Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação

ALEX NEWMAN VELOSO DOS SANTOS

ALGORITMO EVOLUTIVO HÍBRIDO DIFUSO APLICADO A PROBLEMAS DE SEQUENCIAMENTO DE PADRÕES

São Luís
2016

ALEX NEWMAN VELOSO DOS SANTOS

ALGORITMO EVOLUTIVO HÍBRIDO DIFUSO APLICADO A PROBLEMAS DE SEQUENCIAMENTO DE PADRÕES

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Alexandre César Muniz de Oliveira

São Luís

2016

Alex Newman Veloso dos Santos
Algoritmo Evolutivo Híbrido Difuso Aplicado a Problemas de Sequenciamento de Padrões/ Alex Newman Veloso dos Santos. – São Luís, 2016.
57 f.
Orientador: Alexandre César Muniz de Oliveira

Monografia (Graduação) – Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação, 2016.

1. Algoritmo Evolutivo Híbrido Difuso. 2. Sequenciamento de Padrões. 3. Controlador Fuzzy. II. Algoritmo Evolutivo Híbrido Difuso Aplicado a Problemas de Sequenciamento de Padrões.

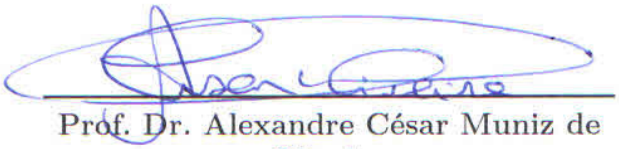
CDU 004.421.2

Alex Newman Veloso dos Santos

Algoritmo Evolutivo Híbrido Difuso Aplicado a Problemas de Sequenciamento de Padrões

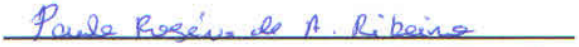
Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Trabalho aprovado. São Luís, 06 de abril de 2016:



Prof. Dr. Alexandre César Muniz de Oliveira
(Orientador)
Universidade Federal do Maranhão

Prof. Dr. Eng. Areolino de Almeida Neto
Universidade Federal do Maranhão



Prof. Dr. Paulo Rogério de Almeida Ribeiro
Faculdade Laboro

São Luís
2016

Para Alcéa Veloso, Lindanir Abreu e Laryssa Costa,
com todo amor.

Agradecimentos

Agradeço aos meus pais, em especial às minhas mães Alcéa Veloso e Lindanir Abreu, por todo o amor, educação, carinho, compreensão e sustento. Tudo o que conquistarei é graças a vocês.

Agradeço à minha grande família, especialmente à minha tia Maria Veloso, seu esposo e filhos, por terem me alimentado e dado teto, todos estes anos, quando precisei. Sou eternamente grato.

Agradeço à minha namorada e melhor amiga, Laryssa Costa, por todos os momentos, conversas, brincadeiras e ajudas. Só você sabe o quanto me faz feliz.

Agradeço aos meus amigos – todos eles – de infância, de escola, da universidade, do PETComp, da TI da Vale, do LACMOR, da vida, etc. Seria necessário um anexo inteiro para listar todos vocês. Sintam-se abraçados.

Agradeço o comprometimento e dedicação de todos os professores e demais colaboradores da UFMA, em especial ao meu orientador e amigo Alexandre.

Por último, agradeço ao PET (Programa de Educação Tutorial) por me permitir a participação de programas de ensino, pesquisa e extensão. Ao CNPq, por ter ampliado os meus horizontes com período de estudos no exterior. À Vale S.A. pela experiência profissional de altíssimo nível.

"A ciência trabalha na fronteira entre
o conhecimento e a ignorância. Não temos medo
de admitir que não sabemos. Não existe vergonha nisto.
A única vergonha é fingir que temos todas as respostas.
(Neil deGrasse Tyson)

Resumo

A alta competitividade industrial tem estimulado a busca por processos de manufatura mais efetivos e baratos. Os planos de produção têm-se tornado cada vez mais complexos, avaliando um número maior de variáveis e envolvendo riscos consideráveis. Neste cenário, a tomada de decisões na linha de produção é fator determinante para se obter alta lucratividade ou amargurar prejuízos no futuro. Problemas de sequenciamento de padrões surgem nas mais variadas searas da indústria. Algoritmos evolutivos híbridos destacam-se por terem sido aplicados com sucesso na solução deste tipo de problema. Este trabalho propõe um controle adaptativo *fuzzy* para a população do Algoritmo de Treinamento Populacional - ATP, específico para aplicações relacionadas ao sequenciamento de padrões. Após o desenvolvimento, uma série de experimentos com problemas-teste foi executada. Gráficos do comportamento da população apontam para um bom funcionamento do controle. Os resultados obtidos são comparáveis a outras abordagens.

Palavras-chaves: Sequenciamento de Padrões; Controlador Fuzzy; Algoritmo Evolutivo Híbrido Difuso.

Abstract

The high competition in industry has broaden the search for more effective and low cost manufacturing processes. Production plans have become more complexes, evaluating a higher number of variables and involving considerable risks. In this scenario, production line decision-making is the game changer to achieve high lucrativity or suffer huge losses in the future. Pattern sequencing problems emerge from the most various industry sectors. Evolutive hybrid algorithms have excelled for being successfully applied to solve this type of problem. This work presents an adaptive fuzzy control for the population of the Training Population Algorithm - TPA, specifically for applications related to pattern sequencing. Following the development, a series of experiments were conducted. The population behaviour plots show a control's proper working. Results obtained are similar to other approaches.

Key-words: Pattern Sequencing; Fuzzy Logic Controller; Evolutive Hybrid Fuzzy Algorithm.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AE	Algoritmo Evolutivo
AG	Algoritmo Genético
AGC	Algoritmo Genético Construtivo
AGC^H	Algoritmo Genético Construtivo com Treinamento Populacional em Heurísticas
ATP	Algoritmo de Treinamento Populacional
ATPD	Algoritmo de Treinamento Populacional Difuso
CI	Circuito Integrado
COG	Center of Gravity
FAM	Fuzzy Associative Memories
FLC	Fuzzy Logic Controller
GMLP	Gate Matrix Layout Problem
MOSP	Minimization of Open Stacks Problem
PBO	Problema Bi-Objetivo
PID	Proporcional Integral Derivativo
PSP	Pattern Sequencing Problem
SC	Sistema de Controle
TPH	Treinamento Populacional em Heurísticas
TSP	Travelling Salesman Problem
VLSI	Very-large-scale Integration Problem
WTA	Winner Takes All

Sumário

1 Introdução

Visando a maximização da eficiência com o menor custo possível, a indústria tem redesenhado seus processos, otimizando as mais variadas etapas da manufatura. A tomada de decisão em um estágio refletirá nos próximos, impactando de maneira positiva ou negativa todo o processo. A escolha de ações, tão sensível para a estratégia da indústria, torna-se mais complexa à medida que a escala de produção cresce.

A larga escala de produção das grandes indústrias faz com que estas criem planos de produção extremamente complexos, envolvendo muitas variáveis e riscos. A tomada de decisão é, portanto, feita com a ajuda de ferramentas especializadas nos problemas relacionados.

Neste contexto, fábricas de vários ramos de negócio lidam com os chamados Problemas de Sequenciamento de Padrões - PSP (FINK; VOSS, 1999). Este tipo de problema busca organizar aspectos relacionados à produção de objetos, sequenciados a outros a partir de um padrão de conexão (LINHARES, 2002).

Motivado pela busca de uma melhor utilização do espaço limitado e, consequentemente, diminuição do custo de armazenagem de semimanufaturados, um dos PSPs mais abordados na literatura é o MOSP - Problema de Minimização de Pilhas Abertas. Ele consiste em organizar uma fila de corte de objetos de maneira que um número mínimo de pilhas – de objetos com o mesmo padrão de corte – permaneça aberto durante o processo.

Em (LINHARES; YANASSE, 2002) aponta-se o Problema de Leiaute de Matriz-Porta - GMLP como correlacionado ao MOSP. Neste problema, tenta-se diminuir a área de impressão de uma placa de circuito a partir do rearranjo das trilhas que conectam as portas digitais.

Muitos métodos computacionais dedicam-se a prover soluções de boa qualidade, em um curto espaço de tempo, para os problemas mencionados acima. Dentre estes, destacam-se os algoritmos genéticos (GOLDBERG et al., 1989).

Embora os algoritmos genéticos sejam eficientes, é comum que gaste-se muito tempo avaliando soluções do vasto espaço de busca de grandes problemas. Em (OLIVEIRA, 2004), uma abordagem com identificação de áreas promissoras é proposta e chamada de Algoritmo de Treinamento Populacional - ATP. O ATP utiliza treinamento populacional em heurísticas para guiar a evolução, avaliando o indivíduo de acordo com o seu *fitness* e aptidão heurística. A população do ATP é dinâmica, de comportamento variável não adaptativo.

Sistemas de controles podem ser aplicados para ajustar o tamanho de uma população

de acordo com um valor ou comportamento desejado. Um sistema de controle (SC) é um mecanismo utilizado para gerenciar o comportamento de um processo (BISHOP, 2002). Um tipo de controle muito comum é o PID – Proporcional, Integral e Derivativo (MINORSKY, 1922). O controlador PID utiliza o erro, seu acúmulo e a sua taxa de variação no tempo para ajustar uma variável de controle do sistema (ASTROM; HAGGLUND, 1995). Estima-se que cerca de 80% dos controladores utilizados na indústria sejam do tipo PID (SIMÕES; SHAW, 2007).

É possível, entretanto, desenvolver um controlador PID utilizando-se lógica difusa (ZADEH, 1965). A lógica difusa é uma alternativa à lógica de Boole, sendo multivalorada. Um dos maiores triunfos da lógica difusa é permitir que valores vagos (*muito claro*, *pouco caro*) sejam interpretados por computadores, de maneira similar à interpretação humana (ZADEH, 1984). Deste modo, pode-se definir regras de funcionamento para um controlador lógico difuso sem a necessidade de conhecimento matemático sobre o problema em questão (GERLA, 2005).

1.1 Objetivos

Diante do exposto, o objetivo deste trabalho é propor um controle adaptativo *fuzzy* para a população do Algoritmo de Treinamento Populacional - ATP.

Desta maneira, pretende-se ajustar o tamanho da população dinâmica adaptativamente, impondo um comportamento definido.

1.2 Organização do Trabalho

Este trabalho está organizado de maneira a dar melhor compreensão do assunto abordado. No Capítulo 2 os principais temas relacionados ao trabalho são apresentados: Algoritmos Evolutivos; Treinamento Populacional em Heurísticas; Problema de Sequenciamento de Padrões; Lógica Difusa; e Controle PID.

O Capítulo 3 contém a metodologia proposta para este trabalho. Os detalhes da integração do controlador lógico difuso ao ATP, como os parâmetros utilizados no controlador, são explicados.

No Capítulo 4, os dados obtidos através dos experimentos são disponibilizados. Os resultados são comparados com a versão original do ATP e mais duas abordagens encontradas na literatura. Os resultados são discutidos ao final do capítulo.

O Capítulo 5 finaliza o trabalho apresentando a conclusão e direções para trabalhos futuros.

2 Fundamentação Teórica

Neste capítulo, apresentam-se conceitos básicos utilizados no desenvolvimento do trabalho. Mecanismos e terminologias pertinentes aos algoritmos evolutivos (AE) são explanados, com foco nos algoritmos genéticos. Aborda-se um algoritmo evolutivo híbrido que utiliza Treinamento Populacional em Heurísticas (TPH) e tópicos de Problemas de Sequenciamento de Padrões (PSP): Minimização de Pilhas Abertas (MOSP) e Problema de Leiaute de Matriz-Porta (GMLP).

Conclui-se este capítulo com uma breve explicação sobre lógica difusa e controladores PID aplicados à construção de controladores do tipo lógico difuso.

2.1 Algoritmos Evolutivos

2.1.1 Descrição

Na natureza, indivíduos dividem um mesmo ambiente e competem entre si pelos recursos disponíveis. Logicamente, aquele indivíduo cujas características - genes de um cromossomo - são mais adequadas ao meio prevalecerá e terá mais chances de se reproduzir, passando parte de seu material genético para as futuras gerações através da reprodução. Durante este processo, um novo indivíduo, contendo informação genética herdada de seus progenitores, será criado e ocasionalmente sofrerá uma alteração em um dos genes de seu cromossomo, chamada de mutação.

Analogamente, os AEs aplicam processos inspirados na evolução natural para resolução de problemas de otimização (RUTKOWSKI, 2008). E não os fazem sozinho. Tantas outras abordagens, bem como a programação genética (KOZA, 1992), a estratégia evolutiva (RECHENBERG, 1965) e a programação evolutiva (FOGEL; OWENS; WALSH, 1966) se encaixam na classe da computação evolutiva.

Em (OLIVEIRA, 1997) define-se um AE como um processo evolutivo onde os indivíduos, formando uma população, são representados por $P = \{x_1, x_2, ..., x_N\}$ de tamanho N. Cada indivíduo x vive durante uma geração t e faz parte da população $P(t) = \{x_1^t, x_2^t, ..., x_N^t\}$. É durante esta geração t que ocorrerão cruzamentos entre membros distintos da população, gerando um novo x^{t+1} através de uma função $R(x_i^t, x_j^t)$ onde:

$$x^{t+1} = R(x_i^t, x_j^t) \text{ para } i \neq j,\ i \leq N \text{ e } j \leq N \tag{2.1}$$

O cruzamento é feito com base na seleção dos pares, que por sua vez pode ser aleatória ou baseada em um critério. Quando aleatória, a probabilidade que um indivíduo

tem de ser escolhido é a mesma dos outros, ou seja $S_r(x_i) = 1/N$ para $i \leq N$. Para o segundo caso de seleção, tem-se a utilização de uma medida de capacidade de um indivíduo: a aptidão ou *fitness* - que na otimização indica o quão bom uma solução é baseada em uma função $F(x_i)$.

Na seleção por aptidão, os indivíduos têm mais chances de cruzarem com outros indivíduos de semelhante *fitness*. O método mais comum é a seleção por roleta, ou *roullette wheel selection*, onde a probabilidade de um membro da população ser escolhido é dada pela proporção de seu *fitness* em relação à população:

$$S_f(x_i) = \frac{F(x_i)}{\sum F(x_k)} \text{ para } k = 1, 2, ...N \text{ e } i \leq N \tag{2.2}$$

Ao utilizar um método por aptidão pode-se tirar vantagem da pressão de seleção (ρ) para guiar a solução para uma convergência mais rápida, muitas vezes limitada a um ótimo local, ou mais lenta mas com maior probabilidade de levar ao ótimo global no futuro. Este dilema é chamado de *exploration - exploitation tradeoff*. Segundo (RUTKOWSKI, 2008), *exploration* preocupa-se menos em manter ou melhorar a qualidade da população, aumentando o espaço de busca. Este modo é alcançado quando se tem menor pressão de seleção. Enquanto *exploitation* - quando há maior pressão de seleção - indica que somente os melhores da população poderão se reproduzir e passar seu padrão adiante. Pode-se então dizer, que em uma população ordenada pela sua aptidão, o número N' de melhores membros que participarão do processo de seleção é dado por:

$$N' = \lceil \rho \cdot N \rceil \text{ para } 0 < \rho \leq 1 \tag{2.3}$$

Não faz sentido, entretanto, adotar uma pressão de seleção baixa quando tem-se uma população muito homogênea. Indivíduos muito similares irão gerar novos x similares, derrubando toda a diversidade da população e estagnando o processo de busca. A mutação é o operador mais utilizado para recompor e introduzir diversidade em uma geração. Ela funciona analogamente como na natureza, alterando um ou mais genes de um cromossomo. Em um AE, esta simples mudança provoca um salto no espaço de busca.

A mutação traz mais uma vez o *exploration - exploitation tradeoff*. É interessante explorar um maior espaço de busca no inicio da execução. Para tanto, utiliza-se uma mutação maior. Enquanto que ao final das gerações, deseja-se focar nos melhores indivíduos para tentar produzir outros de qualidade superior. Neste ponto, a mutação deve ser, logicamente, menor. Pode-se afirmar que *exploration* está muito mais relacionado à mutação, ao mesmo tempo que a seleção liga-se melhor com *exploitation*.

Ao passo que novos indivíduos são criados e melhorados através da seleção e mutação, outros são removidos da população devido a competição por recursos. Esta quantidade de recursos pode ser interpretada como uma população de tamanho N fixo

onde somente os indivíduos com maiores F farão parte dela; ou como um *threshold* H onde indivíduos com aptidão abaixo deste valor serão eliminados.

2.1.2 Formalização

Matematicamente, pode-se representar um AG como uma entidade de nove tuplas (OLIVEIRA, 1997):

$$GA = (P^0, I, N, L, F, S, R, M, T) \tag{2.4}$$

onde P^0 é a população inicial, I é a codificação de um indivíduo, N é o tamanho da população, L é o comprimento do cromossomo de cada indivíduo, F é a função de aptidão (*fitness*), S é a função de seleção, R é a função de reprodução, M é a função de mutação e T é a condição de parada:

$$P^0 = (x_1^0, x_2^0, ..., x_N^0) \in I^N \tag{2.5}$$

$$I = \{0, 1\}^L \tag{2.6}$$

$$N \in \mathbb{N} \tag{2.7}$$

$$L \in \mathbb{N} \tag{2.8}$$

$$F : I \to \mathbb{R} \tag{2.9}$$

$$S : I^N \to I \tag{2.10}$$

$$R : I^2 \to I \tag{2.11}$$

$$M : I \to I \tag{2.12}$$

$$T : I \to \{V, F\} \tag{2.13}$$

Como pode-se observar em (2.6), a abordagem utilizada na formalização é de indivíduos com seus cromossomos codificados em binário. Outras codificações podem ser úteis, ajudando na modelagem do problema. Por exemplo, pode-se codificar os indivíduos utilizando o chamado *Permutation Encoding* (AGGARWAL; GARG; GOSWAMI, 2014), útil quando busca-se resolver um problema de ordenação, como o Problema do Caixeiro Viajante (*Travelling Salesman Problem* - TSP). Na Figura 1 e Figura 2 observa-se exemplos de cromossomos A e B codificados em binário e *permutation encoding*, respectivamente. Para estes casos tem-se que $I = \{0, 1, ..., 9\}^L$.

Figura 1: Cromossomos em binário.

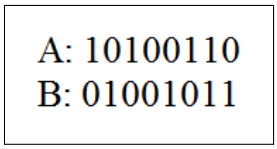
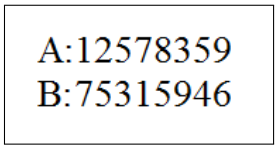


Figura 2: Cromossomos em permutation encoding.



2.1.3 Pseudocódigo

A seguir é apresentada uma simples implementação de um algoritmo genético em pseudocódigo. Note que a cada geração, o melhor indivíduo é armazenado. Ao final da evolução, este indivíduo será apresentado como a solução ótima encontrada.

```
1  População ← Gera População Inicial
2  Melhor Indivíduo ← Melhor(População)
3  enquanto não atinge Condição de Parada faça
4      até Número de Cruzamentos faça
5          Par ← Seleção(População, Pressão de Seleção)
6          Indivíduo ← Reprodução(Par)
7          se Chance Aleatória < Percentual de Mutação então
8              Indivíduo ← Mutação(Indivíduo)
9          fim
10         População ← Insere(População, Indivíduo)
11         Melhor Indivíduo ← Melhor(População)
12     fim
13     População ← Atualiza(População)
14 fim
```

Algoritmo 1: Algoritmo Genético

No Algoritmo 1, linha 1, a rotina `Gera População Inicial` cria indivíduos aleatoriamente e, adicionalmente, calcula a aptidão de cada um. Da mesma forma, `Insere`, na linha 10, calcula o *fitness* antes de inserir o indivíduo na população. Sendo assim, tem-se que a função objetivo do problema é implementada dentro destas duas rotinas.

A aptidão dos indivíduos é utilizada pela função `Melhor` para encontrar a melhor solução dentre os membros da População. Já na linha 13, `Atualiza` aplica a competição por recursos, eliminando os indivíduos menos aptos.

2.2 Treinamento Populacional em Heurísticas

2.2.1 Exploração do Espaço de Busca e Identificação de Regiões Promissoras

O funcionamento de algoritmos evolutivos está diretamente ligado com a exploração do espaço de busca, com cada indivíduo representando uma região visitada. Quanto mais explorado é este espaço, mais fiável considera-se a solução. Entretanto, visitar um elevado número de regiões do espaço implica naturalmente em consumir mais processamento e levar mais tempo para chegar ao critério de parada. Identificar regiões promissoras é, portanto, uma maneira eficaz de se explorar o problema, encurtando o escopo de visita do AE.

As regiões promissoras podem ser representadas utilizando esquemas (GOLDBERG et al., 1989). Um esquema é uma solução incompleta de um problema, a partir do qual outras soluções podem ser derivadas. A ordem de um esquema, que é a quantidade de posições fixas que ele tem, determina o número de soluções derivadas. Na Figura 3 observa-se um esquema de ordem cinco e dois exemplos de soluções derivadas, onde as lacunas podem assumir 0 ou 1.

Figura 3: Esquema (A) e duas soluções derivadas (B e C).

A: #1001##1
B: 11001001
C: 01001101

Esquemas binários são avaliados utilizando a técnica de *templates* competitivos, onde tenta-se obter um ótimo local a partir de uma solução derivada. Diz-se que encontrou o ótimo local quando não é possível melhorar o indivíduo a partir da troca de um único bit do cromossomo (KNJAZEW; GOLDBERG, 2000).

De maneira análoga, pode-se avaliar uma região a partir de um indivíduo utilizando heurísticas específicas do problema (OLIVEIRA, 2004). Heurísticas são métodos, ou critérios, para decidir qual ação, dentre muitas alternativas, promete ser mais efetiva para atingir um objetivo. Uma boa heurística tem critérios simples e distingue bem suas boas escolhas das más (PEARL, 1984).

O Treinamento Populacional em Heurísticas (TPH) identifica as regiões promissoras com base na avaliação da vizinhança da solução que o indivíduo representa (OLIVEIRA, 2004). Basicamente, aplicam-se heurísticas específicas do problema aos indivíduos até o ponto em que ele não possa mais ser melhorado por tais.

2.2.2 Fundamentos

Do ponto de vista computacional, treinamento pode ser interpretado como um exercício de adequação de um modelo a um problema baseado somente em uma função de desempenho. Estes modelos têm parâmetros que são ajustados até se aproximar a uma resposta esperada.

Naturalmente, quando há uma competição por recursos, provocando a escalada do mais sobre o menos aprimorado, já existe um treinamento. Aqueles indivíduos que conseguirão se adequar melhor ao problema serão privilegiados nos processos de seleção e reprodução, perpetuando as suas boas características. Neste ponto, os conceitos de evolução e treinamento se cruzam. Desta forma, pode-se dizer que um AE realiza treinamento em uma população de indivíduos, baseando-se na função de aptidão.

O TPH aplica heurísticas específicas ao problema para induzir as características desejadas à população do AE. Estas heurísticas permitem a avaliação dos indivíduos não somente através do *fitness* (F), mas pela função de aptidão heurística, representado por $g(x)$ - onde x é um indivíduo da população.

A função de aptidão heurística utiliza o conhecimento heurístico para informar o quão bom um indivíduo é (OLIVEIRA, 2004). Em termos simples, é feita uma busca na vizinhança por soluções melhores, desde que de acordo com as restrições. O indivíduo então será classificado de acordo com o encontrado nesta busca: mal adaptado à heurística de treinamento, se um houver uma solução heuristicamente melhor; ou bem adaptado à heurística de treinamento se for o melhor encontrado.

O cálculo desta adaptação ($\wp$) leva em consideração a distância entre o indivíduo em avaliação (x_k) e o melhor indivíduo da vizinhança heurística (x_v). Quando estes são iguais, não é aplicada penalização e o indivíduo tem a melhor avaliação possível. Caso contrário, a adaptação será reduzida na proporção da distância entre x_k e x_v. Afirma-se em (OLIVEIRA, 2004) que a função objetivo F pode ser utilizada como métrica de distância, pois fornece uma avaliação precisa da qualidade de uma solução vizinha a x_k.

Define-se, então, que a adaptação $\wp$ seja dada pela distância entre o *fitness* e a aptidão heurística:

$$\wp(x_k, H) = |F(x_k) - g(x_k)| \tag{2.14}$$

onde H é uma heurística específica utilizada para gerar a vizinhança heurística φ^H de tamanho L:

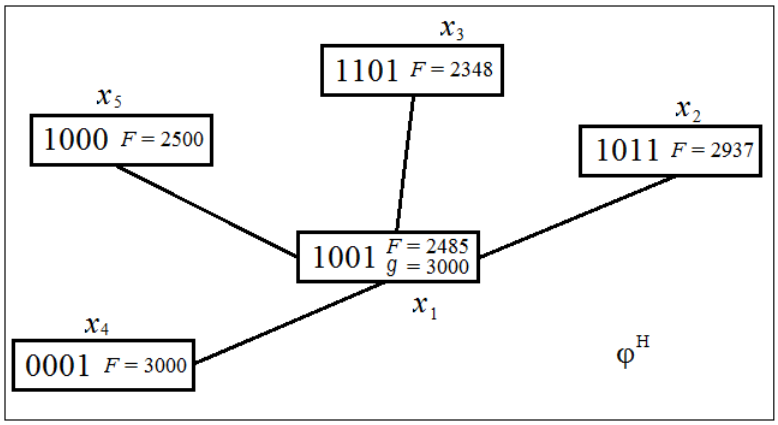
$$\varphi^H(x_k) = \{x_{v1}, x_{v2}, ..., x_{vl}\} \tag{2.15}$$

e a aptidão heurística g é, no caso da maximização, dada por:

$$g(x_k) = F(x_v) \text{ para } x_v \in \varphi^H(x_k), F(x_v) \geq F(x_{vi}), x_{vi} \in \varphi^H(x_k) \text{ e } i = 1, 2, ..., L \tag{2.16}$$

Na Figura 4 visualiza-se a vizinhança $\varphi^H(x_1)$ gerada a partir da heurística H de troca de um bit. É realizada uma busca nesta vizinhança pelo maior F, encontrado no indivíduo x_4. Portanto, a aptidão heurística g de x_1 será $F(x_4)$. Para este exemplo, afirma-se que x_1 não está bem adaptado à heurística de treinamento pois sua adaptação, $\wp$, tem valor elevado.

Figura 4: Vizinhança φ^H a partir de x_1



O TPH possui dois enfoques, sendo um construtivo, baseado no Algoritmo Genético Construtivo - AGC (LORENA; FURTADO, 2001), chamado de Algoritmo Genético Construtivo com Treinamento Populacional em Heurísticas - AGC^H; e um não construtivo, chamado de Algoritmo de Treinamento Populacional - ATP. O primeiro utiliza esquemas e indivíduos completos no processo de treinamento populacional, enquanto o segundo, abordado neste trabalho, usa somente indivíduos completos.

2.2.3 Algoritmo de Treinamento Populacional

O algoritmo de treinamento populacional foi desenvolvido a partir do AGC^H para os problemas onde não é possível aplicar uma competição entre esquemas e indivíduos completos, como os problemas de otimização numérica (OLIVEIRA, 2004).

Quando trabalha-se com esquemas, avalia-se diretamente a qualidade de determinados blocos de genes de um cromossomo, guiando a busca à regiões promissoras. No ATP, algo similar é feito utilizando-se apenas indivíduos completos. A avaliação heurística do indivíduo leva em consideração a sua vizinhança, o que eventualmente preserva determinados blocos do cromossomo.

O ATP, seguindo os fundamentos do TPH, é implementado para solucionar um problema bi-objetivo (PBO) onde busca-se sempre a minimização da adaptação $\wp(x_v, H)$ e a maximização, ou minimização, de $F(x_v)$, de acordo com o problema. Logo na primeira geração, é atribuído uma estimativa G_{max} (ou G_{min}) dos valores que F e g podem assumir durante a evolução. Esta estimativa é fundamental para montar o *ranking*, através do

coeficiente não construtivo δ, e posteriormente permitir a atualização da população. O δ de cada indivíduo é calculado da seguinte maneira:

$$\delta(x_k) = d \cdot [g(x_k) - G] - |F(x_k) - g(x_k)| \tag{2.17}$$

onde G assume G_{min} em caso de um problema de maximização, ou G_{max} para problemas de minimização; e d é uma constante utilizada para equilibrar os componentes. É comum utilizar $1/G$ como valor para d (OLIVEIRA, 2004).

A população é sempre ordenada segundo o coeficiente δ, onde indivíduos mais aptos estão no topo e serão privilegiados durante os processos evolutivos. Um parâmetro adaptativo α é usado para atualizar a população, removendo indivíduos mal avaliados.

Ao início da evolução, à α é associado o δ do último indivíduo da população. Como a população inicial é gerada de maneira aleatória, podendo conter somente soluções de baixa qualidade, este passo evita que todos os indivíduos sejam eliminados precocemente em caso de um alto valor atribuído à α.

Na implementação original do ATP, a cada geração é incrementado o valor de α, fazendo com que a população seja aos poucos eliminada. O incremento é dado por (OLIVEIRA; LORENA, 2002b):

$$\alpha^t = \alpha^{t-1} + k \cdot N^t \cdot \frac{\delta^t(x_1) - \delta^t(x_N)}{T - t} + c \tag{2.18}$$

onde k é uma constante de proporcionalidade; N^t é o tamanho da população atual; T é a condição de parada em número de gerações; t é a geração atual; $\delta^t(x_1)$ e $\delta^t(x_N)$ são, respectivamente, o coeficiente do primeiro e do último indivíduo da população. A constante c determina o mínimo de incremento a ser realizado. Isto evita que α pare de ser incrementado para o caso da população convergir antes do final da execução.

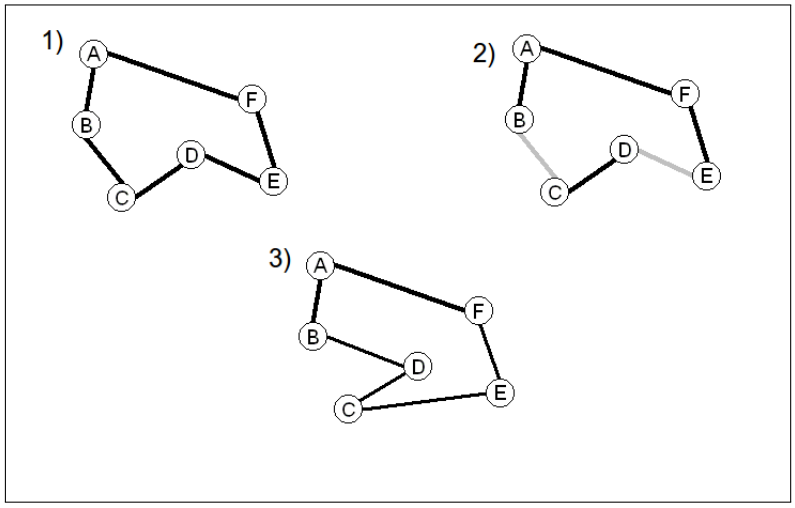
2.2.4 Heurística 2-Opt aplicada ao ATP

Desenvolvido inicialmente como uma heurística para o Problema do Caixeiro Viajante (*Travelling Salesman Problem - TSP*), o 2-Opt (CROES, 1958) é uma busca local que trabalha realizando trocas entre arestas em problemas de permutação.

Em sua proposta original, o 2-Opt realiza modificações de ordem de visita de um grafo, que representa o espaço de soluções de um problema. A partir de uma sequência de visita inicial, o algoritmo realiza operações sucessivas até o ponto em que as possibilidades de permutação sejam esgotadas. Esta sequência de visita resultante é tomada como um ótimo local. As operações realizadas pelo 2-Opt incluem a *quebra* de arestas e a reconexão destas de uma nova maneira.

Na Figura 5 vê-se um exemplo do funcionamento do 2-Opt. Partindo da solução inicial (1), o algoritmo avalia a troca das arestas $\overline{BC}$ e $\overline{DE}$ em (2) pelas arestas $\overline{BD}$ e

Figura 5: Exemplo gráfico do funcionamento da busca 2-Opt com troca das arestas $\overline{BC}$ e $\overline{DE}$ por $\overline{BD}$ e $\overline{CE}$



$\overline{CE}$, resultando em (3). Caso o movimento tenha sido satisfatório para a função objetivo do problema, o novo padrão de visita será mantido.

Outras abordagens para a mesma heurística são encontradas na literatura. O 2-Opt de (AGRAWAL, 2013) implementa uma busca para o TSP multiobjetivo a partir da utilização de duas abordagens diferentes. A primeira, seleciona aleatoriamente, a cada permutação, qual função objetivo será utilizada para calcular a melhoria do movimento. A outra abordagem utiliza a soma das funções objetivo, distância entre os vértices e custos individuais como critério de avaliação.

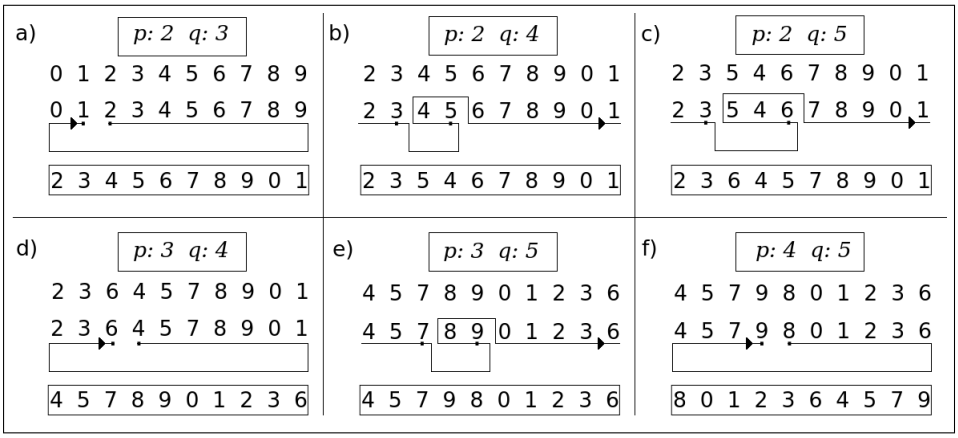
Em (OLIVEIRA; LORENA, 2002a), utiliza-se uma heurística baseada em 2-Opt para gerar indivíduos vizinhos a uma solução inicial em um algoritmo evolutivo. Inicialmente, pega-se um número p aleatoriamente. Este número indica a posição inicial no cromossomo onde, a partir dela, pares de genes serão trocados sucessivamente, até que se atinja um limite definido (l), ou o fim do cromossomo. Após cada troca, é atualizado o valor da aptidão heurística da solução inicial para o *fitness* do melhor indivíduo até então encontrado na vizinhança. A partir de l pode-se ter o tamanho da vizinhança L (número de permutações possíveis mais o indivíduo original):

$$L = \frac{l \cdot (l-1)}{2} + 1 \tag{2.19}$$

Esta heurística 2-Opt ainda implementa um movimento alternativo para posições consecutivas de troca de arestas. Neste caso, no algoritmo original, nenhuma solução diferente seria gerada. Este movimento alternativo consiste na troca da sequência: a partir da posição posterior, até o final do cromossomo; e a partir do início, até a posição inicial de troca. Este movimento, bem como as soluções geradas através desta modificação no 2-Opt, podem ser observadas na Figura 6.

A Figura 6 mostra graficamente o funcionamento da heurística. Parte-se de uma solução x_k de 10 posições, codificada em números naturais positivos. A posição p é escolhida aleatoriamente para dar início ao processo, que terá como escopo de troca até quatro posições à frente. Uma outra posição q é definida, delimitando o intervalo de troca. O conjunto de posições de referência $\{(2,3),(2,4),(2,5),(3,4),(3,5),(4,5)\}$ do tipo (p,q) é montado e representa as arestas que serão rearranjadas em nova disposição. Cada uma destas permutações é um indivíduo x_{vi} da vizinhança $\varphi^H(x_k)$, que terá a sua aptidão avaliada para o cálculo de $g(x_k)$.

Figura 6: Exemplo gráfico do funcionamento do 2-Opt modificado para $p = 2$ e $l = 4$



Ainda através da Figura 6, observa-se nos quadros *a*, *d* e *f* o funcionamento do movimento alternativo, permutando o indivíduo mesmo na troca de arestas com posições consecutivas. Os quadros *b*, *c* e *e* exemplificam visualmente o funcionamento de movimentos 2-Opt.

2.2.5 Mutação 2-Opt

Adota-se um processo elitista para o ATP em que mesmo a inclusão de um ruído na população direciona a busca para uma região promissora. Desta forma, somente soluções onde a mutação melhora o indivíduo original x_k serão adicionadas à população.

O processo de mutação utiliza a heurística 2-Opt para gerar no máximo m vizinhanças de tamanho L. O limite m é usado para evitar que muito custo computacional seja gasto na esperança de encontrar uma solução melhor. A cada iteração, a melhor estrutura encontrada será utilizada como ponto de partida para a próxima.

Pode-se interpretar os indivíduos avaliados na mutação como uma árvore de soluções, de raiz x_k e altura m. Os ramos da árvore são alcançados somente a partir da melhor solução do nível anterior. A ordem da árvore é dada por L, conforme a Equação (2.19).

2.3 Problema de Sequenciamento de Padrões

2.3.1 Definição

Os processos industriais têm se moldado ao longo do tempo visando o aumento da eficiência e a diminuição de custos. Tal condição é alcançada com a adoção de planos de produção cada vez mais complexos, que preveem o funcionamento desde a etapa mais básica até a mais complexa da cadeia. Estes planos de produção podem lidar com problemas de sequenciamento de padrões - PSP.

Define-se um problema de sequenciamento de padrões como qualquer problema que busca uma solução ótima para uma permutação de linhas em uma matriz, respeitando restrições e com avaliação baseada em uma função objetivo (FINK; VOSS, 1999). Diferentemente do problema do caixeiro viajante, não pode-se calcular o sucesso do movimento com base apenas na permutação, sendo necessário, por tanto, o cálculo da função objetivo a cada troca.

O PSP busca minimizar uma função objetivo, que normalmente representa custos associados à produção. Problemas de minimização de pilhas abertas (MOSP) e leiaute de matriz-porta (GMLP) são classificados como problemas de sequenciamento de padrões porque os objetos a serem sequenciados mantêm o mesmo padrão de conexão aos objetos remanescentes (LINHARES, 2002).

2.3.2 Minimização de Pilhas Abertas - MOSP

Indústrias de aço e de móveis, por exemplo, devem planejar a ordem com que objetos serão cortados em suas linhas de produção. Cada corte, de específico padrão, precisa ser armazenado em pilhas no espaço para elas destinado. Estas pilhas são abertas após o corte do primeiro objeto correspondente e permanecem abertas até que o último seja cortado.

O PSP descrito acima é conhecido como problema de minimização de pilhas abertas (*Minimization of Open Stacks Problem - MOSP*). Métodos para a solução exata do MOSP foram propostos na literatura, como por exemplo a utilização de programação linear inteira (YANASSE; LAMOSA, 2007). Tais métodos dependem da quantidade de itens e padrões, tornando a solução exata impraticável à medida que eles crescem. Sabe-se que o MOSP é um problema NP-Difícil (LINHARES; YANASSE, 2002).

Um MOSP pode ser modelado como uma matriz binária A de ordem $I \times J$ com linhas representando os padrões e as colunas os itens, onde o elemento a_{ij} terá valor 1 se o padrão i tem o item j (OLIVEIRA; LORENA, 2002b; LINHARES; YANASSE, 2002). Os padrões e os itens variam de 1 a n_p e 1 a m_o, respectivamente. A permutação π representa a atual disposição da matriz, assim como $\pi(i)$ representa a posição que o padrão i ocupa

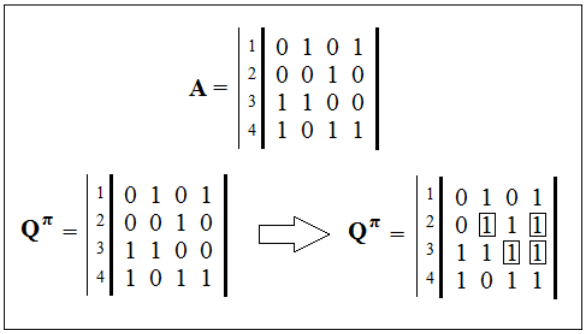
na sequência. O conjunto de todas as permutações π possíveis é Γ.

A partir de A, uma matriz de pilhas abertas (Q^π) é gerada. Basicamente faz-se uma cópia da matriz A, checando para cada coluna se há 0s entre 1s. Caso existam, estes serão substituídos por 1s. Desta maneira, Q^π apresenta uma propriedade conhecida por *1s consecutivos* (GOLUMBIC, 1980). Cada elemento q_{ij}^π de Q^π será preenchido da seguinte maneira:

$$q_{ij}^\pi = \begin{cases} 1 & \text{se } \exists u, \exists v \mid \pi(u) \leq i \leq \pi(v) \text{ e } a_{uj} = a_{vj} = 1 \\ 0 & \text{caso contrário} \end{cases} \tag{2.20}$$

Na Figura 7 vê-se a matriz Q^π sendo gerada para a sequência $\pi = \{1, 2, 3, 4\}$ da matriz A. As posições destacadas se referem aos elementos q_{ij}^π, intercalados por 1s na mesma coluna, que tinham valor 0 e foram substituídos.

Figura 7: Visualização da matriz de pilhas abertas Q^π gerada a partir de A



A cada sequência π, o número máximo de pilhas abertas ao mesmo tempo - $mpa(\pi)$ - é dado pelo maior número de 1s por linha. O $mpa(\pi)$ é dado por:

$$mpa(\pi) = \max_{i \in \{1,\ldots,n_p\}} \sum_{j=1}^{m_o} q_{ij}^\pi \tag{2.21}$$

Portanto, define-se que o objetivo do MOSP é encontrar uma solução que minimize o número de pilhas abertas ao mesmo tempo em uma mesma permutação π:

$$\text{MOSP} = \min_{\pi \in \Gamma} mpa(\pi) \tag{2.22}$$

O exemplo disponível na Figura 8 mostra uma instância do MOSP modelada na matriz A, seguida da matriz Q^π gerada. Note que para a sequência $\pi = \{1, 2, 3, 4, 5\}$, estarão abertas cinco pilhas durante a produção do padrão 2. Abaixo, apresenta-se a permutação ótima de A. Através de $Q^{\pi'}$, onde $\pi' = \{5, 3, 1, 2, 4\}$, verifica-se que o $mpa(\pi')$ é 4.

2.3.3 Problema de Leiaute de Matriz-Porta - GMLP

A integração de circuitos em larga escala (VLSI, do inglês *Very-large-scale integration*) é um processo de combinação de múltiplos componentes, criando um único circuito

Figura 8: Exemplo de matriz A e solução ótima correspondente

A =

item / padrão	1	2	3	4	5	6	7	8
1	0	0	1	0	1	0	0	0
2	1	1	0	0	1	1	0	0
3	1	0	1	0	0	0	0	0
4	0	0	0	1	1	0	0	1
5	0	0	1	0	0	0	1	0

Q^{π} =

	1	2	3	4	5	6	7	8	Σ
1	0	0	1	0	1	0	0	0	2
2	1	1	1	0	1	1	0	0	5
3	1	0	1	0	1	0	0	0	3
4	0	0	1	1	1	0	0	1	4
5	0	0	1	0	0	0	1	0	2

A' =

item / padrão	1	2	3	4	5	6	7	8
5	0	0	1	0	0	0	1	0
3	1	0	1	0	0	0	0	0
1	0	0	1	0	1	0	0	0
2	1	1	0	0	1	1	0	0
4	0	0	0	1	1	0	0	1

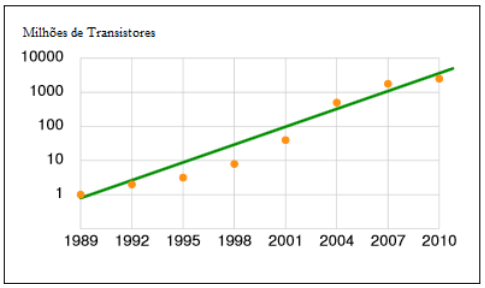
$Q^{\pi'}$ =

	1	2	3	4	5	6	7	8	Σ
5	0	0	1	0	0	0	1	0	2
3	1	0	1	0	0	0	0	0	2
1	1	0	1	0	1	0	0	0	3
2	1	1	0	0	1	1	0	0	4
4	0	0	0	1	1	0	0	1	3

integrado (CI). O avanço da tecnologia permitiu que os componentes de um CI fossem produzidos em tamanhos cada vez menores, levando à criação de *chips* mais complexos, com maior número de componentes por unidade de área.

Os primeiros *chips* continham apenas alguns poucos transistores se comparados aos atuais - que têm mais de um milhão de transistores por mm^2 (HELD et al., 2007). A lei de Moore prevê que o número de transistores dobre a cada dois anos em *chips* VLSI (MOORE, 1965). Visualizando a Figura 9, percebe-se um efeito muito próximo ao apontado por Moore.

Figura 9: Gráfico da evolução do número de transistores por *chip* de 1989 a 2010



Fonte: (HELD et al., 2007)

Estes *chips*, antes criados somente pelas mãos de engenheiros experientes, ficaram cada vez mais complexos e, portanto, mais difíceis de serem desenhados de maneira eficiente por tais. Várias tarefas do processo de desenho VLSI, como posicionamento dos componentes e conexões entre portas, utilizam métodos de otimização. Devido ao grande número de componentes nos circuitos atuais, métodos heurísticos têm se tornado comuns para busca de soluções (OLIVEIRA, 2004; LINHARES, 2002).

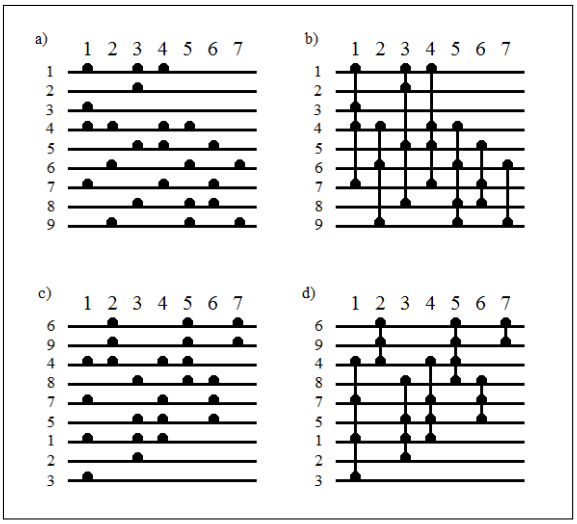
Especificamente na otimização de interconexões entre portas, um problema tem sido bastante abordado na literatura. Trata-se do Problema de Leiaute de Matriz-Porta (GMLP, do inglês *Gate Matrix Layout Problem*). Seu objetivo é organizar as portas do circuito em uma sequência ótima, de maneira que seja possível diminuir a área do leiaute. Em outra palavras, busca-se minimizar o número de trilhas sobrepostas interconectando

as portas (OLIVEIRA; LORENA, 2002a).

No GMLP, há um conjunto de portas com transistores, que são usados para conectar as portas. Uma porção de trilhas - linhas verticais entre pontos em *b)* e *d)* na Fig. 10 - conectam as portas que têm transistores na mesma posição. Estas conexões dão ao GMLP a propriedade dos *1s consecutivos*. Pode-se então modelar o problema de maneira análoga ao MOSP. Cada linha, antes interpretada como um padrão, é agora uma porta (linhas horizontais na Fig. 10), do mesmo modo que as colunas são tratadas como transistores (pontos na Fig. 10).

A Figura 10, adaptada de (OLIVEIRA, 2004), mostra uma instância GMLP em sua configuração inicial, nos pontos *a)* e *b)*, e sua solução ótima abaixo, em *c)* e *d)*. Note que em *b)*, o maior número de pilhas sobrepostas (*mps*) é 7, encontrado nas portas 6 e 7. Enquanto em *d)*, $mps = 5$, encontrado na porta 8. Na Figura 11 encontra-se a matriz A e Q^{π} geradas a partir de *a)* e *b)*, respectivamente.

Figura 10: GMLP- *a)* e *b)* configuração inicial *c)* e *d)* permutação ótima



Fonte: Adaptado de (OLIVEIRA, 2004)

Figura 11: Exemplo de modelagem de GMLP similar ao MOSP

$A =$

p \ t	1	2	3	4	5	6	7
1	1	0	1	1	0	0	0
2	0	0	1	0	0	0	0
3	1	0	0	0	0	0	0
4	1	1	0	1	1	0	0
5	0	0	1	1	0	1	0
6	0	1	0	0	1	0	1
7	1	0	0	1	0	1	0
8	0	0	1	0	1	1	0
9	0	1	0	0	1	0	1

$Q^{\pi} =$

p \ t	1	2	3	4	5	6	7
1	1	0	1	1	0	0	0
2	1	0	1	1	0	0	0
3	1	0	1	1	0	0	0
4	1	1	1	1	1	0	0
5	1	1	1	1	1	1	0
6	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1
8	0	1	1	0	1	1	1
9	0	1	0	0	1	0	1

p - porta
t - transistor

2.4 Lógica Difusa

2.4.1 Noções Básicas

Como uma alternativa à conhecida lógica booleana, a Lógica Difusa, do inglês *fuzzy logic* (ZADEH, 1965), oferece a capacidade de representar valores com termos imprecisos e dar uma resposta em ações precisas. Amplamente utilizada na inteligência artificial e sistemas de controle (LEE, 1990; YEN; LANGARI, 1998), ela formaliza matematicamente valores não convencionais que se aproximam bastante da maneira humana de valorar algo (ZADEH, 1984). Desta maneira, pode-se representar computacionalmente valores vagos como *meio frio* ou *muito cedo*.

Por lidar com graus de pertinência, a lógica difusa trabalha dentro do *range* dos axiomas da lógica comum, geralmente apresentando valores no intervalo entre 0 e 1. Deste modo, utiliza-se rótulos para expressar valores de verdade de subconjuntos de pontos do intervalo (YAGER; ZADEH, 1992), por exemplo *verdadeiro*, *muito verdadeiro*, *não tão verdadeiro* etc.

Por representar um valor *fuzzy* como pontos no intervalo entre 0 e 1, inclusive, pode-se confundi-los com simples valores probabilísticos. Entretanto, a noção de conjunto *fuzzy* é naturalmente não estatística (ZADEH, 1965).

Operações básicas são introduzidas à lógica difusa a partir do cálculo da função de pertinência (μ) de um elemento w a um subconjunto *fuzzy*. Como um conjunto *fuzzy* é composto usualmente de valores reais ($\mathbb{R}$), é inviável listá-los. Logo, a função de pertinência é definida por fórmulas matemáticas. Uma das fórmulas mais conhecidas, devido a sua facilidade de implementação e baixo custo computacional, é a função de pertinência triangular, dada por:

$$\mu_{tri}^{fs}(w) = \max\left(\min\left(\frac{x - a^{fs}}{b^{fs} - a^{fs}}, \frac{c^{fs} - x}{c^{fs} - b^{fs}}\right), 0\right) \tag{2.23}$$

onde a^{fs}, b^{fs} e c^{fs} representam os valores das abscissas dos vértices de um triângulo para o subconjunto *fuzzy* fs, tal que os pontos serão $\{(a^{fs}, 0), (b^{fs}, 1), (c^{fs}, 0)\}$.

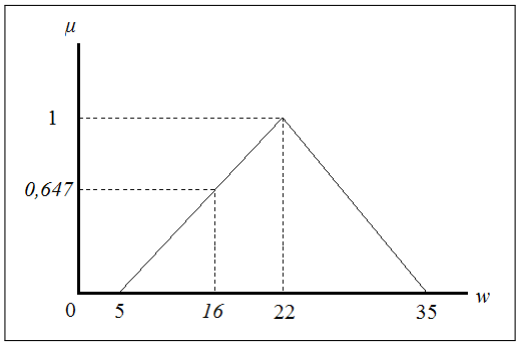
A Figura 12 traz o exemplo de uma função de pertinência triangular onde a, b e c são 5, 22 e 35, respectivamente. Em seguida é calculado $\mu_{tri}(w) = 0,647$, onde $w = 16$.

Em (ZADEH, 1965) define-se algumas operações sobre conjuntos *fuzzy* análogas à lógica booleana. As principais são os operadores: FNOT de negação, na Equação (2.24); FAND de conjunção, Equação (2.25); e ORF de disjunção, na Equação (2.26).

$$FNOT(w) = (1 - \mu(w)) \tag{2.24}$$

$$FAND(w, k) = \min(\mu(w), \mu(k)) \tag{2.25}$$

Figura 12: Função de pertinência triangular.



$$ORF(w,k) = \max(\mu(w), \mu(k)) \tag{2.26}$$

2.4.2 Raciocínio Difuso

O raciocínio difuso consiste de três etapas: *fuzzificação*, inferência *fuzzy* e *defuzzificação*. A primeira delas trata da transformação de elementos comuns para elementos *fuzzy*. A segunda lida com um conjunto de regras das quais se extrai valores verdade. A *defuzzificação* é responsável por transformar o resultado da inferência, um elemento *fuzzy*, em um elemento do mesmo domínio de entrada, ou valor numérico.

Na etapa de *fuzzificação* é feita a conversão do elemento numérico inserido, calculando o grau de pertinência para um subconjunto, ou classe, *fuzzy*. A partir deste momento, o elemento é representado através de um rótulo linguístico.

As regras, elementos necessários para o processo de inferência, são relações condicionais onde uma ou mais premissas *fuzzy* implicam em uma conclusão. Estas regras são definidas linguisticamente utilizando-se sintaxe *SE-ENTÃO* (NATSHEH; BURAGGA, 2010). Só é possível tirar uma conclusão de um elemento *fuzzificado*. A conclusão é a saída de uma regra, classificando o elemento em um dos valores verdade. A saída de todas as regras será agregada, inferindo, desta maneira, a saída geral do processo.

O conjunto de regras pode ser representado através de uma Matriz de Associação *Fuzzy*, ou *Fuzzy Associative Memories* - FAM (KOSKO, 1991). Ela funciona analogamente a uma tabela verdade, mapeando valores de saída para um conjunto de valores de entrada. Os conjuntos de antecedentes (premissas) e consequentes (conclusões) serão combinados utilizando as operações de negação, conjunção e disjunção na etapa seguinte.

A última etapa, *defuzzificação*, é o inverso do processo de *fuzzificação*. Ela consiste em entregar uma resposta no mesmo formato de entrada. Várias técnicas são descritas na literatura. A mais simples é conhecida como *Winner Takes All* - WTA (BISHOP, 2002), ou o Vencedor Leva Tudo. Nesta técnica, o valor verdade ao qual o maior grau de pertinência foi atribuído será retornado como saída da inferência.

Na Figura 13, observa-se um exemplo de FAM com três rótulos de saída e entrada. Os rótulos B, M e A remetem-se à *baixo*, *médio* e *alto*, respectivamente. Note que a combinação das linhas e colunas implicam em um valor de resposta. Por exemplo, para as entradas $w_1 = \text{B}$ e $w_2 = \text{A}$ e técnica WTA de *defuzzificação*, a saída será M.

Figura 13: Exemplo de Matriz de Associação Fuzzy.

w_2 \ w_1	B	M	A
B	A	M	M
M	A	M	B
A	M	B	B

Esta técnica, entretanto, causa perda da continuidade na resposta. A continuidade é uma importante característica da *defuzzificação*. Ela garante que para pequenas alterações no grau de pertinência de cada subconjunto *fuzzy* de saída, não ocorrerão grandes variações na resposta (SALETIC; VELASEVIC; MASTORAKIS, 2002).

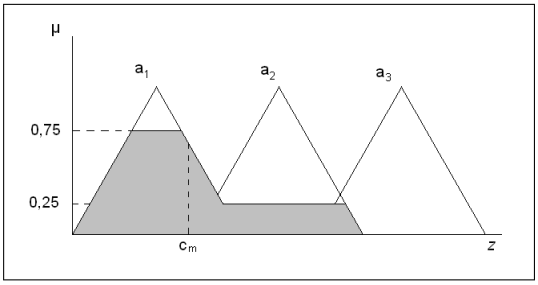
Outra técnica, que não se preocupa com o problema de descontinuidade, é o Centro de Gravidade - ou centro de massa - COG. Calcula-se o centroide da área abaixo dos subconjuntos *fuzzy* envolvidos na resposta (SAADE; DIAB, 2004). Note que esta técnica leva em consideração os limites estabelecidos pelo conjunto de regras. Dada por:

$$c_m = \frac{\int z \cdot \mu(z) dz}{\int \mu(z) dz} \tag{2.27}$$

onde z representa a faixa de valores de saída.

O comportamento da Equação (2.27) pode ser observado através da Figura 14. A inferência conclui dois valores verdade ($0,75$ a_1 e $0,25$ a_2) para as entradas do sistema. Calcula-se, então, o COG a partir da área abaixo de $\mu(z)$ - parte mais escura na imagem. O resultado, c_m é a resposta do processo de *defuzzificação* e, portanto, a resposta do sistema para os valores entrados.

Figura 14: *Defuzzificação* através da técnica de centro de massa



2.5 Controlador PID

2.5.1 Descrição

Amplamente adotado devido a sua simplicidade e eficiência, o controlador Proporcional Integral Derivativo, ou simplesmente controlador PID, é um mecanismo que utiliza erro de determinada referência como entrada para compensar a sua saída, ou retorno. A primeira análise teórica do PID foi publicada somente em 1922 (MINORSKY, 1922), embora exista evidência do seu desenvolvimento na década anterior, aplicado ao controle automático do manche de navios (BENNETT, 1996).

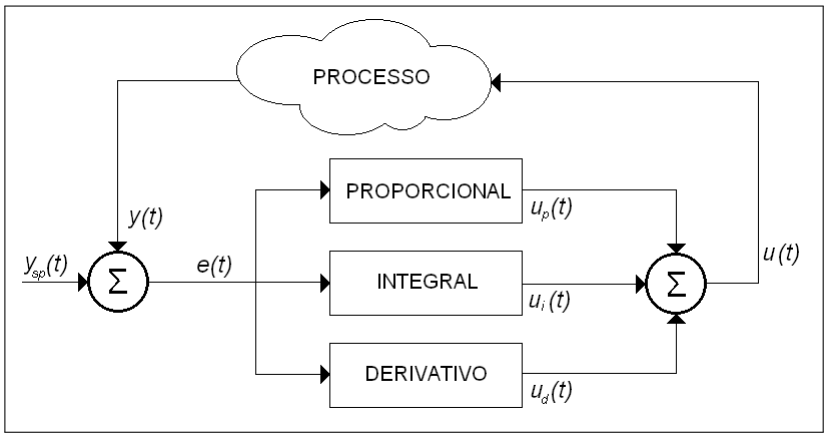
O PID oferece uma vantagem sobre os outros tipos de controle por aplicar pequenos ajustes baseados no erro (e) entre o valor do processo (y) e o valor inserido como referência (y_{sp}). Tal efeito não pode ser observado em outros controladores, como o simples controlador *liga-desliga*, que trabalha com duas faixas de retorno: u_{max} e u_{min}. A má consequência deste simples tipo de controlador é a instabilidade da resposta do sistema, provocando grandes oscilações entre a saída do controlador e a referência.

Observa-se através da Figura 15 que o controlador PID trabalha com três termos. O primeiro deles é o termo Proporcional, responsável por garantir que o retorno (u) seja proporcional ao erro. O termo integral garante que quando o tempo de observação tenda ao infinito, o erro seja zero. O último, termo derivativo, tenta prever o comportamento do sistema e ajusta a saída, estabilizando o sistema (ASTROM; HAGGLUND, 1995).

Ainda na Figura 15, vê-se que o sistema calcula o erro e iterativamente a partir da referência e da observação do processo no momento t:

$$e(t) = y_{sp}(t) - y(t) \tag{2.28}$$

Figura 15: Exemplo de controlador PID



2.5.2 Ação Proporcional

A ação proporcional indica que a correção a ser aplicada ao processo (saída do controlador) deve crescer ou diminuir na mesma proporção que o erro e. Normalmente, o valor do retorno é dado por:

$$u_p(t) = K_p \cdot e(t) + u_b \tag{2.29}$$

onde K_p é o ganho proporcional e u_b é um *bias*. O *bias* é utilizado para que o controlador continue respondendo mesmo quando não houver erro (ASTROM; HAGGLUND, 1995).

Conforme o sistema avança em direção ao valor de referência, o erro fica cada vez menor. Eventualmente, este erro torna-se tão próximo de zero que mesmo a multiplicação pelo ganho proporcional resultará num produto aproximadamente nulo. Isto implica que o controle não aplicará a força e jamais alcançará y_{sp}. Este efeito é chamado de erro em regime permanente (e_{ss}). Em alguns casos, o valor de u_b será ajustado manualmente para que $e_{ss} = 0$ (ASTROM; HAGGLUND, 1995).

Observando a Equação (2.29), percebe-se, empiricamente, que a variação do valor de K_p produz alterações em e_{ss}, mesmo que diminutas. É natural que o aumento do ganho proporcional diminua o valor do erro em regime permanente.

O ganho proporcional K_p induz no retorno do controlador, grosso modo, um maior - ou menor - esforço no alcance do valor de referência. Através do gráfico presente na Figura 16, observa-se o comportamento de u_p ao longo do tempo t. Note que para valores baixos de K_p, o controlador tem mais dificuldade para acompanhar a referência. Valores maiores, porém, provocam oscilações maiores.

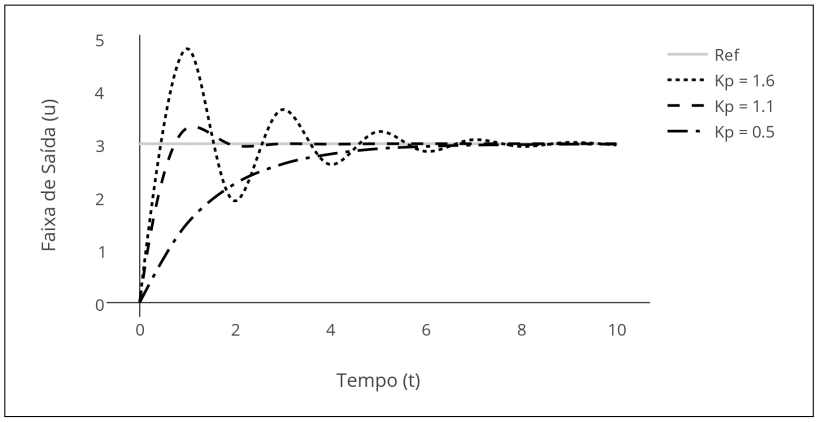
O gráfico da Figura 16 foi gerado a partir da simulação de um controlador P com $u_b = 0$. O processo, simulado para efeito didático, inicia-se zerado e somente é alterado pela resposta do controlador.

Os comportamentos observados para estes três diferentes valores de K_p (Figura 16) diferenciam-se no grau de amortecimento (ζ). O amortecimento, nos controladores, pode ser encarado como a capacidade que o ajuste tem de diminuir gradativamente o erro. Pode-se classificar os controles em: subamortecido, superamortecido e criticamente amortecido.

O controle subamortecido tem conversão mais lenta para a referência, sendo mais oscilatório durante o processo. O sistema de controle superamortecido apresenta aproximação do valor final, entretanto com conversão lenta. O controle criticamente amortecido apresenta o melhor tempo de resposta, conseguindo convergir rapidamente para a referência, sem oscilação.

Na Figura 16, pode-se classificar a reposta simulada para $K_p = \{1,6; 1,1; 0,5\}$ como sub-amortecida, criticamente amortecida e superamortecida, respectivamente.

Figura 16: Gráfico da ação proporcional para $y_{sp} = 3$ e diferentes valores de K_p



2.5.3 Ação Integral

É evidente que o erro em regime permanente é o ponto falho da ação proporcional. A utilização do *bias* com a função de garantir um e_{ss} nulo é uma alternativa que precisa ser atualizada para cada processo, podendo ser inadequada também para um processo onde a referência é dinâmica.

A ação integral é utilizada, portanto, para garantir que a saída do processo (y) assuma o valor de referência (y_{sp}) com o passar do tempo t (ASTROM; HAGGLUND, 1995), dada por:

$$u_i(t) = K_i \cdot \int_0^t e(t)dt + u_b \tag{2.30}$$

onde K_i é o ganho integral e mais uma vez o u_b é tratado como o *bias*. Basicamente, é feita uma acumulação do erro e ao longo do processo.

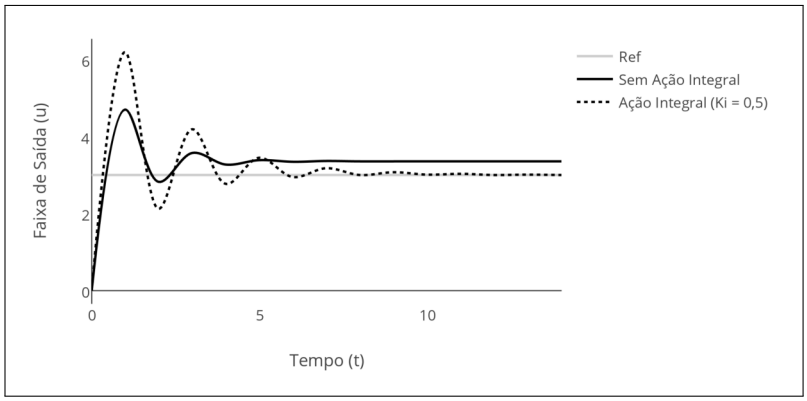
O erro em regime permanente, apesar de pequeno, será somado a cada iteração do controlador. Em determinado momento, este valor causará ruído suficiente para provocar uma resposta da ação integral, guiando y para y_{sp} onde $e = 0$. A partir deste ponto, se não houver variação na referência, o sistema assume $e_{ss} = 0$.

Entretanto, a adoção da ação integral induz um efeito oscilatório na saída do controlador, tornando o sistema mais instável (MOORE; HETTHESSY; BARS, 2005). Tal comportamento pode ser observado a partir do gráfico na Figura 17.

Observa-se no gráfico da Figura 17 o comportamento de dois controladores, para efeito de comparação. O primeiro deles implementa somente a ação proporcional. É possível ver, para este controlador, que o sistema converge rapidamente para um ponto diferente da referência, apresentando, portanto, $e_{ss} \neq 0$. Percebe-se que o controlador PI, em linha pontilhada, eventualmente atinge a referência.

O gráfico, presente na Figura 17, foi obtido com a simulação de dois controladores, P e PI. Os dois utilizaram valores iguais ($u_b = 0,5$, $y_{sp} = 3$ e $Kp = 1,4$), diferenciando-se apenas pela presença da ação integral no segundo controlador.

Figura 17: Gráfico comparativo da saída de um controlador P e um controlador PI



2.5.4 Ação Derivativa

A dinâmica do processo de um controlador faz com que as mudanças na variável de controle somente sejam percebidas depois de certo tempo. Esta lentidão na resposta, associada à instabilidade inserida no sistema pela adição do termo integral, motivou o uso da ação derivativa.

O termo derivativo trabalha tentando predizer o efeito do erro, fazendo com que o sistema acelere ou diminua a correção, mitigando, desta forma, a instabilidade. Basicamente, o erro atual é subtraído do erro anterior a cada iteração do controle. Desta maneira, a taxa de variação de e é computada e será utilizada para estimar a tendência de aumento ou diminuição no futuro.

Em vários trabalhos (FACCIN, 2004; ASTROM; HAGGLUND, 1995), a ação derivativa é dada por:

$$u_d(t) = K_d \cdot \frac{de(t)}{dt} + u_b \tag{2.31}$$

onde K_d é o ganho derivativo e u_b é o *bias*. Note que a ação derivativa somente induzirá uma ação quando houve variação do erro no tempo. Ou seja, em caso de erro em regime permanente, nenhuma correção será aplicada por este termo.

3 Treinamento Populacional com Controle Adaptativo *Fuzzy*

Neste capítulo, as etapas da integração de um controlador lógico difuso ao algoritmo de treinamento populacional, visando o controle da população de indivíduos, são expostas, proporcionando um melhor entendimento dos métodos aplicados.

3.1 Controlador Lógico Difuso

3.1.1 Descrição

Controladores de lógica difusa, do inglês *Fuzzy Logic Controller* - FLC, são sistemas de controle que utilizam o raciocínio difuso para controlar as suas ações (SIMÕES; SHAW, 2007). É possível construir um bom sistema de controle baseado apenas num conjunto de regras do tipo *SE-ENTÃO* introduzidas por um especialista, sem que este tenha algum conhecimento sobre a modelagem matemática do problema em questão (GERLA, 2005).

O processo de raciocínio difuso, por si, já pode ser considerado um método de controle, pois a partir de variáveis de entrada, consegue inferir saídas de acordo com as regras.

O conjunto de regras do FLC pode ser alterado para trabalhar com o erro e entre uma variável do sistema y e a referência y_{sp}. Desta maneira, o controle trabalharia de modo similar a um controlador proporcional: *se erro alto então correção alta.*

Adicionalmente, controles sobre os termos integral e derivativo são incorporados, tornando o processo de correção mais preciso e completo. Este tipo de controlador é chamado de PID FLC. O erro e é calculado conforme a Equação (2.28), enquanto os termos integral (I) e derivativo (D) são dados, em função do momento t, por:

$$I(t+1) = I(t) + e(t) \tag{3.1}$$

$$D(t+1) = e(t+1) - e(t) \tag{3.2}$$

Desenvolver um PID FLC, ou seja com os três modos de controle, implica em especificar uma grande quantidade de regras. Para criar a FAM (*Fuzzy Associative Matrix*), será necessário especificar n^i saídas, onde n é o número de termos linguísticos usados, ou valores verdade, na *fuzzificação* e i é o número de entradas no controlador.

3.1.2 Controlador Difuso PI/PD

Devido a simplicidade envolvida no desenvolvimento, comumente escolhe-se um PD ou PI FLC para ajuste de um sistema de baixa complexidade. A partir de pequenas modificações, pode-se adaptar um controlador PD para assumir um comportamento de um PI ou até mesmo PID.

Em (XU; HANG; LIU, 2000), por exemplo, encontra-se uma estruturação de dois controladores PD FLC e PI FLC paralelamente. A saída destes é combinada de maneira que a resposta seja equivalente a um PID FLC.

Utilizando-se um controlador PD, entretanto, é possível adequar a sua saída de modo que o seu comportamento seja equivalente a um controlador PI (OLIVEIRA, 1997). Isto é feito integrando a saída do controlador a uma variável observável do sistema. A equação de um controlador PD é dada por:

$$u(t) = K_1 \cdot e(t) + K_2 \cdot \frac{de(t)}{dt} \tag{3.3}$$

de modo que, integrando os dois termos, temos que:

$$\int u(t)dt = \int K_1 \cdot e(t)dt + \int K_2 \cdot \frac{de(t)}{dt}dt \tag{3.4}$$

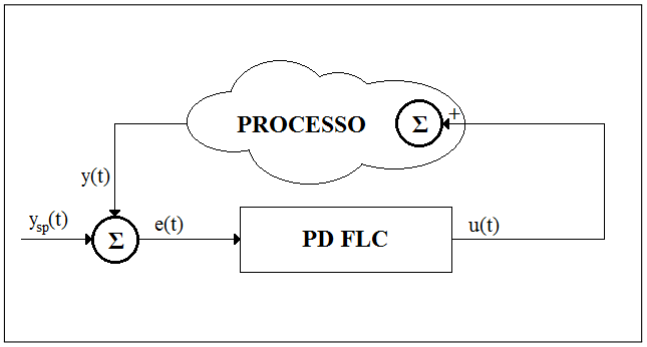
resultando em um controlador PI dado por:

$$\int u(t)dt = K_1 \int e(t)dt + K_2 \cdot e(t) \tag{3.5}$$

onde K_1, originalmente tratado como o ganho proporcional, passa a ser o ganho integral. Da mesma forma, K_2 é agora o ganho proporcional.

Como esta lógica pode ser estendida também aos controladores difusos, tem-se que um controlador PI FLC pode ser representado por um PD FLC, apenas acumulando o resultado do controlador ao processo. A Figura 18 traz o esquema de um controlador PD FLC utilizado como PI FLC. Note que a entrada do controlador continua sendo o erro e e a taxa do erro no tempo, dada pela Equação (3.2).

Figura 18: Estrutura de um controlador difuso PI usando um controlador difuso PD



3.1.3 Inferência

O mecanismo de inferência deste controlador é simples e foi desenvolvido para trabalhar com indicadores populacionais, causando alterações nos parâmetros evolutivos (OLIVEIRA, 1997). Estes indicadores populacionais são variáveis observáveis do processo evolutivo, como a quantidade de membros na população ou o número de indivíduos que foram eliminados.

O PI FLC utilizado neste trabalho foi adaptado para controlar a população de indivíduos, mantendo-a dinâmica, entretanto induzindo um comportamento predefinido. Espera-se que o tamanho da população (N) oscile ao redor da referência.

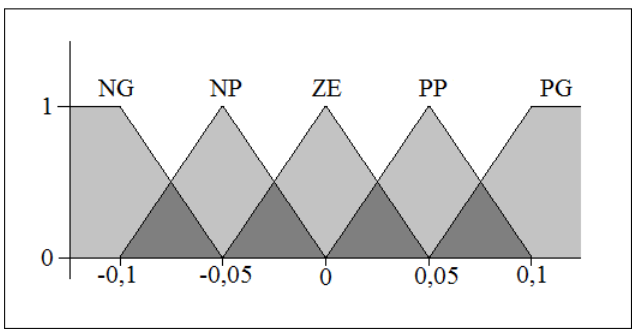
O controlador recebe como entrada os dados usuais para um PD FLC. O erro e e sua derivada são então *fuzzificados* utilizando uma função de pertinência triangular (μ_{tri}) para transformar os valores em cinco valores verdade, ou rótulos, sendo estes:

- NG - Negativo Grande
- NP - Negativo Pequeno
- ZE - Zero
- PP - Positivo Pequeno
- PG - Positivo Grande

Através de tentativa e erro, foram encontrados valores para delimitar (especificar intervalo da função de pertinência) os rótulos acima. Valores de entrada menores ou maiores do que o intervalo são tratados como um dos extremos, evitando que não exista entrada que não possa ser *fuzzificada*.

Na Figura 19 vê-se como estão organizados os subgrupos *fuzzy* no processo de *fuzzificação*. Os rótulos NG, NP, ZE, PP e PG são delimitados pelos intervalos $(-\infty; -0,05)$, $[-0,1; 0)$, $[-0,05; 0,05)$, $[0; 0,1)$, $[0,05; +\infty)$, respectivamente.

Figura 19: Visualização do processo de *fuzzificação* das entradas do PI FLC



As regras de ativação deste controlador são disparadas através de composição de valores *fuzzificados* de entrada. A operação FAND, também conhecida como conjunção difusa e descrita através da Equação (2.25), foi usada para combinar as regras e inferir um resultado. Este operador foi escolhido com base no conhecimento empírico do problema, onde as regras linguísticas, do tipo *SE-ENTÃO*, são estruturadas como o exemplo na regra 3.6.

$$\textbf{\textit{SE}}\ e\ \textbf{\textit{É}}\ PG\ \textbf{\textit{E}}\ \frac{de}{dt}\ \textbf{\textit{É}}\ PG\ \textbf{\textit{ENTÃO}}\ u\ \textbf{\textit{É}}\ NG \tag{3.6}$$

A base de regras é representada por uma matriz de associação *fuzzy*, encontrada na Figura 20. Por se tratar de um problema simples, as regras foram definidas de acordo com a resposta esperada de um controlador.

Figura 20: Matriz de associação *fuzzy* para o controlador PI FLC

	NG	NP	ZE	PP	PG
NG	PG	PG	PP	ZE	ZE
NP	PG	PP	PP	ZE	ZE
ZE	PP	PP	ZE	NP	NP
PP	ZE	ZE	NP	NP	NG
PG	ZE	ZE	NP	NG	NG

Para a última etapa de inferência, *defuzzificação*, utilizam-se os mesmos rótulos presentes no processo de *fuzzificação*. Os valores selecionados para delimitar estes, no entanto, são diferentes. Através de tentativa e erro, os centros de cada subgrupo são definidos em $-0,5$; $-0,1$; 0; $0,1$ e $0,5$; para NG, NP, ZE, PP e PG, respectivamente.

A técnica de centro de massa é utilizada para dar uma resposta no mesmo domínio do problema. Tendo em vista que o controlador será muito utilizado durante a evolução, procurou-se ao máximo simplificar o cálculo para consumir o mínimo possível de poder computacional.

Portanto, uma alternativa discreta à Equação (2.27) é utilizada:

$$c_m = \frac{\sum_{i=1}^{q} z_i \cdot \mu(z_i)}{\sum_{i=1}^{q} \mu(z_i)} \tag{3.7}$$

onde q é o número de subconjuntos *fuzzy*; z_i é o centro do triangulo representando o subconjunto i; μ é o valor retornado pela função de pertinência para o rótulo específico.

No final desta etapa, tem-se que o controlador está pronto para ser incorporado ao ATP, alterando o comportamento da população.

3.2 Algoritmo Evolutivo Híbrido Difuso

Nesta sessão do trabalho, aborda-se detalhes da integração entre o ATP e o controlador PI FLC descrito anteriormente.

3.2.1 Descrição

O Algoritmo de Treinamento Populacional Difuso - ATPD, aqui abordado, é resultante da integração do controlador PI FLC, descrito na sessão anterior, ao Algoritmo de Treinamento Populacional - ATP (OLIVEIRA, 2004) original.

Embora o controlador inserido trabalhe explicitamente com o tamanho da população (N), recebendo uma referência para este valor, é incorreto dizer que o ajuste será aplicado diretamente sobre N.

O ATP calcula um coeficiente não construtivo δ para cada solução, definido pela Equação (2.17), utilizado para organizar os indivíduos na população de acordo com a sua adaptação heurística. Um parâmetro evolutivo α, originalmente dado pela Equação (2.18), é utilizado para atualizar esta população, eliminando os indivíduos que estejam abaixo deste valor. Em momento algum, no ATP, é definido o tamanho de N.

Neste cenário, definir explicitamente um tamanho de população implicaria em alterar rotinas de atualização populacional, culminando na descaracterização do ATP original. Para minimizar as alterações feitas nesta versão do ATP, e consequentemente evitar a invalidação, propõe-se que o controlador PI FLC atue sobre o parâmetro α de acordo com:

$$\alpha^t = \alpha^{t-1} + \Delta\delta^t \cdot u(t) \tag{3.8}$$

Observa-se na Equação (3.8), a presença de um termo de ajuste de passo ($\Delta\delta^t$). Este fora adicionado para adequar a ação de controle, fazendo um ajuste maior ou menor, dependendo da homogeneidade da população. Em momentos, evita-se que o controlador tome uma ação de ajuste drástica, vindo a eliminar toda a população. Em outros, acelera-se o crescimento de α na tentativa de evitar uma explosão populacional.

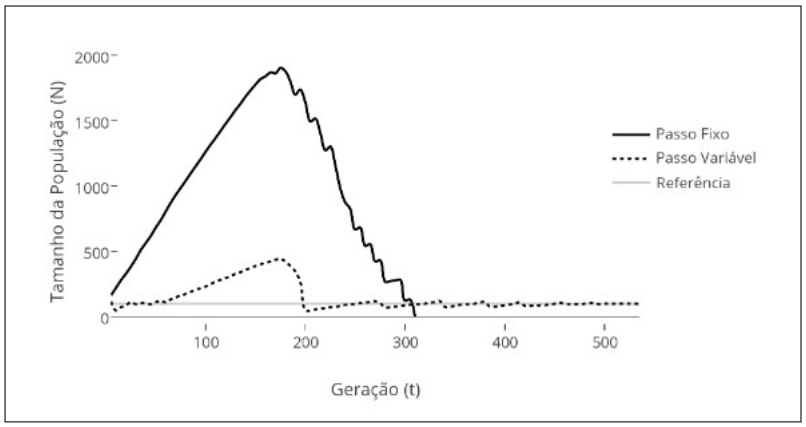
Note que o controlador trabalha somente com o erro e a taxa da variação do erro no tempo. Portanto, não há qualquer ação de controle relacionada à homogeneidade da população. O ajuste de passo é dado por:

$$\Delta\delta^t = \delta^t(x_1) - \delta^t(x_N) + u_\delta \tag{3.9}$$

onde u_δ é um *bias* utilizado para garantir que o passo não será zero quando toda a população tiver o mesmo coeficiente não construtivo; $\delta^t(x_1)$ é o coeficiente não construtivo do indivíduo mais bem avaliado da população. Do mesmo modo, $\delta^t(x_N)$ está em função do indivíduo pior avaliado.

Na Figura 21, vê-se um gráfico comparando a ação de controle com passo fixo e variável. Nas primeiras gerações, o crescimento de α, para o controlador onde o passo é constante (traço sólido escuro), não acompanha a expansão da população, deixando esta livre para crescer. Por volta da geração 300, o passo constante torna a ação do controlador excessiva e elimina toda a população.

Figura 21: Comportamento da população com e sem ajuste de passo para o controlador



O passo variável (traço pontilhado no gráfico da Figura 21), entretanto, fornece um ajuste melhor da ação de controle sobre α. Para as primeiras gerações, quando a disparidade dos coeficientes δ dos indivíduos ainda é grande, o passo do controlador cresce e controla a população de maneira mais eficiente. Com o avanço das gerações, o passo tende a diminuir, dando ajustes menores.

Pode-se dizer que o parâmetro evolutivo α reflete a condição ambiental, sendo esta mais favorável, ou ótima, quando α for baixo. Desta forma, espera-se ter uma população maior, menos especializada, para um valor baixo de α. Enquanto para valores maiores, uma população reduzida, porém de melhor qualidade, é aguardada.

3.2.2 Comportamento da Referência

Três comportamentos diferentes para a referência foram definidos, com a finalidade de avaliar o bom funcionamento do controlador. Estes foram chamados de:

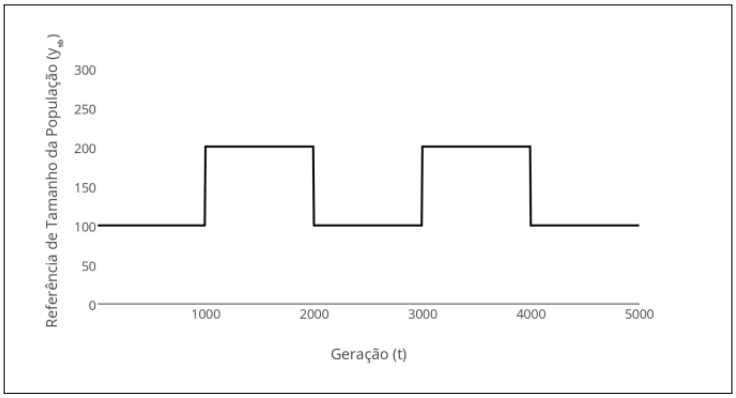
- Comportamento Constante
- Comportamento de Sinal
- Comportamento com Decaimento

O primeiro deles, comportamento constante, foi denominado desta maneira por não apresentar modificações na referência durante a evolução, se mantendo, portanto, constante. O comportamento de sinal tem este nome devido ao gráfico de sua funcionalidade ser muito

similar à representação gráfica de um sinal digital binário. O último, comportamento com decaimento, funciona de maneira similar à atualização de α utilizada na Equação (2.18).

O comportamento de sinal é alcançado a partir da multiplicação do valor de referência por um membro do vetor de troca. Neste trabalho, cada membro do vetor é definido de maneira que a referência alterne-se entre seu valor dobrado e original. A Figura 22 a seguir mostra graficamente o comportamento da referência y_{sb} ao longo das gerações t. É originalmente dado o valor 100 à referência, alterada a cada 1000 gerações.

Figura 22: Variação da referência y_{sp} de acordo com o comportamento de sinal



A fim de verificar se as alterações realizadas no ATP, criando o ATPD, não o descaracterizam como tal, o comportamento com decaimento foi introduzido. O valor da referência, neste caso, decai com a proximidade do número máximo de gerações introduzidas. Isto tende ao esvaziamento da população próximo do limite da evolução. A atualização da referência para a geração t atual é dada por:

$$y_{sp}^{t} = y_{sp}^{t-1} - \theta \tag{3.10}$$

onde θ é o decaimento da referência, dado por:

$$\theta = \frac{N - N_{min}}{T} \tag{3.11}$$

onde N_{min} é o número mínimo exigido de membros na população e T é o número total de gerações.

Observa-se na Equação (2.18), atualização original do ATP, que sempre haverá um aumento em α. Este aumento é realizado de maneira controlada para que a população seja esvaziada próxima do limite da evolução. O comportamento com decaimento, entretanto, permite que α assuma um valor maior ou menor que o anterior, dependendo do tamanho da população e sua referência atual.

Embora o efeito observado na natureza do tamanho de uma população não seja replicado pelo controlador, espera-se que a variação em N introduza um comportamento mais próximo do natural.

4 Experimentos Computacionais

Este capítulo dedica-se a apresentar os experimentos realizados e seus respectivos resultados. O dinamismo populacional decorrente do controlador lógico *fuzzy* é avaliado, bem como a qualidade das soluções produzidas para os problemas de sequenciamento de padrões.

Todos os testes utilizaram instâncias MOSP e GMLP encontradas na literatura[1]. Os experimentos foram realizados em máquinas de hardware diferentes, portanto, o tempo de execução não será avaliado neste trabalho.

4.1 Controle Populacional

4.1.1 Visão Geral

O controle populacional foi testado utilizando-se diferentes comportamentos de referência. A importância destes experimentos está em mostrar o efeito da variação populacional em torno da referência atual, permitindo avaliar o funcionamento do controlador.

Para cada comportamento de referência, foram executas instâncias MOSP com dois valores de referência diferentes ($y_{sp}^1 = 100$ e $y_{sp}^2 = 1000$). Os demais parâmetros foram ajustados (OLIVEIRA, 2004) da seguinte maneira:

- Constante de Equilíbrio $d = \frac{1}{G_{max}}$;
- Atualização de α adaptativa, realizada através de um controlador PI FLC com ajuste de passo $\Delta\delta$;
- Percentual de indivíduos de elite $\rho = 0,2$;
- Percentual de mutação $\rho_m = 0,2$;
- Número máximo de trocas de genes durante o 2-Opt $l = 20$;
- Número máximo de vizinhanças 2-Opt avaliadas $m = 20$.

Em alguns experimentos (Figuras 23, 25, 27 e 32, a seguir) com $y_{sp} = 100$ observou-se um pico de crescimento populacional nas primeiras gerações. Nestas, a diversidade apresenta níveis moderados porém o ajuste de passo $\Delta\delta$ é baixo. Como consequência, permite-se que a população cresca além do desejado mas evita-se que evolução seja interrompida precocemente.

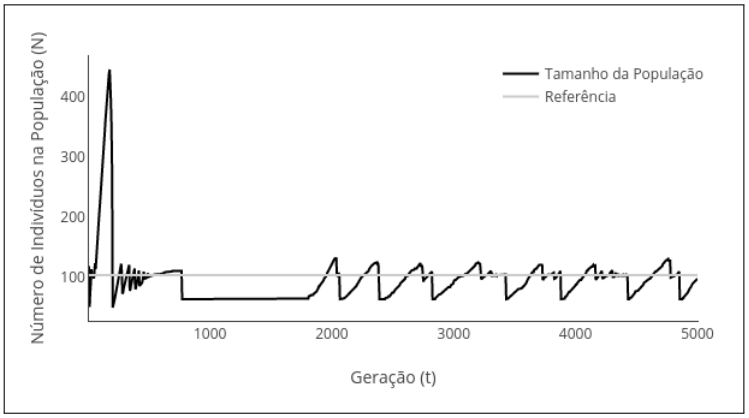
[1] Instâncias disponíveis em <http://www.lac.inpe.br/~lorena/instancias.html>

4.1.2 Comportamento Constante

Uma série de experimentos foi executada para observar-se como o tamanho da população reage à variação de α imposta pelo controlador. Verificou-se nas instâncias uma oscilação de N em torno da referência, efeito este esperado.

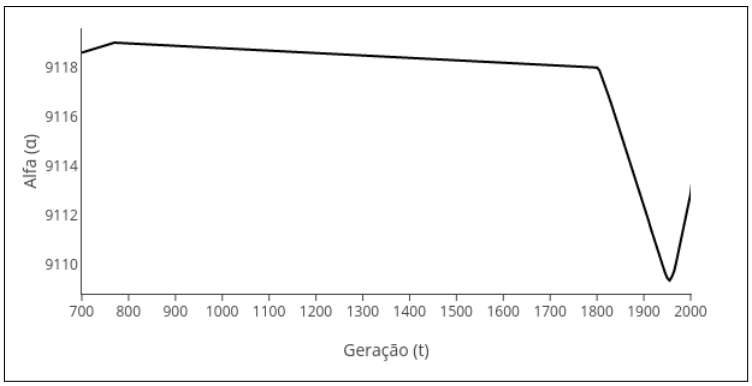
O gráfico da Figura 23 mostra a variação da população durante as gerações para $y_{sp} = 100$. Este gráfico foi escolhido por resumir boa parte do funcionamento do controlador observado nos experimentos. Percebe-se, logo no início da evolução, um ponto onde há diminuição em N. Este permanece constante desde então e, após algumas gerações, volta a variar.

Figura 23: Efeito do controle populacional para referência constante de valor 100



Durante este período onde N foi constante, entretanto, é possível ver através do gráfico de α na Figura 24 que o controlador manteve o esforço, diminuindo a exigência e, consequentemente, aceitando novas soluções.

Figura 24: Gráfico da variação de alfa entre as gerações 700 e 2000



O efeito observado no gráfico da Figura 24 é explicado pelo fato de que a população converge nas proximidades da geração 800. Deste modo, o $\Delta\delta$ é determinado apenas pelo *bias*, pois toda a população apresenta o mesmo coeficiente não construtivo. Próximo da geração 1800, o controlador consegue reduzir α o suficiente para que ele aceite novas

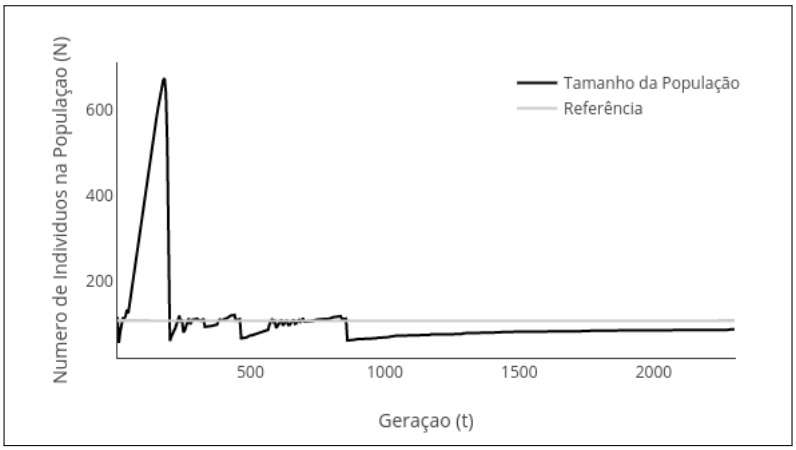
soluções (cruzamento e mutação). A partir deste ponto, $\Delta\delta > u_b$. Estas novas soluções introduzem novamente diversidade à população.

Em alguns casos, esta recuperação após a convergência pode ser muito difícil de ocorrer. Embora o ajuste de passo jamais seja nulo devido a presença do *bias*, o produto de $\Delta\delta$ pela resposta do controlador pode resultar em um valor tão baixo que este será truncado para zero.

O cenário descrito acima ocorrerá quando a convergência populacional acontecer com N próximo da referência. Neste caso, a ação de resposta do controlador terá valor aproximado a zero, anulando a atualização de α. Se não ocorrer nenhuma mutação ou cruzamento de indivíduos, a população permanecerá a mesma até o fim da evolução.

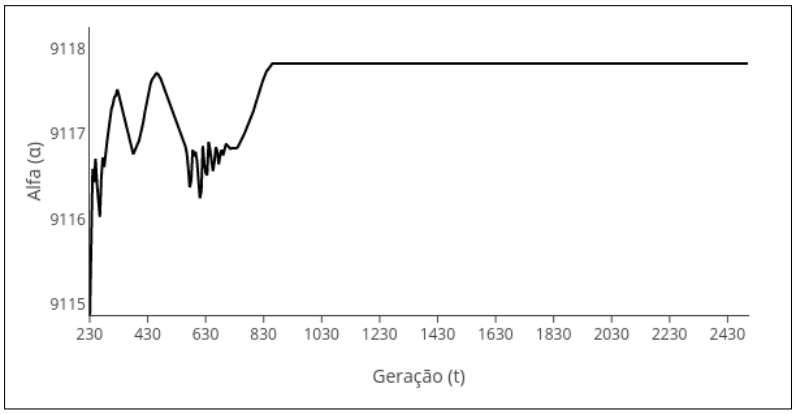
O comportamento descrito pode ser observado através do gráfico na Figura 25. Próximo da geração 1000, a população converge, legando $\Delta\delta$ ao *bias*. A ação de controle do PI FLC, devido ao baixo erro, é baixa, produzindo uma atualização nula de α.

Figura 25: Efeito do controle populacional convergindo com N próximo de $y_{sp} = 100$



A partir do gráfico da Figura 26, percebe-se o momento em que α se torna constante.

Figura 26: Gráfico da variação de alfa a partir da geração 230



Para os experimentos realizados, a maioria apresentou pontos de convergência. Neste grupo de experimentos, 50% das vezes não houve recuperação após a convergência. O comportamento onde $y_{sp} = 1000$ foi semelhante nos experimentos realizados.

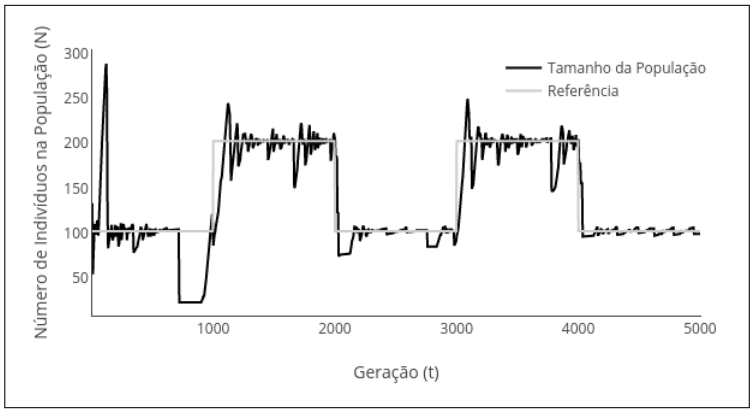
4.1.3 Comportamento de Sinal

As mesmas instâncias executadas para o comportamento constante foram aplicadas ao ATPD com comportamento de sinal. Em geral, os efeitos observados no comportamento anterior, como convergência e recuperação, também foram observados nestes experimentos e, portanto, não necessitarão ser explicados novamente.

A principal finalidade deste comportamento é avaliar a funcionalidade do controlador no que diz respeito à perseguição da referência. Para isto, a referência é alternada entre o seu valor original e o dobro do valor.

Observa-se na Figura 27 o gráfico do controle populacional de um experimento executado utilizando $y_{sp} = 100$. A cada 1000 gerações, entretanto, a referência é trocada. Vê-se que o controle funciona como o esperado, mantendo a população sempre próxima da referência, mesmo quando esta é alterada.

Figura 27: Efeito do controle populacional com comportamento de sinal

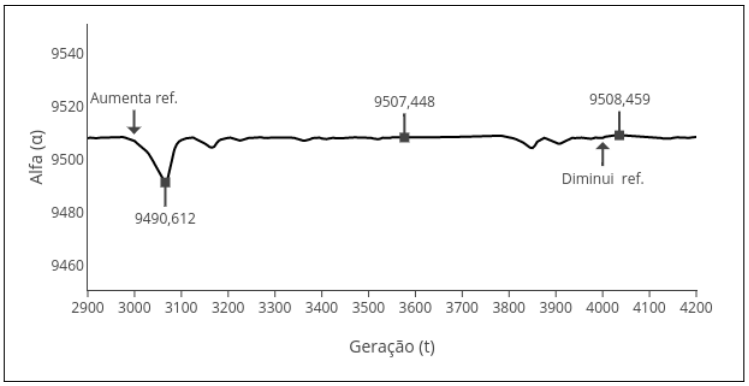


Para alcançar a referência populacional, é esperado que logo após a troca de y_{sp}, o controlador atue aumentando ou diminuindo o valor de α. O efeito esperado é observado no gráfico presente na Figura 28, entretanto vê-se que a variação é maior quando a referência é aumentada.

Embora neste trabalho não tenha sido realizada nenhuma investigação sobre o efeito acima, acredita-se que este esteja relacionado à diversidade da população. Devido a alta taxa de similaridade entre os indivíduos da população, uma pequena variação em α seria suficiente para eliminar metade da população.

Durante as execuções, verificou-se a provável existência de uma relação entre y_{sp} e o tamanho do intervalo entre trocas da referência. Os experimentos realizados, para os

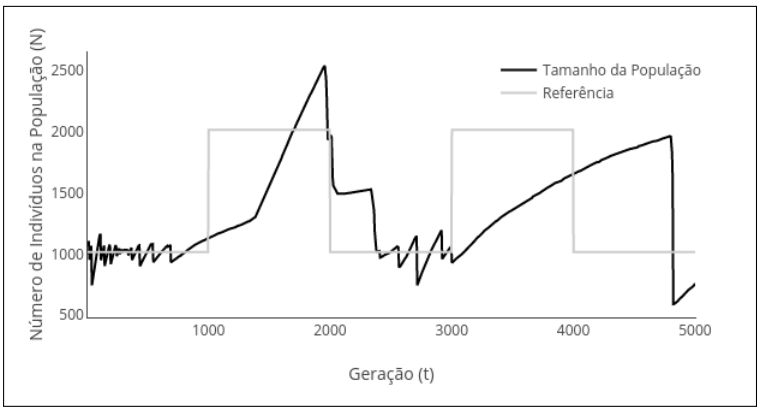
Figura 28: Gráfico da variação de α após troca de referência



quais o período de troca era definido com número de gerações menor ou igual à referência inicial, apresentaram resultados ruins para o controle.

Não foi formulada relação matemática para encontrar o tamanho ideal do intervalo entre trocas, todavia, observou-se que ao aumentar este, um melhor acompanhamento à referência foi imposto pelo controlador. Através dos gráficos presentes na Figura 29 e 30 pode-se observar o controle aplicado à mesma instância MOSP, porém com intervalo de troca diferente. Nota-se que, no primeiro gráfico, há mais dificuldade em manter proximidade da referência quando esta troca, enquanto no segundo aplica-se um melhor controle.

Figura 29: Gráfico da população com troca de y_{sp} a cada 1000 gerações

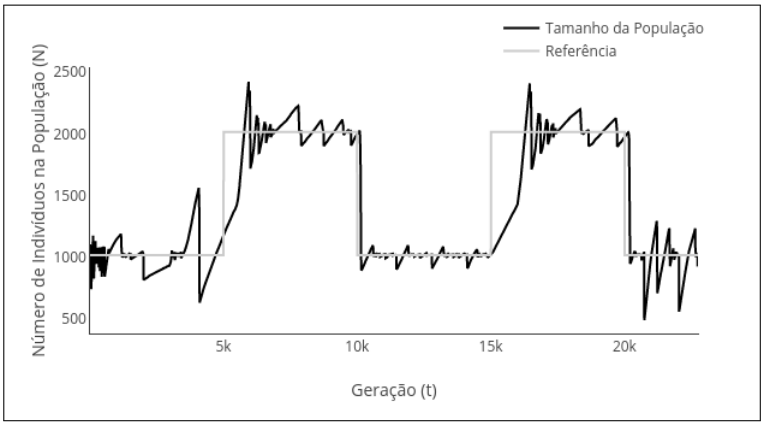


4.1.4 Comportamento com Decaimento

Com a finalidade de aproximar o comportamento da população do ATPD ao efeito observado no ATP, aplicou-se um decaimento à referência. Embora, diferentemente do ATP, o controle aplica atualizações em α decrementando o seu valor, o efeito de aumento da especialização da população foi observado para a maioria dos experimentos realizados.

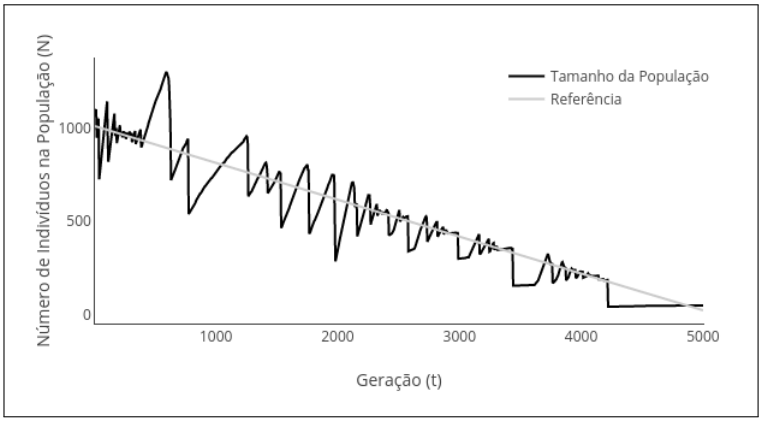
Observou-se nos experimentos um bom desempenho do controlador, no que diz respeito ao acompanhamento da referência. O gráfico da Figura 31 traz o comportamento

Figura 30: Gráfico da população com troca de y_{sp} a cada 5000 gerações



observado em um dos experimentos. O critério de parada, neste caso, foi acionado para o fim da evolução.

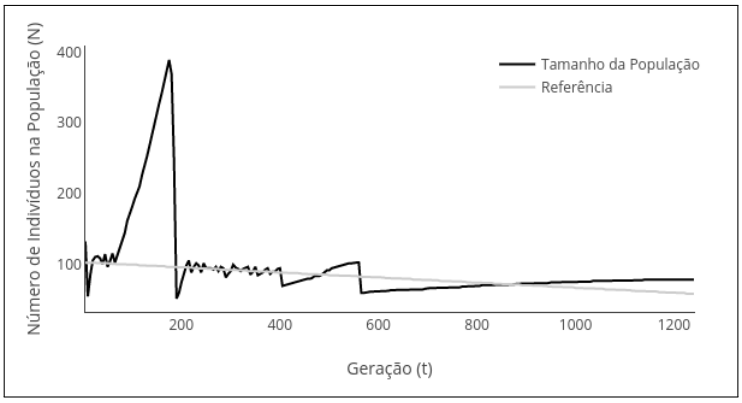
Figura 31: Gráfico da população com decaimento de y_{sp} para 5000 gerações



Com a diminuição de y_{sp}, cada vez menos indivíduos restam na população. Graças a política de poda do ATPD, herdada do ATP, os indivíduos tendem a convergir próximos do fim da evolução. Desta forma, apesar da referência reduzir para N_{min} na última geração, o critério de parada foi somente acionado ou quando a população atingiu seu tamanho mínimo, ou quando atingiu-se o limite de gerações, para os experimentos.

Viu-se que vários experimentos com baixos valores iniciais de y_{sp} convergiram antes do normal. Isto explica-se pela baixa diversidade decorrente do constante aumento da exigência da população (aumento de α). A Figura 32 traz o gráfico de um destes experimentos onde a convergência acontece próxima da geração 600. Posteriormente, após a geração 1200, α cresce o suficiente para eliminar toda a população, encerrando a evolução bem antes do desejado.

Figura 32: Exemplo de comportamento com decaimento onde há convergência



4.2 ATPD Aplicado a Instâncias MOSP

O ATPD como descrito neste trabalho foi aplicado a instâncias MOSP e comparado com outras abordagens disponíveis na literatura: uma busca tabu (TS) e uma busca local generalizada (GLS) sobre uma busca tabu, utilizando-se a heurística de *Faggioli-Bentivoglio* (FAGGIOLI; BENTIVOGLIO, 1998). As principais soluções a se comparar, entretanto, são as encontradas através do ATP, extraídas de (OLIVEIRA, 2004).

O conjunto de problemas MOSP encontrado em (FAGGIOLI; BENTIVOGLIO, 1998) foi utilizado. Tratam-se de 150 instâncias com diferentes números de padrões e itens. Somente problemas com padrões $I \in \{10, 15, 40\}$ e itens $J \in \{10, 20, 30, 40, 50\}$ foram utilizadas nos experimentos. A razão para tal é que as instâncias com quantidade de padrões 15 e 40 são consideradas as mais difíceis.

O ATPD foi ajustado como na seção anterior, com exceção da referência inicial y_{sp} ter sido estabelecida em 50.

A Tabela 1 traz as médias de *mpa* resultantes das abordagens para cada instância. Três colunas são destinadas para as médias resultantes do ATPD: CC, CS e CD. Cada uma delas representa um comportamento de referência. A primeira, remete-se ao comportamento constante; a segunda ao comportamento de sinal e a terceira refere-se ao comportamento com decaimento.

Os resultados são disponibilizados em média, seguindo o modelo de avaliação apresentado nas publicações da área. Vê-se através da Tabela 1 que o ATPD obteve médias similares ao ATP, tendo média diferente em apenas dois conjuntos de instâncias grandes (*p4040* e *p4050*).

As médias obtidas utilizando-se o ATP são menores que as outras abordagens. Para poucas execuções, o ATPD teve médias superiores aos valores do ATP. Entretanto, dada a baixa diferença entre os valores e a utilização da mesma filosofia de evolução nas duas abordagens, pode-se dizer que estas são semelhantes.

Tabela 1: Médias encontradas para cada grupo de instâncias MOSP

IxJ	**TS**	**GLS**	**ATP**	**CC**	**CS**	**CD**
10x10	5,5	5,5	5,5	5,5	5,5	5,5
10x20	6,2	6,2	6,2	6,2	6,2	6,2
10x30	6,1	6,2	6,1	6,1	6,1	6,1
10x40	7,7	7,7	7,7	7,7	7,7	7,7
10x50	8,2	8,2	8,2	8,2	8,2	8,2
15x10	6,6	6,6	6,6	6,6	6,6	6,6
15x20	7,2	7,5	7,2	7,2	7,2	7,2
15x30	7,4	7,6	7,3	7,4	7,3	7,3
15x40	7,3	7,4	7,2	7,2	7,2	7,2
15x50	7,6	7,6	7,4	7,4	7,4	7,4
40x10	8,4	8,4	8,4	8,5	8,4	8,4
40x20	13,1	13,1	13,0	13,1	13,0	13,1
40x30	14,7	14,6	14,5	14,9	14,9	14,5
40x40	15,3	15,3	14,9	15,1	15,1	15,1
40x50	15,3	14,9	14,6	14,7	14,7	14,7

4.3 ATPD Aplicado a Instâncias GMLP

Instâncias GMLP também foram aplicadas ao ATPD, embora não tenham sido testadas no ATP original. Utilizou-se um conjunto de quatro instâncias retiradas de (LINHARES; YANASSE; TORREÃO, 1999). O ajuste de parâmetros foi o mesmo apresentado na primeira seção deste capítulo.

A Tabela 2 detalha as instâncias utilizadas, apresentando o número de portas e redes de cada uma, bem como a sua solução ótima. Estas fazem parte do grupo das maiores instâncias encontradas na literatura.

Tabela 2: Instâncias GMLP utilizadas

Instância	**Portas**	**Redes**	**Solução**
W1	21	18	4
W2	33	48	14
W3	70	84	18
W4	141	202	27

Os resultados obtidos pelo ATPD foram comparados com os do AGC^H. Esta foi escolhida por ser a versão construtiva do TPH, mesma família do ATP original (OLIVEIRA; LORENA, 2002a).

A Tabela 3 apresenta um comparativo entre as soluções obtidas pelo ATPD e o AGC^H para até 10 execuções. Três colunas são destinadas para resultados do ATPD: CC, CS e CD, significando ATPD com comportamento constante, comportamento de sinal e comportamento com decaimento, respectivamente.

Tabela 3: Comparação entre ATPD e AGC^H para instâncias GMLP

Instância	**AGC^H**	**CC**	**CS**	**CD**
W1	4	4	4	4
W2	14	14	14	14
W3	18	18	18	18
W4	27	28	34	28

Embora o ATPD tenha encontrado a maioria das soluções ótimas em até 10 execuções, o acerto é menor ao ser comparado com o AGC^H. Para as instâncias pequenas (até 40 portas), ambos encontram a solução ótima em 100% das vezes. As soluções maiores obtiveram menor número de acertos. Não houve acerto para a instância *W4*. No entanto, para a instância *W3* encontrou-se a solução ótima em 30%, 40% e 20% das vezes para CC, CS e CD, respectivamente.

4.4 Discussão

Os dados expostos na primeira seção deste capítulo corroboram o bom funcionamento do controlador adaptativo *fuzzy* aplicado ao ATP. Para todos os comportamentos avaliados, o controlador conseguiu impor um efeito de acompanhamento da referência.

Durante a realização destes experimentos, muito tempo foi despendido no ajuste por tentativa e erro de alguns poucos parâmetros (intervalo entre trocas, *bias*, etc.). Isto, combinado ao fato de que alguns comportamentos inesperados aconteceram (como por exemplo a convergência próxima da referência), consistiu a dificuldade encontrada.

Viu-se que parte dos problemas encontrados durante os experimentos estão envolvidos com a homogeneidade da população. É provável que a adição de mecanismos que visam o ajuste da diversidade possa contribuir para o controle populacional.

Embora as soluções encontradas pelo ATPD, apresentadas na segunda e terceira seção do capítulo atual, não apresentem nenhuma melhoria em relação ao ATP original, o controle adaptativo incorporou aspectos ao processo evolutivo que podem ser úteis. Por exemplo, observou-se que em vários casos, o controlador conseguiu recuperar a população após uma convergência.

Enfim, o ATPD teve resultados comparáveis ao enfoque original, ATP. Estes resultados sugerem que não houve descaracterização do processo evolutivo original após a introdução do controle e que há semelhança entre estas duas abordagens.

5 Conclusão

Este trabalho apresentou um controle *fuzzy* adaptativo aplicado a um algoritmo de treinamento populacional para solução de problemas de sequenciamento de padrões. Para concluir esta tarefa, foi necessário realizar pequenas alterações ao ATP, sem mudar, entretanto, seu processo evolutivo característico.

No ATPD, utiliza-se uma população dinâmica e ordenada através de um coeficiente não construtivo, também chamado de *ranking*. Posteriormente, a população é atualizada usando um parâmetro adaptativo de exigência. Somente indivíduos que têm *ranking* maior que este parâmetro permanecerão na população da próxima geração. O controlador *fuzzy* adaptativo, incorporado à evolução, trabalha diretamente neste parâmetro de exigência, incrementando ou decrementando o seu valor de acordo com o tamanho da população atual em relação à referência.

O controle foi testado utilizando-se três comportamentos de referência. Os experimentos realizados mostraram uma boa reação do controlador, conseguindo variar a população em torno do valor de referência. Em vários momentos, o controlador, mesmo não trabalhando com a diversidade da população, conseguiu recuperá-la de uma convergência. Este efeito não era esperado e representa um ponto positivo da utilização do controle.

O resultado para as instâncias MOSP é muito similar ao encontrado pelo ATP original, divergindo apenas nas duas maiores instâncias do conjunto testado – a primeira com 40 itens e 40 padrões e a segunda com 40 padrões e 50 itens. Isto indica uma semelhança entre os algoritmos ATPD e ATP.

Apesar do ATP original não ter sido aplicado ao GMLP, executou-se experimentos para algumas das maiores instâncias da literatura. Viu-se que o ATPD encontrou o ótimo para quase todas as instâncias, falhando apenas na maior. Ainda assim, seu número de acertos foi inferior ao AGC^H, da mesma família do ATP.

De modo geral, o ATPD mostrou-se adequado para resolução de problemas MOSP, estando semelhante ao ATP. O controle *fuzzy* adaptativo apresentou bom funcionamento, agindo de forma adequada. Ressalta-se que o objetivo traçado foi atingido.

5.1 Trabalhos Futuros

No contexto do trabalho desenvolvido, algumas melhorias são sugeridas como trabalhos futuros.

Foi observado que muitos dos problemas relacionados ao controle se resumem à

convergência prematura da população. A adição de um controlador adaptativo que trabalhe com a diversidade pode sanar estes efeitos negativos e melhorar a efetividade do controle.

Foi visto que o controle adaptativo consegue acompanhar a referência de maneira satisfatória e que o ATPD encontra resultados diferentes para diferentes comportamentos de referência. Pode-se, então, investigar a qualidade das soluções do ATPD para outros comportamentos de referência.

Muitos outros problemas podem ser classificados como problemas de sequenciamento de padrões. Pode-se avaliar o desempenho e qualidade das soluções do ATPD para outros PSPs.

Referências

AGGARWAL, S.; GARG, R.; GOSWAMI, P. A review paper on different encoding schemes used in genetic algorithms. *International Journal of Advanced Research in Computer Science and Software Engineering*, v. 4, n. 1, p. 596–600, 2014. Citado na página 18.

AGRAWAL, R. Application of the modified 2-opt and jumping gene operators in multi-objective genetic algorithm to solve motsp. *Journal of Advances in Computer Networks*, v. 1, n. 4, p. 320–322, 2013. Citado na página 24.

ASTROM, K. J.; HAGGLUND, T. Pid controllers: Theory, design, and tuning. *Research Triangle Park: 2nd Ed. Instrumentation, Systems and Automatic Society*, 1995. Citado 5 vezes nas páginas 15, 33, 34, 35 e 36.

BENNETT, S. A brief history of automatic control. *IEEE Control Systems Magazine*, v. 16, n. 3, p. 17–25, 1996. Citado na página 33.

BISHOP, R. H. *The Mechatronics Handbook, -2 Volume Set*. [S.l.]: CRC Press, 2002. Citado 2 vezes nas páginas 15 e 31.

CROES, G. A. A method for solving traveling-salesman problems. *Operations research*, INFORMS, v. 6, n. 6, p. 791–812, 1958. Citado na página 23.

FACCIN, F. *Abordagem inovadora no projeto de controladores PID*. 2004. Citado na página 36.

FAGGIOLI, E.; BENTIVOGLIO, C. A. Heuristic and exact methods for the cutting sequencing problem. *European Journal of Operational Research*, Elsevier, v. 110, n. 3, p. 564–575, 1998. Citado na página 50.

FINK, A.; VOSS, S. Applications of modern heuristic search methods to pattern sequencing problems. *Computers & Operations Research*, Elsevier, v. 26, n. 1, p. 17–34, 1999. Citado 2 vezes nas páginas 14 e 26.

FOGEL, L. J.; OWENS, A. J.; WALSH, M. J. Intelligent decision making through a simulation of evolution. *Behavioral science*, Wiley Online Library, v. 11, n. 4, p. 253–272, 1966. Citado na página 16.

GERLA, G. Fuzzy logic programming and fuzzy control. *Studia Logica*, Springer, v. 79, n. 2, p. 231–254, 2005. Citado 2 vezes nas páginas 15 e 37.

GOLDBERG, D. E. et al. *Genetic algorithms in search optimization and machine learning*. [S.l.]: Addison-wesley Reading Menlo Park, 1989. Citado 2 vezes nas páginas 14 e 20.

GOLUMBIC, M. C. Algorithmic graph theory and perfect graphs. 1980. *Academic, New York*, 1980. Citado na página 27.

HELD, S.; KORTE, B.; RAUTENBACH, D.; VYGEN, J. *Combinatorial optimization in VLSI design*. [S.l.]: Forschungsinst. für Diskrete Mathematik, 2007. Citado na página 28.

KNJAZEW, D.; GOLDBERG, D. E. Large-scale permutation optimization with the ordering messy genetic algorithm. In: SPRINGER. *Parallel Problem Solving from Nature PPSN VI*. [S.l.], 2000. p. 631–640. Citado na página 20.

KOSKO, B. Fuzzy associative memories. In: *NASA, Lyndon B. Johnson Space Center, Proceedings of the 2nd Joint Technology Workshop on Neural Networks and Fuzzy Logic*. [S.l.]: NASA, United States, 1991. v. 1, p. 3–58. Citado na página 31.

KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. [S.l.]: MIT press, 1992. v. 1. Citado na página 16.

LEE, C. C. Fuzzy logic in control systems: fuzzy logic controller. ii. *Systems, Man and Cybernetics, IEEE Transactions on*, IEEE, v. 20, n. 2, p. 419–435, 1990. Citado na página 30.

LINHARES, A. *Industrial pattern sequencing problems: some complexity results and new local search models*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais, 2002. Citado 3 vezes nas páginas 14, 26 e 28.

LINHARES, A.; YANASSE, H. H. Connections between cutting-pattern sequencing, vlsi design, and flexible machines. *Computers & Operations Research*, Elsevier, v. 29, n. 12, p. 1759–1772, 2002. Citado 2 vezes nas páginas 14 e 26.

LINHARES, A.; YANASSE, H. H.; TORREÃO, J. R. Linear gate assignment: a fast statistical mechanics approach. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, IEEE, v. 18, n. 12, p. 1750–1758, 1999. Citado na página 51.

LORENA, L. A. N.; FURTADO, J. C. Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, MIT Press, v. 9, n. 3, p. 309–327, 2001. Citado na página 22.

MINORSKY. Directional stability of automatically steered bodies. *Journal of the American Society for Naval Engineers*, Blackwell Publishing Ltd, v. 34, n. 2, p. 280–309, maio 1922. Citado 2 vezes nas páginas 15 e 33.

MOORE, C.; HETTHESSY, J.; BARS, R. 2.5 control modes—closed-loop response. *Instrument Engineers' Handbook, Volume Two: Process Control and Optimization*, CRC Press, v. 2, p. 135, 2005. Citado na página 35.

MOORE, G. *Cramming More Components Onto Integrated Circuits, Electronics,(38) 8*. 1965. Citado na página 28.

NATSHEH, E.; BURAGGA, K. A. Comparison between conventional and fuzzy logic pid controllers for controlling dc motors. *International Journal of Computer Science Issues*, IJCSI, v. 7, n. 5, p. 128–134p, 2010. Citado na página 31.

OLIVEIRA, A. C. M. de. *Uma Contribuição ao Estudo de Algoritmos Evolutivos Difusos*. 1997. Citado 4 vezes nas páginas 16, 18, 38 e 39.

OLIVEIRA, A. C. M. de. *Algoritmos Evolutivos Híbridos com Detecção de Regiões Promissoras em Espaços de Busca Contínuos e Discretos*. Tese (Doutorado) — Instituto Nacional de Pesquisas Espaciais, 2004. Citado 10 vezes nas páginas 14, 20, 21, 22, 23, 28, 29, 41, 44 e 50.

OLIVEIRA, A. C. M. de; LORENA, L. A. N. 2-opt population training for minimization of open stack problem. In: *Advances in Artificial Intelligence*. [S.l.]: Springer, 2002. p. 313–323. Citado 3 vezes nas páginas 24, 29 e 51.

OLIVEIRA, A. C. M. de; LORENA, L. A. N. Real-coded evolutionary approaches to unconstrained numerical optimization. In: *Advances in Logic, Artificial Intelligence and Robotics*. [S.l.]: Plêiade, 2002. v. 2, p. 10–15. Citado 2 vezes nas páginas 23 e 26.

PEARL, J. Heuristics: intelligent search strategies for computer problem solving. Addison-Wesley Pub. Co., Inc., Reading, MA, 1984. Citado na página 20.

RECHENBERG, I. Cybernetic solution path of an experimental problem. Ministry of Aviation, Royal Aircraft Establishment, 1965. Citado na página 16.

RUTKOWSKI, L. *Computational Intelligence: Methods and Techniques*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2008. ISBN 9783540762874. Citado 2 vezes nas páginas 16 e 17.

SAADE, J. J.; DIAB, H. B. Defuzzification methods and new techniques for fuzzy controllers. *Iranian journal of electrical and computer engineering*, ELECTRICAL ENG. RES. INST. JAHAD, v. 3, n. 2, p. 161, 2004. Citado na página 32.

SALETIC, D.; VELASEVIC, D.; MASTORAKIS, N. Analysis of basic defuzzification techniques. In: *Proceedings of the 6th WSES international multiconference on circuits, systems, communications and computers*. [S.l.: s.n.], 2002. p. 7–14. Citado na página 32.

SIMÕES, M. G.; SHAW, I. S. Controle e modelagem fuzzy. Edgard Blucher, 2007. Citado 2 vezes nas páginas 15 e 37.

XU, J.-X.; HANG, C.-C.; LIU, C. Parallel structure and tuning of a fuzzy pid controller. *Automatica*, Elsevier, v. 36, n. 5, p. 673–684, 2000. Citado na página 38.

YAGER, R. R.; ZADEH, L. A. *An introduction to fuzzy logic applications in intelligent systems*. [S.l.]: Springer US, 1992. v. 165. Citado na página 30.

YANASSE, H. H.; LAMOSA, M. J. P. An integrated cutting stock and sequencing problem. *European Journal of Operational Research*, Elsevier, v. 183, n. 3, p. 1353–1370, 2007. Citado na página 26.

YEN, J.; LANGARI, R. *Fuzzy logic: intelligence, control, and information*. [S.l.]: Prentice-Hall, Inc., 1998. Citado na página 30.

ZADEH, L. A. Fuzzy sets. *Information and control*, Elsevier, v. 8, n. 3, p. 338–353, 1965. Citado 2 vezes nas páginas 15 e 30.

ZADEH, L. A. Making computers think like people. *Spectrum, IEEE*, IEEE, v. 21, n. 8, p. 26–32, 1984. Citado 2 vezes nas páginas 15 e 30.