COORDENAÇÃO DE RELÉS DE SOBRECORRENTE EM SISTEMAS RADIAIS UTILIZANDO ALGORITMO GENÉTICO

Escola Politécnica
Universidade Federal do Rio de Janeiro

Gilvan Rodrigues de Oliveira Junior

RIO DE JANEIRO, RJ - BRASIL
JANEIRO DE 2008

COORDENAÇÃO DE RELÉS DE SOBRECORRENTE EM SISTEMAS RADIAIS UTILIZANDO ALGORITMO GENÉTICO

Gilvan Rodrigues de Oliveira Junior

PROJETO SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO ELETRICISTA.

Aprovada por:

Sandoval Carneiro Jr, Ph.D.
(Orientador)

Paulo Augusto Nepomuceno Garcia, D.Sc.
(Co-orientador)

Sergio Sami Hazan, Ph.D.

Paulo Roberto Maisonnave, Eng.

RIO DE JANEIRO, RJ - BRASIL
JANEIRO DE 2008

OLIVEIRA JUNIOR, GILVAN RODRIGUES DE

Coordenação de Relés de Sobrecorrente em Sistemas Radiais Utilizando Algoritmo Genético [Rio de Janeiro] 2008

VII, xx p. 29,7 cm (Escola Politécnica / UFRJ, Engenharia Elétrica, 2008)

Projeto Final (graduação) – Universidade Federal do Rio de Janeiro, Escola Politécnica, Departamento de Engenharia Elétrica.

1. Proteção Elétrica
2. Seletividade
3. Algoritmo Genético

A Deus, pela força e por
nunca ter me deixado
caminhar sozinho.

AGRADECIMENTOS

Agradeço a minha mãe e minha irmã, pelo carinho e pelo esforço prestado. Em especial, agradeço à minha mãe, Ana Maria, por tudo que fez para que eu pudesse ter a oportunidade de estudar, me dando apoio e sempre fazendo com que eu mantivesse o foco no meu objetivo.

Agradeço às minhas tias: Ângela Maria, Vanda Pereira e Araci de Oliveira, por tudo que fizeram por mim, me dando sempre muito carinho, apoio e durante algum tempo até um teto.

Agradeço também a minha namorada Milena Lima, que sempre esteve comigo nos momentos difíceis, pela paciência, pelo carinho e por sempre me fazer buscar o equilíbrio.

Aos meus amigos, Edilberto, Bruno, Alex, Vítor, Luís Antônio, Denison, Eric e Alexandre, que sempre estiveram ao meu lado e que já passaram muitos momentos tristes e felizes comigo.

Aos engenheiros Paulo Maisonnave, Paulo Fróes e Leonardo Soares pela sua paciência, orientação e ensinamentos.

Aos professores Sandoval Carneiro Junior e Paulo Augusto Nepomuceno pelos seus conselhos e dedicação ao me orientar nesse projeto.

Ao professor Sérgio Sami Hazan pela disponibilidade e esforço em participar da banca examinadora desse projeto final.

Aos meus amigos de faculdade, pela amizade desenvolvida ao longo de toda esta caminhada acadêmica, principalmente pelo apoio nos momentos difíceis e incentivos na superação de todos os obstáculos.

Resumo do Projeto Final apresentado a Poli/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro Eletricista (Eng.º)

COORDENAÇÃO DE RELÉS DE SOBRECORRENTE EM SISTEMAS RADIAIS UTILIZANDO ALGORITMO GENÉTICO

Gilvan Rodrigues de Oliveira Junior

Janeiro/2008

Orientador: Sandoval Carneiro Jr.

Co-orientador: Paulo Augusto Nepomuceno Garcia

Departamento: Engenharia Elétrica

Nos dias de hoje é cada vez mais comum a implementação de sistemas automatizados para a melhoria de processos em geral. Para a realização de estudos de seletividade da proteção elétrica, as ferramentas encontradas no mercado não possuem nenhum tipo de automatização, fazendo com que a experiência do operador seja decisiva para a obtenção de um bom resultado para o estudo. Logo, o desenvolvimento de uma ferramenta baseada em métodos de inteligência artificial, que realize de forma automática a otimização dos tempos de coordenação das proteções elétricas de um determinado sistema será importante para o setor elétrico, pois trará de maneira mais rápida e eficiente os resultados esperados neste tipo de estudo.

Este trabalho explora a utilização do Algoritmo Genético como o método de inteligência artificial a ser aplicado à otimização dos tempos de coordenação de relés de sobrecorrente. Foi desenvolvido no MATLAB um código computacional que possui como saída as curvas corrente/tempo dos relés de sobrecorrente incluídos no estudo de seletividade, procurando trazer ao operador, de forma automática, uma resposta otimizada e dentro dos padrões esperados.

Abstract of Final Project presented to Poli/UFRJ as a partial fulfillment of the requirements for the degree of Electrical Engineer (Engº)

COORDENAÇÃO DE RELÉS DE SOBRECORRENTE EM SISTEMAS RADIAIS UTILIZANDO ALGORITMO GENÉTICO

Gilvan Rodrigues de Oliveira Junior

Janeiro/2008

Orientador: Sandoval Carneiro Jr.

Co-orientador:

Departamento: Engenharia Elétrica

Nowadays, it is becoming more common the implementation of automatic systems to improve the processes quality in general. To achieve the electrical protection coordination studies, the available tools in the market do not present any type of automatism, as such studies are based on the operator's experience to obtain consistent results. Thus, the development of a tool based on artificial intelligence methods to automatically find the coordination times of protection devices that compose a system will be important to the electrical sector bringing faster and efficiently the results expected from this type of study.

This work explores the application of Genetic Algorithm as an artificial intelligence method to be applied to the overcurrent relays coordination time optimization. A computational code was developed in the MATLAB that has as output the current/time curves of the overcurrent relays included in the coordination study, trying to bring to the operator an optimized answer and in accordance to the expected standards, automatically.

LISTA DE TABELAS

LISTA DE QUADROS

ÍNDICE

1. INTRODUÇÃO

1.1. Motivação

A seletividade das proteções elétricas de um sistema é necessária para evitar o corte desnecessário de cargas [23]. O estudo de seletividade consiste na procura da melhor parametrização das proteções elétricas que compõem o sistema, de forma que suas curvas de corrente/tempo não se cruzem ao longo de um determinado intervalo de valores de corrente.

Hoje em dia, os programas utilizados no mercado para este tipo de estudo não utilizam nenhum tipo de automação e dependem muito da experiência do usuário. Portanto, o desenvolvimento de um programa que utilize técnicas de inteligência artificial para otimização dos resultados seria de grande importância para o setor elétrico.

1.2. Objetivo

O objetivo deste trabalho é verificar o potencial de aplicação de Algoritmos Genéticos (AG) no problema COORDENAÇÃO DE RELÉS DE SOBRECORRENTE EM SISTEMAS RADIAIS.

Neste projeto nos limitaremos a estudar a seletividade apenas para relés de sobrecorrente, que são dispositivos de proteção largamente empregados na proteção de sistemas elétricos industriais radiais.

2. SELETIVIDADE DE RELÉS DE SOBRECORRENTE

2.1. Introdução

O estudo de seletividade consiste na coordenação dos tempos de operação das proteções elétricas que compõem um determinado sistema, evitando que suas curvas de corrente/tempo se cruzem ao longo de um determinado intervalo de valores de corrente.

Isto significa que a seletividade da proteção elétrica é uma estratégia de proteção, onde para um ponto n considerado, há um degrau crescente de tempo no sentido do relé próximo desse ponto até os demais relés a jusante, ou seja $t_n > t_{n-1} > t_{n-2} > ... > t_1$, de modo a garantir e permitir a seletividade nos desligamentos do sistema [20].

Essa seletividade consiste de uma seqüência de desligamentos onde o relé mais próximo do defeito atua primeiro. Se esse falhar, deve atuar o primeiro relé a montante com um degrau de tempo, ou tempo de coordenação. Na ocorrência de falha dessa última deve atuar o próximo a montante e assim, sucessivamente [20].

2.2. Tempo de Coordenação (t_c)

Para melhor entendermos os tempos que compõem a coordenação dos relés de sobrecorrente, utilizaremos o sistema mostrado na Figura 2.1 abaixo.

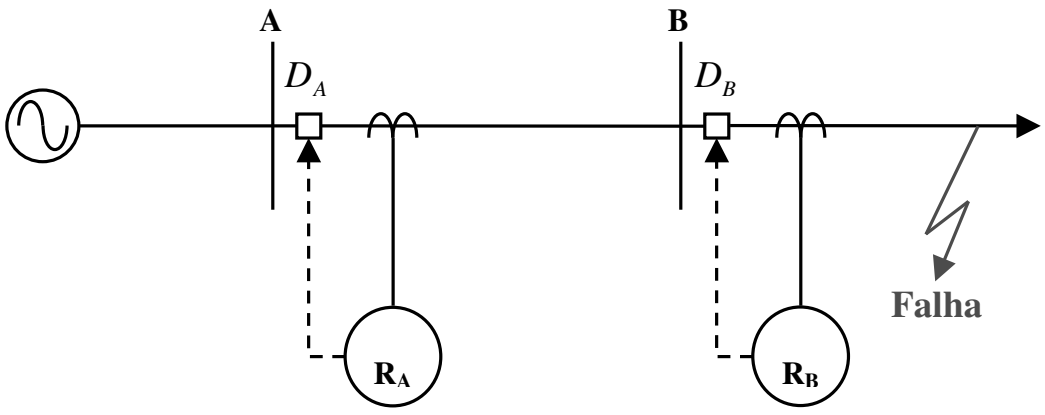


Figura 2.1 – Sistema Elétrico com 2 barras [20]

Nesta figura são mostrados dois relés de sobrecorrente, onde, o relé R_A é responsável por proteger o trecho compreendido entre as barras A e B, e o relé R_B protege o circuito conectado à barra B.

Segundo os conceitos discutidos anteriormente, no caso de falha no ponto indicado na Figura 2.1, o relé responsável pela extinção do defeito deve ser o relé R_B, o mais próximo do ponto da falha, cabendo ao relé R_A atuar apenas se o Disjuntor D_B não abrir de maneira correta.

Neste caso, o tempo total gasto para o relé R_A atuar, caso falhe o conjunto formado pelo relé R_B e o Disjuntor D_B, deverá ser o somatório dos tempos de abertura do disjuntor D_B, tempo de "reset" do relé R_A, tempo de operação do relé R_B e do tempo de segurança definido pelo responsável pela parametrização dos relés de sobrecorrente.

A Figura 2.2 ilustra a composição do tempo de coordenação, segundo suas parcelas.

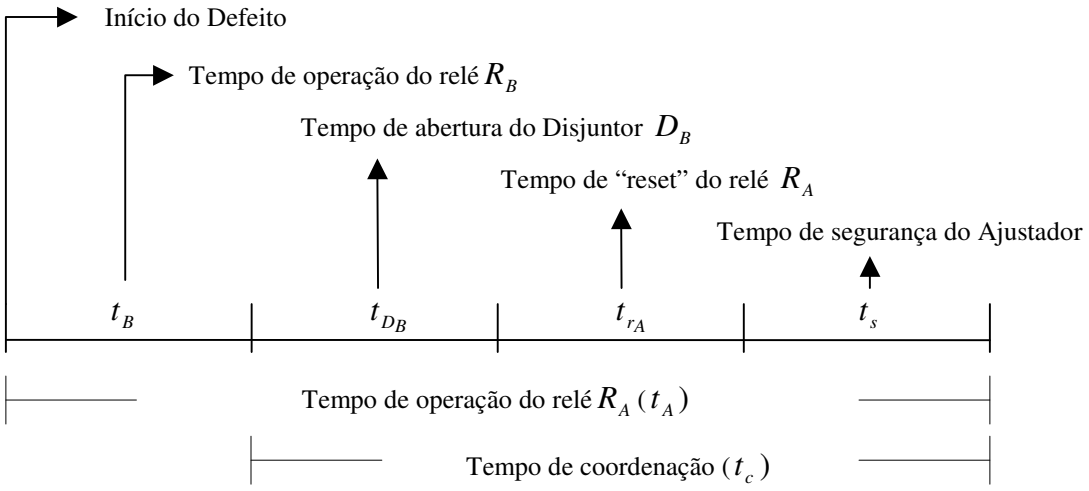


Figura 2.2 – Tempo de Coordenação [20]

Da Figura 2.2, temos que o tempo de coordenação é o somatório dos tempos t_{D_B}, t_{r_A} e t_s, ou seja:

$$t_c = t_{D_B} + t_{r_A} + t_s \tag{2.1}$$

ou

$$t_c = t_A - t_B \tag{2.2}$$

Na prática, podemos utilizar um valor de tempo de coordenação t_c compreendido entre 0,3 e 0,5 segundos [20]. Se no circuito protegido forem utilizados relés de sobrecorrente estáticos microprocessados, cujos tempos de "reset" são iguais a zero, utilizamos t_c igual a 0,3 segundos. Quando utilizados relés de sobrecorrente eletromecânico, cujo tempo de "reset" é de 0,13 segundos, os valores do tempo de coordenação compreendidos entre 0,4 e 0,5 segundos podem ser empregados.

De acordo com a norma IEC 60255-3 [24] o tempo de operação de um relé de sobrecorrente é dado por:

$$t(s) = \frac{k \cdot \beta}{\left(\frac{I}{I>}\right)^{\alpha} - 1} \tag{2.3}$$

onde

t : Tempo de operação, em segundos

k : Dial de tempo, em segundos

I : Corrente de fase, em pu

I>: *Pick up* ajustado, em pu. Ajuste de 2 a 20 x In, onde In é a corrente nominal do sistema

α e β : são constantes

Substituindo (2.3) em (2.2), temos que o tempo de coordenação t_c é dado por:

$$t_c = \frac{k_A \cdot \beta_A}{\left(\frac{I}{I>}\right)^{\alpha_A} - 1} - \frac{k_B \cdot \beta_B}{\left(\frac{I}{I>}\right)^{\alpha_B} - 1} \qquad (2.4)$$

A Tabela 2.1 mostra os valores de α e β que definem os diferentes tipos de curva tempo/corrente [24].

Tipo de Curva Tempo/Corrente	α	β
Normalmente Inversa	0,02	0,14
Muito Inversa	1,0	13,5
Extremamente Inversa	2,0	80,0
Longamente Inversa	1,0	120,0

Tabela 2.1 – Tipos de curva com relação aos valores de α e ß

As figuras a seguir mostram as características dos tipos de curvas apresentadas na Tabela 2.1 segundo alguns valores de k. Estas curvas são as principais ferramentas para se realizar a coordenação de relés de sobrecorrente.

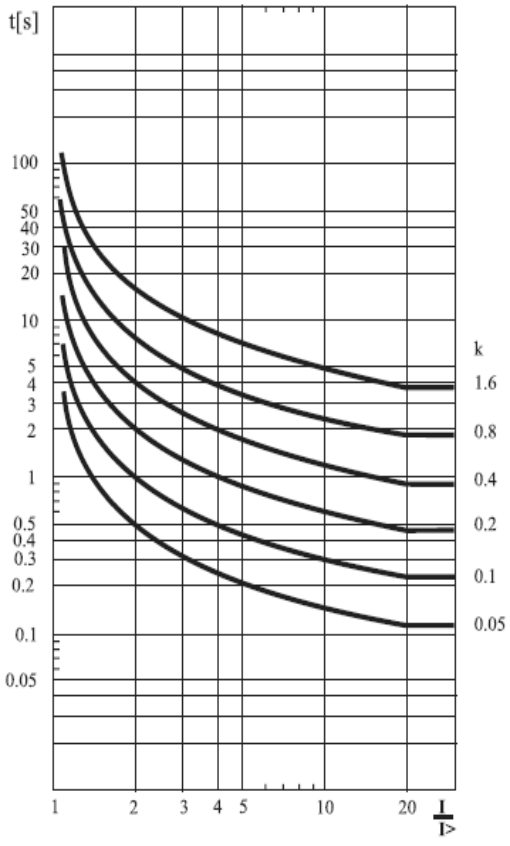


Figura 2.3 – Curva Normalmente Inversa [21]

Figura 2.4 – Curva Muito Inversa [21]

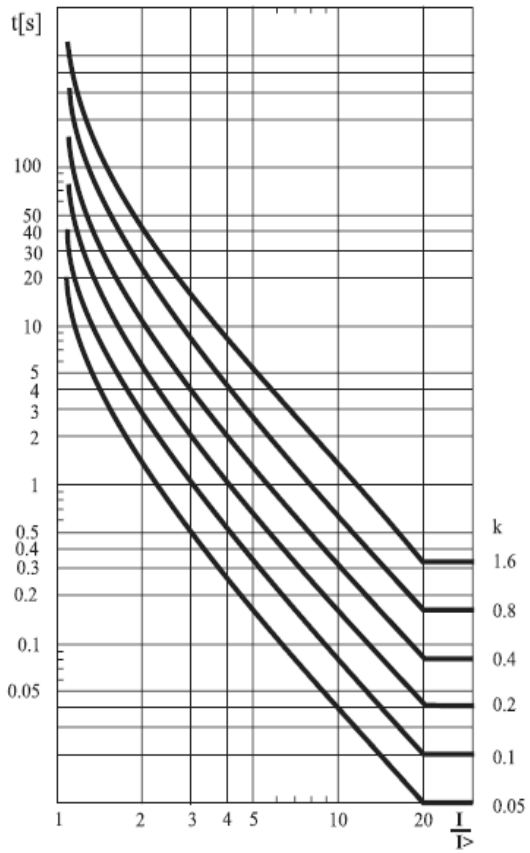


Figura 2.5 - Curva Extremamente Inversa [21]

Figura 2.6 – Curva Longamente Inversa [21]

A Figura 2.7 mostra como poderia ser feita a coordenação dos relés apresentados na Figura 2.1 utilizando-se curvas tempo/corrente.

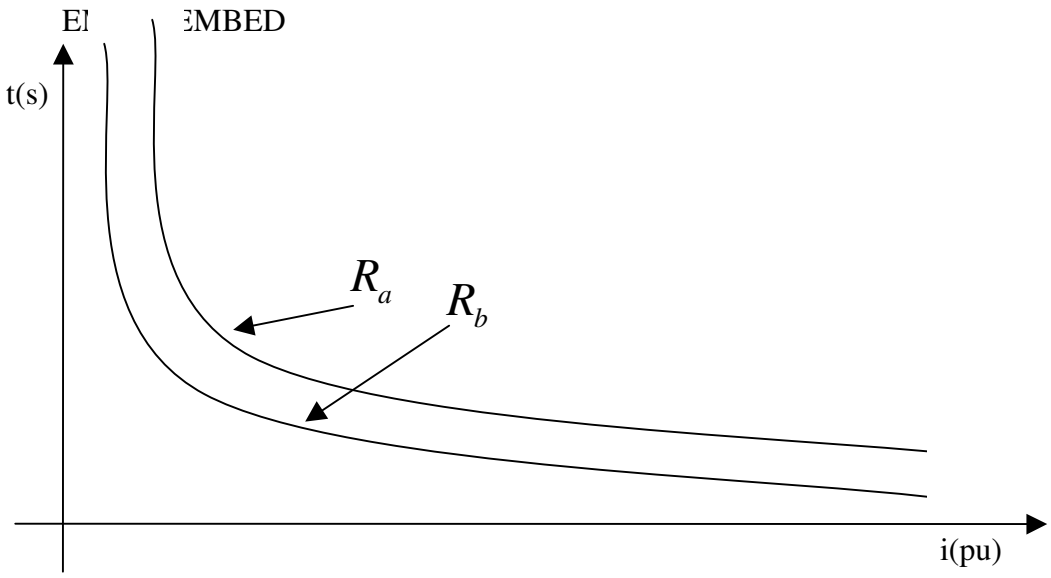


Figura 2.7 – Coordenação de Relés [20]

Na Figura 2.7, a parametrização dos valores de k, α, β e I> fazem com que o tempo de operação do Relé B (t_{R_B}) seja sempre menor que o tempo de operação do Relé A (t_{R_A}), garantindo portanto, que o Relé B sempre opere antes do Relé A, independentemente do valor da corrente I, ou seja, garantindo a seletividade do sistema da Figura 2.1.

3. ALGORITMO GENÉTICO APLICADO À SELETIVIDADE DE RELÉS DE SOBRECORRENTE

3.1. Algoritmo Genético

O Algoritmo Genético é um algoritmo de otimização global e é baseado na teoria da evolução. Utiliza busca paralela e estruturada de forma aleatória, efetivada por indivíduos com alto ou baixo valor de aptidão para a maximização ou minimização de uma função objeto [25].

Os conceitos relativos à teoria e implementação do Algoritmo Genético são mostrados no Apêndice A.

O fluxograma básico do Algoritmo Genético, mostrado na Figura 3.1, foi elaborado com base nas seguintes referências da literatura, [1], [17], [16], [2], [5], [19].

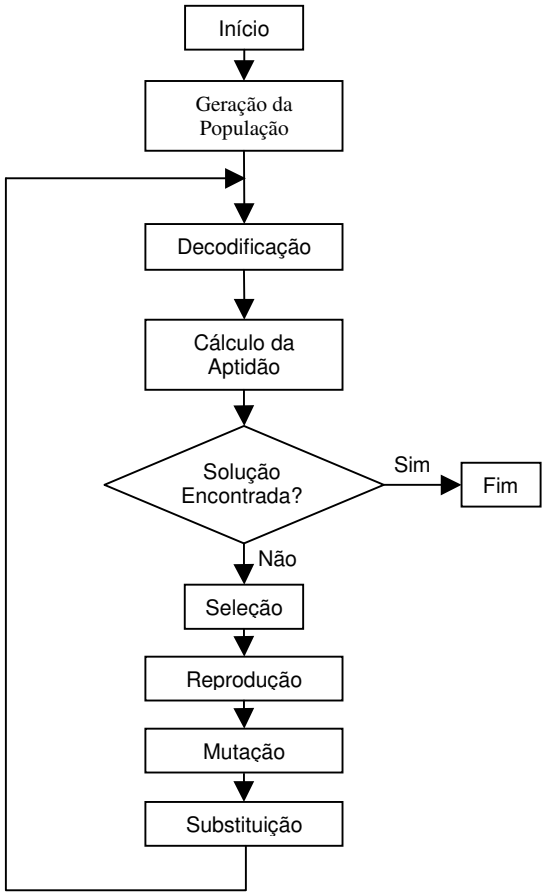


Figura 3.1 – Diagrama de Blocos de um Algoritmo Genético

O algoritmo genético básico envolve seis passos: tendo início na geração da população, avaliação da população, teste de convergência ou critério de término para a otimização, seleção e aplicação dos operadores do AG, e criação de uma nova geração.

A primeira etapa é a definição de qual será a função para representar o problema, esta é a "chave" para um resultado satisfatório do AG. Após a definição da função pode-se acrescentar a parametrização do sistema, ou seja, é neste momento que as variáveis são inicializadas. Alguns exemplos de variáveis que devem ser iniciadas são: tamanho da população; quantidade de geração; taxa de cruzamento; taxa de mutação; tamanho do indivíduo entre outros.

O segundo passo é a geração da população de forma aleatória com a quantidade de indivíduos e seu tamanho determinado em parâmetro, com distribuição uniforme pelo espaço de busca. A terceira etapa é analisar o critério da parada do sistema, que pode ser

pelo número de gerações estabelecidas, pelo tempo de execução ou algum outro indicador. A seleção dos indivíduos da população é um passo importante, ou seja, os indivíduos selecionados têm a maior probabilidade de participarem do processo de escolha para uma nova geração. O melhor material genético tem maior chance de ser selecionado. Após a escolha, são aplicados os operadores de cruzamento e/ou mutação. Primeiramente, o cruzamento, dividindo a população em pares de cromossomos da população atual e a cada novo indivíduo aplica-se o processo de mutação, gerando desta forma uma nova população em substituição a anterior. Os indivíduos da nova geração são novamente avaliados pela função de adaptação, começando assim um novo ciclo a partir da quinta etapa, até a condição de término ser atendida.

Finalmente, quando a condição de término é atendida o indivíduo mais apto é considerado como um forte candidato para ser a solução que estava sendo procurada. Em última análise, a impressão do resultado final é feita de acordo com a necessidade do usuário, podendo mostrar somente o melhor indivíduo da última geração, ou mostrar a última população inteira, ou qualquer indivíduo que se possa extrair as informações desejadas.

3.2. Coordenação Otimizada de Relés de Sobrecorrente Utilizando Algoritmo Genético

Devido à natureza não-linear da função objetivo apresentada na Equação 2.4, com seis diferentes valores para o dial de tempo (κ), quatro diferentes valores para a corrente de *pick up* (I>) e quatro diferentes valores para as constantes α e β, há uma enorme combinação de soluções possíveis para o problema, o que dificulta o uso de algoritmos de otimização clássicos. Por isso, optamos pela utilização do algoritmo genético para efetuar a coordenação otimizada de relés de sobrecorrente.

A seguir, apresentaremos os parâmetros implementados no desenvolvimento do algoritmo genético desenvolvido, assim como as rotinas elaboradas no MATLAB, para compilação do Algoritmo Genético.

Rotina Principal

É a rotina responsável por seguir o fluxograma de compilação do Algoritmo Genético, conforme a Figura 3.1.

Abaixo segue o algoritmo da Rotina Principal.

```
%_____________________________________________________________________
% Ferramenta Computacional para Otimização do Tempo de Coordenação de Relés
%                de Sobrecorrente em Sistemas Radiais
%---------------------------------------------------------------------

%_____________________________________________________________________
% Desenvolvido por:

%    Gilvan Rodrigues de Oliveira Junior

%                                        Janeiro/2008
%---------------------------------------------------------------------

clear
clc;

%_____________________________________________________________________
% Setup
%---------------------------------------------------------------------
npop=input('Número de Indivíduos: ');
nger=input('Número de Gerações: ');
txcruz=input('Taxa de Cruzamento: ');
```

```
txmut=input('Taxa de Mutação: ');
tol=1;
nprogen=round(txcruz*npop);
elitsm=round(nprogen*.4);
F=[];
stpop=nger;

%_____________________________________________________________________
% Gerando a População Inicial
%---------------------------------------------------------------------
[pop]=inipop(npop);

%_____________________________________________________________________
% Algoritmo Genético
%---------------------------------------------------------------------
while (tol>0.0001) && (nger>=0)

flagplot=nger/stpop;

%_____________________________________________________________________
%  Decodificando os Indivíduos
%---------------------------------------------------------------------

 [sAB1 sI1 sK1 sAB2 sI2 sK2 sAB3 sI3 sK3] = separa(pop);

%_____________________________________________________________________
% Cálculando a Função de Aptidão
%---------------------------------------------------------------------

fApt=@aptidao;

[T1 T2 T3 FA]=calculaFA(pop);

%_____________________________________________________________________
% Verifica Melhor Indivíduo da População
%---------------------------------------------------------------------

[mApt bestIndbin]=melhor(FA,pop);

[mAB1 mI1 mK1 mAB2 mI2 mK2 mAB3 mI3 mK3] = separa(bestIndbin);

[mT1 mT2 mT3] = TOP(mAB1,mI1,mK1,mAB2,mI2,mK2,mAB3,mI3,mK3);

T1T2=mT1-mT2;
T2T3=mT2-mT3;
TT=T1T2+T2T3;

%_____________________________________________________________________
% Analisa Convergência
%---------------------------------------------------------------------

mediaFA=sum(FA)/size(FA,2);
tol=abs(mApt-mediaFA);

%_____________________________________________________________________
% Plota Populções
```

```
%---------------------------------------------------------------------

if flagplot==1

   figure
   plot(T3,T1,'.');

elseif nger==0

   figure
   plot(T3,T1,'.');

elseif (tol<=0.0001)

   figure
   plot(T3,T1,'.');

end

%_____________________________________________________________________
% Reprodução, Substituição e Mutação
%---------------------------------------------------------------------

if (tol>0.0001) && (nger~=0)

%_____________________________________________________________________
% Selecionando para a Reprodução
%---------------------------------------------------------------------

[Rank]=fitscalingrank(FA,nprogen);

[progen]=selectiontournament(Rank,nprogen,[],4);

Gencomp=size(pop,2);

options = gaoptimset('EliteCount',elitsm);

%_____________________________________________________________________
% Reprodução
%---------------------------------------------------------------------

[Kids]  = crossoverscattered(progen,[],Gencomp,fApt,options,pop);

[KT1 KT2 KT3 KFA]=calculaFA(Kids);

%_____________________________________________________________________
% Substituição 1
%---------------------------------------------------------------------

[newpop NFA]=substituir(FA,KFA,pop,Kids);

pop=newpop;

%_____________________________________________________________________
% Mutação
%---------------------------------------------------------------------
```

```
[MFA]=calculaFA(pop);

[mRank]=fitscalingrank(MFA,nprogen);

[mprogen]=selectiontournament(mRank,nprogen,[],4);

[mKids]=mutationuniform(mprogen,options,Gencomp,fApt,[],MFA,pop,txmut);

mKids=round(mKids);

[mmT1 mmT2 mmT3 mmFA]=calculaFA(mKids);

%_____________________________________________________________________________
% Substituição 2
%-----------------------------------------------------------------------

[mnewpop MFA]=substituir(MFA,mmFA,pop,mKids);

pop=mnewpop;

nger = nger-1

else

nger = nger-1;

end

end

%_____________________________________________________________________________
% Finalizando
%-----------------------------------------------------------------------

[vT1 vT2 vT3]=plotar(mAB1,mI1,mK1,mAB2,mI2,mK2,mAB3,mI3,mK3,mT1,mT2,mT3);
```

Geração da População Inicial

A geração da população inicial é formada por valores aleatórios de *diais* de tempo dos relés de sobrecorrente, corrente de *pick up* e tipos aleatórios de curva de tempo/corrente.

Abaixo segue o algoritmo desenvolvido para geração da população inicial.

```
%_____________________________________________________________________________
%                  Incialização da População
%-----------------------------------------------------------------------

%_____________________________________________________________________________
% Desenvolvido por:
```

```
%    Gilvan Rodrigues de Oliveira Junior

%                                    Janeiro/2008
%------------------------------------------------------------------------

function [pop]=inipop(npop)

pop=round(rand(npop, 21));
```

Codificação

Utilizaremos a codificação binária, principalmente pela fácil implementação computacional. O conjunto formado pelos relés 1, 2 e 3 será tratado com um indivíduo, onde cada indivíduo possuirá nove cromossomos: três com dois genes, representando o tipo de curva de tempo/corrente do relé, três com dois genes, representando o valor *pick up* de corrente do relé e por fim, três com três genes, representando o *dial* de tempo do relé. A Figura 3.2 abaixo ilustra a composição dos indivíduos.

A codificação do *pick up* de corrente foi feita a partir de uma sobrecarga de 20% (vinte por cento) sobre o valor da corrente nominal do alimentador, e com um degrau de 0,5 pu para os Relés 1 e 2 e com um degrau de 0,05 pu para o Relé 3.

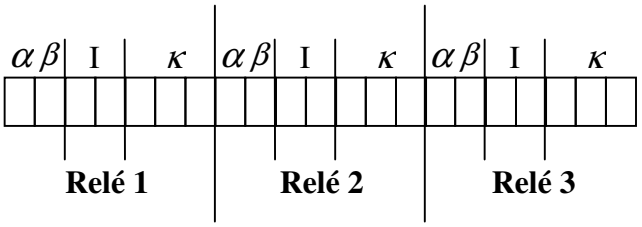


Figura 3.2 – Representação dos Indivíduos segundo seus cromossomos

Nas Tabelas 3.1, 3.2 e 3.3 abaixo, seguem as codificações das variáveis apresentadas acima.

Tipo de Curva Tempo/Corrente	α	β
Normalmente Inversa	0	0
Muito Inversa	0	1
Extremamente Inversa	1	0
Longamente Inversa	1	1

Tabela 3.1 – Codificação do Tipo de Curva Tempo/Corrente [24]

Corrente de *Pick up*		I (pu)		
		Relé 1	**Relé 2**	**Relé 3**
0	0	1,20	0,60	0,120
0	1	1,25	0,65	0,125
1	0	1,30	0,70	0,130
1	1	1,35	0,75	0,135

Tabela 3.2 – Codificação da Corrente de p*ick up* [24]

Dial de tempo (s)	κ		
0,05	0	0	0
0,1	0	0	1
0,2	0	1	0
0,4	0	1	1
0,8	1	0	0
1,6	1	0	1

Tabela 3.3 – Codificação do Dial de tempo [21]

No cromossomo κ foi implementado o método de ajustes mínimos, que não permite que indivíduos gerados com genes 110 e 111, que não estão codificados segundo a tabela 3.3, se percam. Neste método, a decodificação para estes genes é feita de forma que seus valores κ podem assumir, aleatoriamente, valores pré-determinados de κ apresentados na Tabela 3.3.

Abaixo segue o algoritmo desenvolvido para decodificação dos indivíduos.

```
%_____________________________________________________________________
%                  Decodificação dos Indivíduos
%---------------------------------------------------------------------

%_____________________________________________________________________
% Desenvolvido por:

%    Gilvan Rodrigues de Oliveira Junior

%                                   Janeiro/2008
%---------------------------------------------------------------------

function [sAB1 sI1 sK1 sAB2 sI2 sK2 sAB3 sI3 sK3]=separa(pop)
```

```
%_____________________________________________

%Relé 1
%_____________________________________________

% Alpha e Beta

AB1=pop(:,[1 2]);

% Corrente de Fase

I1=pop(:,[3 4]);

% Dial de Tempo

K1=pop(:,[5 6 7]);

%_____________________________________________

%Relé 2
%_____________________________________________

% Alpha e Beta

AB2=pop(:,[8 9]);

% Corrente de Fase

I2=pop(:,[10 11]);

% Dial de Tempo

K2=pop(:,[12 13 14]);

%_____________________________________________

%Relé 3
%_____________________________________________

% Alpha e Beta

AB3=pop(:,[15 16]);

% Corrente de Fase

I3=pop(:,[17 18]);

% Dial de Tempo

K3=pop(:,[19 20 21]);

%_____________________________________________

% Leitura

%_____________________________________________
```

```
% Alpha e Beta
k=size(pop,1);
for i=1:1:k
if isequal(AB1(i,:),[0 0])
        sAB1(i,:)=[0.02 0.14];
    elseif isequal(AB1(i,:),[0 1])
        sAB1(i,:)=[1 13.5];
    elseif isequal(AB1(i,:),[1 0])
        sAB1(i,:)=[2 80];
    elseif isequal(AB1(i,:),[1 1])
        sAB1(i,:)=[1 120];
end
if isequal(AB2(i,:),[0 0])
        sAB2(i,:)=[0.02 0.14];
    elseif isequal(AB2(i,:),[0 1])
        sAB2(i,:)=[1 13.5];
    elseif isequal(AB2(i,:),[1 0])
        sAB2(i,:)=[2 80];
    elseif isequal(AB2(i,:),[1 1])
        sAB2(i,:)=[1 120];
end
if isequal(AB3(i,:),[0 0])
        sAB3(i,:)=[0.02 0.14];
    elseif isequal(AB3(i,:),[0 1])
        sAB3(i,:)=[1 13.5];
    elseif isequal(AB3(i,:),[1 0])
        sAB3(i,:)=[2 80];
    elseif isequal(AB3(i,:),[1 1])
```

```
        sAB3(i,:)=[1 120];
end
% Corrente de Fase
if isequal(I1(i,:),[0 0])
        sI1(i,:)=1.2;
    elseif isequal(I1(i,:),[0 1])
        sI1(i,:)=1.25;
    elseif isequal(I1(i,:),[1 0])
        sI1(i,:)=1.3;
    elseif isequal(I1(i,:),[1 1])
        sI1(i,:)=1.35;
end

if isequal(I2(i,:),[0 0])
        sI2(i,:)=.6;
    elseif isequal(I2(i,:),[0 1])
        sI2(i,:)=.65;
    elseif isequal(I2(i,:),[1 0])
        sI2(i,:)=.7;
    elseif isequal(I2(i,:),[1 1])
        sI2(i,:)=.75;
end
if isequal(I3(i,:),[0 0])
        sI3(i,:)=.12;
    elseif isequal(I3(i,:),[0 1])
        sI3(i,:)=.125;
    elseif isequal(I3(i,:),[1 0])
        sI3(i,:)=.13;
```

```
    elseif isequal(I3(i,:),[1 1])

        sI3(i,:)=.135;

end

% Dial de tempo

if isequal(K1(i,:),[0 0 0])

        sK1(i,:)=0.05;

    elseif isequal(K1(i,:),[0 0 1])

        sK1(i,:)=0.1;

    elseif isequal(K1(i,:),[0 1 0])

        sK1(i,:)=0.2;

    elseif isequal(K1(i,:),[0 1 1])

        sK1(i,:)=0.4;

    elseif isequal(K1(i,:),[1 0 0])

        sK1(i,:)=0.8;

    elseif isequal(K1(i,:),[1 0 1])

        sK1(i,:)=1.6;

    elseif isequal(K1(i,:),[1 1 0])

        n1=round(rand(1));
        n2=round(rand(1));
        n=n1+n2;

        if n==1

            sK1(i,:)=0.2;

        else

            sK1(i,:)=0.8;

        end

    elseif isequal(K1(i,:),[1 1 1])

        n1=round(rand(1));
        n2=round(rand(1));
        n=n1+n2;

        if n==1
```

```
            sK1(i,:)=0.4;
        else
            sK1(i,:)=1.6;
        end
end
if isequal(K2(i,:),[0 0 0])
        sK2(i,:)=0.05;
    elseif isequal(K2(i,:),[0 0 1])
        sK2(i,:)=0.1;
    elseif isequal(K2(i,:),[0 1 0])
        sK2(i,:)=0.2;
    elseif isequal(K2(i,:),[0 1 1])
        sK2(i,:)=0.4;
    elseif isequal(K2(i,:),[1 0 0])
        sK2(i,:)=0.8;
    elseif isequal(K2(i,:),[1 0 1])
        sK2(i,:)=1.6;
        elseif isequal(K2(i,:),[1 1 0])
        n1=round(rand(1));
        n2=round(rand(1));
        n=n1+n2;
        if n==1
            sK2(i,:)=0.2;
        else
            sK2(i,:)=0.8;
        end
    elseif isequal(K2(i,:),[1 1 1])
        n1=round(rand(1));
        n2=round(rand(1));
        n=n1+n2;
```

```
        if n==1
            sK2(i,:)=0.4;
        else
            sK2(i,:)=1.6;
        end
end
if isequal(K3(i,:),[0 0 0])
        sK3(i,:)=0.05;
    elseif isequal(K3(i,:),[0 0 1])
        sK3(i,:)=0.1;
    elseif isequal(K3(i,:),[0 1 0])
        sK3(i,:)=0.2;
    elseif isequal(K3(i,:),[0 1 1])
        sK3(i,:)=0.4;
    elseif isequal(K3(i,:),[1 0 0])
        sK3(i,:)=0.8;
    elseif isequal(K3(i,:),[1 0 1])
        sK3(i,:)=1.6;
    elseif isequal(K3(i,:),[1 1 0])
        n1=round(rand(1));
        n2=round(rand(1));
        n=n1+n2;
        if n==1
            sK3(i,:)=0.2;
        else
            sK3(i,:)=0.8;
        end
    elseif isequal(K3(i,:),[1 1 1])
        n1=round(rand(1));
        n2=round(rand(1));
        n=n1+n2;
```

```
    if n==1
       sK3(i,:)=0.4;
    else
       sK3(i,:)=1.6;
    end
end
end
```

Restrições

As restrições implementadas são:

- Tempo mínimo de coordenação entre dois relés consecutivos ($t_{c_{\min}}$) [21];
- Tempo máximo de coordenação entre dois relés consecutivos ($t_{c_{máx}}$) [20];
- Tempo mínimo de operação dos relés;
- Tempo máximo de operação dos relés.

As Tabelas 3.4 e 3.5 mostram os valores atribuídos a cada uma das restrições utilizadas.

$t_{c_{\min}}$	200 ms
$t_{c_{máx}}$	250 ms

Tabela 3.4 – $t_{c_{\min}}$ e $t_{c_{máx}}$

Tempo de Operação	Mínimo	Máximo
T1	700ms	1600 ms
T2	500ms	1200 ms
T3	300 ms	800 ms

Tabela 3.5 – T1, T2 e T3

Os indivíduos que não atenderem estas restrições receberão uma penalização no cálculo da função de aptidão, fazendo com que este indivíduo dificilmente seja escolhido para a reprodução.

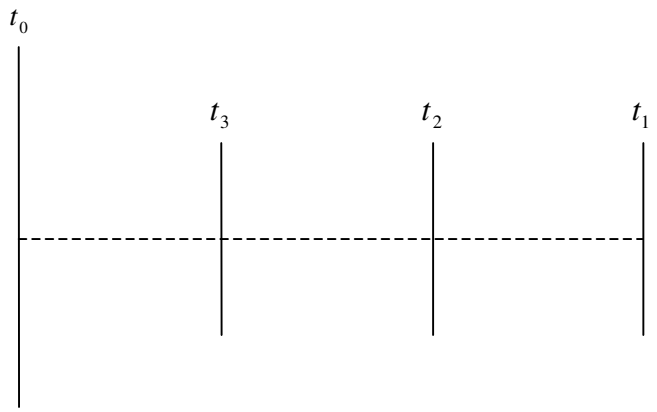
Abaixo segue o algoritmo implementado para o cálculo das penalizações referentes às restrições.

```
%_____________________________________________________________________
%                   Cálculo da Matriz de Restrições
%---------------------------------------------------------------------

%_____________________________________________________________________
% Desenvolvido por:

%    Gilvan Rodrigues de Oliveira Junior

%                                          Janeiro/2008
%---------------------------------------------------------------------

function [R]=restricoes(T1,T2,T3)

Ar=[1 -1 0;0 1 -1;1 0 0;0 1 0;0 0 1];
br=[0.2;0.25;0.2;0.25;0.7;1.6;0.5;1.2;0.3;0.8];

k=size(T1,2);

for i=1:1:k

   x=[T1(:,i);T2(:,i);T3(:,i)];
   b=Ar*x;

   if (b(1)<br(1))

      R(i)=1000;

   elseif b(1)>br(2)

      R(i)=1000;

   else

      R(i)=0;

   end

   if b(2)<br(3)

      R(i)=R(i)+1000;

   elseif b(2)>br(4)

      R(i)=R(i)+1000;
```

```
    else
        R(i)=R(i)+0;
    end
    if b(3)<br(5)
        R(i)=R(i)+600;
    elseif b(3)>br(6)
        R(i)=R(i)+600;
    else
        R(i)=R(i)+0;
    end
    if b(4)<br(7)
        R(i)=R(i)+400;
    elseif b(4)>br(8)
        R(i)=R(i)+400;
    else
        R(i)=R(i)+0;
    end
    if b(5)<br(9)
        R(i)=R(i)+200;
    elseif b(5)>br(10)
        R(i)=R(i)+200;
    else
        R(i)=R(i)+0;
    end
end
```

Função de Aptidão

A Figura 3.3 mostra como é feita a seletividade dos relés de sobrecorrente que compõem o sistema apresentado na Figura 3.1.



t_0 : Início da falha

t_1 : Tempo de operação do relé 1

t_2 : Tempo de operação do relé 2

t_3 : Tempo de operação do relé 3

Figura 3.3 – Seletividade dos relés de sobrecorrente que compõem o sistema

Para cada indivíduo da população é calculado um valor para função de aptidão, e este valor é o que determinará as suas chances da permanência nas futuras gerações.

Neste trabalho procuraremos minimizar o tempo de coordenação entre os Relés 1 e 3, sem que isto afete a seletividade do sistema. Logo, a função de aptidão é dada por:

$$f_{apt} = \frac{1}{t_1 - t_3 + C_{\min}} + R \tag{4.1}$$

Substituindo a Equação 2.4 na Equação 4.1, temos que:

$$f_{apt} = \frac{1}{\left(\frac{k_1 \cdot \beta_1}{\left(\frac{I_1}{I_1 >} \right)^{\alpha_1} - 1} - \frac{k_3 \cdot \beta_3}{\left(\frac{I_3}{I_3 >} \right)^{\alpha_3} - 1} \right) + C_{\min}} + R \tag{4.2}$$

onde

f_{apt} : Função de Aptidão

k: Dial de tempo do relé em segundos

I: Corrente de fase

I>: *Pick up* ajustado

α e β: são constantesEMBEDEMBED

$C_{\min}$: Constante de Continuidade

R: Penalidade aplicada pelas restrições

Abaixo segue o algoritmo implementado para o cálculo da função de aptidão.

```
%_____________________________________________________________________
%                  Cálculo da Função de Aptidão
%---------------------------------------------------------------------

%_____________________________________________________________________
% Desenvolvido por:

%    Gilvan Rodrigues de Oliveira Junior

%                                        Janeiro/2008
%---------------------------------------------------------------------
function [FA]=aptidao(T1,T2,T3,R)

k=size(T3,2);

T1T2=T1-T2;
T2T3=T2-T3;

T=T1T2+T2T3;

for i=1:1:k

   FA(i)=(1/(T(i)+1))+R(i);

end
```

Seleção para Reprodução

Para selecionar os indivíduos que participarão da reprodução, utilizamos um método chamado Torneio, descrito no Apêndice 1. Essencialmente, este método consiste em comparar uma quantidade pré-estabelecida de indivíduos da população, selecionando para reprodução o mais apto deles.

Abaixo segue o algoritmo desenvolvido para efetuarmos a seleção.

```
%_____________________________________________________________________
%                   Função de Seleção de Reprodução
%---------------------------------------------------------------------

%_____________________________________________________________________
% Desenvolvido por:

%    Gilvan Rodrigues de Oliveira Junior

%                                        Janeiro/2008
%---------------------------------------------------------------------

function parents = selectiontournament(expectation,nParents,options,tournamentSize)

if(nargin < 4)
   tournamentSize = 4;
end

players = ceil(length(expectation) * rand(nParents,tournamentSize));

scores = expectation(players);

[unused,m] = max(scores');

parents = zeros(1,nParents);
for i = 1:nParents
   parents(i) = players(i,m(i));
end
```

Reprodução

Para efetuarmos a reprodução, utilizamos o método de reprodução uniforme. Outro critério utilizado foi o Elitismo, que consiste em selecionar automaticamente para a próxima população os n indivíduos mais aptos da população, onde n é um número inteiro e pré-determinado.

Abaixo segue o algoritmo desenvolvido para efetuarmos a reprodução

```
%_____________________________________________________________________
%                   Função de Reprodução
%---------------------------------------------------------------------

%_____________________________________________________________________
% Desenvolvido por:

%    Gilvan Rodrigues de Oiveira Junior

%                                        Janeiro/2008                  8
%---------------------------------------------------------------------
```

```
function xoverKids  = crossoverscattered(parents,options,GenomeLength,...
                    FitnessFcn,unused,thisPopulation)

nKids = length(parents)/2;

constr = false;
if isfield(options,'LinearConstr')
   alpha = rand;
   linCon = options.LinearConstr;
   A = linCon.A;
   LB = linCon.L;
   UB = linCon.U;
   IndEqcstr = linCon.IndEqcstr;
   TolBind = sqrt(options.TolCon);
   constr = ~isempty(A);
end

xoverKids = zeros(nKids,GenomeLength);

index = 1;

for i=1:nKids

   r1 = parents(index);
   index = index + 1;
   r2 = parents(index);
   index = index + 1;

    for j = 1:GenomeLength
      if(rand > 0.5)
         xoverKids(i,j) = thisPopulation(r1,j);
      else
         xoverKids(i,j) = thisPopulation(r2,j);
      end
   end

   if constr
      feasible  = isfeasible(xoverKids(i,:)',A,LB,UB,TolBind,IndEqcstr);
      if ~feasible

         xoverKids(i,:) = alpha*thisPopulation(r1,:) + ...
            (1-alpha)*thisPopulation(r2,:);
      end
   end
end
```

Mutação

No algoritmo implementamos a mutação por troca de bit, citada no Apêndice A deste trabalho como um dos diferentes tipos de mutação utilizados. Neste tipo de mutação, escolhe-se aleatoriamente uma posição de um descendente e nesta posição fazemos a troca do valor do bit. As Figuras 3.4 e 3.5 mostram, para uma melhor visualização, a

representação de um cromossomo de 8 bits, onde se utiliza código binário e introduz-se a mutação no terceiro gene.

Figura 3.4 – Cromossomo de um dos Descendentes

Figura 3.5 – Cromossomo Após Mutação

Abaixo segue o algoritmo desenvolvido para efetuar a reprodução.

```
%_____________________________________________________________________
%                    Função de Mutação
%---------------------------------------------------------------------

%_____________________________________________________________________
% Desenvolvido por:

%    Gilvan Rodrigues de Oliveira Junior

%                                        Janeiro/2008                                    8
%---------------------------------------------------------------------
function mutationChildren =
mutationuniform(parents,options,GenomeLength,FitnessFcn,state,thisScore,thisPopulation,mutationRate)

if(nargin < 8)
   mutationRate = 0.01;
end

if(strcmpi(options.PopulationType,'doubleVector'))
   mutationChildren = zeros(length(parents),GenomeLength);
   for i=1:length(parents)
      child = thisPopulation(parents(i),:);

      mutationPoints = find(rand(1,length(child)) < mutationRate);

      range = options.PopInitRange;

      [r,c] = size(range);
      if(c ~= 1)
         range = range(:,mutationPoints);
```

```
        end
        lower = range(1,:);
        upper = range(2,:);
        span = upper - lower;


    child(mutationPoints) = lower + rand(1,length(mutationPoints)) .* span;
    mutationChildren(i,:) = child;
    end
elseif(strcmpi(options.PopulationType,'bitString'))

    mutationChildren = zeros(length(parents),GenomeLength);
    for i=1:length(parents)
        child = thisPopulation(parents(i),:);
        mutationPoints = find(rand(1,length(child)) < mutationRate);
        child(mutationPoints) = ~child(mutationPoints);
        mutationChildren(i,:) = child;
    end

end
```

Substituição

Após os processos de reprodução e mutação, são geradas sub-populações com novos indivíduos. É necessário que seja feita a substituição dos indivíduos menos aptos da população pelos indivíduos mais aptos das sub-populações geradas após a geração e a mutação, formando assim a nova população.

Abaixo segue o algoritmo desenvolvido para efetuarmos a reprodução

```
%_____________________________________________________________________________
%                    Função de Substituição
%-----------------------------------------------------------------------------

%_____________________________________________________________________________
% Desenvolvido por:

%    Gilvan Rodrigues de Oliveira Junior

%                                          Janeiro/2008
%-----------------------------------------------------------------------------

function [newpop NFA]=substituir(FA,KFA,pop,Kids)

[oFA,IXFA] = sort(FA);
[oKFA,IXKFA] = sort(KFA);

p=size(pop,2);

noFA=size(oFA,2);
noKFA=size(oKFA,2);
```

```
k=1;
l=noFA;

for i=1:1:noFA

   if k<=noKFA

   if oKFA(1,i)<oFA(1,l)

      for j=1:1:p

         newpop(i,j)=Kids(IXKFA(k),j);
         NFA(i)=KFA(IXKFA(k));

      end

   else

      for j=1:1:p

         newpop(i,j)=pop(IXFA(l),j);
         NFA(i)=FA(IXFA(l));

      end

   end

   else

      for j=1:1:p

        newpop(i,j)=pop(IXFA(l),j);
        NFA(i)=FA(IXFA(l));

      end

   end

   k=k+1;

   l=l-1;

end
```

Critério de Parada

O algoritmo será executado até o número máximo de n gerações, onde n é um número inteiro pré-determinado, ou até que o valor da diferença entre o valor da função de aptidão encontrado do melhor indivíduo e o valor da média das funções aptidão da população seja menor que a tolerância de 0,0001.

4. ESTUDO DE CASO

4.1. Configuração do Sistema

Dado o sistema radial apresentado na Figura 4.1, são mostrados os valores de corrente, em pu, através dos alimentadores e um defeito fase-terra em um alimentador da Barra 3.

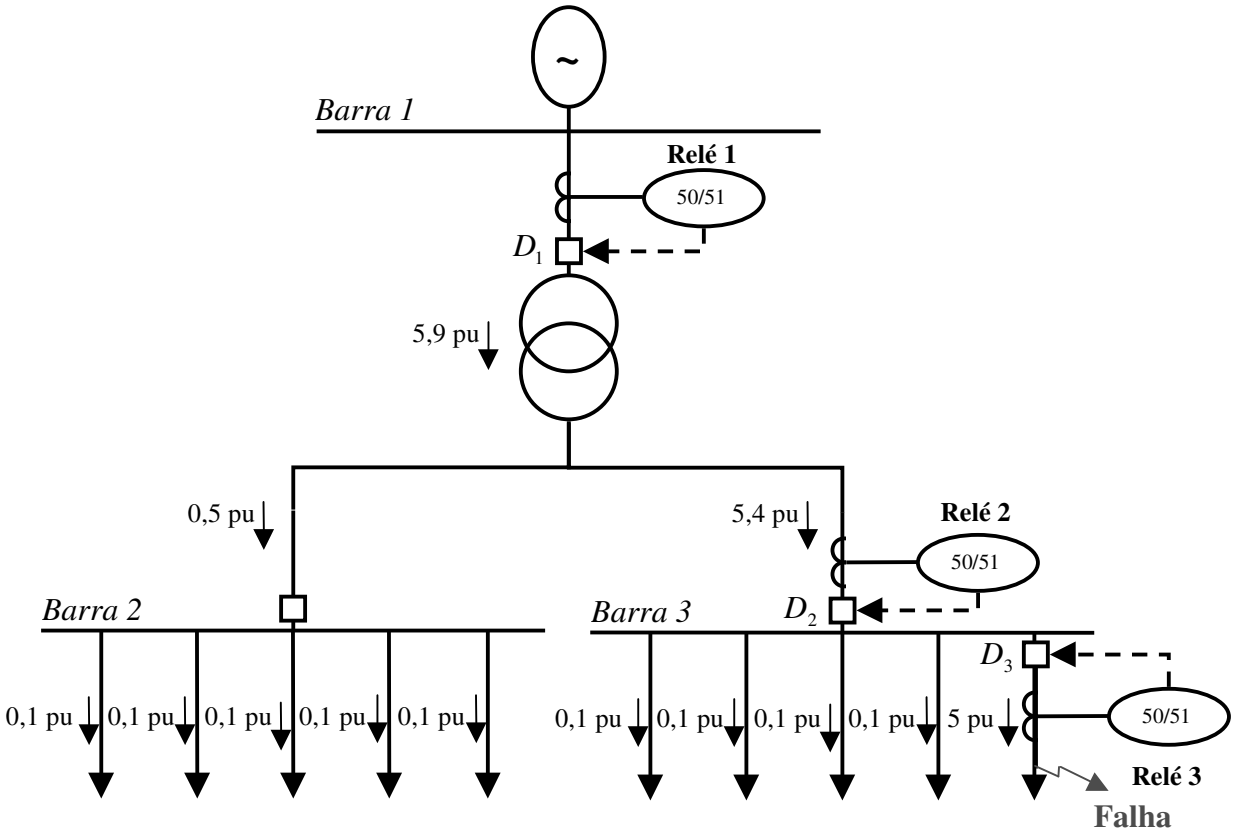


Figura 4.1 – Sistema Radial Proposto para Estudo

Devemos garantir a seletividade do sistema, ou seja, devemos garantir que o primeiro relé a operar seja o Relé 3. Mas caso o disjuntor D_3 não opere, o Relé 2 deve operar e abrir o disjuntor D_2, ainda sim, se o disjuntor D_2 não operar, como último recurso, o Relé 1 deve operar e fazer com que o disjuntor D_1 abra, extinguindo assim o defeito.

Analisando-se a Figura 4.1, podemos perceber mais claramente a importância de garantir a seletividade da proteção, pois quanto mais a montante estiver o disjuntor que

abriu, mas cargas serão cortadas, por conseqüência, maior prejuízo para as distribuidoras e consumidores.

Neste estudo de caso procuraremos minimizar o tempo de coordenação entre os Relés 1 e 3. Para efetuarmos a coordenação otimizada da proteção foi utilizada a ferramenta computacional apresentada no capítulo anterior.

4.2. Resultados

Uma das características do Algoritmo Genético é a necessidade da busca da melhor parametrização dos operadores genéticos. Após a conclusão da ferramenta computacional, foram testadas várias configurações para estes operadores, sendo que a configuração que obteve o melhor desempenho é apresentada na Tabela 4.1.

Operadores Genéticos	
Tamanho da População (Nº de Indivíduos)	1500
Nº de Gerações	100
Taxa de cruzamento	80%
Taxa de Mutação	2%

Tabela 4.1 – Operadores Genéticos utilizados

Cabe destacar o tamanho da população, que devido a grande quantidade de indivíduos, proporcionou a diversidade necessária para a obtenção da melhor solução para o problema. A discretização adotada para as variáveis do problema fez com que os indivíduos ficassem, apesar de distribuídos, localizados dentro de seis faixas, que correspondem aos seis diferentes *dials* de tempos utilizados, fazendo com que o espaço amostral não fosse preenchido uniformemente. Na Figura 4.2, é apresentada a população inicial para o caso onde se obteve o menor valor para o tempo de coordenação entre os Relés 1 e 3.

O valor utilizado para o número de gerações foi suficiente para garantir a convergência do algoritmo, enquanto que os valores utilizados para a taxa de cruzamento e mutação foram parametrizados de forma garantir a convergência para o ponto que apresentava menor tempo de coordenação entre os Relés 1 e 3.

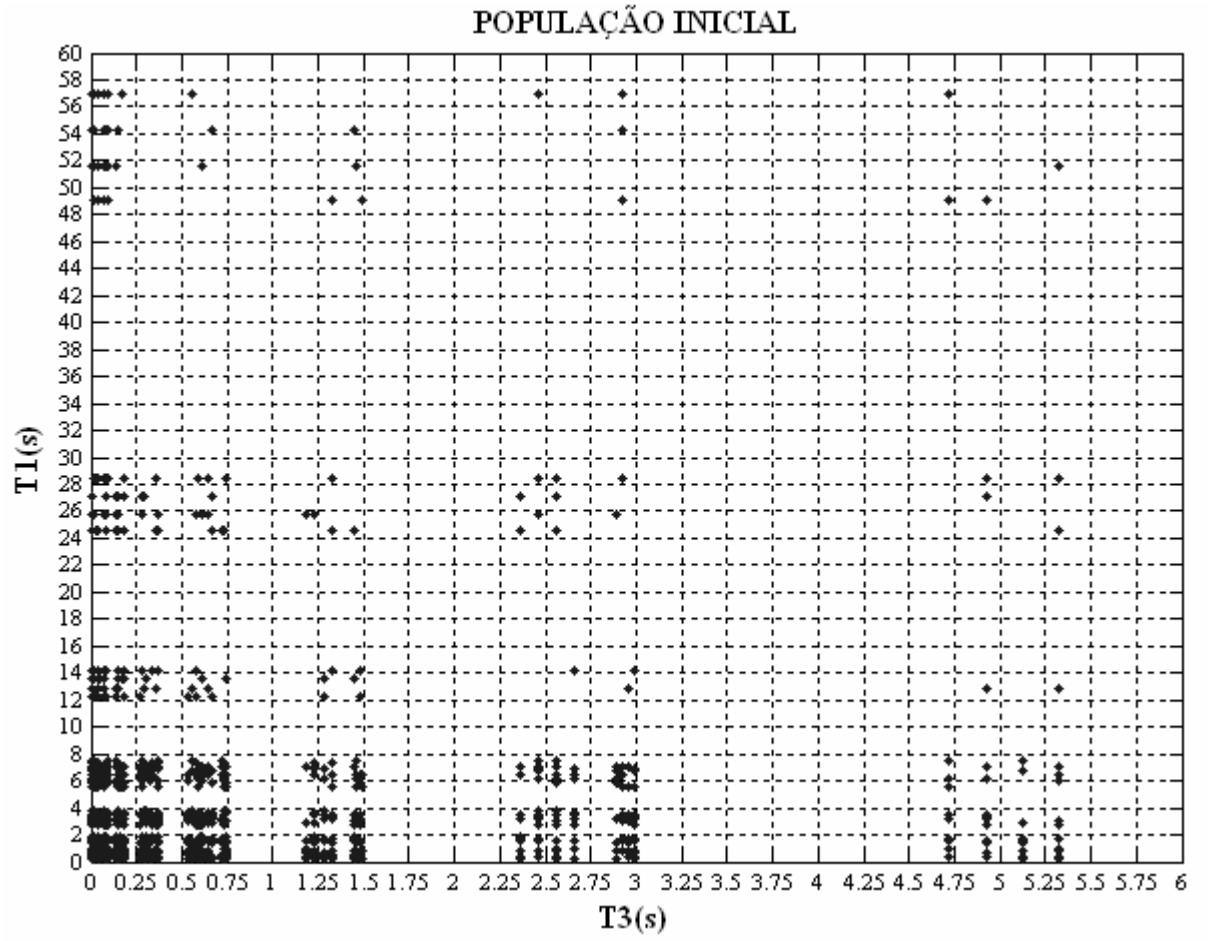


Figura 4.2 – População Inicial

Após 42 (quarenta e duas) iterações o problema convergiu para uma solução, mostrada na Figura 4.3, a qual apresenta o valor da função de aptidão do indivíduo mais apto, assim como a média do valor da função de aptidão da população.

Convergência	
Média	**Indivíduo mais Apto**
0,6924	0,69239

Tabela 4.2 – Valores da Função de Aptidão

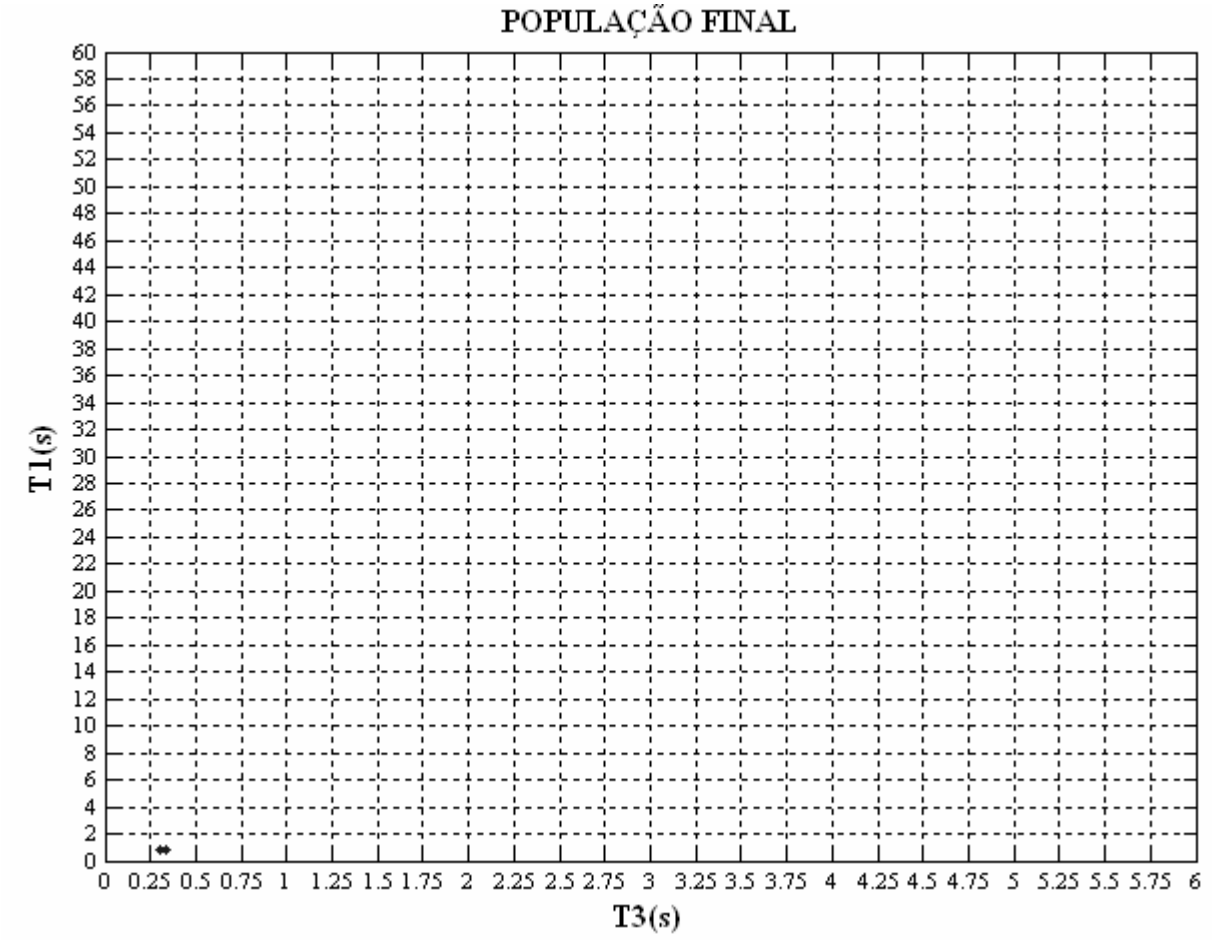


Figura 4.3 – População Final

Na Tabela 4.3, é mostrado o indivíduo que apresentou a melhor aptidão ao meio, ou seja, foi o que apresentou, respeitando as restrições impostas à população, o menor tempo de coordenação entre os Relés 1 e 3.

Indivíduo	α_1	β_1	I_1		κ_1			α_2	β_2	I_2		κ_2			α_3	β_3	I_3		κ_3		
Binário	1	0	0	1	0	1	0	1	0	1	0	0	1	1	1	1	0	1	0	0	1
Decodificado	2	80	1,25		0,20			2	80	0,70		0,40			1	120	0,125		0,10		

Tabela 4.3 – Melhor Indivíduo

Após a obtenção do melhor indivíduo, foram calculados os tempos de operação e coordenação para os relés 1, 2 e 3, apresentados nas Tabela 4.4 e 4.5.

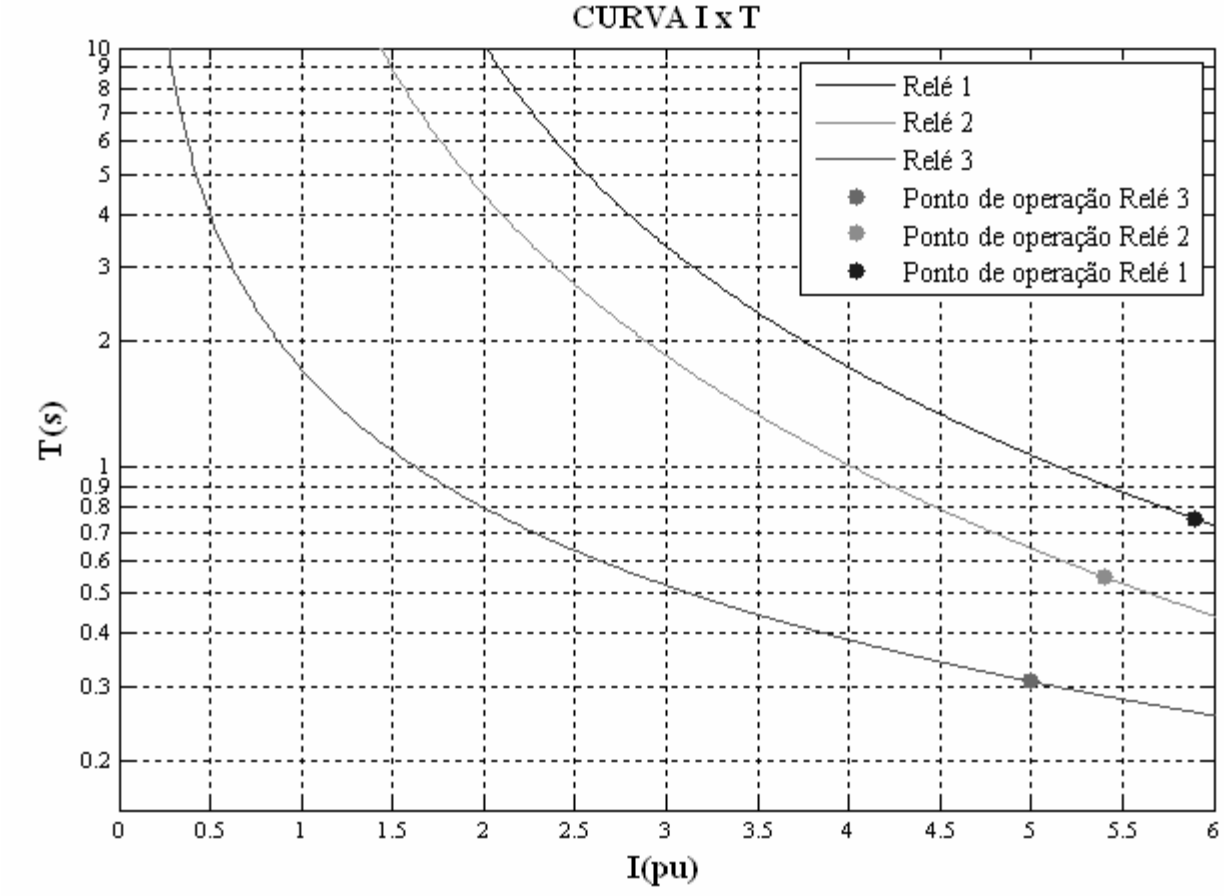
Tempo de Operação (ms)	
T_1	751,9
T_2	546,9
T_3	307,7

Tabela 4.4 – Tempos de operação

Tempo de Coordenação (ms)	
T_1 - T_2	205,0
T_2 - T_3	239,2
T_1 - T_3	444,2

Tabela 4.5 – Tempos de coordenação

Para finalizar, com base nos parâmetros fornecidos pelo melhor indivíduo, foram traçadas as curvas corrente/tempo dos Relés 1, 2 e 3, apresentadas na Figura 4.4.



MERGEFORMAT Figura 4.4 – Curva i x t

5. CONCLUSÃO

O Algoritmo Genético é uma técnica de otimização global, baseado na teoria da evolução, largamente utilizado para problemas de otimização não-lineares. Na coordenação de relés de sobrecorrente, pela quantidade de variáveis envolvidas, não utilizamos métodos de otimização clássicos.

Neste trabalho conseguimos comprovar que o Algoritmo Genético apresenta potencial para a otimização do tempo de coordenação entre relés de sobrecorrente, sendo assim uma boa opção para estudos nesta área.

Os resultados encontrados estão dentro dos padrões esperados. O indivíduo encontrado como melhor solução atendeu a todas as restrições impostas pelo meio e a minimização do tempo de coordenação entre os Relés 1 e 3 foi alcançada.

Em trabalhos futuros, seria interessante a aplicação de uma melhor discretização das variáveis que compõem os indivíduos. Isto traria uma diversidade maior à população inicial, o que provavelmente trará uma solução mais otimizada ao problema.

APÊNDICE 1

Este Apêndice foi criado baseado no Relatório Técnico – Marco 7 [25], desenvolvido em conjunto pela UFRJ e UFJF, apresentado à Endesa Geração Brasil como parte do projeto de P&D “Sistemas Inteligentes Aplicados à Confiabilidade e Seletividade de Plantas Termelétricas”.

ALGORITMO GENÉTICO

A.1. Histórico

Em meados da década de 70, John H. Holland propôs a técnica de algoritmo genético (AG), inspirada nas teorias darwinianas. Este estudo foi responsável pela publicação "*Adaptation in Natural and Artificial Systems"* [1], [2].

O AG é baseado na teoria da evolução. Utiliza busca paralela e estruturada de forma aleatória, efetivada por indivíduos com alto ou baixo valor de aptidão para a maximização ou minimização de uma função objeto. O AG é utilizado basicamente para resolver problemas em pesquisas numéricas, otimização de funções e aprendizagem de máquinas, dentre outras áreas [3]. A aplicação do AG é destacada em sistemas classificadores de dados que ocorrem geralmente na ordenação destes dados para um propósito, por exemplo, para uma simples recuperação ou para efetuar uma análise de dados [4]. Whitley [5] cita que o AG é freqüentemente descrito como um método de busca global, não utilizando gradiente de informação e podendo ser combinado com outros métodos para refinamento de buscas quando há aproximação de um máximo ou mínimo local.

Atualmente o AG é muito empregado na resolução de problemas de bioinformática, como exemplo, a descoberta da estrutura de RNA [6], para a predição de "*non-coding*" RNA (ncRNA) [7], para a construção de mapas de DNA [8] e para a descoberta de regiões regulatórias [9], [10].

A.2. Computação Evolutiva

A computação evolucionária se inspira na teoria evolutiva para o desenvolvimento de métodos computacionais. Na computação evolucionária o algoritmo mais conhecido é o algoritmo genético [11], [12]. O algoritmo genético é uma subdivisão do algoritmo evolucionário, onde também se encontra a programação evolucionária (PE) e a estratégia evolucionária (EE). Todos partilham de uma base conceitual comum, que consiste na simulação da evolução de estruturas individuais, via processo de seleção e os operadores de busca, referidos como operadores genéticos (OG), tais como mutação e *crossove*r

(cruzamento ou recombinação). Todo o processo depende do grau de adaptação, ou seja, do *fitness* (aptidão), do indivíduo frente ao ambiente. A seleção, inspirada na seleção natural das espécies preconiza que os indivíduos mais aptos ou com melhor grau de adaptação ao meio terão maiores chances de repassar o seu material genético para as próximas gerações. Assim, quanto maior a aptidão do indivíduo, maiores são as chances do material genético deste estar presente na próxima geração.

A.3. Algoritmo Genético e sua Inspiração Biológica

A primeira teoria sobre evolução das espécies foi proposta em 1809, pelo naturalista francês Jean Baptiste Pierre Antoine de Monet, conhecido como Lamarck. Para Lamarck as características que um animal adquire durante sua vida podem ser transmitidas hereditariamente. Este estudo ficou conhecido pela ciência como a "lei do uso e desuso" [13]. Charles Darwin vem debater a teoria de Lamarck, de forma agressiva tentando de forma científica explicar como as espécies evoluem. A seleção natural é um processo de evolução, geralmente aceito pela comunidade científica como a melhor explicação para a adaptação. O meio ambiente seleciona os seres mais aptos, em geral, só estes conseguem reproduzir-se e os menos adaptados são eliminados ou pelo menos reduzidos em um primeiro momento a uma minoria. Assim, só as diferenças que facilitam a sobrevivência são transmitidas à geração seguinte [14].

A seleção natural depende muito das condições ambientais, podendo selecionar características de um determinado organismo ajudando na reprodução e sobrevivência deste, os organismos que não possuem tal característica podem vir a morrer antes que se reproduzam ou serem menos prolíficos que os organismos que apresentam a característica. À medida que as condições ambientais não variem essas características continuam sendo adaptativas tornando-se comum na população. Certas características são preservadas devido à vantagem seletiva que conferem, permitindo que o indivíduo reproduza-se, conseqüentemente, deixando mais descendentes através de diversas interações os organismos podem vir a desenvolver características adaptativas muito complexas [15], [14].

A teoria da seleção natural não é aplicada somente a organismos biológicos, podendo ser aplicada a qualquer organismo que se reproduz de modo a envolver tanto a

hereditariedade como a variação [16]. Assim, pode ocorrer seleção natural no reino não biológico.

Alguns pesquisadores buscaram na natureza a inspiração necessária para pesquisar e desenvolver novas técnicas de busca de soluções para determinados problemas. Na natureza o processo de seleção natural demonstra que seres mais preparados (aptos) compete com os recursos naturais impostos, tendo assim uma maior probabilidade de sobreviver, conseqüentemente, disseminar o seu código genético [2]. Com o passar das gerações, através de sucessivos cruzamentos e mutações que ocorrem com as espécies, estes tendem a estar cada vez mais adaptados ao meio ambiente em que vivem.

O AG trabalha com uma população no qual cada indivíduo é um candidato a solução do problema. A função de otimização representa o ambiente no qual a população inicial encontra-se. Emprega-se no AG a mesma terminologia e os mesmos princípios da teoria evolutiva e da genética [1].

O AG relacionado com a seleção natural pode ser expressado como [13].

1 – SE há organismo que se reproduzem;

2 – SE os descendentes herdam as características de seus genitores;

3 – SE há variação nas características;

4 – SE o ambiente não suporta todos os indivíduos de uma população em crescimento;

5 – ENTÃO os indivíduos que apresentarem menor adaptação (determinadas pelo ambiente) morrerão;

6 – ENTÃO os indivíduos que apresentarem maior grau de adaptação (determinadas pelo ambiente) prosperarão.

Como resultado desse processo tem-se a evolução das espécies.

A.4. Parâmetros Genéticos

Alguns parâmetros influenciam diretamente no desempenho e até mesmo na convergência do AG. O grau de influência pode variar para mais ou para menos, dependendo da aplicação dos mesmos.

Dentre os diversos tipos de parâmetros, tem-se:

- Tamanho da população;
- Codificação;
- Função de Aptidão
- Taxa de cruzamento;
- Taxa de Mutação;
- Substituição;
- Convergência;
- Elitismo.

A seguir descritas as principais características de cada parâmetro para uma melhor compreensão do algoritmo.

a) Tamanho da População

O tamanho da população influi diretamente no desempenho e eficiência do AG. Quando a população é muito pequena, o algoritmo não abrange um espaço de busca satisfatório e como conseqüência pode resultar em convergência prematura. Para população muito grande, o espaço de busca fica muito bem representado no domínio do problema, mas em compensação o algoritmo consome um elevado tempo computacional. Portanto, o ajuste deste parâmetro é realizado por um operador com bastante experiência para que o algoritmo tenha um funcionamento adequado.

b) Taxa de Cruzamento

Este parâmetro indica quantos indivíduos da população irão reproduzir. Com uma taxa de cruzamento muito baixa, o algoritmo se torna direto, pois existirá pouca diversidade da população a cada geração. Para uma taxa de cruzamento alta, a população poderá ter uma perda no seu material genético, já que quase toda a população será substituída pelos seus descendentes.

c) Convergência

A convergência é um parâmetro que indica quando o algoritmo chegou a uma solução próxima do ótimo global. Basicamente, existem três tipos de critério de convergência. O primeiro é designado pelo número máximo de gerações. Neste caso, o algoritmo vai evoluir até um determinado número fixo de gerações, independentemente de ter ou não chegado a uma solução satisfatória. No segundo critério, são calculados, a cada geração, a média e o maior valor da função aptidão na população e se a diferença entre eles for menor que uma determinada tolerância, cujo valor deverá ser próximo de zero, o algoritmo convergiu, chegando assim em uma solução satisfatória. Por último, é determinado em tempo fixo de parada, independente da evolução do algoritmo.

d) Elitismo

O elitismo foi introduzido por Kenneth de Jong (1975), para contribuir em uma convergência satisfatória do algoritmo. Este método faz com que a melhor solução de uma determinada geração não seja perdida pela não seleção à reprodução.

A cada geração, alguns dos melhores indivíduos passam a fazer parte automaticamente para as próximas gerações, até que encontrem outros indivíduos melhores.

e) Função de Aptidão

Os valores de aptidão de cada indivíduo são calculados basicamente pela função objetivo. Quando se deseja maximizar, a função de aptidão se confunde com a função objetivo e quando se deseja minimizar a função aptidão deve ser invertida na forma mostrada pela Equação (1) ou sofrer um redimensionamento em relação a um dado valor

$C_{máx}$ mostrado pela Equação (2). Entretanto o processo de inversão pode causar um problema que é o da descontinuidade. Para resolvê-lo, deve ser somado à função objetivo um incremento menor que o passo da população para que este incremento não interfira no resultado. No redimensionamento, o valor de $C_{máx}$ pode ser constante ou variável ao longo das gerações, e deve ser escolhido de forma a manter a função objetivo sempre com valores positivos.

De uma maneira mais clara, tem-se a função de aptidão apresentada na Equação (1) com seus limites na Equação (2):

$$f_{apt}(x) = \frac{1}{f_0(x) + C_{\min}} \tag{1}$$

onde:

f_{apt} - Função de aptidão;

f_0 - Função objetivo;

$C_{\min}$ - Constante que evita a descontinuidade.

$$\begin{cases} f_{apt}(x) = C_{máx} - f_0(x) & se \quad f_0(x) < C_{máx} \\ f_{apt}(x) = 0 & se \quad f_0(x) \geq C_{máx} \end{cases} \tag{2}$$

onde:

$C_{máx}$ - Variável de redimensionamento.

A construção da função objetivo depende diretamente do tipo de problema a ser trabalhado, sendo que para cada problema tem-se que construir uma função objetivo diferente. Esta função é constituída de parâmetro que se deseja maximizar ou minimizar. Entretanto, o AG por ser um método probabilístico, não aceita que a função de aptidão tenha valores negativos. Para se resolver este problema, deve ser somada à função de

aptidão uma constante tal que seus valores sejam sempre positivos, conforme indicado na Equação (3).

$$f_{apt}(x) = f_0(x) + f_{\min} \tag{3}$$

onde:

$f_{\min}$ - Constante que torna positivo os valores da função aptidão.

Para a maioria dos problemas, existe ainda a inserção de restrições na função de aptidão. Este processo se faz adicionando-se à função objetivo um fator de penalidade, multiplicado por uma função penalidade, na qual são incorporadas as restrições do problema. Todas as vezes que algum parâmetro for violado, a função de penalidade atua tornado inviável o indivíduo responsável pela violação.

Uma demonstração deste método pode ser visto pela Equação (4):

$$f_0^{atual}(x) = f_0^{antiga}(x) + r \cdot Q(x) \tag{4}$$

f_0^{antiga} - Função objetivo real;

f_0^{atual} - Função objetivo falsa;

r - Fator de Penalidade;

Q - Função de Penalidade

f) Codificação

Os AGs trabalham com os indivíduos codificados em uma estrutura cromossômica ou cadeia de caracteres, sendo esta uma de suas características. Cada cromossomo representa uma variável na formação do indivíduo. A codificação é necessária para a aplicação dos operadores genéticos e existem diversos tipos de codificação, cuja escolha depende do tipo de problema a ser utilizado.

Dentre os mais utilizados, tem-se:

- Código decimal (base 10);
- Código binário (base 2);
- Código octal (base 8);
- Código hexadecimal (base 16);
- Código gray.

Nos sistemas de bases convencionais (decimal, binário, octal, hexadecimal, etc), cada dígito é compreendido entre 0 e (N-1). Onde N é a base de cada código. Qualquer número de cada sistema pode ser expresso em potência de base deste número.

A seguir tem-se uma pequena descrição de cada tipo para um melhor entendimento de cada sistema.

- **Código decimal**

Normalmente este tipo de codificação é usado para problemas de ordenação. Este código é o mais utilizado no sistema de ensino, representado pelo sistema real de numeração.

- **Código binário**

Este é um dos tipos mais utilizados pela sua simplicidade de tratamento. Neste caso cada cromossomo é representado por uma cadeia de genes ou bits que podem ter valores “0” ou “1”. Este é um processo importante já que a reprodução e a mutação trabalham com o parâmetro codificado.

O tamanho do cromossomo depende diretamente dos limites e da discretização de cada variável, conforme exemplo abaixo.

Parâmetros da variável.

$l_{\inf}$ - 0 Limite inferior

$l_{\sup}$ - 30 Limite Superior

pva - 1 Passo da variável

$$n_{discreto} = \frac{l_{\sup} - l_{\inf}}{pva} + 1 = 31 \tag{2.5}$$

onde $n_{discreto}$ é o espaço do domínio que a variável pode assumir.

$$l_{cromossomo} = 1 + \text{int}\left(\frac{\log\left(\frac{l_{\sup} - l_{\inf}}{pva} + 1 \right)}{\log(2)} \right) = 5 \tag{2.6}$$

onde $l_{cromossomo}$ é o número de bits que cada variável representa e "int" é o operador pára números inteiros.

Pode-se observar também que a variável com 5 bits pode assumir um valor máximo 2^5 = 32 e a variável no exemplo possui limite superior igual a 30. Para resolver este problema, tem-se que fazer uma mudança de escala para se enquadrar dentro dos limites de cada variável e em seguida aplicar os operadores genéticos.

- **Código Octal**

Este sistema é o menos usado por não ser tão trivial como o binário, mas tem sua importância em problemas específicos os quais podem facilitar em muito a aplicação dos operadores genéticos. Para determinados problemas, esse tipo de código pode melhorar em muito a convergência do algoritmo em relação aos outros códigos.

- **Código Hexadecimal**

Assim como no sistema octal este sistema é menos usado por não ser tão trivial, mas possui sua importância para determinadas aplicações.

- **Código Gray**

Este código surgiu para resolver um determinado problema que apareceu no código binário. O problema é que no código binário uma distância muito grande entre duas variáveis decimais vizinhas, ou seja, uma variável de valor 7 em binário representada por 0111 e uma variável 8 representada por 1000 tem diversos bits diferentes um do outro, mesmo sendo números vizinhos. Esta situação pode provocar, em determinadas problemas de otimização, fatos indesejados. Então este tipo de código surgiu como uma adaptação ao código binário, mas de forma a aproximar esta distância.

A transformação do código gray para decimal, começa com o número zero e um do decimal que são iguais aos do código gray. Para encontrar o número 2 e 3 refletem-se os números 0 e 1, no código gray, conforme a Figura A.1, sendo que os números refletidos são antecedidos pelo número 1.

Gray	Decimal
0 0	0
0 1	1
1 1	2
1 0	3

Figura A.1 – Transformação de Código Gray para Decimal

Para encontrar os números 4, 5, 6 e 7, basta repetir o processo descrito acima com os números 0, 1, 2 e 3 no código gray, conforme ilustrado na Figura A.2.

Gray	Decimal
0 0 0	0
0 0 1	1
0 1 1	2
0 1 0	3
1 1 0	4
1 1 1	5
1 0 1	6
1 0 0	7

Figura A.2 – Transformação de Código Gray para Decimal

O processo é repetido para a formação dos demais números. Assim, tem-se a transformação do código gray para decimal e vice-versa.

A.5. Operadores Genéticos

Os operadores genéticos são responsáveis pelo processo e otimização, que compreende a seleção e diversificação da espécie durante várias gerações (ou seja, pelo processo de otimização). Eles fazem com que indivíduos mais aptos tenham uma maior probabilidade de cruzamento e desta forma conservem suas melhores características de adaptação para seus descendentes. Em conseqüência, indivíduos com baixa adaptabilidade se perdem durante as gerações. Esta diversificação faz com que os indivíduos se adaptam melhor ao seu meio e conduz a uma população com valores de aptidão na maioria das vezes ótimos.

Basicamente os operadores genéticos são divididos em três categorias:

- Seleção;
- Cruzamento;
- Mutação.

a) Operador de Seleção

O operador de seleção tem como objetivo selecionar os indivíduos mais adaptados ao seu meio ambiente para sofrer a ação dos operadores de cruzamento e mutação e conseqüentemente gerar uma população mais adaptada ao seu nicho ecológico. Deste modo, as gerações futuras terão uma menor probabilidade de serem extintas.

Abaixo são apresentados alguns dos mais importantes tipos de seleção.

- **Roleta**

O método de seleção roleta assemelha-se a uma roleta de cassino. O processo é feito avaliando-se cada indivíduo através da adaptabilidade em relação ao meio ambiente. Depois da avaliação de cada indivíduo calculam-se os valores percentuais de aptidão e com estes valores monta-se uma roleta de área proporcional. Roda-se a roleta com o mesmo

número de vezes que o tamanho da população (N), com isso os indivíduos que possuem maior adaptabilidade tem uma maior chance de serem selecionados para reprodução.

Esta probabilidade é dada pela Equação (7):

$$P_i = \frac{f_i}{f_m} \tag{7}$$

onde f_i é o valor da função avaliação (ou aptidão) e f_m é a média dos valores da função de avaliação dada pela Equação (8):

$$f_m = \frac{\sum_{i=1}^{N} f_i}{N} \tag{8}$$

- **Ranking**

Neste método de seleção, faz-se a ordenação decrescente dos indivíduos conforme seus valores de adaptabilidade. Em seguida, seleciona-se o número de indivíduos que se deseja realizar o cruzamento, de acordo com a ordenação realizada.

- **Torneio**

Este método é derivado do método da roleta, pois seu processo retorna o melhor indivíduo, em relação a sua adaptabilidade, de dois resultados obtidos pelo método roleta. Os indivíduos com baixa adaptabilidade, neste método, possuem poucas chances de serem escolhidos para a aplicação dos operadores genéticos, mesmo que estas sejam muito baixas. Por beneficiar em muito os indivíduos com maior adaptabilidade, este processo pode favorecer a convergência prematura.

b) Operador de Cruzamento

O operador de cruzamento tem como objetivo realizar a troca de material genético dos progenitores escolhidos pelo operador de seleção durante o cruzamento. Desta forma, seus descendentes herdarão parte das características de um progenitor e parte do outro. Com isso, as características dos progenitores mais adaptados serão conservadas de geração em geração pelos seus descendentes, fazendo com que estes se adaptem melhor ao meio em que vive. Este operador é realizado em um número fixo de indivíduos regidos por uma taxa de cruzamento (t_c).

Dentre os vários tipos de cruzamento, temos:

- Cruzamento de um ponto de corte;
- Cruzamento de "x" pontos de corte;
- Cruzamento uniforme;
- Cruzamento variável;
- Cruzamento entre vários indivíduos;

A seguir serão descritas as principais características de cada cruzamento para uma melhor compreensão do algoritmo.

- **Cruzamento de Um Ponto de Corte**

Neste tipo de cruzamento, escolhe-se aleatoriamente um ponto no cromossomo. Um dos descendentes recebe parte do cromossomo de um dos progenitores e parte do outro, sendo que o segundo descendente recebe as partes restantes dos progenitores. Sendo assim, o ponto de corte pode variar de um até L-1, onde L é o comprimento do cromossomo de cada indivíduo.

Uma representação deste tipo de cruzamento é mostrada nas Figuras A.3 e A.4, com o tamanho do cromossomo de 8 bits, codificação binária e com o ponto de corte escolhido aleatoriamente na posição 4.

Figura A.3 – Cromossomo dos Genitores

Figura A.4 – Cromossomo dos Descendentes

- **Cruzamento de "x" Pontos de Corte**

Neste tipo de cruzamento, o procedimento é muito semelhante ao do cruzamento de um ponto. A principal diferença é que neste tipo escolhe-se "x" pontos de corte aleatoriamente. Os cromossomos dos progenitores são divididos nos "x" pontos de corte e seus descendentes recebem estes "pedaços" de modo alternado.

Do mesmo modo que o anterior, as Figuras A.5 e A.6 mostram uma representação com o tamanho do cromossomo de 8 bits, codificação binária e pontos de corte em número de 2, escolhidos aleatoriamente nas posições 2 e 6.

Figura A.5 – Cromossomo dos Genitores

Figura A.6 – Cromossomo dos Descendentes

- **Cruzamento Uniforme**

Neste tipo de cruzamento, cria-se uma máscara aleatória com o mesmo tamanho dos cromossomos dos progenitores. O primeiro descendente é formado de tal forma que, quando a máscara tiver bit ativo ou igual a um, este descendente herda o bit do primeiro progenitor e quando a máscara tiver bit inativo ou igual a zero, o progenitor herda o bit do segundo progenitor. Para formar o segundo descendente, o processo ocorre de modo semelhante, mas inverso. Quando a máscara tiver bit ativo, o segundo descendente herda o bit do segundo progenitor e quando a mascar tiver bit inativo, o descendente herda o bit do primeiro progenitor.

Uma representação para este tipo de cruzamento será mostrada através das Figuras A.7, A.8 e A.9 com o tamanho do cromossomo de 8 bits e com codificação binária.

Figura A.7 – Cromossomo dos Genitores

Figura A.8 – Máscara Criada Aleatoriamente

Figura A.9 – Cromossomo dos Genitores

- **Cruzamento por Variável**

Este tipo de cruzamento possui também uma semelhança muito grande com o cruzamento de um ponto de corte. A diferença é que realiza a troca de material genético em um ponto de corte, mas não apenas por indivíduo e sim por cada variável ou gene do indivíduo.

Uma melhor apresentação deste método será mostrada nas figuras A.10 e A.11, sendo que cada indivíduo utiliza código binário, os indivíduos possuem duas variáveis, o

tamanho destas variáveis são de 3 bits e os pontos de corte escolhidos aleatoriamente são em número de 2 e nas posições 2 e 1.

Figura A.10 – Cromossomo dos Genitores

Figura A.11 – Cromossomo dos Descendentes

- **Cruzamento entre Vários Indivíduos**

Neste tipo de cruzamento, escolhe-se um indivíduo base e, para cada variável deste, tem-se que escolher um outro indivíduo e um ponto para o cruzamento, ou seja, cada variável do indivíduo base realiza o cruzamento com um indivíduo diferente, escolhido aleatoriamente. Desta forma, tenta-se aproveitar um maior número de características na população. Outro fato importante é que este método de cruzamento gera apenas um único descendente.

Para um melhor entendimento, uma representação será mostrada nas Figuras A.12 e A.13, sendo que cada indivíduo utiliza código binário, os indivíduos possuem duas variáveis, o tamanho destas variáveis são de 3 bits e os pontos de corte escolhidos aleatoriamente são em número de 2 e nas posições 2 e 1.

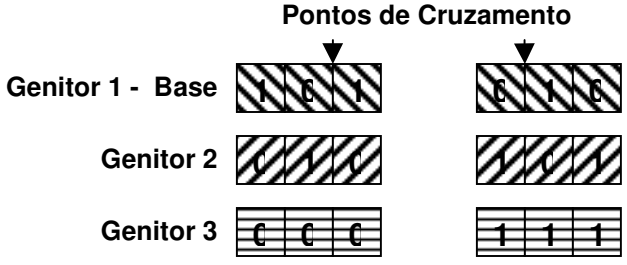


Figura A.12 – Cromossomo dos Genitores

Descendente

Figura A.13 – Cromossomo do Descendente

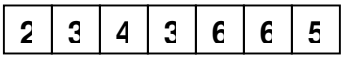
- **Cruzamento por Média**

Neste tipo de cruzamento, utiliza-se código real. Para realizar o cruzamento faz-se a média dos bits dos genitores formando assim um descendente. Caso uma média não seja valor inteiro, realiza-se uma escolha aleatória para saber se o valor será arredondado para cima ou para baixo.

Nas Figuras A.14 e A.15, tem-se a representação deste tipo de cruzamento, sendo que as posições 4 e 7 não obtiveram valores inteiros. Na posição 4, o valor foi arredondado para baixo, escolhido de forma aleatória, e na posição 7 o valor foi arredondado para cima, também de forma aleatória.

Genitor 1 **Genitor 2**

Figura A.14 – Cromossomo dos Genitores



Descendente

Figura A.15 – Cromossomo dos Genitores

c) Operador de Mutação

O operador de mutação tem como objetivo inserir novas características nos descendentes e até mesmo restaurar as características perdidas a cada geração. O processo faz com que alguns descendentes de cada geração, regidos por um percentual denominado taxa de mutação (t_m), sofram uma troca no valor de um de seus bits. A posição do bit é escolhida de forma aleatória.

Dentre os tipos de mutação, temos:

- Mutação por troca de bit;
- Mutação uniforme;
- Troca de posições;
- Adição ou subtração em um bit.

B. REFERÊNCIAS BIBLIOGRÁFICAS

[1] Dias, J.S.; Barreto, J.M. (1998), "Algoritmo genético: inspiração biológica na solução de problemas - uma introdução", Revista Marítima Brasileira - Suplemento Especial, Pesquisa Naval, nº 11, p. 105-128.

[2] M. Srinivas, Lalit M. Patnaik, "Genetic algorithms: A Survey," Computer, vol. 27, no. 6, pp. 17-26, Jun., 1994 [3] Fogel, D.B. (1995), "Evolutionary Computation", IEEE Press: Piscataway, NJ.

[3] Fogel, D.B. (1995), "Evolutionary Computation", IEEE Press: Piscataway, NJ.

[4] Furtado, J.C.; Lorena, L.A.N. (1998), "Algoritmo genético construtivo na otimização de problemas combinatórios de agrupamentos", III Oficina de cortes e empacotamento, lac.inpe.br.

[5] Whitley, D. (1994), "*A genetic algorithm tutorial*", *Springer Science + Business Media* B.V., Formerly Kluwer Academic. p. 65-85.

[6] Fogel, G.B.; Porto, V.W.; Weekes, D.G.; Griffey, R.H.; Mcneil, J.A.; Lesnik, E.; Ecker, D.J.; Sampath, R. (2002), "*Discovery of RNA structural elements using evolutionary computation*", *Nucleic Acids Research*, v.30, nº 23, p. 5310-5317.

[7] Satrom, P.; Sneve, R.; Kristiansen, K.I; Snove, O.; Grünfeld, T.; Rognes, T.; Seeberg, E. (2005), "*Predicting non-coding RNA genes in Escherichia Coli with boosted genetic programming*", *Nucleic Acids Research,* v.33, nº 10, p. 3263-3270.

[8] Walker J.D.; File, P.E.; Miller, C.J.; Samson, W.B. (1994), "*Building DNA maps, a genetic algorithm based approach*", *Advances in molecular bioinformatics*. S. Schulze-Kremer, IOS Press. p. 179-199.

[9] Aerts, S.; Loo, P.V.; Moreau, Y.; Moor, D.B. (2004), "*A genetic algorithm for the detection of new cis-regulatory modules in sets of coregulated genes*", *Bioinformatics*, *Oxford University Press*.

[10] Fogel, G.B.; Weekes, D.G.; Varga, G.; Dow, E.R; Harlow, H.B.; Onyia, J.E.; Su, C. (2004) "*Discovery of sequence motifs related to coexpression of genes using evolutionary computation*", *Nucleic Acids Research*, v. 32, nº 13, p. 3826-3835.

[11] Azevedo, F.M. (1999), "Algoritmos genéticos em redes neurais artificiais", V Escola de Redes Neurais, promoção: Conselho Nacional de Redes Neurais – ITA, São José dos Campos, SP.

[12] Barreto, J.M. (1996), "Conexionismo e a Resolução de Problemas", *Titular Professor Contestissertation*, Departamento de Informática e Estatística, UFSC, Florianópolis, SC.

[13] Darwin, C. (2004), "A origem das espécies". Rio de Janeiro: Ediouro.

[14] Stearns, S.C. (2003), "Evolução: uma introdução", Atheneu: São Paulo.

[15] Futuyama, D.J. (2003), "Biologia evolutiva", Ribeirão Preto: FUNPEC-RP.

[16] Michalewicz, Z. (1996), "*Genetic algorithm + data structures = evolution programs*", 3º ed. *Springer-Verlag*.

[17] Haupty, R.L.; Haupty, S.E. (2004), "*Practical genetic algorithm*", 2º ed. *A John Wiley & Sons, Inc., Publications.*

[18] Koza, J.R. (1995), "*Survey of genetic algorithms and genetic programming*", *Proc. of Wescon* 95, IEEE *Press*, p. 589-594.

[19] Zuben, F.J.V. (2000), "Computação evolutiva: uma abordagem pragmática", Tutorial: Notas de aula da disciplina IA707, DCA/FEEC/Unicamp.

[20] SILVA, H. A. (2007), "Treinamento – Seletividade de Sistemas Elétricos", Relatório Marco 8, Projeto P&D Endesa Fortaleza\UFJF e UFRJ

[21] VAMP (2007), "VAMP 210 – Relé de Proteção de Geradores – Instruções de Operação e Configuração"

[22] Institute of Electrical and Electronic Engineers (IEEE) (2001), “IEEE Std 242-2001 – Recommended Practice for Protection and Coord. of Industrial and Commercial Power Systems - IEEE Buff BookT”

[23] MANTOVANI, Jose Eduardo (2006), “Sistema Inteligente para Alocação, Especificação, Coordenação e Seletividade da Proteção em Redes Aéreas de Distribuição de Energia Elétrica”,XVII Seminário Nacional de Distribuição de Energia Elétrica – Belo Horizonto/MG

[24] International Electrotechnical Commission (IEC) (1989),” IEC 60255-3 – Electrical relays - Part 3: Single input energizing quantity measuring relays with dependent or independent time”

[25] GARCIA, P. A. N. (2007), “Sistemas Inteligentes Aplicados à Confiabiliade e Seletividade de Plantas Termelétricas”, Relatório Marco 7, Projeto P&D Endesa Fortaleza\UFJF e UFRJ