eSOM_2$	60	1	0,26	98,4	96,3

abc do estator, defasamentos tensão/corrente abc, potência ativa, deslize do rotor, valor absoluto da corrente de seqüência negativa, e ângulo da corrente de seqüência positiva; e 3 variáveis de saída nomeadas porcentagem de espiras em curto-circuito nos enrolamentos abc do estator;
$*$ Dados de treinamento/teste: 60/100%.

Para comparação de performance considerou-se os seguintes modelos: redes neurais Perceptron Multi-Camadas (MLP) e Elman, Mapa Auto-Organizável (SOM), Fuzzy C-Means (FCM), além de eMLP e eSOM.

3.3 Resultados

Para avaliar o resultado de diferentes parametrizações, as abordagens eMLP e eSOM consideraram dois limiares de alcance dos neurônios que são $\rho_1 = 0.3$ e $\rho_2 = 0.2$. Cada experimento foi realizado cinco vezes com os mesmos parâmetros. Em cada experimento, os dados foram misturados para induzir diferentes ordens no fluxo de dados. Os dados foram apresentados aos modelos seqüencialmente apenas uma vez para emular um fluxo. eMLP e eSOM começam a aprender com uma base de regras vazia e sem pré-treinamento. Isto é feito para simular um processo evolutivo real. A Tabela 1 mostra o desempenho dos modelos considerados.

Os resultados da Tabela 1 mostram que o modelo eMLP sugerido foi o mais preciso para a classificação de padrões de falta em máquinas elétricas. Usando uma estrutura compacta, 43 protótipos, o modelo eSOM proposto atingiu a performance de $97,8\%$. eSOM se apresentou como a alternativa mais rápida e com menos requerimentos de memória para a modelagem de processos evolutivos. Note que eMLP e eSOM usam algoritmos de um passo, o que naturalmente apela por suas aplicações na resolução de problemas em tempo real que demandam adaptabilidade incremental.

4 Conclusão

Variações evolutivas de algoritmos de aprendizado de sistemas conexionistas foram sugeridas. O aprendizado evolutivo provê redes neurais com habilidade de modelagem de processos adaptativos em tempo real a partir de fluxos de dados e algoritmos incrementais. Experimentos em detecção de falta em máquinas elétricas têm confirmado a efetividade da abordagem evolutiva quando comparada a técnicas não-lineares de aprendizado de máquina. Trabalhos futuros incluem o desenvolvimento de algoritmos evolutivos para induzir o aprendizado de modelos baseados em novas estruturas conexionistas e em grânulos de informação. Aplicações em séries temporais e controle são também direções de desenvolvimento.

Agradecimento

O primeiro autor agradece a CAPES pelo apoio financeiro. O segundo autor é grato a CEMIG pelo suporte P&D178. O último autor agradece ao CNPq pelo suporte 304857/2006-8.

Referências

Bouchachia, A.; Gabrys, B.; Sahel, Z. (2007). "Overview of some incremental learning algorithms". *IEEE Fuzzy Systems Conf.*, pp: 1-6.

Da Deng; Kasabov, N. (2000). "ESOM: an algorithm to evolve self-organizing maps from online data streams". *IEEE International Joint Conference on Neural Networks*, Vol. 6, pp: 3-8.

Deza, M.; Deza, E. (2006). Dictionary of Distances. Elsevier Science, 1^a edição, pp: 412p.

Fayyad, U.; Uthurusamy, R. (1996). "Data mining and knowledge discovery in databases". *Communications ACM*, Vol. 39, pp: 24-27.

Inmon, W. H. (1996). "The data warehouse and data mining". *Communications ACM*, Vol. 39, pp: 49-50.

Kasabov, N.; Song, Q. (2002). "DENFIS: Dynamic evolving neuro-fuzzy inference system and its application for time-series prediction". *IEEE Tran. on Fuzzy Systems*, Vol. 10-2, pp: 144-154.

Kasabov, N. (2003). Evolving Connectionist Systems. Springer Verlag, Londres, 1^a ed. pp: 307p.

Kasabov, N. (2007). Evolving Connectionist Systems: The Knowledge Engineering Approach. Springer Verlag, Londres, 2^a edição, pp: 465p.

Leite, D. F.; et al. (2007). "Real-Time Model-Based Fault Detection and Diagnosis for Alternators and Induction Motors". *IEEE Electric Machines and Drives Conf.*, Vol. 1, pp: 202-207.

Mitra, S.; Pal, S. K.; Mitra, P. (2002). "Data mining in soft computing framework: a survey". *IEEE Tran. on Neural Nets.*, Vol. 13-1, pp: 3-14.

Nandi, S.; Toliyat, A. (2005). "Condition Monitoring and Fault Diagnosis of Electrical Motors - A Review". *IEEE Transaction on Energy Conversion*, Vol. 20, No. 4, pp: 719-729

AUTONOMOUS EVOLUTION OF HARDWARE USING AN EVOLUTIONARY ALGORITHM

CELSO DE LA CRUZ CASAÑO AND TEODIANO FREIRE BASTOS

Electrical Engineering Department, Universidade Federal do Espírito Santo, Vitória, Brasil
Av. Fernando Ferrari, 514, 29075-910, Vitória-ES, BRASIL
E-mails: celsodelacruz@gmail.com, tfbastos@ele.ufes.br

HÉCTOR D. PATIÑO AND RICARDO CARELLI

Instituto de Automática, Universidad Nacional de San Juan,
1109 San Martín Oeste, San Juan, Argentina
E-mails: dpatino@inaut.unsj.edu.ar, rcarelli@inaut.unsj.edu.ar

Abstract— This work presents the analysis of the evolution in nature of species. From this analysis, some evolution properties are extracted which are considered to design an evolutionary algorithm for evolvable hardware. These properties reduce the risk of malfunctions in a physical system when it is evolving. This algorithm is designed specially to evolve hardware, i.e., through the use of evolutionary computation methods, the electrical and mechanical hardware systems are automatically designed, adapted, and reconfigured. An experiment was carried out in order to evaluate the performance of the evolutionary algorithm.

Keywords— Intelligent systems, evolutionary algorithms, evolvable hardware, robot systems, optimization methods, species evolution.

Resumo— Este artigo apresenta a análise da evolução de espécies na natureza. A partir desta análise, algumas propriedades evolutivas são extraídas, as quais são consideradas no projeto de um algoritmo evolutivo para evolucionar hardware. Estas propriedades reduzem o risco de mau funcionamento do sistema físico quando evoluciona. Este algoritmo é desenvolvido principalmente para evolucionar hardware, o que significa que, mediante o uso de métodos computacionais evolutivos, os sistemas físicos elétricos e mecânicos são automaticamente projetados, adaptados e reconfigurados. Foi realizado um experimento para avaliar o desempenho do algoritmo evolutivo.

Palavras-chave— Sistemas inteligentes, algoritmos evolutivos, hardware evolutivo, sistemas robóticos, métodos de optimização, evolução de espécies.

1. INTRODUCTION

The evolution process experienced by a species in nature turns their members more efficient in regard to survival in a changing and complex environment. These processes can be compared to the minimization (or maximization) of a cost function in a mathematical optimization problem. Numerous works on evolutionary algorithms have been presented that are based on the theory of natural evolution, from which the most widely known ones are: Evolution Strategies (ES) [Rechenberg, 1973; Schwefel, 1981], Evolutionary Programming (EP) [Fogel et al., 1966] and Genetic Algorithm (GA) [Holland, 1975].

Evolvable hardware [Lohn and Hornby, 2006] is an exciting and emerging field. It lies at the intersection of evolutionary computation and physical design. Through the use of evolutionary computation methods, the electrical and mechanical hardware systems are automatically designed, adapted, and reconfigured. There is plenty of research in this field. For example, in [Thompson, 1997] a field programmable gate array was evolved to discriminate a tone; in [Lipson and Pollack, 2000] an evolutionary system to automatically design both the robot morphology and controller was developed; in [Hornby, 2005] an autonomous evolution of a dynamic gait on Sony's entertainment robot was developed; and in [Cai et al., 2008] a normalized steady state genetic algorithm is proposed and used to design a ST5 antenna (X-band antenna for NASA´s Space Technology 5 spacecraft).

The present work considers the fact that almost all animals of a species in nature are able to survive and only a few animals die because of malfunctions in their organisms. The aim of this work is to obtain similar properties when evolving hardware. On the basis of the properties of the species in nature, a new computational evolutionary algorithm is developed. The proposed algorithm is characterized by gradual changes in the mathematical representation of the species at the time of evolution. In addition, this algorithm is

also characterized by showing a bounded mathematical representation of the species within a parameters space. The last two properties do not help the evolution to escape from a local minimum. However, they give good convergence properties and are good for evolvable hardware, because they can make the risk of malfunctions of the real system null during the evolution. An experiment was performed using a unicycle-like mobile robot which has a tracking control system. This experiment consisted of letting some control parameters (dynamic model parameters of the robot) of the control system evolve by themselves while it executes a repetitive task.

2. EVOLUTION OF A SPECIES

2.1 Evolution in nature

The animals of a species in nature share similar characteristics. If these characteristics could be put in a vector space, a set of the animal characteristic vectors of the species would be grouped in a zone. In many cases, the evolution of a species in nature is gradual [Darwin, 1995] and it would take many generations for a noticeable change to take place.

From the sexual selection studied in [Darwin, 1995], the following can be concluded: the number of children in a group of parents will show greater probability of having a large value when those parents are more similar in characteristics to the fittest parent. Conversely, the number of children in a group of parents will have greater probability of having a low value when those parents differ too much in characteristics from the fittest parent, and/or differ too much in characteristics from each other. The last analysis considers the difficulty in bearing children between two biologically different animals (different with respect to their characteristics). Indeed, they could be incompatible to have descendants and, consequently, will have no descendant with probability 1. The mutants, deformed or mutilated children have extremely different features from the animals of their own species, which makes them most unlikely to have children. Even if these animals were capable of reproducing, they would most likely be rejected for reproduction by the members of their own species.

2.2 Evolution of a physical system

Let us consider three things: 1) an individual is a physical system (a robot, an industrial plant, etc.), 2) the species' individuals have similar characteristics, and 3) the evolution of this species is gradual. Then, the following analysis is done. If the physical system (individual) of a species works well, then the physical systems of the same species sharing similar characteristics will work well too with high probability. Hence, the risk that the physical system breaks down in an evolutionary process is very low or null. The eliminated individuals are those that do not work as well as the rest of the group; those individuals eliminated do not necessarily work badly. There is a great probability that some children will work better than their parents, consequently, the species will have a tendency to a minimum of the cost function.

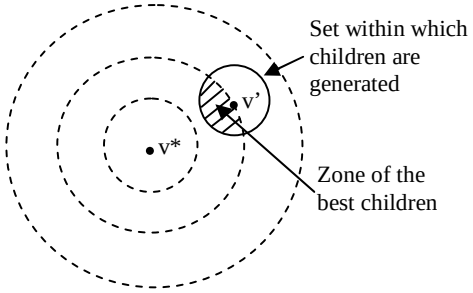


Fig. 1. Way to evolve a species. The points v' and v* are, respectively, the vector of characteristics of the fittest individual and the optimal individual.

2.3 Example of evolution

For example, let us consider an evolution process with a convex function $F(x)=\|x\|^2$, and a uniform random distribution in the mutation operator. In this example, if the children dispersion is small enough, the probability that the fittest individual creates a child with less cost than that of itself is in a range close to 0.5 (see Fig. 1).

3. DEFINITIONS FORMULATED TO BE USED IN THIS WORK

1) *Species dispersion:* A term that refers to the location, in a vector space, of species' individuals related to a central value.

2) *Space S:* It is a normed space, whose elements or points are characteristic vectors of individuals. A normed space is a vector space with a norm defined on it (to delve into this subject refer to [Kreyszig, 1978]).

3) *Cost functional F and cost of an individual*: cost functional F is a functional with domain in the space S and range in the positive real numbers; thus, $F:S\rightarrow R^+$. This function assigns a cost to each individual.

4) *Evolution stage or generation*: It is the average time interval between the creation of parents and the creation of their children.

4. EVOLUTIONARY ALGORITHM (μ,1+λ)-E1E

In the evolutionary algorithm (μ,1+λ)-E1E, the term (μ,1+λ) means that the μ parents will have λ children. Then, the λ children and the fittest of the μ parents will be evaluated in order to obtain μ individuals with the smallest cost values to form the new population of the species. E1E indicates the evolution of only one species.

In the algorithm (μ,1+λ)-E1E, it should be complied with (λ+1)≥μ, μ≥2 and λ≥2. The first condition is aimed at ensuring that the new population has μ individuals. The second and third ones are necessary for self-adjustment of the species' dispersion, which will be covered further below in this section.

4.1 Algorithm (μ,1+λ)-E1E

Let the function $d(.,.)$ be the metric induced by the norm of space S (for example $d(x,y)=\|x-y\|_\infty$ [Kreyszig, 1978]), and let the closed ball $\tilde{B}(.,.)$ be defined as $\tilde{B}(x,r)=\{y\in S : d(y,x)\le r\}$. Let us consider that the individuals $a_i^n,\ i=1,2,...,\mu$ form the vector

$$a^n=\left(a_1^n,a_2^n,...,a_\mu^n\right)\in S^\mu,$$

and v'^n is an element of a^n such that

$$F\left(v'^n\right)=\min_i\left(F\left(a_i^n\right)\right).$$

The procedure of the algorithm is as follows:

1. Initialize the first individual a_1^0, then initialize a^0 such that $a_i^0\in\tilde{B}\left(a_1^0;r/c_o\right);\ i=2,..,\mu.$
2. Determine the fittest individual v'^0 and initialize $n=1$.
3. While the finalization criteria are not yet met, perform the following:
   a. Select $\bar{\mu}^{n-1}$ parents. These selected parents are the elements of a^{n-1} contained by the closed ball $\tilde{B}(v'^{n-1},r)$.
   b. Modify the order of a^{n-1} such that:
   $a_i^{n-1}\in\tilde{B}\left(v'^{n-1},r\right)$, for $i=1,...,\bar{\mu}^{n-1}$; and
   $$F\left(a_1^{n-1}\right)\le F\left(a_2^{n-1}\right)\le...\le F\left(a_{\bar{\mu}^{n-1}}^{n-1}\right)$$
   c. If $\bar{\mu}^{n-1}=1$, create the children in the following way:
   $$Z_{1k}^n=v'^{n-1}+X_{1k}^n r/c_o$$
   where $1\le k\le\lambda$ and $X_{jk}^n\in S$ is a random variable having uniform probability on the closed ball $\tilde{B}(0,1)$, and r and c_0 are real constants (algorithm parameters).
   If $\bar{\mu}^{n-1}\ge 2$, create the children in the following way:
   $$Z_{jk}^n=v'^{n-1}+X_{jk}^n d\left(a_j^{n-1},v'^{n-1}\right)/c_o$$
   where $j\ge 2$, and Z_{jk}^n is the k-th child created using the first and j-th selected parent. The number of children created by the first and j-th selected parent is calculated as follows:
   $$\bar{\lambda}_j^n=\begin{cases}\lambda-fix\left(\dfrac{\lambda}{\bar{\mu}^{n-1}-1}\right)\left(\bar{\mu}^{n-1}-2\right) & \text{if } j=2\\ fix\left(\dfrac{\lambda}{\bar{\mu}^{n-1}-1}\right) & \text{if } j=3,4,..,\bar{\mu}^{n-1}\end{cases}$$
   where $fix(.)$ is a function that returns the integer part of its argument.
   d. Form the vector b^n with all the created children in the following way:
   $$b^n=\left(b_1^n,...,b_l^n,...,b_\lambda^n\right)\in S^\lambda;\ b_l^n=Z_{jk}^n\in S$$
   e. Calculate a^n:
   $$a^n=T_2^n\left(a^{n-1},b^n\right).$$
   f. Increment $n=n+1$.

Mapping $T_2^n:S^\mu\to S^{\mu+\lambda}$ evaluates v'^{n-1} and the elements from b^n, selects μ characteristic vectors which give the smallest cost values and forms a vector with the μ selected vectors.

Possible spaces S can be: a real vector space with 1-norm; a real vector space with 2-norm, or a real vector space with ∞-norm.

4.2 Self-adaptation of the species dispersion

In the creation of children, the distance $d\left(a_j^{n-1},v'^{n-1}\right)$ is used, which is the distance between v'^{n-1} and another element of a^{n-1}. Constant c_0 can be taken as $c_o=\sqrt[N]{P_{co}}$ if considering S equal to the real vector space of dimension N with ∞-norm. This constant makes $\left(X_{jk}^n/c_o\right)\in\tilde{B}(0;1)$ be complied with a probability P_{co}. Therefore, it is expected that $100\times P_{co}$% of the children will be less dispersed than the parents, and the remaining children will be more dispersed than the parents. The dispersions are measured with respect to v'^{n-1}. The self-adaptation of the species dispersion can be controlled by the parameter P_{co}.

Choosing Pco=0.5, the dispersion of the species tends to increase when such species is relatively far from a minimum of the costs function (far with respect to their distance on space S). The reason is that the most distanced children to the fittest one of their parents will have greater probabilities of having the smallest cost of the species than the least distanced children. Figure 2 shows this fact when using the same example of section 2.3.

Conversely, choosing again Pco=0.5, the dispersion of the species tends to decrease when such species is relatively close to a minimum of the costs function, because, now, the least distanced children to the fittest one of their parents will have greater probabilities of having a smaller cost than the most distanced children. Figure 3 shows this fact when using the same example of section 2.3 again.

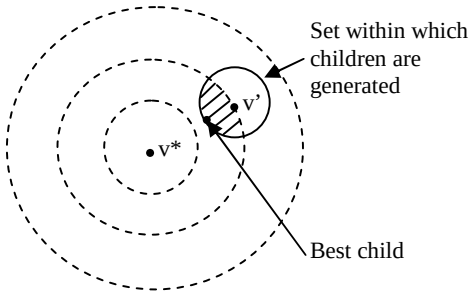


Fig. 2. Best child when the species is relatively far from the minimum point. The points v' and v* are, respectively, the vector of characteristics of the fittest individual and the optimal individual.

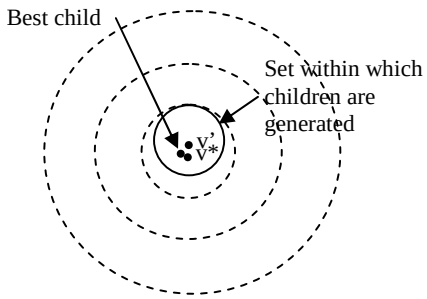


Fig. 3. Best child when the species is relatively close to the minimum point. The points v' and v* are, respectively, the vector of characteristics of the fittest individual and the optimal individual.

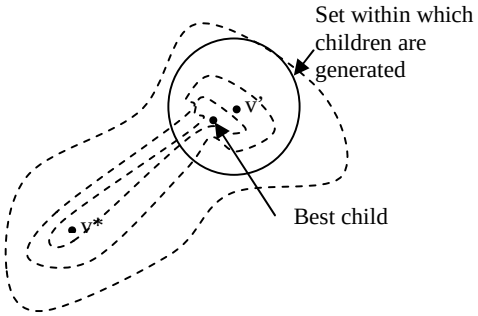


Fig. 4. Special case of self-adaptation. The points v' and v* are, respectively, the vector of characteristics of the fittest individual and the optimal individual.

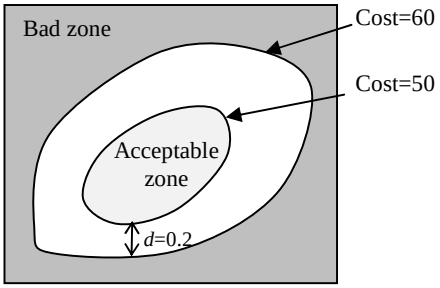


Fig. 5. Graph of Acceptable and Bad values of the characteristic vectors of individuals. Here *d* is the minimum distance between points of acceptable and bad zones.

Only some special cases do not have these properties. One example is illustrated in Fig. 4. What happens in this example is a decrease of the species dispersion while it goes through the strait zone.

This self-adaptation of the dispersion of a species increases the convergence rate towards a minimum of the costs function. The reason lies in the fact that the species will be concentrated close to a minimum point of the costs function and, therefore, increasing the probability of creating children that may be closer still to such minimum point.

4.3 Making null the risk to break down a physical system

Let us consider two zones in space *S*: The first one, the *Bad Zone*, whose points are characteristic vectors of individuals prone to collapse (for example points that have costs equal or greater than 60 in Fig. 5). This means that the physical system works very badly or that it has a high risk of breaking down. The second, the *Acceptable Zone*, whose points are characteristic vectors of individuals that acceptably perform their task (for example points that have costs equal or less than 50 in Fig. 5).

The dispersion of parents that will have children is limited by the closed ball $\widetilde{B}(v'^{n-1},r)$. Considering this fact, the vector of characteristics of the children will be placed within the closed ball $\widetilde{B}(v'^{n-1};r/c_o)$, on account of the nature of the probability density function with which the children are created.

The aim here is to avoid the characteristic vector of a child getting into the Bad Zone. Algorithm (μ,1+λ)-E1E accepts only a vector v'^{n} with equal or less cost than that of v'^{n-1}. Let us consider that algorithm (μ,1+λ)-E1E begins with a_1^0 within the Acceptable Zone. Every vector v'^{n} with $n\geq 0$ is equal to a_1^0 or equal to a vector with a smaller cost; therefore, v'^{n} is in the acceptable zone $\forall n\geq 0$. Then, if r/c_o has low enough value, the vector of characteristics of a child will never get into the bad zone. For example, let us consider $r/c_o=0.1$, then, the distance between the characteristic vector of a child and v'^{n-1} is smaller than or equal to 0.1. If we consider Fig. 5 in this example, the characteristic vector of any child will never get into the bad zone; because, to happen, the distance between the vector of characteristics of the child and v'^{n-1} has to be equal or greater than 0.2. Therefore, the constant *r* has to be defined with a value that makes null the risk to break down a physical system. The value of *r* can be defined based on simulation results. If the mathematical model does not accurately describe the real system, a smaller value of *r* than that defined by simulations can be used.

In Sections 4.2 and 4.3, the properties of algorithm (μ,1+λ)-E1E are reviewed. These properties improve the performance of the evolutionary algorithm (i.e., good convergence rate and better chances of implementing it on evolvable hardware, as will be seen in the experiments). However, these properties can make the evolution converge to a sub-optimal solution. For numerous engineering problems, especially to evolve hardware with security, a sub-optimal solution will be an acceptable solution whenever that solution improves the performance of the system.

5. EXPERIMENT

Let us consider as an individual a mobile robot Pioneer with a tracking control. The elements of the characteristic vector of this individual are the estimated parameters of the dynamic model of the robot. These estimated parameters are used to compensate the non-linearities in the control system. Better estimated parameters imply better control performance. The problem here can be formulated as to obtain the estimated parameters that give the least control

errors. Although, there is no evolution of circuits or mechanical parts, it is considered that the physical brain of the robot is evolving according to the changes of the physical environment.

The tracking control of the robot system is obtained from [De La Cruz et al., 2008]. This tracking control is:

$$\nu_{ref} = \hat{D} M (\upsilon - N) + T_a \hat{\theta}$$

$$\upsilon = \ddot{h}_d + K_1 \dot{\tilde{h}} + K_2 \tilde{h}, \qquad \tilde{h} = h_d - h,$$

where:

$$\nu_{ref} = \begin{bmatrix} u_{ref} \\ \omega_{ref} \end{bmatrix}, \quad \hat{D} = \begin{bmatrix} \hat{\theta}_1 & 0 \\ 0 & \hat{\theta}_2 \end{bmatrix},$$

$$M = \begin{bmatrix} \cos\psi & \sin\psi \\ -\frac{1}{a}\sin\psi & \frac{1}{a}\cos\psi \end{bmatrix},$$

$$N = \begin{bmatrix} -u\omega\sin\psi - a\omega^2\cos\psi \\ u\omega\cos\psi - a\omega^2\sin\psi \end{bmatrix}, \quad h = \begin{bmatrix} x \\ y \end{bmatrix},$$

$$T_a = \begin{bmatrix} 0 & 0 & -\omega^2 & u & 0 & 0 \\ 0 & 0 & 0 & 0 & u\omega & \omega \end{bmatrix},$$

$$\hat{\theta} = \begin{bmatrix} \hat{\theta}_1 & \hat{\theta}_2 & \dots & \hat{\theta}_6 \end{bmatrix}^{\mathrm{T}},$$

$$K_1 = \begin{bmatrix} 2\sqrt{0.5} & 0 \\ 0 & 2\sqrt{0.5} \end{bmatrix}, \quad K_2 = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix},$$

$h_d(t)$ defines the desired trajectory, u and ω are the linear and angular velocity, u_{ref} and ω_{ref} are the linear and angular reference velocity (these references are sent to the robot), $\hat{\theta}_i$ is the i-th estimated parameter, and the rest of variables are shown in Fig. 6.

In the experiment, the cost of an individual is calculated by evaluating the performance of the individual executing a task. The task considered in this experiment is that the robot must follow a circular trajectory of 0.5m of radius with a velocity of 0.4m/s. The cost function that evaluates the performance of the tracking control is:

$$F(\hat{\theta}) = \int_{t_1}^{t_2} \sqrt{\left(\tilde{h}^{\mathrm{T}}\tilde{h} + \dot{\tilde{h}}^{\mathrm{T}}\dot{\tilde{h}}\right)} dt$$

where t_1 and t_2 are the initial and final time instant of the evaluation of one individual. In the experiment $t_2 - t_1 = 4$ seconds.

The initial model parameters of the mobile robot, obtained by an identification method, are [De La Cruz and Carelli, 2008]:

$$\hat{\theta} = \begin{bmatrix} 0.3037 & 0.2768 & -0.0004 & 0.9835 & 0.0038 & 1.0725 \end{bmatrix}^{\mathrm{T}}$$

One can observe that some parameters have greater values than others. Therefore, a normalized vector of parameters is considered as an individual. The space S is $\mathbf{R}^6$ with ∞-norm.

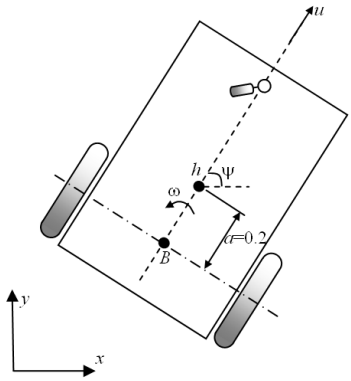


Fig. 6. Schematic of the mobile robot.

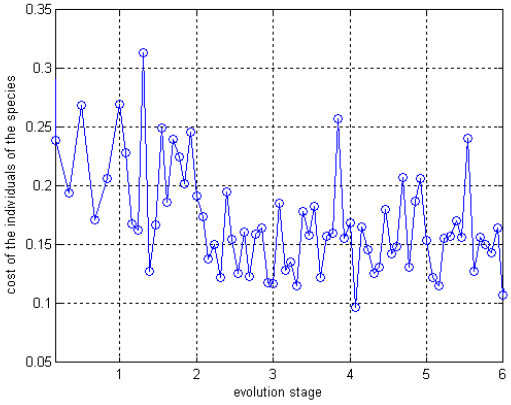


Fig. 7. Experimental result: cost of all individuals of the species per evolution stage.

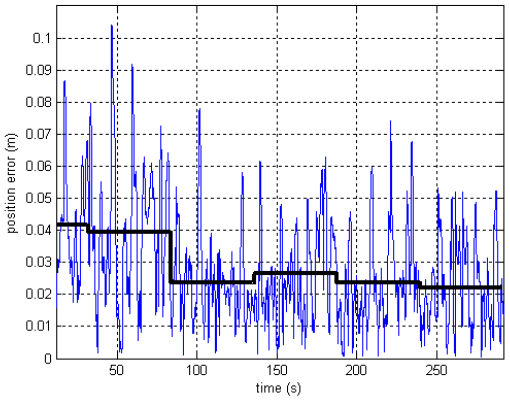


Fig. 8. Position error of the tracking control during evolution. The thick line represents the mean of the error in every evolution stage.

The parameters of the evolutionary algorithm (μ,1+λ)-E1E are:

r=0.2, μ=6, λ=12, $c_o = \sqrt{P_{co}}$, P_{co}=0.5.

The value of r was chosen using the experiences that the authors have in this kind of tracking control.

In Fig. 7 the cost of all individuals of the species per evolution stage is shown. One can see that the mean of the

individuals' cost of the species tends to decrease and the cost variance present a limited behavior. This result demonstrates that the physical system works better and better during the evolution. Figure 8 shows the position error of the tracking control during the evolution. Notably, the mean of the error in every evolution stage tends to decrease. Finally, figure 9 presents the variation of the estimated parameters in time. A tendency to some value can be observed specially in 3-th and 6-th parameters.

It is important in the experiment that function *F* only depends on the estimated parameters. This is difficult to reach at, because *F* depends on the initial position, the random uncertainties, and other factors. However, there is no problem if *F* varies slowly through time.

Remark: This experiment was performed only to demonstrate the functionality of the evolutionary algorithm (μ,1+λ)-E1E in a real situation. Better results (faster adaptation and less control errors) can be obtained using adaptive control and stability theories [De La Cruz et. al., 2008]. However, these theories can not be applied when a simple and exact enough model of the real system is not available. In contrast, to apply the evolutionary algorithm presented in this work, only an approximated model of the real system is required.

6. CONCLUSIONS

The evolutionary algorithm (μ,1+λ)-E1E differs from the other algorithms developed so far on the fact that it shows good properties for hardware evolution. This is so because it can be guaranteed that the algorithm will not lead the system, when evaluating the costs function, to instability or to undesirable states. Another property of the evolutionary algorithm (μ,1+λ)-E1E is that the self-adaptation of the species dispersion concentrates such species around the solution for tuning up the results. The experimental results show that a real system can evolve without any risk of work badly while it is executing a repetitive task. It is important that the system executes a repetitive task because function F(.) must depend only on the characteristic vector of an individual.

Future work will analyze evolving hardware with function F(.) perturbed by random uncertainties. Besides, future work will analyze the use of multiple species in order to obtain a better search of the global minimum.

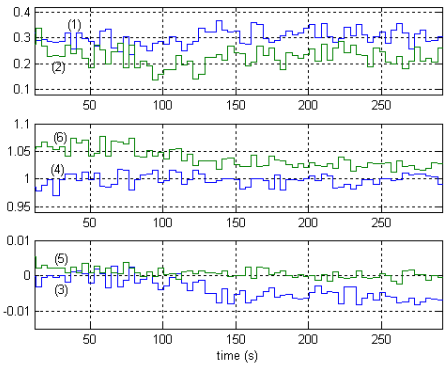


Fig. 9. Estimated parameters of the dynamic model during evolution.

7. ACKNOWLEDGMENTS

The authors would like to thanks to FAPES-Brazil for financing the project: Adaptive Dynamic Controller Applied to a Robotic Wheelchair Commanded by Brain Signals, process 39385183/2007. This work was also supported by the National Council for Scientific and Technical Research (CONICET), the National Agency for Scientific and Technological Promotion (ANPCyT) under grants PICT/04 No. 21592, PICT/05 Start-up No. 35398, and National University of San Juan under grants 21/I 842-2007.

8. REFERENCES

[1] Cai, Z., Zeng, S., Yang, Y. and Kang, L. (2008), "Automated antenna design using normalized steady state genetic algorithm," *NASA/ESA Conference on Adaptive Hardware and Systems*, June, pp. 125-132.

[2] Darwin, C. (1995), *The Origin of Species*, Gramercy.

[3] De La Cruz, C. and Carelli, R. (2008), "Dynamic model based formation control and obstacle avoidance of multi-robot systems", *Robotica*, vol. 26, pp. 345-356.

[4] De La Cruz, C., Carelli, R. and Bastos, T.F. (2008), "Switching Adaptive Control of Mobile Robots", *IEEE International Symposium on Industrial Electronics – ISIE08*, Cambridge, UK, July.

[5] Fogel, L.J., Owens, A.J. and Walsh, M.J. (1966), *Artificial Intelligence through Simulated Evolution*, New York, John Wiley.

[6] Holland, J.H. (1975), *Adaptation in natural and artificial systems*, Ann Arbor, The University of Michigan Press.

[7] Hornby, G.S., Takamura, S., Yamamoto, T. and Fujita, M. (2005), "Autonomous evolution of dynamic gaits with two quadruped robots," *IEEE Transactions on Robotics*, vol. 21, no 3, pp. 402-410.

[8] Kreyszig, E. (1978), *Introductory Functional Analysis with Applications*, New York, John Wiley & Sons.

[9] Lipson, H. and Pollack, J.B. (2000), "Automatic design and manufacture of robotic lifeforms," *Nature*, vol. 406, pp. 974-978.

[10] Lohn, J.D. and Hornby, G.S. (2006), "Evolvable Hardware: Using Evolutionary Computation to Design and Optimize Hardware Systems," *IEEE Computational Intelligence*, vol. 1, no. 1, February, pp. 19-27.

[11] Rechenberg, I. (1973), *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Stuttgart, Frommann-Holzboog Verlag.

[12] Schwefel, H.P. (1981), *Numérical optimization of computer models*, Chichester, Wiley.

[13] Thompson, A. (1997), "An evolved circuit, intrinsic in silicon, entwined with physics," *1st International Conference on Evolvable Systems 1996*, Springer Verlag.

Tuning a Multivariable Predictive Control of a Hot Rolling Mill With Genetic Algorithm

G. M. Almeida, J.L. Salles, J. Denti Filho and F. Rossomando
Universidade Federal do Espirito Santo
Laboratorio de Controle e Instrumentação
UFES-CT2 CEP 29060-970

Abstract

Multivariable Model Predictive Control algorithm is a powerful control design method widely applied to industrial processes. However, there is not easy way to tune the design parameters in order to get good output time response. In this work, a genetic algorithm is used to find the optimal values for these parameters and is applied in the real process of a hot rolling mill. Results of the controlled model are tested by simulation and compared with real data obtained from an industrial process controlled by a conventional technique.

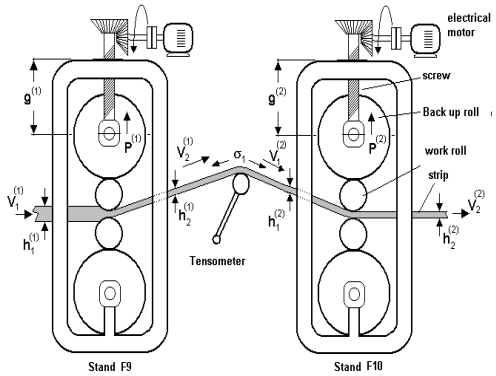


Figure 1. Rolling mill stands escheme

1. Introduction

The rolling mill process consists of introducing a strip metal under a set of rotating rolls that cause a permanent deformation on the strip, that is called thickness reduction. The stands and rotary rolls are the machines that make the rolling mill process. Figure 1 shows a scheme of two simple rolling mill stands with four rolls each and with thickness adjustment systems by screw positioner (see section 2 for information of the notation). The rolling mill stands are constituted by four rolls and two work rolls, which are in contact with the strip (the work rolls), and two back-up. The strip plate from the previous processes (flat product or a coil), is introduced in the gap of the work rolls, which is smaller than the thickness of the strip. This gap is determined by the set point of the screws that positions the rolls. These rolls drag the strip to the Stand roll bite, reducing its thickness. This strip has to leave the actual stand and enter in the next until the wished thickness is reached and is coiled in the output of the last Stand.

Our main motivation in this work is the reduction of the output thickness variations despite of the disturbances that enter in the process: strip temperature and input strip thickness.

For the considered system the control inputs are the gap variations on stands F9 ($g_{in}^{(1)}$) and F10 ($g_{in}^{(2)}$) and the rolls tip speed variation of the Stand F9 ($\Delta V_{in}^{(1)}$). The controlled outputs are the strip output thickness ($\Delta h_2^{(1)},\ \Delta h_2^{(2)}$), and the tension between the stands ($\Delta\sigma_1$). The main control objective is to maintain the strip output thickness variation (Δh_2) closer to zero, for any temperature or strip input thickness variation in the system.

This control problem is solved in the industry using the FFF control technique which is based in the detection of hardness fluctuations in the material that passes through the first finishing stand, sending this information to the set up model. In [1] and [2] was applied, respectively, the Linear Optimal Control and the Adaptive Neural Control techniques, and they showed that the performance of the control system is increased with respect to the FFF controller. However, they didn´t take account the restrictions on the inputs, states and outputs of the process as we can do in the Multivariable Generalized Predictive Control method (MGPC)[3]. According to [4], a successful controller for the process industries must maintain the system as close as possible to constraints without violating them, and this is

the main raison for the increasing applications of Predictive Control technics in the industries nowadays.
These note focus on the MPGC method to control the output thickness in a plate rolling mill process. Genetic Algorithm is used to tuning the parameters of MGPC which are: the control and prevision horizons, the controlled variable weights and the movie suppression coefficients. In fact, the trial and error method used to tuning these parameters are tedious for multivariable systems and doesn´t furnish good output responses (transient and steady state response). In other hand we can see in [5],[6],[7] that GA is an excellent framework for tuning the GPC parameters for SISO systems.

This paper is organized as follows: section 2 the model is presented;section 3 describes the classical formulation of MGPC; section 4 makes an overview of GA; the applications of GA for tuning the parameters is the subject in section 5; the section 6 presents the results of applications in the rolling mill process and we make the comparison between the MGPC and the FFF controllers with real data from the industry. Finally, section 7 presents the conclusions of the applications.

2 Process Modeling

The implemented mathematical model considers the last two rolling stands, since FFF control makes the thickness adjustment in the last ones. In our case we make the thickness adjustment by GPC technique, using the mathematical model expressed in the CARIMA model. The output thickness of each Stand is a function of the rolls gap and the stand streching

$$h_2^{(i)} = g^{(i)} + \frac{P^{(i)}}{K^{(i)}} \tag{1}$$

where:

- (i): Rolling mill stand i=1 (F9), i=2 (F10);
- $h_2^{(i)}$: Output strip thickness of the stand (i);
- $g^{(i)}$: Rolls gap of the stand (i);
- $P^{(i)}$: Roling mill force of the stand (i).
- $K^{(i)}$: Elasticity Modulus of the Stand (i).

The derived strip tension between the Stands is a function of the difference between the strip output and input speed on Stands F9 and F10:

$$\frac{d\sigma_1}{dt} = \frac{E}{L}(V_2^{(1)} - V_1^{(2)}) \tag{2}$$

where:

- σ_1 : Strip tension between stands F9 and F10;
- E : Strip Young modulus 21.000 N/m;
- L : distance between Stands (i). 5.486 m;
- $V_2^{(1)}$: output strip velocity of the Stand F9;
- $V_1^{(2)}$: input strip velocity of the Stand F10.

The volume continuity of the strip in the rolls gap is defined as:

$$V_1^{(i)} h_1^{(i)} = V_2(i) h_2^{(i)} \tag{3}$$

The volume continuity of the strip in the rolls gap is defined as:

$$P^{(i)} = P(h_1^{(i)}, h_2^{(i)}, \sigma_1^{(i)}, \sigma_2^{(i)}, S^{(i)}, \mu^{(i)}, T^{(i)}) \tag{4}$$

where:

- $\sigma_2^{(i)}$: output strip tension on the Stand (i);
- $\mu^{(i)}$: friction Coefficient on the Stand (i);
- $S^{(i)}$: Yield stress on the Stand (i).

Sliping function is defined like:

$$f^{(i)} = f(h_1^{(i)}, h_2^{(i)}, \sigma_1^{(i)}) \tag{5}$$

2.1 Basic Moldel Considerations

i) The main consideration is that the control is applied when a strip exists between both stands, this means, that the previous moment to the entrance of the strip in the last stand is not considered;

ii) We have considered the finite variations of the nominal values. For example variable x, being Δx the finite variation and $x*$ the nominal value;

iii) We have also considered that the rolling force is a function of the input thickness, output thickness, strip temperature and strip tension between Stands [8], [9]. The sensitivity curves show the rolling mill force variation:

$$\begin{aligned} \Delta P^{(1)} &= \beta_1 \Delta\sigma_1 + \beta_1 \Delta h_2^{(1)} + \beta_3 \Delta h_1^{(1)} + \beta_4 \Delta T^{(1)} \\ \Delta P^{(2)} &= \beta_5 \Delta\sigma_1 + \beta_6 \Delta h_2^{(2)} + \beta_7 \Delta h_1^{(2)} + \beta_8 \Delta T^{(2)} \end{aligned} \tag{6}$$

The coefficients of the linear functions are:

$$\begin{aligned} \beta_1 &= \frac{\partial P^{(1)}}{\partial \sigma_1};\ \beta_2 = \frac{\partial P^{(1)}}{\partial h_2^{(1)}};\ \beta_3 = \frac{\partial P^{(1)}}{\partial h_1^{(1)}};\ \beta_4 = \frac{\partial P^{(1)}}{\partial T^1} \\ \beta_5 &= \frac{\partial P^{(2)}}{\partial \sigma_1};\ \beta_6 = \frac{\partial P^{(2)}}{\partial h_2^{(2)}};\ \beta_7 = \frac{\partial P^{(2)}}{\partial h_1^{(2)}};\ \beta_8 = \frac{\partial P^{(2)}}{\partial T^2} \end{aligned} \tag{7}$$

iv) The sliding variation is a function of the input and the output thickness, and the strip tension which is represented by equations (8) and (9):

$$\Delta f^{(1)} = \alpha_1 \Delta\sigma_1 + \alpha_2 \Delta h_2^{(1)} + \alpha_3 \Delta h_1^{(1)} \quad (8)$$
$$\Delta f^{(2)} = \alpha_4 \Delta\sigma_1 + \alpha_5 \Delta h_2^{(2)} + \alpha_6 \Delta h_1^{(2)}$$

The coefficients are:

$$\alpha_1 = \frac{\partial f^{(1)}}{\partial \sigma_1};\ \alpha_2 = \frac{\partial f^{(1)}}{\partial h_2^{(1)}};\ \alpha_3 = \frac{\partial f^{(1)}}{\partial h_1^{(1)}}; \quad (9)$$
$$\alpha_4 = \frac{\partial f^{(2)}}{\partial \sigma_1};\ \alpha_5 = \frac{\partial f^{(2)}}{\partial h_2^{(2)}};\ \alpha_6 = \frac{\partial f^{(2)}}{\partial h_1^{(2)}};$$

v) The strip output speed variation is expressed by the sliding variation coefficients and by the roll tip speed:

$$\Delta V_2^{(1)} = (1 + f^{*(1)})\Delta V^{(1)} + V^{*(1)}\Delta f^{(1)} \quad (10)$$
$$\Delta V_2^{(2)} = (1 + f^{*(2)})\Delta V^{(2)} + V^{*(2)}\Delta f^{(2)}$$

vi) The controlled variables are not directly affected by the tip speed of the last Stand. Then we have $\Delta V_2 = 0$;

vii) Each actuator has its own dynamic that can be approximated by a first order system expressed as:

$$T_g = \frac{dg^{(i)}}{dt} = g_{in}^{(i)} - g^{(i)} \quad (11)$$

$$T_v = \frac{dV^{(i)}}{dt} = V_{in}^{(i)} - V^{(i)} \quad (12)$$

viii) The variables $\Delta h_1^{(1)}$ $\Delta T^{(1)}$ and $\Delta T^{(2)}$ will be considered as system disturbances;

ix) Considering the transport delay between F9 and F10, the output strip thickness variation on the Stand F9 is equal to the input strip thickness variation on the Stand F10. The delay time is equal to $Td = L/(V + \Delta V)$ then $\Delta h_2^{(1)} = \Delta h_1^{(2)}.exp(-s.Td)$. where **s** is the Laplace operator.

Discretizing the linear equations above we get the following multivariable CARIMA model:

$$\begin{aligned}\mathbf{A}_i(z^{-1}) &= \mathbf{B}^{i,1}u_1(t-1) + \mathbf{B}^{i,2}u_2(t-1) + \mathbf{B}^{i,3}u_3(t-1) \\ &+ \mathbf{D}^{i,1}d_1(t-1) + \mathbf{D}^{i,2}d_2(t-1) + \mathbf{D}^{i,3}d_3(t-1) \\ &+ \mathbf{D}^{i,4}d_4(t-1) + \frac{e(t)}{\Delta} \end{aligned} \quad (13)$$

Where $\Delta = 1 - z^{-1}$ and the polynomials $\mathbf{A}_i$, $\mathbf{B}^{i,k}$ and $\mathbf{D}^{i,l}$, $i = 1, 2, 3,$, $k = 1, 2, 3,$ and $l = 1, 2, 3, 4$ have na_i and $nb_{i,k}$ and $nd_{i,l}$ order respectively, and are obtained from the discretized matricial transfer function; $e(t)$ is a random noise with zero mean; the inputs , outputs and disturbances ($u_i,\ y_i\ i = 1, 2, 3$ and $d_i, i = 1, 2, 3, 4$) are represented by the vectors:

$$\bar{u} = \begin{bmatrix} \Delta g_{in}^{(1)} \\ \Delta V_{in}^{(1)} \\ \Delta g_{in}^{(2)} \end{bmatrix}; \bar{y} = \begin{bmatrix} \Delta h_2^{(1)} \\ \Delta\sigma_1 \\ \Delta h_2^{(2)} \end{bmatrix}; \bar{d} = \begin{bmatrix} \Delta h_1^{(1)} \\ \Delta T^{(1)} \\ \Delta h_1^{(2)} \\ \Delta T^{(2)} \end{bmatrix} \quad (14)$$

3 Multivariable Generalized Predictive Control

In this section we present he MGPC problem applied to the hot rolling mill process studied here. Let $\hat{y}_i(t + j)$ the minimum variance predictor of the output j steps ahead from the current time t; $\hat{ui}(t + j)$ the input predictor j steps ahead from the current time t; ; $r_i(t + j)$, is the future reference trajectory j steps ahead; and finally, let $w_i(t + j)$ the filtration of this reference given by:

$$\begin{aligned} w_i(t) &= y_i(t) \\ w_i(t+j) &= \alpha w_i(t+j-1) + (1-\alpha) r_i(t+j) \\ j &= 1...h_p \end{aligned} \quad (15)$$

where α is a parameter that lies between 0 and 1, h_p is the prevision horizon. The main principle of MGPC is to calculate a sequence of future control signals along the control horizon h_c, i.e. calculate $\hat{ui}(t + j)$, for $i = 1, 2$ and $j = 1 \ldots h_c$, in such way that the following cost function is minimized:

$$\sum_{i=1}^{3}\sum_{j=1}^{h_p} \delta_i(\hat{y}_i(t+j) - w_i(t+j))^2 + \sum_{j=0}^{h_c-1} \lambda_i \Delta\hat{ui}^2(t+j) \quad (16)$$

where δ_i is and λ_i are, respectively, the controlled variable weights and the movie suppression coefficients and $\Delta\hat{ui}(t+j) = \hat{ui}(t+j) - \hat{ui}(t+j-1)$, is the estimate of the control signal variation. In addition the following restrictions are considered:

$$\begin{aligned} \Delta u_{imin} \le \Delta\hat{ui}(t+j) \le \Delta u_{imax}\ j = 0, \ldots, h_c - 1, \\ u_{imin} \le \hat{ui}(t+j) \le u_{imax}\ j = 0, \ldots, h_c - 1 \\ y_{imin} \le \hat{y}_i(t+j) \le y_{imax}\ j = 1, \ldots, h_p \end{aligned} \quad (17)$$

To formulate this problem as a standard quadratic programming problem, it is necessary to define some matrices that are represented with a bar over the letter, for example, $\bar{x}$, or $\bar{x}_{j,k}$, where j and k are respectively the rows and the column of the matrix. Let us define:

$$
\begin{aligned}
\Delta\hat{\hat{u}}_i &:= [\Delta\hat{u}_i(t)\ \Delta\hat{u}_i(t+1)\dots\Delta\hat{u}_i(t+h_c-1)]^T\\
\hat{\hat{u}}_i &:= [\hat{u}_i(t)\ \hat{u}_i(t+1)\dots\hat{u}_i(t+h_c-1)]^T\\
\hat{\hat{y}}_i &:= [\hat{y}_i(t+1)\ \hat{y}_i(t+2)\dots\hat{y}_i(t+h_p)]^T\\
\Delta\hat{\hat{u}} &:= [\Delta\hat{\hat{u}}_1\ \Delta\hat{\hat{u}}_2\ \Delta\hat{\hat{u}}_3]^T\\
\hat{\hat{u}} &:= [\hat{\hat{u}}_1\ \hat{\hat{u}}_2\ \hat{\hat{u}}_3]^T\ \ \hat{\hat{y}} := [\hat{\hat{y}}_1\ \hat{\hat{y}}_2\ \hat{\hat{y}}_2]^T\\
\Delta\bar{u}_{max} &:= [\Delta u_{1max}\ \Delta u_{2max}\ \Delta u_{3max}]^T\\
\Delta\bar{u}_{min} &:= [\Delta u_{1min}\ \Delta u_{2min}\ \Delta u_{3min}]^T\\
\bar{u}_{max} &:= [u_{1max}\ u_{2max}\ u_{3max}]^T\\
\bar{u}_{min} &:= [u_{1min}\ u_{2min}\ u_{3min}]^T\\
\bar{y}_{max} &:= [y_{1max}\ y_{2max}\ y_{3max}]^T\\
\bar{y}_{min} &:= [y_{1min}\ y_{2min}\ y_{3min}]^T\\
\bar{u}(t-1) &= [u_1(t-1)\ u_2(t-1)\ u_3(t-1)]^T
\end{aligned}
\tag{18}
$$

Using the results in [3] we can represent the output prevision by the following expression:

$$
\hat{\hat{y}} = \bar{H}\Delta\hat{\hat{u}} + \bar{f} \tag{19}
$$

where $\bar{H}$ $(3h_p \times 3h_c)$ and $\bar{f}$ $(3h_p \times 1)$. The first term in equality (19) is called the forced response , since it depends on the future inputs, and the second term is denoted by free response, since it depends on the past inputs and outputs. Let $\bar{I}_{j,j}$ an identity matrix and $\bar{O}_{j,k}$ a null matrix. Let us define the weight matrices:

$$
\begin{aligned}
\bar{\delta} &= \begin{bmatrix} \delta_1\bar{I}_{h_p,h_p} & \bar{0}_{h_p,h_p} & \bar{0}_{h_p,h_p} \\ \bar{0}_{h_p,h_p} & \delta_2\bar{I}_{h_p,h_p} & \bar{0}_{h_p,h_p} \\ \bar{0}_{h_p,h_p} & \bar{0}_{h_p,h_p} & \delta_3\bar{I}_{h_p,h_p} \end{bmatrix}\\
\bar{\lambda} &= \begin{bmatrix} \lambda_1\bar{I}_{h_c,h_c} & \bar{0}_{h_c,h_c} & \bar{0}_{h_c,h_c} \\ \bar{0}_{h_c,h_c} & \lambda_2\bar{I}_{h_c,h_c} & \bar{0}_{h_c,h_c} \\ \bar{0}_{h_c,h_c} & \bar{0}_{h_c,h_c} & \lambda_3\bar{I}_{h_c,h_c} \end{bmatrix}
\end{aligned}
\tag{20}
$$

Taking (20) and (19), we can represent the cost (16) by the following expression:

$$
\frac{1}{2}\Delta\hat{\hat{u}}^T\bar{\mathcal{H}}\Delta\hat{\hat{u}} + \bar{\mathcal{B}}^T\Delta\hat{\hat{u}} + \bar{\mathcal{F}} \tag{21}
$$

where: $\bar{\mathcal{H}} = 2(\bar{H}^T\bar{\delta}\bar{H} + \bar{\lambda})$, $\bar{\mathcal{B}}^T = 2(\bar{f} - \bar{w})^T\bar{\delta}\bar{H}$ e $\bar{\mathcal{F}} = (\bar{f} - \bar{w})^T\bar{\delta}(\bar{f} - \bar{w})$.

Let the vector of ones $\bar{\Gamma}_{j,1}$, and the inferior triangular matrix $\bar{T}_{j,j}$, which are defined by:

$$
\bar{\Gamma}_{j,1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \quad \bar{T}_{j,j} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ \vdots & & & & \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix} \tag{22}
$$

By considering the matrices $\bar{\mathcal{T}}_{3j,3j}$ and $\bar{\Gamma}_{3j,3}$, defined by:

$$
\bar{\mathcal{T}}_{3j,3j} := \begin{bmatrix} \bar{T}_{j,j} & \bar{O}_{j,j} & \bar{O}_{j,j} \\ \bar{O}_{j,j} & \bar{T}_{j,j} & \bar{O}_{j,j} \\ \bar{O}_{j,j} & \bar{O}_{j,j} & \bar{T}_{j,j} \end{bmatrix} \bar{\Gamma}_{3j,3} := \begin{bmatrix} \bar{\Gamma}_{j,1} & \bar{O}_{j,1} & \bar{O}_{j,1} \\ \bar{O}_{j,1} & \bar{\Gamma}_{j,1} & \bar{O}_{j,1} \\ \bar{O}_{j,1} & \bar{O}_{j,1} & \bar{\Gamma}_{j,1} \end{bmatrix} \tag{23}
$$

we can join the restrictions (17) in the following formulation:

$$
\begin{bmatrix} \bar{I}_{3h_c,3h_c} \\ -\bar{I}_{3h_c,3h_c} \\ \bar{\mathcal{T}}_{3h_c,3h_c} \\ -\bar{\mathcal{T}}_{3h_c,3h_c} \\ \bar{H}_{3h_p,3h_c} \\ -\bar{H}_{3h_p,3h_c} \end{bmatrix} \Delta\hat{\hat{u}} \leq \begin{bmatrix} \bar{\Gamma}_{3h_c,3}\Delta\bar{u}_{max} \\ -\bar{\Gamma}_{3h_c,3}\Delta\bar{u}_{min} \\ \bar{\Gamma}_{3h_c,3}(\bar{u}_{max} - \bar{u}(t-1)) \\ \bar{\Gamma}_{3h_c,3}(u(t-1) - \bar{u}_{min}) \\ \bar{\Gamma}_{3h_p,3}\bar{y}_{max} - \bar{f}_{2h_p,1} \\ \bar{f}_{3h_p,1} - \bar{\Gamma}_{3h_p,3}\bar{y}_{min} \end{bmatrix} \tag{24}
$$

The solution of the MGPC problem is obtained by calculating the prevision of the control vector $\Delta\hat{\hat{u}}$ that minimizes the quadratic cost (21) with the restrictions (24). The control signals applied in the process are $u_1(t) = \hat{u_1},\ u_2(t) = \hat{u_2}(t), u_3(t) = \hat{u_3}$ and the remainder is discarded. The performance of the output response depends on the tuning parameters defined in the matrices $\alpha, h_p, h_c, \delta_i, \lambda_i$ for $i = 1, 2, 3$. The optimum choice of these parameters is made in this work by the Genetic Algorithm.

4 GENETIC ALGORITHMS

Genetic algorithms are search algorithms based on the mechanics of natural selection of Darwin and natural genetics of Mendel. They combine survival of the fittest among string structures with a structures yet randomized information exchange to form a search algorithm with some of the innovative flair of human search. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure. While randomized, genetic algorithms are no simple random walk. They efficiently exploit historical information to speculate on new search points with expected improved performance.

GAs were first presented by [10]. GAs have been used in many diverse areas such as function optimization, image processing, signal processing and system identification [11]. A GA is a parallel global search technique that emulates natural genetic operators and works on a population representing different parameter vectors whose optimal value with respect to some (fitness) criterion is searched. This technique includes operations such as reproduction, crossover and mutation. These operators work with a number of artificial creatures called generation. By exchanging information from each individual in a population. GAs preserve a better individual and yield higher fitness generation by generation such that the performance can be improved. Next, we will briefly describe the basic operators in a GA.

4.1 Reproduction

Reproduction is a process in which a new generation of population is formed by selecting individuals from an existing population, according to their fitness. This process results in individuals with higher fitness values obtaining one or more copies in the next generation, while low fitness individuals may have none. Note, however, that reproduction does not generate new individuals but only favours the percentage of fit individuals in a population of given size. The Figure 2 shows the reproduction operator.

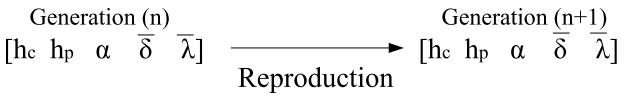


Figure 2. Reproduction Operation

where $\bar{\delta}$ and $\bar{\lambda}$ are given by matrices (20).

4.2 Crossover

This operation provides a mechanism for individual to exchange information via probabilistic process. This operation takes two "parents" individuals and produces two "offspring" who are new individual whose characteristics are a combination of those of their parents. The operation of crossover can be seen in the Figure 3

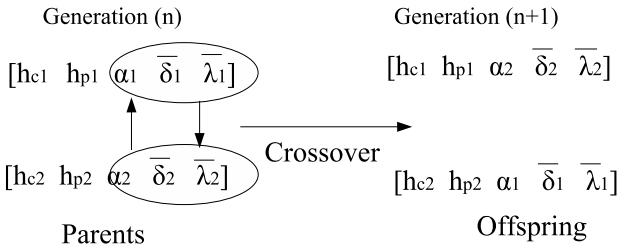


Figure 3. Crossover Operation

4.3 Mutation

Mutation is an operation where some characteristics of an individuals are randomly modified, yielding a new individual. Here, the operation simply consists in randomly changing the value of one bit of the string representing an individual. This operation is shown in the Figure 4.

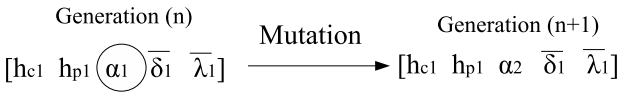


Figure 4. Mutation Operation

5 GENETIC ALGORITHMS APPLIED TO GPC

In this section, GA described above is used to optimize the parameters of a GPC controller. Firstly, GA will create the population of 50 individuals that contain the parameters necessary to minimize the objective function. The individuals of the GA in the real coding are showed as:

$$[h_c,\ h_p,\ \alpha,\ \delta_i,\ \lambda_i] \tag{25}$$

After that, the GA algorithm will compute the fitness function of each individual in the population and select the best individuals according to the roulette wheel method. The fitness functions presented in this work is:

$$Fit(h_c, h_p, \alpha, \delta_i, \lambda_i) = \frac{1}{\sum abs(y-w)^2} \tag{26}$$

After that, the genetic operators (reproduction, crossover and mutation) will be carried through in each individual to create a new generation. GA runs iteratively during 100 generations, presenting at the end of the procedure the best individual that is considered the result of simulation.

6 Results

In order to make the control technique evaluation, it was collected real data of the rolling mill of the steel coil used, with No 982 1612 rolled in the SIDERAR S.A. plant in Argentina.

The population generated by the GA program has 50 individuals and we consider the crossover and mutation rates be equal to 0.8 and 0.01 respectively.

The Figure 5 shows the measured input thickness variation , the temperature process variation of the real rolling mill that are applied to the model to test the proposed control law. Only variations concerned with the operation point were introduced. Simulation results show the output thickness variations $(\Delta h_2^{(1)})$ and $\Delta h_2^{(2)}$ in the Figure 6.

The Figure 6 shows the output thickness variations for Stands F9 and F10, being the output expected reference variation equal to zero. The optimal control error is lower than $50\mu m$ and in the case of conventional control (FFF) it goes up to $100\mu m$. The same results for different data collected in the rolling mill process were obtained and the

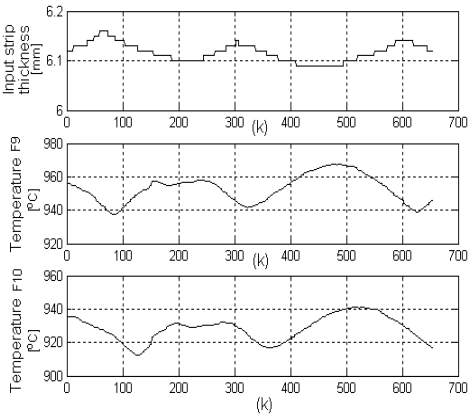


Figure 5. Strip thickness and temperature variation of stands F9 and F10.

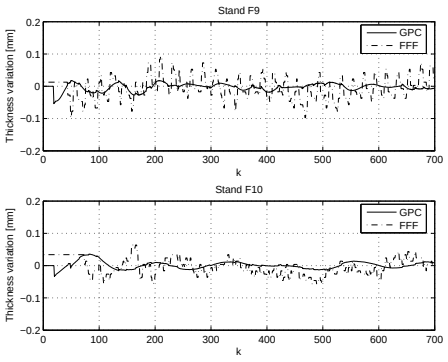


Figure 6. Strip output thickness variation of stands F9 and F10 with FFF and GPC.

thickness error in the case of the GPC is always smaller than the error of the control FFF, which is the real controller used in the plant.

7. Conclusions

The results showed that the predictive control technique of the controlled process evaluated by computational simulation, had a smaller thickness variation than the real force feed forward FFF controller, which was taken for comparison. FFF presented a greater thickness dispersion than the GPC (the variance in the stand F9 by FFF was 0.0012 and the GPC $1.0983e^{-4}$, in the stand F10, FFF $6.2787e^{-4}$ and the GPC $1.1581e^{-4}$). These results were a consequence of the tuning of the parameters h_c, h_p, α, δ_i λ_i which were supplied by the GA. If these parameters were adjusted by using trail and error method, it would give bad output thickness. The good performance of the transient response were obtained by the efficient tuning of the parameters h_c, h_p, α, and the adequate choice of the α parameter decreased the horizons h_c and h_p, in such way that it reduced the computational effort to solve the optimization problem.

References

[1] Rossomando, F.G., Filho, J.D. "Modelling and Control of a Hot Rolling Mill"*Latim American Research Applied* 36 (2006) 199–204

[2] Rossomando, F.G., Ferreira, E.P., Filho, J.D. "Controle Neural Adaptativo de um Trem de Laminação a Quente", *VII Simpósio Brasileiro de Automação Inteligente* Maranhão, 2005.

[3] Camacho, E.F., and Bordons, C. *Model Predictive Control.*Springer, New York, 2004.

[4] Quin, S.J., Badgwell, T.A. "A Survey of Industrial Model Predictive Control Technology"*Control Engineering Pratice* 11 (2003) 733–764

[5] Almeida, G.M., Salles, J.L.F., Filho, J.D. "Using Genetic Algorithms for Tuning the Parameters of Generalized Predictive Control", *VII Conferência Internacional de Aplicações Industriais INDUSCON* Recife, 2006.

[6] Almeida, G.M., Salles, J.L.F., Filho, J.D. "Controlador Preditivo Generalizado com Restrições Sintonizado via Algoritmos Genéticos", *XVI Congresso Brasileiro de Automática* Salvador, 2006.

[7] Filalit, S., Wertz, V. "Using genetic algorithms to optimize the design parameters of generalized predictive controllers"*International Journal of Systems Science* 32 (2001) 503–512

[8] Pedersen, L. "Modeling and identification of hot rolling mill", *ln proceedings of the American Control Conference* 1995, 3674–3680.

[9] Filho, J.D. "Um método de Controle Dinâmico de Laminadores reversíveis", *Dsc Thesis UFMG* 1994

[10] Holland, J.H. *Adaptation in natural and artificial systems*The University of Michigan Press, Ann Arbor, Berlin, 1975.

[11] Dejong, K.A. *An analysis of the behavior of a class of genetic adaptative systems.* PhD Dissertation, University of Michigan, Ann Arbor, Michigan, 1975.

UM ALGORITMO EVOLUTIVO MULTIOBJETIVO DUAS FASES COM MELHORAMENTO DO CONJUNTO PARETO-ÓTIMO APLICADO AO PROBLEMA DE FLUXO MULTIPRODUTO COM BALANCEAMENTO DA REDE

Fábio Pires Mourão*, Sérgio Ricardo de Souza*, Marcone J. F. Souza†

*Programa de Mestrado em Modelagem Matemática e Computacional
Centro Federal de Educação Tecnológica de Minas Gerais
30510-000, Belo Horizonte, MG, Brasil

†UFOP - Departamento de Computação
Campus Universitário
35400-000, Ouro Preto, MG, Brasil

Emails: fabiomourao@terra.com.br, sergio@dppg.cefetmg.br, marcone@iceb.ufop.br

Abstract— This paper proposes an evolutionary multiobjective method applied to the Multiobjective Multicommodity Flow Problem (MMFP). The proposed method consists of two phases, being the former a Genetic algorithm (GA), with which it attempts to obtain feasible solutions for the second phase, and the second the Nondominated Sorting Genetic Algorithm II method (NSGA II). The objective is to determine the minimum cost for the best balance of the network, in order to guarantee minimum congestion in the arcs of the network. The first phase of the method Phase I of the method works to generate the largest number of feasible solutions to be used in Phase 2. The NSGA II method, then, produces front solutions, using criteria of dominated solutions ordering and diversity in a population. After the second phase execution, a method based on neighborhood structures is applied to the Pareto-Optimal candidate solutions, in order to improve the Pareto-Optimal found set. Computational results show the efficiency of the proposed methodology.

Keywords— Multiobjective Multicommodity Flow Problem, Genetic Algorithm, NSGA II.

Resumo— O objetivo deste trabalho é propor um método evolutivo multiobjetivo aplicado ao Problema Multiobjetivo de Fluxo Multiproduto (PMFM). O método proposto é constituído de duas fases, sendo a primeira um Algoritmo Genético (AG), com o qual busca-se obter soluções factíveis para a segunda fase, constituída pelo método *Nondominated Sorting Genetic Algorithm II* (NSGA II). O objetivo é determinar o custo mínimo para o melhor balanceamento da rede, de modo a garantir congestionamento mínimo pelos arcos da rede. A Fase I do método busca gerar o maior número de soluções factíveis para serem usadas na Fase 2. O método NSGA II, então, produz fronteiras de soluções, utilizando critérios de ordenamento de soluções dominadas e diversidade da população. Após a execução da segunda fase, é aplicado, às soluções candidatas a ótimas de Pareto, um método baseado numa estrutura de vizinhança, cujo objetivo é melhorar o conjunto Pareto-Ótimo encontrado. Resultados computacionais mostram a eficiência da metodologia proposta.

Palavras-chave— Problema Multiobjetivo de Fluxo Multiproduto, Algoritmo Genético, NSGA II.

1 Introdução

Problemas de Fluxo Multiproduto (PFM) em geral apresentam um grande número de variáveis e restrições. Diante desse fato, a utilização de métodos exatos pode se tornar inviável, motivando a utilização de métodos heurísticos aplicados à resolução do problema. Tais métodos heurísticos podem, mesmo sem garantir otimalidade, gerar bons resultados e em menor tempo computacional quando comparados a métodos exatos. O PFM é modelado por meio de uma rede identificada por um grafo, na qual vários produtos trafegam pelos arcos capacitados a um determinado custo e competem pela capacidade destes arcos. Os nós da rede representam pontos de oferta e demanda para os vários produtos e o problema é, então, o de determinar o fluxo dos vários produtos ao menor custo possível de forma a atender basicamente dois conjuntos de restrições: restrições de conservação de fluxo, que gerenciam o fluxo dos produtos pela rede e restrições de capacidade, que limitam o tráfego de produtos nos arcos. Os primeiros trabalhos sobre PFM datam do início da década de 60, com contribuições iniciais de (Hu, 1963) e (Fulkerson and Ford, 1962). O PFM possui uma larga variedade de aplicações, principalmente nas áreas de telecomunicações e transportes. Dentre essas aplicações podem ser citadas: roteamento de tráfego na internet, (Buriol, 2003); roteamento de mensagens em redes de telecomunicações, (Hu, 1969), seqüenciamento de operações em refinarias de petróleo, (Milidiu et al., 2001); seqüenciamento de carga, (Shan, 2007); otimização em redes de fibra ótica, (Ozdaglar and Bertsekas, 2003) e alocação de trens em uma rede ferroviária, (Mendes, 1999). A abordagem multiobjetivo consiste em otimizar a função de custo, característica da formulação básica de um PFM, e uma função de balanceamento na rede, baseada na função para determinar o congestionamento mínimo, tratada em trabalhos de otimização em redes de comunicação, como em (Buriol, 2003), (Fortz and Thorup, 2000) e (Fortz et al., 2000). O método proposto neste trabalho é constituído por duas fases, uma mono-objetivo e outra multiobjetivo. A

Fase 1 é mono-objetiva e nela foi aplicado um Algoritmo Genético (AG), com o objetivo de encontrar soluções factíveis para a Fase 2, constituída pelo *Nondominated Sorting Genetic Algorithm II* (NSGA II). Após a obtenção das soluções candidatas, pela execução da Fase 2 do algoritmo, um método heurístico baseado em uma estrutura de vizinhança definida para cada solução é aplicado sucessivamente, a fim de melhorar a qualidade das soluções ótimas de Pareto.

Segundo (Castro, 2001), os métodos evolucionários apresentam características mais apropriadas para a resolução de problemas multiobjetivos, principalmente na obtenção do Conjunto Ótimo de Pareto, já que tais métodos utilizam um conjunto de soluções e, por conseguinte, exploram melhor o espaço de busca. O método NSGA II, proposto em (Deb et al., 2002), vem se constituindo na principal referência de métodos evolucionários multiobjetivos.

Este artigo é organizado como segue: a próxima seção apresenta o problema objeto de interesse. A seção 3 discute a metodologia de solução adotada, enquanto a seção 4 descreve a metodologia *Nondominated Sorting Genetic Algorithm* II. A seção 5 mostra a estrutura de melhoria das soluções de Pareto na forma aplicada ao PMFM. A seção seguinte 7 apresenta os resultados computacionais referentes à aplicação da metodologia proposta. A última seção apresenta as conclusões referentes ao trabalho.

2 Problema Multiobjetivo de Fluxo Multiproduto

O modelo do Problema Multiobjetivo de Fluxo Multiproduto (PMFM) em questão envolve a formulação do Problema de Fluxo Multiproduto (PFM), acrescentada pela função de balanceamento de carga. A estrutura do PFM em questão considera que cada unidade de cada produto é indivisível e que os fluxos destes produtos devem fazer uso apenas de uma rota. Deste modo, as variáveis envolvidas no PMFM são do tipo binário. A função de balanceamento de carga tem como intuito determinar o congestionamento na rede, sendo inicialmente apresentada em (Fortz and Thorup, 2000) e utilizada, dentre outros trabalhos, em (Fortz et al., 2000) e (Buriol, 2003). O problema em tela é modelado por uma rede identificada por um grafo, com n nós, a arcos e p produtos e ainda com $\mathcal{N}$, $\mathcal{A}$ e $\mathcal{P}$, os conjuntos de nós, arcos e produtos, respectivamente. O modelo matemático, então, é dado pela expressão (1).

$$\min \quad F = (f, \Phi) \tag{1a}$$

$$\text{s. a} \quad \sum_{(i,j)\in\mathcal{A}} x_{ij}^k - \sum_{(j,i)\in\mathcal{A}} x_{ji}^k = b_i^k, \ \forall i \in \mathcal{N}, \ k \in \mathcal{P} \tag{1b}$$

$$\sum_{k\in\mathcal{P}} x_{ij}^k \leqslant u_{ij} \tag{1c}$$

$$x_{ij}^k \in \mathcal{Z}_+, \quad \forall (i,j) \ \in \mathcal{A}, \ \ \forall k \ \in \mathcal{P} \tag{1d}$$

A expressão (1b) representa as restrições de conservação de fluxo, responsáveis pelo gerenciamento do fluxo de produtos na rede, de forma que x_{ij}^k representa o fluxo do produto k que flui pelo arco (i,j) (variáveis de decisão); b_i^k é igual a d^k, se o nó i for de oferta do produto k, $-d^k$, se o nó i for de demanda do produto k ou 0, em caso contrário, sendo d^k a quantidade de produto k que deve fluir do nó origem até o nó destino, associados a cada produto k. A carga total de produtos no arco, dada por l_{ij}, é definida como:

$$l_{ij} = \sum_{k\in\mathcal{P}} x_{ij}^k \tag{2}$$

de modo que a expressão (1c) representa a restrição de capacidade dos arcos, sendo u_{ij} a capacidade máxima do arco (i,j). Por dim , a expressão (1d) surge da própria formulação matemática do problema, representado as restrições de integralidade e não-negatividade.

A função f determina o custo do fluxo dos produtos na rede, definida como:

$$f = \sum_{k=1}^{p} \sum_{(i,j)\in\mathcal{A}} c_{ij}^k x_{ij}^k \tag{3}$$

para c_{ij}^k correspondente ao custo para o produto k trafegar pelo arco (i,j). A função Φ mede o congestionamento total da rede, sendo definida como:

$$\Phi = \sum_{(i,j)\in\mathcal{A}} \Phi_{ij}(l_ij) \tag{4}$$

para $\Phi_{ij}(l_{ij})$ representação do congestionamento em cada arco em função de sua carga, de modo que um arco com maior taxa de utilização será mais penalizado, possuindo um valor funcional mais elevado quando comparado a arcos com menores taxas de utilização. A função $\Phi_{ij}(l_{ij})$ é convexa e linear por partes, de forma que o modelo possui como parâmetros as definições dos intervalos e das diferenciais em cada intervalo. Os intervalos são definidos em função da capacidade de cada arco e as diferenciais são utilizadas para penalizar de acordo com a carga do arco, portanto, para cada intervalo existe uma diferencial associada.

$$\Phi_{ij}(l_{ij}) = \begin{cases} \delta_1 l_{ij} + \lambda_1 u_{ij}, & \text{se } 0 \leq l_{ij} < w_1 u_{ij}; \\ \delta_2 l_{ij} + \lambda_2 u_{ij}, & \text{se } w_1 u_{ij}; \leq l_{ij} < w_2 u_{ij}; \\ \vdots \quad ; & \\ \delta_{m-2} l_{ij} + \lambda_{m-2} u_{ij}, & \text{se } w_{m-3} u_{ij} \leq l_{ij} < u_{ij}; \\ \delta_{m-1} l_{ij} + \lambda_{m-1} u_{ij}, & \text{se } u_{ij} \leq l_{ij} < w_{m-1} u_{ij}; \\ \delta_m l_{ij} + \lambda_m u_{ij}, & \text{se } l_{ij} \geq w_{m-1} u_{ij}. \end{cases} \quad (5)$$

Assim, a função $\Phi_{ij}(l_{ij})$ é, então, dada pela expressão (5) para m intervalos. Nesta expressão, cada δ_i, com $1 \leq i \leq m$, corresponde ao valor da diferencial no intervalo i, responsável direto pela penalização; w_j, com $1 \leq j \leq (m-1)$, é um fator de multiplicação, responsável por limitar, inferior ou superiormente, o intervalo; e $\lambda_k u_{ij}$, com $1 \leq k \leq m$, é o termo independente, ou coeficiente angular, dados como parâmetros os intervalos e as diferenciais.

3 Metodologia de Solução

A idéia básica de um AG é aplicar o processo de evolução natural como um paradigma de solução de um problema, partindo de uma população inicial, gerada normalmente de forma aleatória e, por meio de operadores presentes no algoritmo, chegar a uma população contendo melhores soluções, quando comparadas às soluções das gerações iniciais.

A abordagem na Fase 1 é feita por meio do AG e a justificativa consiste em encontrar uma população contendo soluções factíveis, para ser usada como população inicial na Fase 2 e, com o uso dos operadores característicos do AG, é possível obter diversidade dentre os indivíduos. No AG, um indivíduo representa um cromossomo, que contém a codificação (genótipo) de uma possível solução (fenótipo). Em particular, cada cromossomo representa uma solução x na formulação dada anteriormente, sendo a implementação desta em forma de uma matriz de dimensão $a \times p$. Os atributos de uma solução tratada neste trabalho são os fluxos correspondentes a cada produto. Portanto, cada coluna da matriz solução, onde estão os fluxos para cada produto, é denominada gene. Os possíveis fluxos que podem ser determinados para cada produto são denominados alelos. Obviamente, a definição de gene e alelo pode variar de acordo com o problema tratado.

Os parâmetros e os operadores utilizados em um AG são responsáveis diretos pelo sucesso do algoritmo. Podem ser citados como principais operados da estrutura básica de um AG os seguintes operadores: **Operador de crossover**: responsável pela formação de novos indivíduos (filhos), partindo-se de duas ou mais soluções (pais); **Operador de mutação**: responsável por diversificar a população, alterando a codificação de um filho gerado após o crossover; **Seleção de indivíduos**: este operador define como será o processo de seleção de indivíduos para a população seguinte. Neste trabalho, o método é elitista, mantendo sempre as melhores soluções.

4 *Nondominated Sorting Genetic Algorithm* II - NSGA II

Este método, utilizado na Fase 2, foi propostos por (Deb et al., 2002) e apresenta, como principais vantagens quando comparado a seu antecessor, o NSGA, as seguintes características principais: menor complexidade quando ao ordenamento por não dominância; melhor controle da diversidade da população e uma abordagem elitista.

Duas questões são fundamentais nesse algoritmo, são elas: **Ordenamento por não dominância**: este procedimento cria as fronteiras formadas por soluções de acordo com seus níveis de não-dominância. Essas fronteiras são geralmente denominadas *fronts* e este procedimento, entitulado *Fast Nondominated Sort*, utiliza o número de soluções que dominam uma determinada solução p e o conjunto de soluções dominadas por p, ao fim da execução do algoritmo, o *front 1* corresponde às soluções pertecentes à aproximação do Conjunto Pareto-Ótimo; **Diversidade da população de soluções**: para manter a diversidade, bem como a característica elitista do método, é atribuída a cada solução um valor correspondente à sua *crowding distance* (distância de agrupamento), que é definida como o perímetro do cubóide criado a partir de um ponto (solução p) que não esteja no extremo do *front*, tendo como vértices as soluções adjacentes a esse ponto e pertencentes ao mesmo *front*.

Uma melhor descrição do método, bem como dos procedimentos citados, pode ser encontrada em (Deb et al., 2002).

5 Método de Melhoramento do Pareto Baseado numa Estrutura de Vizinhança

Após a execução da segunda fase, é aplicado, às soluções candidatas a ótimas de Pareto, um método baseado na geração de vizinhos de cada solução, com o intuito de obter mais soluções candidatas, ou de encontrar soluções que dominem as soluções candidatas encontradas pelo NSGA II.

Este procedimento, para cada soluções s pertencente ao conjunto ótimo de Pareto, gera vizi-

nhos a partir de um movimento que consiste na troca de uma coluna na matriz solução.

O procedimento é baseado no Método da Descida Randômica e consiste em, dada uma solução x, candidata a ótima de Pareto, é gerada uma nova solução x', definida como vizinha de x e gerada a partir da alteração de uma coluna em x, que corresponde à troca do fluxo de um produto, escolhido aleatoriamente. Caso x' seja factível, então ela é armazenada em uma outra lista de soluções, caso ela seja infactível, é incrementado um contador, chamado neste trabalho de *iter*. No caso de x' ser factível, então o contador *iter* é zerado. A cada vizinho gerada é feita uma verificação para eliminar a geração de soluções repetidas, e o contador *iter* não é zerado no caso de uma solução repetida, somente no caso de infactibilidade. O processo aplicado a cada solução é interrompido até *itermax*, parâmetro que representa o número máximo de iterações sem ser encontrada uma solução factível vizinha de x. Em nenhum momento é efetuada a troca de x por x', pois o objetivo é encontrar o maior número de soluções próxima de x, somente é efetuada a troca de conjuntos de soluções candidatas, após serem aplicados os procedimentos do NSGA II. Para evitar um esforço computacional muito elevado, o número de vizinhos de x é limitado por um parâmetro denominado *max_neigh*.

Todas as soluções geradas por este procedimento são armazenadas em uma lista, na qual também são armazenadas todas as soluções candidatas, a partir das quais as novas soluções foram geradas. Em seguida, o procedimentos *Fast Nondominated Sort* e o cálculo da *crowding distance* são aplicados às soluções dessa lista e, então, uma nova fronteira de soluções candidatas é gerada. Para essa nova fronteira e para as seguintes, o processo de melhoramento do Pareto baseado em estrutura de vizinhança é repetido até um número máximo de vezes, denominado neste trabalho de *trial*.

6 Aplicação do método proposto ao PMFM

A representação de uma solução x se dá por meio de uma matriz de dimensão $a \times p$ e os custos para os produtos trafegarem nos arcos da rede foram representados por uma matriz c de dimensão $p \times a$. Como o objetivo da Fase 1 é encontrar um conjunto contendo soluções factíveis, a função de aptidão nessa fase consiste no somatório das violações das restrições de capacidade em todos os arcos da rede. A função de aptidão do AG (Fase 1) é definida como:

$$f_{apt} = \sum_{(i,j) \in A} \nu_{ij} \qquad (6)$$

Se o arco (i, j) não estiver violado, então $\nu_{ij} = 0$ e, em caso contrário, $\nu_{ij} = (\sum_{k=1}^{p} x_i j^k) - u_{ij}$. Na Fase 1 as soluções iniciais são geradas aleatoriamente, conforme descrito em (Mourão et al., 2008). A população é representada por meio de uma estrutura, que contém os campos que armazenam a solução (matriz) e o somatório da violação. Um processo de busca local (Método da Descida Randômica) é aplicado aos indivíduos da população inicial, tendo como objetivo reduzir as violações (f_{apt}). A descrição de tal método não pertence ao escopo deste trabalho. A fim de otimizar o tempo computacional, o processo de busca é interrompido no momento em que é encontrada uma solução tendo f_{apt} nula, ou seja, uma solução factível.

Na Fase 1 (AG) o operador de *crossover* escolhido foi o *Crossover* Uniforme, cuja descrição detalhada pode ser encontrada em (Syswerda, 1989).A operação de mutação pode ser aleatória, dada certa probabilidade (*taxa_mutação*), ou o indivíduo poderá passar pelo método da Descida Randômica, caso ele seja infactível, sendo eliminados os $nind/2$ piores indivíduos, sendo $nind$ o tamanho da população. Durante toda a execução da Fase 1, se uma solução encontrada é factível, então ela é copiada para outra lista que armazena as soluções iniciais para a Fase 2. Se a solução é infactível, ela também pode ser copiada para essa lista, dada certa probabilidade, no entanto, ela será penalizada e colocada em um *front* mais elevado.

Na Fase 2 (NSGA II), a função de custo é definida como $\mathbf{Tr}(cx)$ e a função de balanceamento é definida conforme Equação 5, utilizando-se os mesmo intervalos e diferenciais utilizados por (Fortz and Thorup, 2000). Sendo assim, o vetor contendo as diferenciais é dado por $(1, 3, 10, 70, 500, 5000)$ e os intervalos são $[0, 1/3u_{ij})$, $[1/3u_{ij}, 2/3u_{ij})$, $[2/3u_{ij}, 9/10u_{ij})$, $[9/10u_{ij}, u_{ij})$, $[u_{ij}, 11/10u_{ij})$ $[11/10u_{ij}, \infty)$. O operador de *Crossover* utilizado na Fase 2 também é o *Crossover* Uniforme e a mutação pode ser feita somente de forma aleatória. O elitismo aplicado à Fase 2 é característico do NSGA II.

Após o fim do procedimento NSGA II, é aplicado o algoritmo de melhoramento, conforme descrito em 5.

7 Experimentos Computacionais

Os testes computacionais incidem sobre instâncias geradas aleatoriamente pelo ***GenMCF***, desenvolvido por (Alvelos, 2005). O método proposto foi implementado utilizando-se a linguagem de programação C e os gráficos foram gerados por meio do MatLab 6.5. Para as instâncias bl01-02, foram utilizados, durante a Fase 1, *60* indivíduos, *80* gerações, taxa de mutação aleatória igual a

Inst.	N	A	P	P/A	T1	T2	T3
bl01	32	96	48	0.5	474.38	609.63	1084.00
bl02	32	96	48	0.5	493.844	173.578	667.42
bl03	32	96	48	0.5	-	-	-
bl04	32	96	48	0.5	-	-	-
bl05	32	320	48	0.15	252.95	318.39	571.34
bl06	32	320	48	0.15	234.63	319.67	554.30
bl07	32	320	48	0.15	273.81	486.91	760.72
bl08	32	320	48	0.15	259.50	228.03	487.53

Tabela 1: Instâncias Testadas

30% e no máximo *200* iterações sem melhora no método de busca local que pode ser aplicado aos filhos gerados com o crossover (*itermax1*). Já para as instâncias bl05-08, durante a Fase 1, foram *40* indivíduos, *60* gerações e taxa de mutação de *30%*, além de no máximo *100* iterações sem melhora no método de busca local desenvolvido para a Fase 1. No caso do melhoramento do conjunto ótimo de Pareto, para as instâncias bl01-02, foram utilizados *itermax* igual a *500* e *max_neigh* igual a *300*, e para as instâncias bl05-08, foram utilizados *itermax 100* e *max_neigh* igual a *70*. Na Tabela 1 a coluna ***Inst.*** representa o nome da instância testada, ***N***, ***A*** e ***P*** representam a quantidade nós, arcos e produtos, respectivamente. A coluna P/A apresenta a relação entre o total de produtos e o total de arcos da rede testada e as colunas ***T1***, ***T2*** e ***T3*** representam, em segundos, os tempos do ***AG + NSGA II***, do **melhoramento** e a soma destes dois tempos, respectivamente. Os resultados são apresentados pelas Figuras 1-6 e os pontos representados por um $*$ representam o conjunto Pareto-Ótimo sem a aplicação do algoritmo de melhoramento. Já os pontos representados por um *o* representam os pontos do conjunto de soluções candidatas melhorado.

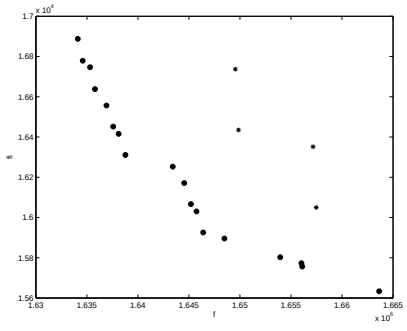


Figura 1: Instância bl01.

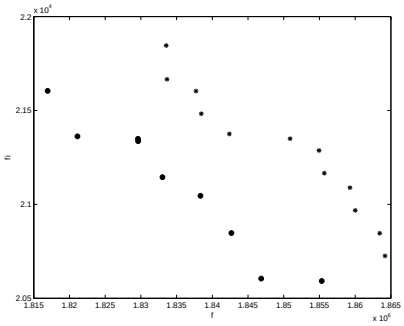


Figura 2: Instância bl02.

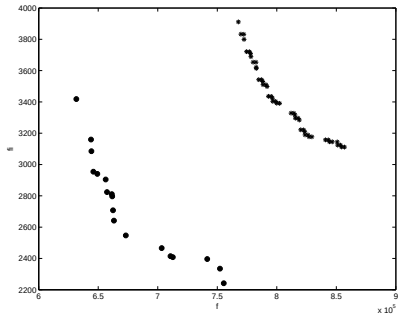


Figura 3: Instância bl05.

8 Conclusões

Nos gráficos gerados, é possível perceber que existem pontos que não admitem hiperplano suporte. Essa característica é comum quando pelo menos uma das funções não é convexa e como a função f está definida sobre os inteiros e, sendo o conjunto dos inteiros não-convexo, a função f não é convexa. A função Φ também está definida sobre o conjunto dos inteiros, embora possa assumir valores reais. Devido às características citadas, pode-se afirmar que o algoritmo conseguiu obter soluções esperadas. O NSGA II foi capaz de gerar soluções candidatas a ótimas de Pareto sempre que partiu de um conjunto de soluções factíveis, porém o AG não se mostrou eficiente para as instâncias *bl03* e *bl04*. O método de melhoramento também obteve sucesso, pois sempre gerou melhores conjuntos Pareto-Ótimos, partindo dos conjuntos gerados pelo NSGA II, ou seja, sempre foram geradas novas soluções que dominam as soluções geradas pelo NSGA II. As instâncias que apresentam menor relação P/A tiveram mais soluções candidatas. Pode ser percebida, também, a natureza conflitante dos objetivos tratados, pois uma redução no custo sempre implicou

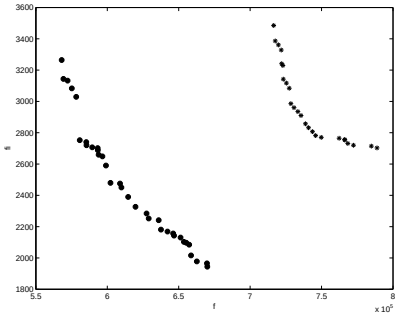


Figura 4: Instância bl06.

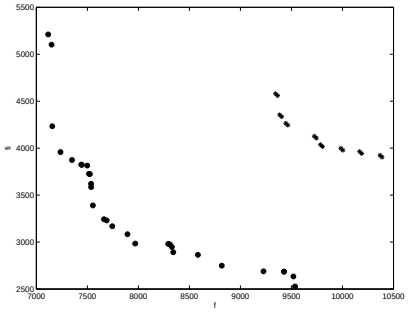


Figura 5: Instância bl07.

num maior valor da função de congestionamento (Φ). Conclui-se, então, que o NSGA II e o método de melhoramento são eficientes, porém o AG não foi capaz de gerar boas soluções para algumas instâncias e isso se deve, principalmente, ao fato do problema tratado ser um problema combinatório, com um elevado número de variáveis e restrições. Não foram feitas comparações com resultados presentes na literatura por não ter sido encontrada uma abordagem semelhante ao problema de fluxo multiproduto.

Referências

Alvelos, F. P. (2005). *Branch-And-Price and Multicommodity Flows*, PhD thesis, Universidade do Minho.

Buriol, L. S. (2003). *Roteamento do Tráfego na Internet: algoritmos para projeto e operação de redes com protocolo OSPF*, PhD thesis, UNICAMP.

Castro, R. E. (2001). *Otimização de estruturas com multi-objetivos via algoritmos genéticos*, PhD thesis, UFRJ.

Deb, K., Agrawal, S., Pratab, A. and Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: Nsga-ii, *IEEE - Transactions on Evolutionary Computation*, Vol. 6, pp. 182–197.

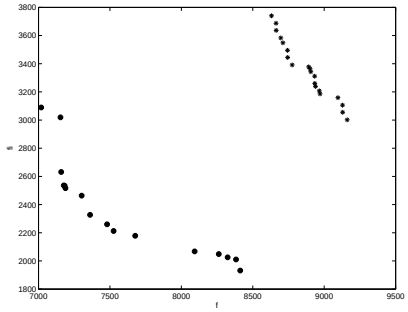


Figura 6: Instância bl08.

Fortz, B., Rexford, J. and Thorup, M. (2000). Traffic engineering with traditional ip routing protocols.

Fortz, B. and Thorup, M. (2000). Increasing internet capacity using local search.

Fulkerson, L. R. and Ford, D. R. (1962). Flows in networks, Princeton University, USA.

Hu, T. C. (1963). Multicommodity network flows, *Operations Research* **11**: 344–360.

Hu, T. C. (1969). *Integer Programming and Network Flows*, Addison-Wesley.

Mendes, R. R. (1999). *Programação Matemática Fuzzy Aplicada a um Problema de Transporte Multiproduto em Ferrovias*, PhD thesis, UNICAMP.

Milidiu, R. L., Pessoa, A. A., Braconi, V., Laber, E. S. and Rey, P. A. (2001). Um algoritmo grasp para o problema de transporte de derivados de petróleo em oleodutos, *XXXIII Simpósio Brasileiro De Pesquisa Operacional*, pp. 237–246.

Mourão, F., Souza, S. and Silva, C. (2008). Uma aplicação do algoritmo genético e metaheurística iterated local search a problemas de fluxo multiproduto, *Simpósio de Pesquisa Operacional e Logística da Marinha*.

Ozdaglar, A. E. and Bertsekas, D. P. (2003). Optimal solution of integer multicommodity flow with application in optical networks, *Symposium on Global Optimization*.

Shan, Y. S. (2007). *A Dynamic Multicommodity Network Flow Model for Real Time Optimal Rail Freight Car Management*, PhD thesis, Princeton University.

Syswerda, G. (1989). Uniform crossover in genetic algorithms, *in* J.D. (ed.), *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, pp. 2–9.