

PAULO DE TARSO E P. COUTINHO

**MINIMIZAÇÃO DE PERDAS EM SISTEMAS ELÉTRICOS INDUSTRIAIS
UTILIZANDO CAPACITORES *SHUNT* DIMENSIONADOS POR ALGORITMO
GENÉTICO**

Dissertação apresentada ao Programa
de Pós-Graduação em Engenharia Elétrica -
PPGEE, da Universidade Federal da Paraíba -
UFPB, como requisito para a obtenção do título
de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Yuri Percy Molina

JOÃO PESSOA

2017

PAULO DE TARSO E P. COUTINHO

**MINIMIZAÇÃO DE PERDAS EM SISTEMAS ELÉTRICOS INDUSTRIAIS
UTILIZANDO CAPACITORES *SHUNT* DIMENSIONADOS POR ALGORITMO
GENÉTICO**

Dissertação apresentada ao Programa
de Pós-Graduação em Engenharia Elétrica -
PPGEE, da Universidade Federal da Paraíba -
UFPB, como requisito para a obtenção do título
de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Yuri Percy Molina

JOÃO PESSOA

2017

C871m Coutinho, Paulo de Tarso e P.

Minimização de perdas em sistemas elétricos industriais utilizando capacitores shunt dimensionados por algoritmo genético / Paulo de Tarso e P. Coutinho. – João Pessoa, 2017.

144 f.: il. -

Orientador: Yuri Percy Molina Dissertação (Mestrado) - UFPB/
PPGEE

1. Sistemas Elétricos Industriais. 2. Capacitores. 3. Engenharia Elétrica. I. Título.

UFPB/BC

CDU: 621.3(043)

UNIVERSIDADE FEDERAL DA PARAÍBA – UFPB
CENTRO DE ENERGIAS ALTERNATIVAS E RENOVÁVEIS – CEAR
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA -
PPGEE

A Comissão Examinadora, abaixo assinada, aprova a Dissertação

MINIMIZAÇÃO DE PERDAS EM SISTEMAS ELÉTRICOS INDUSTRIAIS UTILIZANDO
CAPACITORES *SHUNT* DIMENSIONADOS POR ALGORITMO GENÉTICO

Elaborada por

PAULO DE TARSO E P. COUTINHO

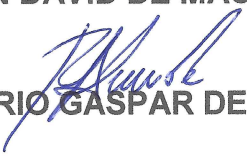
Como requisito para obtenção do grau de
Mestre em Engenharia Elétrica.

COMISSÃO EXAMINADORA


DR. YURI PERCY MOLINA RODRIGUEZ (Presidente)

DR. ALEXANDRE CÉZAR DE CASTRO (Examinador Interno)

DR. HELON DAVID DE MACEDO BRÁZ (Examinador Interno)


DR. ROGÉRIO GASPARD DE ALMEIDA (Examinador Externo)

João Pessoa, 24 de março de 2017.

AGRADECIMENTOS

Ao professor Yuri, eternos agradecimentos pelas frequentes e sempre sábias orientações.

A todos os meus mestres e a todos os meus colegas, eternos agradecimentos pelos ensinamentos que me propiciaram.

RESUMO

MINIMIZAÇÃO DE PERDAS EM SISTEMAS ELÉTRICOS INDUSTRIAIS UTILIZANDO CAPACITORES *SHUNT* DIMENSIONADOS POR ALGORITMO GENÉTICO

Este trabalho propõe um procedimento para minimização de perdas elétricas presentes em sistemas industriais, através de um programa específico desenvolvido para este fim, estruturado como Algoritmo Genético, e escrito usando instruções *MATLAB®*. A minimização é realizada alocando capacitores *shunt* conectados às barras e a performance do programa desenvolvido, considerando exatidão e velocidade, é qualificada através da comparação de seus resultados com os resultados produzidos por programas tradicionais. O programa evita fator de potência capacitivo na barra conectada com a empresa que faz a distribuição de energia e o usuário pode determinar as barras onde ele não quer que sejam instalados capacitores. O valor de aptidão que guia o processo de otimização do Algoritmo Genético é a perda elétrica ativa presente no sistema elétrico. Como encerramento o programa é aplicado em um sistema elétrico industrial de 38 barras.

Palavras-chave: Minimização de Perdas, Sistemas Elétricos Industriais, Algoritmo Genético, Capacitores.

ABSTRACT

MINIMIZING INDUSTRIAL LOOSES USING SHUNT CAPACITORS CALCULATED BY GENECTIC ALGORITHM

This work proposes a procedure to minimize electrical losses present in industrial systems, through a specific program developed for this purpose, based on Genetic Algorithm, and written using MATLAB instructions. The minimization is realized with the placement of shunt capacitors connected to the buses. For validation of the proposed method, it is considered accuracy and speed, and the developed program's results are compared with the results produced by traditional programs. The program avoids leading power factor at the bus connected with the power distribution company and the user may determine buses where he does not want the installation of capacitors. The Genetic Algorithm's fitness, responsible for guiding the optimization process, is the active power loss. Finally, the program is applied to minimize active losses present at an industrial system, 38 buses.

Keywords: Looses, Minimization, Industrial Electric Systems, Genetic Algorithm, Capacitors.

LISTA DE ILUSTRAÇÕES

FIGURA 1: ESTRUTURA BÁSICA DOS ALGORITMOS GENÉTICOS	22
FIGURA 2: ESTRUTURA DE ALGORITMO GENÉTICO DEDICADO	24
FIGURA 3: FUNÇÃO $F(X) = (X * \text{SEN}(10 * \pi * X) + 1)$	28
FIGURA 4: SELEÇÃO TIPO ROLETA	30
FIGURA 5: MODELO PI	39
FIGURA 6: INTERFACE GRÁFICA DO PROGRAMA <i>POWERFACTORY</i> ®	43
FIGURA 7: INTERFACE DO PROGRAMA <i>POWERFACTORY</i> ® RELATIVA À ENTRADA DE DADOS	44
FIGURA 8: EXEMPLO DE RELATÓRIO DO PROGRAMA <i>POWERFACTORY</i> ®	45
FIGURA 9: $F(X) = (X * \text{SENO}(10 * \pi * X) + 1)$ DENTRO DO ESPAÇO DE BUSCA -1 A 2	48
FIGURA 10: ERROS RELATIVOS INCORRIDOS NAS OTIMIZAÇÕES REALIZADAS COM REPRESENTAÇÃO BINÁRIA E BUSCA PELO MÁXIMO	49
FIGURA 11: ERROS RELATIVOS INCORRIDOS NAS OTIMIZAÇÕES REALIZADAS COM REPRESENTAÇÃO BINÁRIA E BUSCA PELO MÍNIMO	49
FIGURA 12: ERROS RELATIVOS INCORRIDOS NAS OTIMIZAÇÕES REALIZADAS COM REPRESENTAÇÃO REAL E BUSCA PELO MÁXIMO	50
FIGURA 13: ERROS RELATIVOS INCORRIDOS NAS OTIMIZAÇÕES REALIZADAS COM REPRESENTAÇÃO REAL E BUSCA PELO MÍNIMO	50
FIGURA 14: ESTRUTURA MACRO DO PROGRAMA OBJETO DA DISSERTAÇÃO (PAGIND)	55
FIGURA 15: ESTRUTURA DETALHADA DO PROGRAMA OBJETO DA DISSERTAÇÃO (PAGIND)	57
FIGURA 16: EXEMPLO DE EVOLUÇÃO DA PERDA ELÉTRICA EM FUNÇÃO DA POTÊNCIA CAPACITIVA INSTALADA	62
FIGURA 17: EVOLUÇÃO DA VARIÁVEL “INTENSIDADE DE MUTAÇÃO”	65
FIGURA 18: FUNÇÃO $F_1(X) = 1,5 * X + \text{SEN}(20 * \pi * X) + 1$	71
FIGURA 19: FUNÇÃO $F_2(X) = X * \text{SEN}(10 * \pi * X) + 1$	71
FIGURA 20: FUNÇÃO $F_3(X) = (1/2) * (X - 0,5)^2 + 1$	72
FIGURA 21: FUNÇÃO $F_4(X) = -5 * X + 1,5 * X^3$	72
FIGURA 22: FUNÇÃO $F_5(X) = 0,52 * X^3 + 9,9 * X^2 - 37 * X + 121$	72
FIGURA 23: UNIFILAR DO SISTEMA ELÉTRICO UTILIZADO PARA VALIDAÇÃO DO BLOCO “ROTINAS FC”	78
FIGURA 24: RELATÓRIO EMITIDO PELO PROGRAMA <i>POWERFACTORY</i> ® COM RESULTADOS RELATIVOS ÀS PERDAS INTERNAS	82
FIGURA 25: RELATÓRIO EMITIDO PELO PROGRAMA <i>POWERFACTORY</i> ® COM RESULTADOS RELATIVOS ÀS BARRAS ELÉTRICAS E RAMOS ELÉTRICOS	82
FIGURA 26: UNIFILAR DO SISTEMA ELÉTRICO DE 38 BARRAS	88
FIGURA 27: COMPLEMENTAÇÃO DO UNIFILAR DO SISTEMA ELÉTRICO DE 38 BARRAS	89
FIGURA 28: CURVA DE MUTAÇÃO ADAPTATIVA E PERDAS ELÉTRICAS	91
FIGURA 29: VALOR DE APTIDÃO E POTÊNCIA TOTAL DE CAPACITORES	91

FIGURA 30: VALOR TOTAL DA POTÊNCIA CAPACITIVA CALCULADA À CADA MINIMIZAÇÃO	92
FIGURA 31: PERDA TOTAL REMANESCENTE APÓS CADA MINIMIZAÇÃO	92
FIGURA 32: VALOR TOTAL DA POTÊNCIA CAPACITIVA CALCULADA COM DIFERENTES CARREGAMENTOS.....	95
FIGURA 33: PERDA ELÉTRICA (KW) CALCULADA COM DIFERENTES CARREGAMENTOS.....	95

LISTA DE TABELAS

TABELA 1: POPULAÇÃO COM 2 INDIVÍDUOS, REPRESENTAÇÃO REAL.....	26
TABELA 2: POPULAÇÃO COM 2 INDIVÍDUOS, REPRESENTAÇÃO BINÁRIA	26
TABELA 3: INDIVÍDUOS BINÁRIOS E SEUS VALORES DE APTIDÃO	29
TABELA 4: SELEÇÃO TIPO ROLETA, BUSCA POR UM PONTO DE MÁXIMO	30
TABELA 5: POPULAÇÃO A SER SUBMETIDA AO PROCESSO DE SELEÇÃO TIPO TORNEIO.....	31
TABELA 6: POPULAÇÃO GERADA PELA SELEÇÃO TIPO TORNEIO, BUSCA PELO MÍNIMO	31
TABELA 7: PAIS/INDIVÍDUOS SELECIONADOS PARA CRUZAMENTO SIMPLES.....	32
TABELA 8: FILHOS, CRUZAMENTO SIMPLES, TROCA APÓS O TERCEIRO BIT	32
TABELA 9: PAIS SELECIONADOS PARA CRUZAMENTO EM DOIS PONTOS	33
TABELA 10: FILHOS, RESULTANTES DO CRUZAMENTO EM DOIS PONTOS.....	33
TABELA 11: CÁLCULOS ENVOLVIDOS NO CRUZAMENTO DE INDIVÍDUOS REAIS	33
TABELA 12: POPULAÇÃO DE QUATRO INDIVÍDUOS ANTES DA MUTAÇÃO	34
TABELA 13: POPULAÇÃO DE QUATRO INDIVÍDUOS APÓS A MUTAÇÃO	34
TABELA 14: POPULAÇÃO REAL ANTES DA MUTAÇÃO	35
TABELA 15: POPULAÇÃO REAL APÓS A MUTAÇÃO	35
TABELA 16: RELAÇÃO CONTENDO TIPO DE VARIÁVEIS EM FUNÇÃO DO TIPO DE BARRAS	36
TABELA 17: SÍNTESE DAS OTIMIZAÇÕES REALIZADAS COM POPULAÇÕES DE 220 INDIVÍDUOS, $F(X) = [X * \text{SENO}(10 * \pi * X) + 1]$, ESPAÇO DE BUSCA -1 A 2	51
TABELA 18: SÍNTESE DAS OTIMIZAÇÕES REALIZADAS COM POPULAÇÕES DE 660 INDIVÍDUOS, $F(X) = [X * \text{SENO}(10 * \pi * X) + 1]$, ESPAÇO DE BUSCA -1 A 2	52
TABELA 19: RELAÇÃO DE FUNÇÕES E “SCRIPT” EM RELAÇÃO À APLICAÇÃO DENTRO DA ESTRUTURA DO PAGIND	56
TABELA 20: MATRIZ “PI_BIN”, CODIFICADA POR 5 BITS.....	66
TABELA 21: MATRIZ “FAIXA_BUSCA”, CODIFICADA COM 5 BITS, CAPACITORES EM MVAR.....	67
TABELA 22: MATRIZ “PI_MVAR, CAPACITORES EM MVAR, VALORES NÃO COMERCIAIS.....	67
TABELA 23: MATRIZ “TABELA_COMERCIAL, VALORES EM KVAR	68
TABELA 24: MATRIZ “PI_MVAR_COMERCIAL”, MOSTRANDO PARTE DE SEUS INDIVÍDUOS, JÁ COM VALORES COMERCIAIS	68
TABELA 25: RESUMO DOS PARÂMETROS EXISTENTES NO PAGIND	69
TABELA 26: ESTATÍSTICA DOS RESULTADOS PRODUZIDOS PELO BLOCO “ROTINAS AG”	74
TABELA 27: DADOS ALIMENTADOS NO PROGRAMA <i>POWERFACTORY</i> ®	79
TABELA 28: DADOS ALIMENTADOS NO BLOCO “ROTINAS FC”, CORRESPONDENTES ÀS BARRAS ELÉTRICAS.....	80
TABELA 29: DADOS ALIMENTADOS NO BLOCO “ROTINAS FC”, CORRESPONDENTES AOS RAMOS ELÉTRICOS	81
TABELA 30: RESULTADOS GERADOS PELO BLOCO “ROTINAS FC”, RELATIVOS ÀS BARRAS ELÉTRICAS.....	83
TABELA 31: RESULTADOS GERADOS PELO BLOCO “ROTINAS FC”, RELATIVOS AOS RAMOS ELÉTRICOS	83
TABELA 32: RESULTADOS DO PROGRAMA <i>POWERFACTORY</i> ® E DO BLOCO “ROTINAS FC”	84

TABELA 33: RESULTADOS GERADOS PELO PROGRAMA <i>POWERFACTORY</i> ® E PELO BLOCO “ROTINAS FC”	84
TABELA 34: DADOS DAS BARRAS DO SISTEMA ELÉTRICO SUBMETIDO À MINIMIZAÇÃO	90
TABELA 35: DADOS DOS RAMOS DO SISTEMA ELÉTRICO SUBMETIDO À MINIMIZAÇÃO.....	90
TABELA 36: RESULTADOS PRODUZIDOS PELO PAGIND SEM CAPACITORES E COM CAPACITORES	93

LISTA DE ABREVIATURAS E SIGLAS

AG	Algoritmo Genético
AG's	Algoritmos Genéticos
FA	Função aptidão (valor de aptidão)
FA's	Funções de aptidão
FC	Fluxo de Carga
FO	Função Objetivo
F.P	Fator de Potência
F.P's	Fatores de Potência
kW	Quilowatt
MW	Megawatt
MVar	Mega volt ampere reativo
<i>MATLAB®</i>	Ambiente de programação <i>MATLAB®</i>
<i>POWERFACTORY®</i>	Programa para cálculos elétricos
PAGIND	Programa Algoritmo Genético Industrial

SUMÁRIO

RESUMO	III
ABSTRACT	IV
LISTA DE ILUSTRAÇÕES	5
LISTA DE TABELAS	7
LISTA DE ABREVIATURAS E SIGLAS	9
1 INTRODUÇÃO	15
1.1 MOTIVAÇÃO.....	16
1.2 OBJETIVO	16
1.2.1 Objetivo Geral	16
1.2.2 Objetivos Específicos	17
1.2.3 Organização do trabalho	17
2 REVISÃO DE LITERATURA	21
2.1 ALGORITMOS GENÉTICOS	21
2.1.1 A teoria de Darwin, pedra fundamental dos Algoritmos Genéticos	21
2.1.2 Funcionamento do Algoritmo Genético Básico	22
2.1.3 Aplicações dos AG's	23
2.1.4 Algoritmos Genéticos "Dedicados"	24
2.1.5 Componentes e parâmetros dos Algoritmos Genéticos.....	25
2.2 FLUXO DE CARGA.....	36
2.2.1 Expressões gerais relativas ao fluxo de carga	37
2.2.2 Cálculo de fluxo de carga através do método <i>Newton Raphson</i>	40
2.3 O PROGRAMA <i>POWERFACTORY</i> ®.....	42
2.3.1 Desenhando o diagrama unifilar no programa <i>POWERFACTORY</i>	42
2.3.2 Introduzindo dados no programa <i>POWERFACTORY</i>	43
2.3.3 Obtendo resultados gerados pelo programa <i>POWERFACTORY</i>	44
3 REPRESENTAÇÃO BINÁRIA OU REAL?	47
3.1 OTIMIZAÇÕES REALIZADAS PARA AVALIAR AS FORMAS DE REPRESENTAÇÃO DOS INDIVÍDUOS	48
3.1.1 Resultados das otimizações em busca do máximo, representação binária ..	49
3.1.2 Resultados das otimizações em busca do mínimo, representação binária ...	49
3.1.3 Resultados das otimizações em busca do máximo, representação real	50
3.1.4 Resultados das otimizações em busca do mínimo, representação real.....	50
3.2 RESUMO DAS 400 OTIMIZAÇÕES REALIZADAS COM POPULAÇÃO DE 220 INDIVÍDUOS	51

3.3	CONCLUSÕES SOBRE AS OTIMIZAÇÕES COM 220 INDIVÍDUOS	51
3.3.1	Tempo de execução.....	51
3.3.2	Eliminando erros de otimizações realizadas com Algoritmos Genéticos	52
3.4	TOMADA DE DECISÃO SOBRE A FORMA DE REPRESENTAÇÃO A SER UTILIZADA NO PROGRAMA PAGIND	53
4	DESENVOLVIMENTO DO PROGRAMA OBJETO DA DISSERTAÇÃO.....	55
4.1	O FLUXOGRAMA DO PAGIND	56
4.1.1	“Script” “A_iniciar”	58
4.1.2	Função “B_dados_gerais”	58
4.1.3	Função “C_dados_38b”	58
4.1.4	Função “D_pi_bin”	59
4.1.5	Função “E_conv_pi_bin_MVAr”	59
4.1.6	Função “F_tab_comercial”	59
4.1.7	Função “G_fc_mestre”	59
4.1.8	Função “G_fc_escravo”	60
4.1.9	Função “I_aptidão”	60
4.1.10	Função “J_penal”	62
4.1.11	Função “K_selec”	63
4.1.12	Função “L_cruz”	64
4.1.13	Função “M_mut”	64
4.2	CONVERSÃO DE POPULAÇÃO BINÁRIA PARA POPULAÇÃO COMERCIAL.....	66
4.3	PARÂMETROS EXISTENTES NO PAGIND	68
5	VALIDAÇÃO DO BLOCO “ROTINAS AG”	71
5.1	VALIDANDO O BLOCO “ROTINAS AG”	73
5.2	RESUMO DOS RESULTADOS PRODUZIDOS PELO BLOCO “ROTINAS AG” QUANDO REALIZANDO MINIMIZAÇÕES.....	74
5.3	CONCLUSÕES SOBRE OS RESULTADOS PRODUZIDOS PELO BLOCO “ROTINAS AG”	74
5.4	TOMADA DE DECISÃO SOBRE A EFICIÊNCIA DO BLOCO “ROTINAS AG”	75
6	VALIDAÇÃO DO BLOCO “ROTINAS FC”	77
6.1	O SISTEMA ELÉTRICO UTILIZADO PARA VALIDAÇÃO DO BLOCO “ROTINAS FC”	77
6.2	DADOS INSERIDOS NO PROGRAMA POWERFACTORY®	78
6.3	DADOS INSERIDOS NO BLOCO “ROTINAS FC”	79
	FONTE: DO AUTOR	81

6.4	RESULTADOS GERADOS PELO PROGRAMA VALIDADOR (POWERFACTORY®).....	81
6.5	RESULTADOS GERADOS PELOS PROGRAMAS SOB VALIDAÇÃO (ROTINAS FC).....	81
6.6	COMPARANDO RESULTADOS	83
6.6.1	Comparando resultados relativos às barras (tensões e ângulos)	83
6.6.2	Comparando resultados relativos às perdas	84
6.6.3	Comparando tempos de execução.....	84
6.7	TOMADA DE DECISÃO SOBRE A EFICIÊNCIA DO BLOCO “ROTINAS FC”	85
7	APLICAÇÃO DO PROGRAMA OBJETO DA DISSERTAÇÃO	87
7.1	UNIFILAR DO SISTEMA ELÉTRICO 38 BARRAS SUBMETIDO À MINIMIZAÇÃO DE PERDAS.....	87
7.2	PARÂMETROS AJUSTADOS NO PROGRAMA PAGIND.....	91
7.3	MINIMIZAÇÕES REALIZADAS À PLENA CARGA	92
7.3.1	Resultados das minimizações, carregamento nominal	92
7.3.2	Estatísticas relativas às otimizações com carregamento nominal	94
7.4	VARIANDO O CARREGAMENTO	94
7.5	PERFORMANCE DO PROGRAMA DESENVOLVIDO.....	95
7.5.1	Pontos Positivos do programa desenvolvido	95
7.5.2	Limite do programa desenvolvido.....	96
8	CONCLUSÕES.....	98
	REFERÊNCIAS	101
	APÊNDICE 1	104
	SCRIPT “A_INICIAR”	104
	FUNÇÃO “B_DADOS_GERAIS”	109
	FUNÇÃO “C_DADOS_38B”	111
	FUNÇÃO “D_PI_BIN”.....	115
	FUNÇÃO “E_CONV_PI_BIN_MVAR”	116
	FUNÇÃO “F_TAB_COMERCIAL”	118
	FUNÇÃO “G_FC_MESTRE”	119
	FUNÇÃO “H_FC_ESCRAVO”	120
	FUNÇÃO “I_APTIDAO”.....	132
	FUNÇÃO “J_PEN”	133
	FUNÇÃO “K_SELEC”	135
	FUNÇÃO “L_CRUZ”.....	137

<i>FUNÇÃO “M_MUT”</i>	139
-----------------------------	-----

1 INTRODUÇÃO

1 INTRODUÇÃO

Todos os recursos existentes na natureza são finitos, inclusive os econômicos, e esta consciência mantém a sociedade contemporânea em constante busca por meios de produção mais eficientes.

As redes elétricas não escapam da incessante busca por eficiência e as empresas envolvidas na transmissão e distribuição de energia tentam e buscam novas formas para reduzir suas perdas. Existem dezenas de trabalhos acadêmicos que tratam da minimização de perdas em redes elétricas, e eles diferem, caso a caso, no tipo de algoritmo, no dispositivo físico corretor e, também, na forma de quantificar a melhoria.

Quanto aos dispositivos físicos, alguns estudiosos (DAS, 1999), (ABUL'WAFI, 2013) e (MOHAMMAD H. MORADI, 2014) desenvolveram trabalhos buscando a minimização das perdas através da implantação de capacitores *shunt*, com e sem chaveamento de fontes de geração distribuída; obstante a existência de trabalhos (MUSIRIN, 2014) optando pela reconfiguração de alimentadores associada à geração distribuída. Ainda tratando do dispositivo físico, alguns autores (Q. A. SALIH, 2015) avaliaram somente os resultados propiciados pela geração distribuída, ao passo que outros (CHING-TZONG SU, 2000) quantificaram os benefícios consequentes exclusivamente da reconfiguração de alimentadores.

Quanto às estruturas computacionais utilizadas nos estudos, em (DAS, 1999), (MOHAMMAD H. MORADI, 2014) e (MUSIRIN, 2014) as otimizações foram desenvolvidas utilizando-se algoritmos genéticos, enquanto que nos demais trabalhos acima citados foram utilizados outros e diferentes tipos de algoritmos.

As expressões numéricas que guiam os procedimentos de otimização também são diversificadas: em (MUSIRIN, 2014) o estudo se baseou em perdas ativas, com restrição de geração distribuída e restrição de tensão; em (DAS, 1999) o estudo considerou o custo das perdas ativas combinado ao custo de capacitores; em (Q. A. SALIH, 2015) o estudo se fundamentou exclusivamente em perdas ativas, sem contar com nenhuma restrição; em (ABUL'WAFI, 2013) o estudo foi elaborado em perdas reativas, também sem contar com nenhuma restrição; em (MOHAMMAD H. MORADI, 2014) o estudo foi formulado em perdas ativas, combinada a custo de geração distribuída, utilizando-se fatores de balanceamento definidos pelo próprio

autor; e em (CHING-TZONG SU, 2000) o estudo foi baseado em perdas ativas com restrição de tensão.

A busca por eficiência também ganha destaque no meio consumidor, o que se expressa, dentre outros estudos, pela edição de literatura específica sobre perdas elétricas incidentes em ambientes comerciais e industriais (IEEE , 1995). Nela, dentre outros mecanismos utilizados para redução de perdas elétricas, também é estudado o uso de capacitores *shunt*.

1.1 MOTIVAÇÃO

Durante várias décadas o cálculo dos capacitores necessários para correção de fator de potência de instalações industriais foi desenvolvido visando evitar penalidades financeiras, advindas de questões contratuais, e a quantidade e locais onde deveriam ser inseridos os capacitores era resultante desta visão pontual, que, contudo, subsidiariamente, também promovia redução de perdas elétricas.

A motivação deste trabalho é desenvolver o conceito oposto, focando prioritariamente a redução de perdas elétricas, e sua execução utiliza Algoritmos Genéticos, com sua expressão numérica fundamentada em perda ativa, aliada a restrição de Fator de Potência capacitivo na barra de conexão com a empresa distribuidora de energia elétrica.

1.2 OBJETIVO

1.2.1 Objetivo Geral

Esta dissertação tem como objetivo desenvolver um programa computacional específico para redução de perdas elétricas incidentes em consumidores industriais, sendo a redução imposta pela inserção de capacitores *shunt*. O programa baseia-se na estrutura dos Algoritmos Genéticos e, como saída, dois resultados são gerados: valor comercial da potência reativa dos capacitores necessários para efetuar a minimização das perdas e listagem contendo o número dos barramentos elétricos onde eles devem ser instalados.

1.2.2 Objetivos Específicos

São objetivos específicos do presente trabalho:

- Desenvolver, em ambiente *MATLAB*®, programa voltado para minimização de perdas elétricas em sistemas industriais, tomando, como recurso, a instalação de capacitores “*shunt*”;
- Estruturar o programa como Algoritmo Genético, fazendo a otimização com base em perdas elétricas, contudo sob restrição de Fator de Potência;
- Validar o programa desenvolvido tomando como referência funções cujos pontos de mínimo global já sejam conhecidos;
- Validar programa especialista em cálculo de fluxo de carga/potência, já existente, desenvolvido com instruções *MATLAB*®, tomando como referência o programa *POWERFACTORY*®;
- Utilizar o programa especialista em cálculo de fluxo de carga/potência para gerar os dados (perdas, tensões, ângulos e potências) demandados pelo programa estruturado com Algoritmo Genético;
- Utilizar o programa desenvolvido para determinar os capacitores necessários para efetuar a redução de perdas em sistema elétrico industrial, de 38 barras, em diferentes níveis de carregamento (50, 75, 100, 125, 175 e 200 %), determinando, inclusive, os barramentos onde devem ser instalados cada um dos capacitores calculados.

1.2.3 Organização do trabalho

Este trabalho é composto por 8 capítulos, sendo o conteúdo de cada um deles sintetizado a seguir.

Primeiro capítulo: Contém introdução e detalhamento de objetivos, evidenciando que o propósito do trabalho é o desenvolvimento de programa específico, estruturado como Algoritmo Genético, especialista em minimização de perdas elétricas industriais através do uso de capacitores *shunt*. Este programa, ao longo da dissertação, é referenciado como PAGIND (Programa Algoritmo Genético Industrial).

Segundo capítulo: Contém revisão literária, abrangendo os assuntos base da dissertação: Algoritmos Genéticos, Fluxo de Carga e programa *POWERFACTORY*®.

Maior aprofundamento é feito em relação aos Algoritmos Genéticos, visto que tal assunto é o cerne do programa computacional desenvolvido ao longo da dissertação.

Terceiro capítulo: Neste capítulo é avaliado se no PAGIND deverá ser utilizada representação binária ou real. Para a tomada de decisão são feitas diversas otimizações, utilizando programas simples, editados e estruturados como Algoritmos Genéticos.

Quarto capítulo: Contém definições sobre a estrutura macro do programa objeto da dissertação (PAGIND). As definições são tomadas com base no conteúdo teórico contido nos capítulos anteriores. Conforme será definido neste quarto capítulo, o PAGIND é constituído por 3 blocos: “ROTINAS AG” (responsável por executar a estrutura básica do Algoritmo Genético), “ROTINAS FC” (responsável por executar cálculos relativos ao fluxo de carga) e “ROTINAS DE APOIO” (responsável pelo gerenciamento e integração dos demais blocos).

Quinto capítulo: Nele é tratado o processo de validação do bloco “ROTINAS AG”. A validação é feita com base em expressões numéricas cujos pontos de mínimo global já são conhecidos. Para cada função são realizadas 100 otimizações.

Sexto capítulo: Este capítulo trata do processo de validação do bloco “ROTINAS FC”, que contém programa de fluxo de carga, desenvolvido com instruções em *MATLAB®*. A validação toma como referência o programa *POWERFACTORY®* e, considerando que o bloco “ROTINAS FC” e o programa utilizado para sua validação são ferramentas de comportamento determinístico, o processo é desenvolvido com base em um único evento.

Sétimo capítulo: Trata da aplicação do PAGIND em um sistema elétrico industrial, de 38 barras, sob diferentes níveis de carregamentos. Para cada carregamento são realizadas 100 otimizações, permitindo-se, desta forma, identificar e quantificar a progressão da redução das perdas elétricas determinada pelo programa desenvolvido.

Oitavo capítulo: Contém conclusões, fazendo também confronto entre objetivos definidos e resultados efetivamente alcançados.

Para todo e qualquer capítulo, as atividades dependentes de computador para execução de programas; seja a título de simulações, testes, validações ou otimizações, foram desenvolvidas utilizando-se Notebook Dell, Windows 7 “Ultimate”,

“Service Pack 1”, processador Intel® Core™ i5-2410M CPU 2.3 GHz, RAM 8 GB, 64 bits.

2 REVISÃO DE LITERATURA

Neste capítulo é apresentada revisão bibliográfica dos assuntos relacionados ao tema e objetivos da dissertação. Em primeiro lugar são apresentados conceitos relativos aos Algoritmos Genéticos. Em seguida é feita abordagem sobre os conceitos embutidos no programa *MATLAB*® que, operando como ferramenta auxiliar do Algoritmo Genético, executa cálculos de fluxo de carga. O capítulo se encerra fazendo abordagem sobre as características do programa *POWERFACTORY*®, destacando também algumas interfaces utilizadas por este programa para entrada de dados e/ou obtenção de resultados.

2.1 ALGORITMOS GENÉTICOS

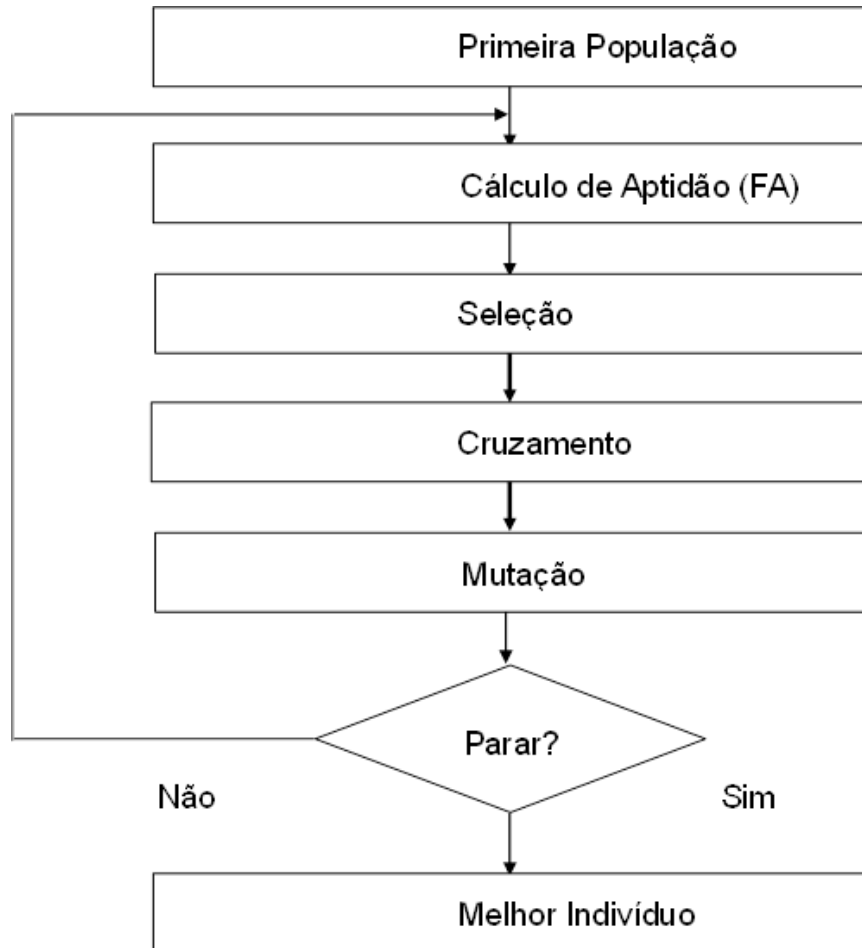
No geral, qualquer tarefa abstrata pode ser entendida como um processo de procura através do espaço de soluções potenciais e, uma vez que se busca a solução ótima, pode-se, então, classificar a tarefa como um processo de otimização. Para pequenos espaços de busca métodos clássicos podem ser suficientes, mas, para grandes espaços, técnicas especiais devem ser utilizadas, dentre elas os Algoritmos Genéticos (MICHALEWICZ, 1996). Além disto, o tipo do problema a ser resolvido também influi pela escolha do Algoritmo Genético.

2.1.1 A teoria de Darwin, pedra fundamental dos Algoritmos Genéticos

Charles Robert Darwin desenvolveu, em 1858, a “Teoria da Evolução”. Conforme tal teoria, todos os organismos vivos estão em lento, mas constante processo de evolução, tendendo a se diferenciar com a passagem das eras. As populações se diferenciam gradualmente, de geração em geração, e cada nova geração é colocada à prova pelas condições do ambiente em que vive. Nas novas populações os indivíduos mais adaptados ao ambiente têm maior probabilidade de sobreviverem e de se reproduzirem. Ao longo das reproduções, ocasionalmente, e de forma aleatória, algumas mutações podem ocorrer, melhorando ou piorando as características dos descendentes. Findo determinado número de eras tem-se, como resultado, o melhor indivíduo.

A estrutura básica dos Algoritmos Genéticos é representada pela Figura 1.

FIGURA 1: ESTRUTURA BÁSICA DOS ALGORITMOS GENÉTICOS



Fonte: Do autor

2.1.2 Funcionamento do Algoritmo Gen  tico B  sico

O Algoritmo Gen  tico    um processo iterativo que mant  m uma popula  o contendo candidatos    solu  o de um problema. O Algoritmo come  a com uma popula  o, criada geralmente aleatoriamente, e durante cada etapa da itera  o, que    referenciada como uma nova gera  o, a estrutura da popula  o corrente    avaliada. Com base nesta avalia  o uma nova popula  o de candidatos    formada e, atrav  s deste procedimento, o Algoritmo usa a popula  o corrente para criar uma nova, que, em m  dia,    melhor que a popula  o que a criou.

Tr  s processos - sele  o, cruzamento e muta  o; que s  o usados para fazer a transi  o de uma gera  o para a pr  xima, s  o repetidos a cada nova gera  o, e o

algoritmo continua desta forma até que alguma condição de parada seja alcançada (máximo número de gerações ou nova população estabilizada em relação às anteriores) (DAS, 1999).

A aptidão de cada indivíduo ao ambiente é representada por um valor numérico, que é o “valor de aptidão”, ou “Fitness”, ou “FA”, do indivíduo em relação ao ambiente.

Pelo descrito, fica evidente que os blocos responsáveis pela seleção, reprodução e mutação guardam extrema similaridade com a teoria evolutiva de Darwin.

A representação dos indivíduos é usualmente feita através de números binários, sendo, contudo, vantajoso, em alguns casos, a representação através de números reais.

2.1.3 Aplicações dos AG's

O Algoritmo Genético (AG) possui enorme flexibilidade, pois não carece de particularização de suas etapas. Portanto, funciona de forma padronizada, guiado somente pela definição do resultado a ser gerado pela solução, e não propriamente por ela. Ele pode ser utilizado para resolver questões tão diferentes como:

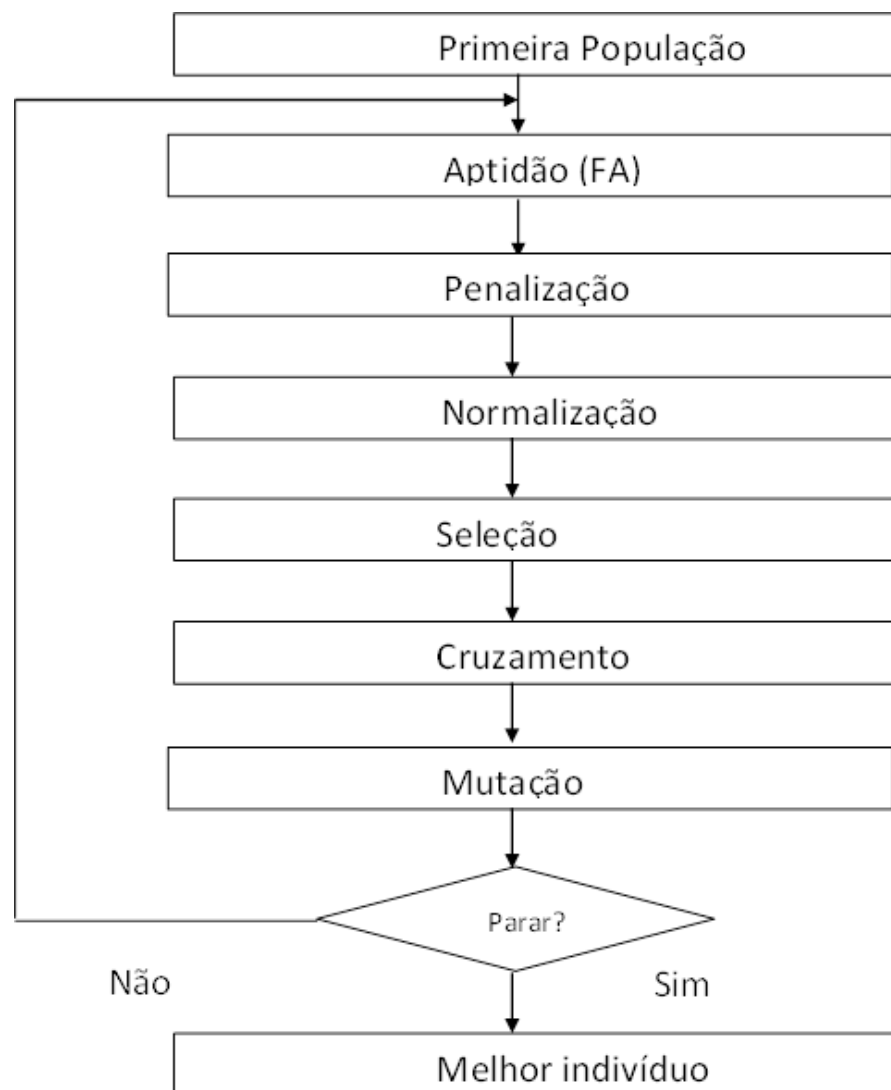
- Previsão de congestionamento de tráfego em uma via expressa (PEDRO LOPEZ GARCIA, 2016);
- Otimização da qualidade dos sinais a serem transmitidos por satélites (MATTEO ANEDDA, 2015);
- Otimização do tempo/estado de sinaleiros para controle de tráfego (KISEOK SUNG, 1997);
- Otimização do deslocamento de um vendedor (TANOMARU, 1995);
- Estimativa da eficiência de motor de indução (KHALIL BANAN, 2007);
- Otimização de expressões numéricas.

Neste trabalho o interesse é por otimização de expressões numéricas, dentre elas as expressões relacionadas com as leis que regem circuitos elétricos, e especificamente a que executa cálculo de perdas elétricas. Por isto, em itens posteriores, são apresentadas várias otimizações deste tipo de expressão.

2.1.4 Algoritmos Genéticos “Dedicados”

Dependendo da aplicação poderá ser necessário, ou conveniente, incrementar o fluxograma básico, adicionando a ele blocos especiais, sendo o mais usual a inserção do bloco “Normalização”, ou do bloco “Penalização”, ou de ambos, conforme mostrado na Figura 2.

FIGURA 2: ESTRUTURA DE ALGORITMO GENÉTICO DEDICADO



Fonte: Do autor

O bloco “Penaliza o” degrada, especificamente, o valor de aptid o do indiv duo que provoca viola o de alguma restri o. A penaliza o favorece os bons

indivíduos, o que, por consequência, imprime maior velocidade à convergência do algoritmo, e funciona conforme Equação 2.1.

$$FA_{penal} = (1 + K1 * \Delta P) * FA \quad (2.1)$$

Onde:

FA_{penal} = Vetor contendo o novo valor de aptidão de todos os indivíduos;

FA = Vetor contendo o valor de aptidão de todos os indivíduos, procedentes do bloco “aptidão”;

ΔP = Vetor contendo a intensidade da violação de todos os indivíduos. Para os não infratores o valor é nulo;

$K1$ = Variável, cujo valor pode ser feito constante e definido arbitrariamente, ou calculado com base em alguma formulação numérica.

O bloco “Normalização” tem recursos para acelerar, ou desacelerar, a convergência do algoritmo, promovendo, ou não, os indivíduos originalmente mais fracos. A normalização funciona conforme Equação 2.2.

$$FA_{norm} = K2 * FA_{penal} + K3 \quad (2.2)$$

Onde:

FA_{norm} = Vetor contendo o novo valor de aptidão de todos os indivíduos;

FA_{penal} = Vetor com o valor de aptidão, procedente do bloco “penalização”;

$K2$ = Variável, cujo valor pode ser feito constante e definido arbitrariamente, ou calculado com base em alguma formulação numérica;

$K3$ = Variável, cujo valor pode ser feito constante e definido arbitrariamente, ou calculado com base em alguma formulação numérica.

2.1.5 Componentes e parâmetros dos Algoritmos Genéticos

2.1.5.1 Indivíduo

Indivíduo é o membro de uma população (THOMAS BACK, 2000) e cada indivíduo é candidato natural à solução do problema. Em otimizações que buscam

por respostas numéricas o “indivíduo” pode ser representado em formato binário, ou em formato real.

2.1.5.2 População

População é um grupo de indivíduos que podem interagir, acasalando e gerando descendentes (THOMAS BACK, 2000). Conceitualmente, algum, dentre os diversos indivíduos da última população gerada pelo AG, representa a solução do problema. Se um, dentre os indivíduos de uma população, for representado na forma binária, todos os demais indivíduos deverão ter esta mesma forma de representação. A mesma regra se aplica caso um dos indivíduos tenha representação implementada através de números reais.

Uma população “real”, composta por 2 indivíduos, é representada pela Tabela 1; enquanto que uma população “binária”, composta por 2 indivíduos, é representada na Tabela 2.

TABELA 1: POPULAÇÃO COM 2 INDIVÍDUOS, REPRESENTAÇÃO REAL

Indivíduo	Genes							
1	32	80	64	48	64	48	64	48
2	96	12	77	93	77	93	77	93

Fonte: Do autor

TABELA 2: POPULAÇÃO COM 2 INDIVÍDUOS, REPRESENTAÇÃO BINÁRIA

Indivíduo	Genes																					
1	1	1	0	0	1	1	1	0	0	1	1	0	0	0	1	0	0	1	1	1	1	0
2	0	0	1	0	1	0	1	1	0	1	0	1	1	0	0	1	0	1	1	0	0	1

Fonte: Do autor

2.1.5.3 Gene

Um indivíduo é constituído por genes (S.N.DEEPA, 2008). Na representação binária, o gene é o próprio bit e, desta forma, os indivíduos representados pela Tabela 2, mostrada anteriormente, possuem 22 genes.

2.1.5.4 Espaço de busca

O espaço de busca faz parte da especificação do problema a ser resolvido e contém todas as possíveis soluções, sendo proporcionalmente incrementado com o aumento da precisão desejada.

Caso uma determinada expressão numérica, definida em função da variável “x”, seja otimizada dentro do intervalo que vai de -3 até 12, e caso esta otimização seja feita com precisão de 4 casas decimais, existirão 150.000 possíveis soluções, sendo este o tamanho do espaço de busca (MICHALEWICZ, 1996).

O número de possíveis soluções (150.000) é calculado com base no intervalo da variável e com base na precisão desejada, ou seja: $150.000 = [(12) - (-3) * 10^4]$. No caso de representação binária, para criar o número de possíveis soluções, são necessários 18 bits; visto que (2^{17}) é menor que 150.000 e que 150.000 é menor que (2^{18}).

2.1.5.5 Tamanho da população

A intuição indica que quanto maior o tamanho da população, mais fácil será a identificação do melhor indivíduo. Em última análise, com uma população infinita, cobrindo assim todo espaço de busca, a solução ótima seria garantidamente obtida na primeira iteração (MICHALEWICZ, 1996). Na prática, devido à necessidade de se limitar a demanda computacional, temos que nos ater a populações finitas, e valores da ordem de 50 a 200 indivíduos resolvem a maior parte dos problemas. Contudo, populações maiores podem ser necessárias para problemas mais complexos (TANOMARU, 1995).

2.1.5.6 Valor de Aptidão (ou fitness, ou FA)

A cada Indivíduo de uma população é atribuído um valor, dito de aptidão, equivalente ao resultado de sua interação com o ambiente. O valor de aptidão é a força motora da seleção Darwiana e, desta forma, dos Algoritmos Genéticos (KOZA, 1990). No caso de otimização de uma expressão numérica, o valor de aptidão é o resultado desta expressão para determinado indivíduo, sendo que a cada indivíduo corresponde um único valor de aptidão. Imaginando que a expressão $f(x) = [x *$

$seno(10 * \pi * x) + 1]$, cuja representação gráfica é mostrada na Figura 3, seja usada para fazer avaliação de indivíduos dentro do intervalo de -1 a 2 com precisão de 6 casas decimais (o que demanda 22 bits); apresenta-se na Tabela 3, coluna 3, o valor resultante do FA para cada indivíduo considerado. O cálculo do efetivo valor de cada indivíduo, já inserido dentro do espaço de busca, e conforme registrado na segunda coluna da Tabela 3, é determinado usando-se a equação 2.3.

$$Ind_{real_ins} = \frac{Faixa_{Inferior} + Ind_{real_nao_inserido} * [Faixa_{Superior} - Faixa_{Inferior}]}{Max_{real}} \quad (2.3)$$

Onde:

Ind_{real_ins} : Valor do indivíduo, originalmente binário, já convertido em número real, e já inserido no espaço de busca;

$Ind_{real_nao_inserido}$: Valor do indivíduo, originalmente binário, já convertido para número real, mas ainda não inserido no espaço de busca;

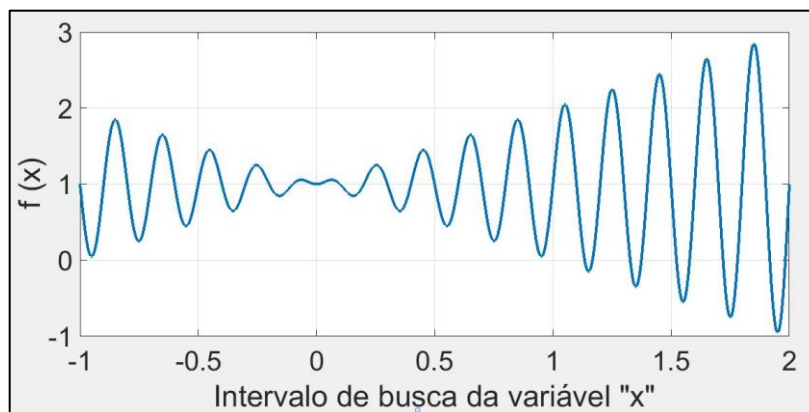
$Faixa_{Superior}$: Valor do limite superior do intervalo de avaliação;

$Faixa_{Inferior}$: Valor do limite inferior do intervalo de avaliação;

Max_{real} : Valor máximo possível, em representação real, para qualquer indivíduo binário representado por 22 bits (4.194.203).

Complementando a explanação sobre a conversão e inserção do indivíduo binário no espaço de busca, o cálculo do FA de cada um deles, conforme registrado na terceira coluna da Tabela 3, é determinado adentrando o " Ind_{real_ins} " na expressão numérica $f(x) = (x * seno(10 * \pi * x) + 1)$.

FIGURA 3: FUNÇÃO $F(x) = (X * SENO(10 * \Pi * X) + 1)$



Fonte: Do autor

TABELA 3: INDIVÍDUOS BINÁRIOS E SEUS VALORES DE APTIDÃO

Indivíduo, em formato binário, 22 bits, fora do intervalo de busca	Indivíduo, em formato real, já no espaço de busca (Ind_{real_ins})	Valor de aptidão (FA)
10 0011 1000 1001 0100 1000	0,665	1,585
10 0001 0110 1101 0100 1010	0,566	0,511
00 1001 0011 0000 0111 0100	-0,569	0,531

Fonte: Do autor

2.1.5.7 Restrição e penalidade

Restrição é um limite estabelecido para alguma variável correlata à otimização do problema. Caso este limite seja ultrapassado deverá ser desencadeada penalização do indivíduo que a provocou. Existe uma variedade de formas para se aplicar a penalidade, e em algumas delas a intensidade da penalidade depende da própria intensidade da violação (MICHALEWICZ, 1996). Genericamente, a penalização deve atuar no valor de aptidão do indivíduo infrator, reduzindo suas chances de sobrevivência.

2.1.5.8 Seleção tipo roleta

Na seleção tipo roleta os indivíduos são selecionados com probabilidade que é diretamente proporcional a seus valores de aptidão, e a cada indivíduo corresponde uma porção no tabuleiro da roleta. A área total do tabuleiro é igual à soma de todos os valores de aptidão, e quando a roleta é girada todos os indivíduos podem ser selecionados, com a maior probabilidade recaindo sobre o indivíduo mais adaptado (NORAINI MOHD RAZALI, 2011).

Especificamente, o processo de seleção tipo roleta se desenvolve da seguinte forma: a roleta deverá ser acionada quantas vezes for o número de indivíduos da população a ser selecionada e, para cada evento, o indivíduo com melhor FA terá, proporcionalmente em relação ao demais, maior chance de ser selecionado. Pelo exposto, fica evidente que este tipo de seleção não funciona corretamente quando os valores de FA são negativos, ou nulos.

No exemplo representado pela Tabela 4, onde busca-se por um máximo, a roleta deverá ser girada 3 vezes, selecionando 3 indivíduos. O mesmo indivíduo poderá ser selecionado repetidas vezes, mas não existe garantia de que, à cada rodada, o indivíduo com maior probabilidade seja efetivamente o selecionado.

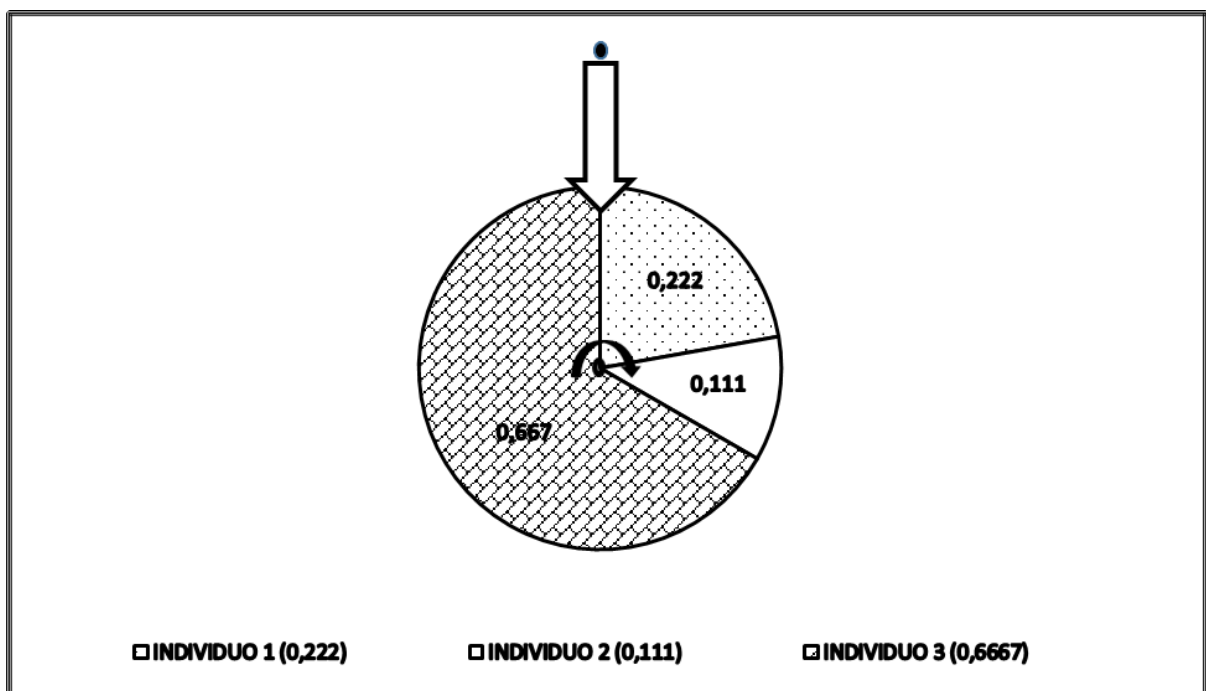
TABELA 4: SELEÇÃO TIPO ROLETA, BUSCA POR UM PONTO DE MÁXIMO

Indivíduo	Valor de aptidão (FA)	Área da roleta
1	1,5858	$(1,5858 / 7,1361) = 0,2222$
2	0,7929	$(0,7929 / 7,1361) = 0,1111$
3	4,7574	$(4,7574 / 7,1361) = 0,6667$
	Soma (7,1361)	

Fonte: Do autor

Os dados contidos na Tabela 4 correspondem à roleta representada pela Figura 4, onde se busca por um ponto de máximo, ficando evidenciado que o indivíduo com maior FA (4,7574) possui a maior porção da área total do tabuleiro da roleta (66,67 %); o que lhe confere alta probabilidade de seleção.

FIGURA 4: SELEÇÃO TIPO ROLETA



Fonte: Do autor

2.1.5.9 Seleção tipo Torneio

O processo de seleção tipo torneio promove várias competições entre os indivíduos da população, e o número de competições é igual ao tamanho da população. Para cada competição, aleatoriamente é definida a quantidade de competidores. Já estando definido a quantidade de competidores, também aleatoriamente são escolhidos os próprios jogadores, sendo selecionado para sobreviver na próxima geração o melhor elemento de cada competição (THOMAS BACK, 2000). Este tipo de seleção é muito abrangente, pois opera, sem restrições, com números negativos, nulos e positivos. Para exemplificar o funcionamento da seleção tipo torneio, vamos lançar mão de uma população existente antes da seleção, conforme mostrada na Tabela 5; sendo os indivíduos selecionados conforme Tabela 6, onde se verifica repetida seleção do indivíduo “E”; que é o mais adaptado para o corrente exemplo, onde a otimização busca pelo valor mínimo.

TABELA 5: POPULAÇÃO A SER SUBMETIDA AO PROCESSO DE SELEÇÃO TIPO TORNEIO

Indivíduo	Valor de aptidão
A	2,3
B	9,0
C	4,3
D	5,0
E	0,5
F	5,0
G	6,0
H	3,0

Fonte: Do autor

TABELA 6: POPULAÇÃO GERADA PELA SELEÇÃO TIPO TORNEIO, BUSCA PELO MÍNIMO

Torneio	Jogadores selecionados	Vencedor
1	A, B,	Indivíduo A
2	A, E, F,	Indivíduo E
3	A, B, B, D, G,	Indivíduo A
4	B, C, D, D, D, G,	Indivíduo C
5	B, C, D, F, F, G, G,	Indivíduo C
6	A, A, B, C, E, F, G, G,	Indivíduo E
7	A, B, C, D, E, F, G, H,	Indivíduo E
8	A, B, C, D, E, F, G, H,	Indivíduo E

Fonte: Do autor

2.1.5.10 Cruzamento tipo “Simples” para indivíduos binários

O cruzamento simples (“*one point crossover*”) opera com base no parâmetro “pc” (probabilidade de cruzamento), que define a quantidade de indivíduos da população que será submetida a cruzamento. Para cada indivíduo da população gera-se, aleatoriamente, um número real (“r”), dentro da faixa [0...1], e se o valor de “r” é menor que “pc”, o correspondente indivíduo está selecionado para participar do cruzamento. De forma também aleatória, todos os indivíduos selecionados formam pares (casais) e, para cada par (casal) é gerado, aleatoriamente, um número inteiro (“pos”), dentro da faixa [1, ..., m-1]; onde “m” é o número de genes de cada indivíduo. O número “pos” é o ponto onde se inicia a transferência dos genes entre os indivíduos que formam o casal (MICHALEWICZ, 1996).

No exemplo mostrado nas Tabelas 7 e 8, dois indivíduos se cruzam, gerando dois filhos; sendo que o ponto de cruzamento foi ajustado para iniciar a transferência dos genes após o terceiro bit (gene).

TABELA 7: PAIS/INDIVÍDUOS SELECIONADOS PARA CRUZAMENTO SIMPLES

País/Indivíduos	Genes							
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	1

Fonte: Do autor

TABELA 8: FILHOS, CRUZAMENTO SIMPLES, TROCA APÓS O TERCEIRO BIT

Filhos/Indivíduos	Genes							
1	0	1	1	1	1	0	0	1
2	1	0	1	0	0	0	0	0

Fonte: Do autor

2.1.5.11 Cruzamento em dois pontos para indivíduos binários

O cruzamento em dois pontos (“*two points crossover*”) para indivíduos binários funciona de forma similar ao método de cruzamento simples, porém possui mais pontos referenciais para início da transferência de genes.

Considerando como exemplo a Tabela 9 (pais) e a Tabela 10 (filhos), temos 2 pontos de cruzamento: o primeiro está ajustado para iniciar após o terceiro bit e o segundo está ajustado para iniciar após o sétimo bit.

TABELA 9: PAIS SELECIONADOS PARA CRUZAMENTO EM DOIS PONTOS

Filhos/Indivíduos	Genes							
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	1

Fonte: Do autor

TABELA 10: FILHOS, RESULTANTES DO CRUZAMENTO EM DOIS PONTOS

Filhos/Indivíduos	Genes							
1	0	1	1	1	1	0	0	0
2	1	0	1	0	0	0	0	1

Fonte: Do autor

2.1.5.12 Cruzamento de indivíduos reais

No caso de representação dos indivíduos através de números reais, dois pais (“x” e “y”) produzem dois filhos (“p” e “q”), sendo os filhos definidos conforme as relações “ $p = [a \cdot x + (1 - a) \cdot y]$ ” e “ $q = [(1 - a) \cdot x + a \cdot y]$ ” (SCHOENAUERY, 1996).

Exemplo deste tipo de cruzamento é mostrado na Tabela 11, onde são tomados 2 indivíduos (os pais); respectivamente com valores iguais a 30 e 80, que, com taxa de cruzamento “a” igual a 60%, geram “Filho-1” igual a 50 e “Filho-2” igual a 60.

TABELA 11: CÁLCULOS ENVOLVIDOS NO CRUZAMENTO DE INDIVÍDUOS REAIS

Filhos	Cálculos	Resultados
1	$(0,6) \cdot 30 + (1 - 0,6) \cdot 80$	50
2	$(0,6) \cdot 80 + (1 - 0,6) \cdot 30$	60

Fonte: Do autor

2.1.5.13 Mutação de indivíduos binários

A mutação deve alterar somente uma pequena quantidade de bits/genes da população pois, caso contrário, as características dos indivíduos poderão ser

significativamente descaracterizadas, perdendo correlação com o problema. Se isto acontece, o algoritmo se torna incapaz de produzir resultado aproveitável. Para evitar este inconveniente, a mutação tipicamente deverá ficar limitada a 5 % do número total de bits/genes constituintes de toda população. O processo de mutação opera com base no parâmetro “pm” (probabilidade de mutação), que define a quantidade de bits/genes da população que será submetida à mutação. Cada bit/gene de toda a população tem a mesma chance de se submeter ao processo, e para cada um deles gera-se, aleatoriamente, um número real (“r”), dentro da faixa [0...1]. Se o valor de “r” é menor que “pm”, o correspondente bit/gene está selecionado para ser alterado e o processo se encerra invertendo-se o estado dele (MICHALEWICZ, 1996).

No exemplo representado pelas Tabelas 12 e 13, com probabilidade de mutação igual a 3,125 %; o sexto bit do segundo indivíduo foi o gene selecionado para sofrer a mutação.

TABELA 12: POPULAÇÃO DE QUATRO INDIVÍDUOS ANTES DA MUTAÇÃO

Indivíduo	Genes							
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	1
3	1	1	1	1	1	0	0	0
4	1	1	0	0	1	0	0	0

Fonte: Do autor

TABELA 13: POPULAÇÃO DE QUATRO INDIVÍDUOS APÓS A MUTAÇÃO

Indivíduo	Genes							
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	1	0	1
3	1	1	1	1	1	0	0	0
4	1	1	0	0	1	0	0	0

Fonte: Do autor

2.1.5.14 Mutação de indivíduos reais

No processo de mutação de indivíduos reais, cada elemento da matriz que representa a população tem igual chance de ser submetido à mutação, e ela se processa alterando o valor do elemento sorteado, atribuindo-lhe, aleatoriamente, um

novo valor, limitado ao espaço de busca (MICHALEWICZ, 1996). Definindo de outra forma, também para o caso de representação através de números reais, a quantidade de elementos da população que sofrerá mutação é definida por parâmetro, e ela não pode ser muito grande, pois, caso contrário, incorre-se no risco de que o algoritmo perca capacidade de produzir correto resultado. Por isto, usualmente a faixa de mutação é limitada em 5 % do número de elementos. Dentro da quantidade definida, os elementos são escolhidos aleatoriamente, e o novo valor também é aleatório; porem limitado ao espaço de busca. Considerando, como exemplo, mutação de 4,17 % imposta à população contida na Tabela 14, as células a serem mutadas são escolhidas aleatoriamente e o novo valor destas células também é aleatório, limitado ao espaço de busca, que neste exemplo será considerado entre 0 e 100. Imaginando que seja escolhida a célula que corresponde à linha 2 e à coluna 4, e que o novo valor seja aleatoriamente definido como 55,1; temos, na Tabela 15, a população resultante do processo de mutação.

TABELA 14: POPULAÇÃO REAL ANTES DA MUTAÇÃO

Indivíduo	Genes					
1	11,6	19,6	41,2	12,7	49,2	17,3
2	39,7	64,0	61,0	29,9	31,0	13,0
3	24,5	55,5	22,2	18,8	25,0	85,0
4	15,4	99,2	59,2	49,0	76,2	19,2

Fonte: Do autor

TABELA 15: POPULAÇÃO REAL APÓS A MUTAÇÃO

Indivíduo	Genes					
1	11,6	19,6	41,2	12,7	49,2	17,3
2	39,7	64,0	61,0	55,1	31,0	13,0
3	24,5	55,5	22,2	18,8	25,0	85,0
4	15,4	99,2	59,2	49,0	76,2	19,2

Fonte: Do autor

2.1.5.15 Parada do Algoritmo

A parada pode ser realizada com base no número de iterações ou com base em alguma performance específica, que pode ser, por exemplo, a repetição da mesma

população durante determinado número de iterações. Considerando que a solução já esteja identificada, o algoritmo pode ser terminado a qualquer momento, e o único e melhor indivíduo pode ser considerado como a saída do algoritmo (KOZA, 1990). Em outras palavras, uma vez que a solução já esteja identificada, ela corresponde a um único indivíduo; que é o mais adaptado ao ambiente.

2.2 FLUXO DE CARGA

O estudo de fluxo de carga é uma análise numérica que determina as variáveis correlatas ao fluxo de potência dentro de um sistema elétrico. No estudo normalmente usa-se notação simplificada, adotando-se sistema p.u. e diagrama unifilar; sendo usual considerar o sistema em regime permanente e com cargas balanceadas. As informações geradas são usadas em planejamento e operação. O programa que executa o estudo de fluxo de carga começa sua análise identificando as variáveis conhecidas e as variáveis desconhecidas; o que, por sua vez, depende do tipo da barra, conforme detalhado na Tabela 16, onde P = Potência Ativa, Q = Potência reativa, V = Magnitude da tensão e θ = Ângulo da tensão.

TABELA 16: RELAÇÃO CONTENDO TIPO DE VARIÁVEIS EM FUNÇÃO DO TIPO DE BARRAS

Tipo	Conhecidas		Não conhecidas	
Carga (PQ)	P	Q	θ	V
Geração (PV)	P	V	θ	Q
Referência (V θ)	V	θ	Q	P

Fonte: Do autor

Então, num sistema com NPQ barras de carga e NPV barras de geração, existem $2 \cdot (NPQ + NPV + 1)$ variáveis desconhecidas e $2 \cdot (NPQ + NPV + 1)$ variáveis conhecidas; significando que o sistema de equações é determinado e admite solução. A partir das variáveis conhecidas, as relações possíveis de serem usadas são as correspondentes aos balanços de potência, e conforme expressas pelas Equações 2.4 e 2.5.

$$0 = -P_i + \sum_{k=1}^n |V_i| |V_k| (G_{ik} \cos \theta_{ik} + B_{ik} \sin \theta_{ik}) \quad (2.4)$$

$$= -Q_i + \sum_{k=1}^n |V_i||V_k|(G_{ik} \sin \theta_{ik} - B_{ik} \cos \theta_{ik}) \quad (2.5)$$

Onde:

P_i = Potência ativa líquida da barra “i”;

Q_i = Potência reativa líquida da barra “i”;

G_{ik} = Parte real da matriz admitância de barras (YBUS);

B_{ik} = Parte imaginária da matriz admitância de barras (YBUS);

θ_{ik} = Diferença entre os ângulos das tensões das barras “ik”.

Para resolver as equações não lineares, apresentadas em (2.4) e (2.5), é utilizado o método iterativo de Newton e, como primeiro passo, são calculadas as variáveis “ θ ” e “ V ” das barras de carga e a variável “ θ ” para as barras de geração. Como segundo passo é calculada a variável “ P ” da barra de referência, sendo, a seguir, calculadas as variáveis “ Q ” das barras de geração e da barra de referência.

2.2.1 Expressões gerais relativas ao fluxo de carga

Equações básicas são obtidas impondo-se o conceito de conservação das potências ativa e reativa em cada nó. Isto equivale a determinar a lei de *Kirchhoff* que trata das correntes que chegam e saem dos nós elétricos. As equações são para linha de transmissão (MONTICELLI, 1983), conforme Figura 5. Nas equações relacionadas à Figura 5, temos:

z_{km} = Impedância elétrica existente entre as barras “k” e “m”;

r_{km} = Componente ativa (parte real) da impedância elétrica existente entre as barras “k” e “m”;

x_{km} = Componente reativa (parte imaginária) da impedância elétrica existente entre as barras “k” e “m”;

E_k = Tensão, barra k;

V_k = Magnitude de tensão nodal, barra k;

θ_k = Ângulo da tensão nodal, barra k;

P_{km} = Fluxo de potência ativa, sentido de “k” para “m”;

Q_{km} = Fluxo de potência reativa, sentido de “k” para “m”;

P_k = Geração líquida (geração menos carga) de potência ativa na barra “k”;

Q_k = Geração líquida (geração menos carga) de potência reativa na barra “k”;

k = Número inteiro positivo, vai de 1 até NB;

m = Número inteiro positivo, vai de 1 até NB;

NB = Número de barras da rede elétrica;

S_{km}^* = Fluxo de Potência complexa conjugado, sentido de “ k ” para “ m ”;

I_{km} = Corrente total fluindo de “ k ” para “ m ”;

I_{mk} = Corrente total fluindo de “ m ” para “ k ”;

g_{km} = Condutância (parte real) da admitância série;

b_{km} = Susceptância (parte imaginária) da admitância série;

b_{km}^{sh} = Susceptância “*shunt*” ligada à barra “ k ” e à barra “ m ”;

Q_k^{sh} = Componente de injeção de potência reativa devido ao elemento “*shunt*” da barra “ k ”, onde: $Q_k^{sh} = b_k^{sh} V_k^2$;

Y_{km} = Matriz admitancia;

G_{km} = Condutância (parte real) da matriz admitância;

B_{km} = Susceptância (parte imaginária) da matriz admitância.

A matriz admitância nodal é dada pela Equação 2.6.

$$Y_{km} = (G_{km} + j * B_{km}) \quad (2.6)$$

O fluxo de potência complexo é dado pela Equação 2.7 e a corrente total, da barra “ k ”, é conforme Equação 2.8.

$$S_{km}^* = (P_{km} - jQ_{km}) = E_{km}^* I_{km} \quad (2.7)$$

$$I_{km} = (y_{km} + jb_{km}^{sh})E_k - (y_{km}E_m) \quad (2.8)$$

Temos então que a potência complexa é conforme Equações 2.9 e 2.10.

$$S_{km}^* = E_k^* [y_{km}(E_k - E_m) + jb_{km}^{sh}E_k] \quad (2.9)$$

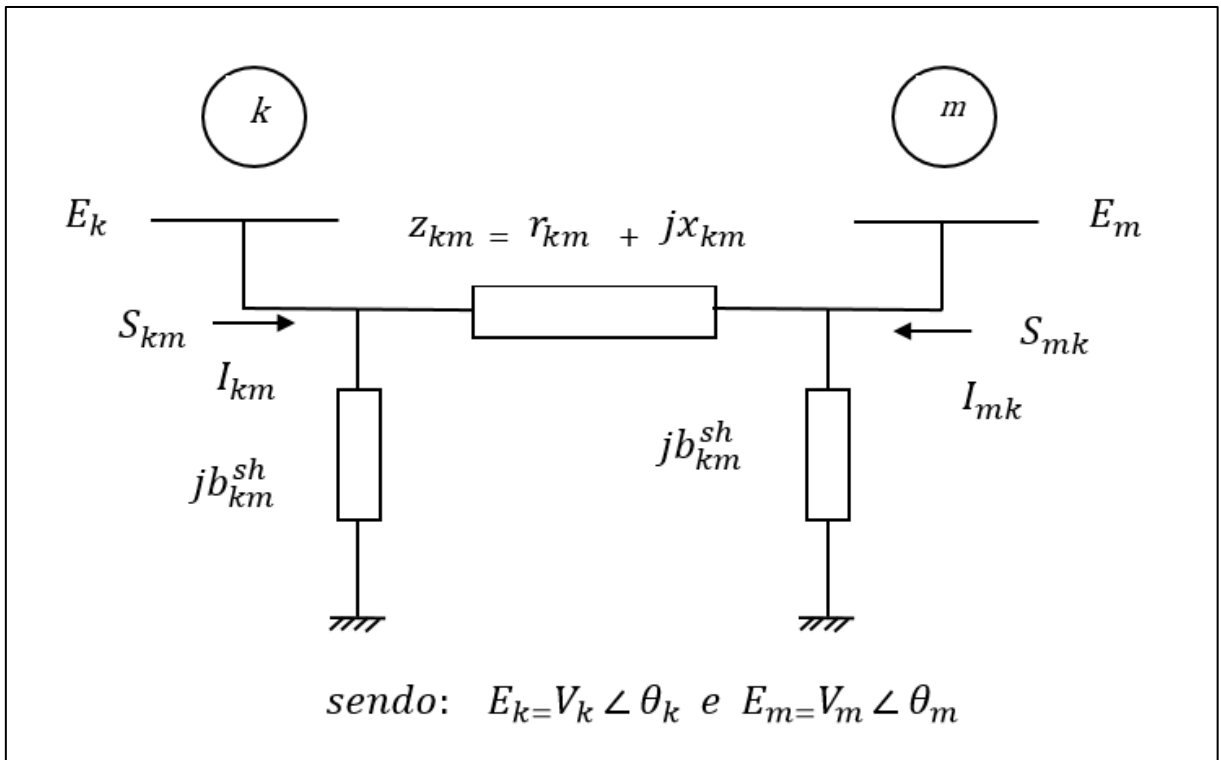
$$S_{km}^* = Y_{km} V_k e^{-j\theta_k} [V_k e^{j\theta_k} - V_m e^{j\theta_m}] + jb_{km}^{sh} V_k^2 \quad (2.10)$$

Nas Equações 2.7 e 2.8, isolando a parte real e imaginária, obtemos o fluxo de potência ativa e reativa do ramo “km”, conforme Equações 2.11 e 2.12.

$$P_{km} = V_k^2 g_{km} - V_k V_m g_{km} \cos \theta_{km} - V_k V_m b_{km} \sin \theta_{km} \quad (2.11)$$

$$Q_{km} = -V_k^2 (b_{km} + b_{km}^{sh}) + V_k V_m b_{km} \cos \theta_{km} - V_k V_m g_{km} \sin \theta_{km} \quad (2.12)$$

FIGURA 5: MODELO PI



Fonte: Do autor

De maneira semelhante, pode-se obter os fluxos de potência ativa e reativa saindo da barra “m” para a barra “k”, conforme Equações 2.13 e 2.14.

$$P_{mk} = V_m^2 g_{km} - V_k V_m g_{km} \cos \theta_{km} + V_k V_m b_{km} \sin \theta_{km} \quad (2.13)$$

$$Q_{mk} = -V_m^2 (b_{km} + b_{km}^{sh}) + V_k V_m b_{km} \cos \theta_{km} + V_k V_m g_{km} \sin \theta_{km} \quad (2.14)$$

O cálculo de perdas ativas é conforme Equações 2.15, 2.16 e 2.17 e o cálculo de perdas reativas é conforme Equações 2.18, 2.19 e 2.20.

$$P_{perdas} = P_{km} + P_{mk} \quad (2.15)$$

$$P_{perdas} = g_{km}(V_k^2 + V_m^2 - 2V_k V_m \cos \theta_{km}) \quad (2.16)$$

$$P_{perdas} = g_{km}|E_k - E_m|^2 \quad (2.17)$$

$$Q_{perdas} = Q_{km} + Q_{mk} \quad (2.18)$$

$$Q_{perdas} = -b_{km}^{sh}(V_k^2 + V_m^2) - b_{km}(V_k^2 + V_m^2 - 2V_k V_m \cos \theta_{km}) \quad (2.19)$$

$$Q_{perdas} = -b_{km}^{sh}(V_k^2 + V_m^2) - b_{km}|E_k - E_m|^2 \quad (2.20)$$

Onde:

" $-b_{km}|E_k - E_m|^2$ " = perdas reativas no elemento série;

" $-b_{km}^{sh}(V_k^2 + V_m^2)$ " = geração de potência reativa nos elementos "shunt".

2.2.2 Cálculo de fluxo de carga através do método *Newton Raphson*

O método *Newton-Raphson* é um método geral para a determinação de raízes reais de equações não-lineares. Na essência, o método trabalha utilizando série de *Taylor*, a partir de uma aproximação inicial. A fórmula básica do problema considera quatro variáveis: V, Θ , P e Q (MONTICELLI, 1983).

O método *Newton-Raphson* tem como objetivo a determinação do fluxo de potência ativa e reativa em cada ramo, e faz seus cálculos de forma iterativa. Os desvios dos valores de referência são calculados à cada iteração, sendo dadas pelas relações de balanço de potência, conforme Equações 2.21 e 2.22.

$$\Delta P_k = P_k^{esp} - P_k = P_k^{esp} - V_k \sum_{j=1}^{n+m+1} V_m (G_{km} \cos \theta_{km} + B_{km} \sin \theta_{km}) \quad (2.21)$$

$$\Delta Q_k = Q_k^{esp} - Q_k = Q_k^{esp} - V_k \sum_{j=1}^{n+m+1} V_m (G_{km} \sin \theta_{km} - B_{km} \cos \theta_{km}) \quad (2.22)$$

O índice "esp" indica valores especificados, ou "injeção menos carga", de potência ativa e reativa; enquanto G e B são a condutância e a susceptância entre as

barras “ k ” e “ m ”; calculadas na matriz de admitância Y . As barras de carga já têm definido seus valores para P_{esp} e Q_{esp} , e assim os valores de ΔP e ΔQ tendem a zero na solução. O mesmo acontece para ΔP das barras de geração. O problema representado nas Equações 2.21 e 2.22 pode ser formulado em termos de um sistema matricial, representado pelas Equações 2.23 e 2.24, em que os elementos H , N , M e L compõem a matriz Jacobiana.

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix} = \begin{bmatrix} H & N \\ M & L \end{bmatrix} x \begin{bmatrix} \Delta \theta \\ \Delta V \end{bmatrix} \quad (2.23)$$

$$J = - \begin{bmatrix} \frac{\partial P}{\partial \theta} & \frac{\partial P}{\partial V} \\ \frac{\partial Q}{\partial \theta} & \frac{\partial Q}{\partial V} \end{bmatrix} \quad (2.24)$$

Assim, os componentes ΔV e $\Delta \theta$ da tensão nodal podem ser escritos em termos dos desvios ΔP , ΔQ e da matriz Jacobiana. Atualizam-se, então, tensões das barras e os valores das potências injetadas. Se o critério de convergência; que é dado pela Equação 2.25, for satisfeito, o processo iterativo é então interrompido. A tolerância (ε) é um número arbitrariamente pequeno.

$$\max(\Delta P, \Delta Q) < \varepsilon \quad (2.25)$$

Os resultados obtidos após a convergência são as tensões (magnitude e ângulo) inicialmente desconhecidos, permitindo, agora, calcular as perdas de potência ativa e reativa do sistema, conforme Equações 2.26 e 2.27; em que P_{km} e Q_{km} são, respectivamente, os fluxos ativo e reativo no ramo de “ k ” para “ m ”:

$$P_{perdas}^* = P_{km} + P_{mk} = G_{km}(V_k^2 + V_m^2 - 2V_k V_m \cos \theta_{km}) \quad (2.26)$$

$$Q_{perdas}^* = Q_{km} + Q_{mk} = -B_{km}(V_k^2 + V_m^2 - 2V_k V_m \cos \theta_{km}) \quad (2.27)$$

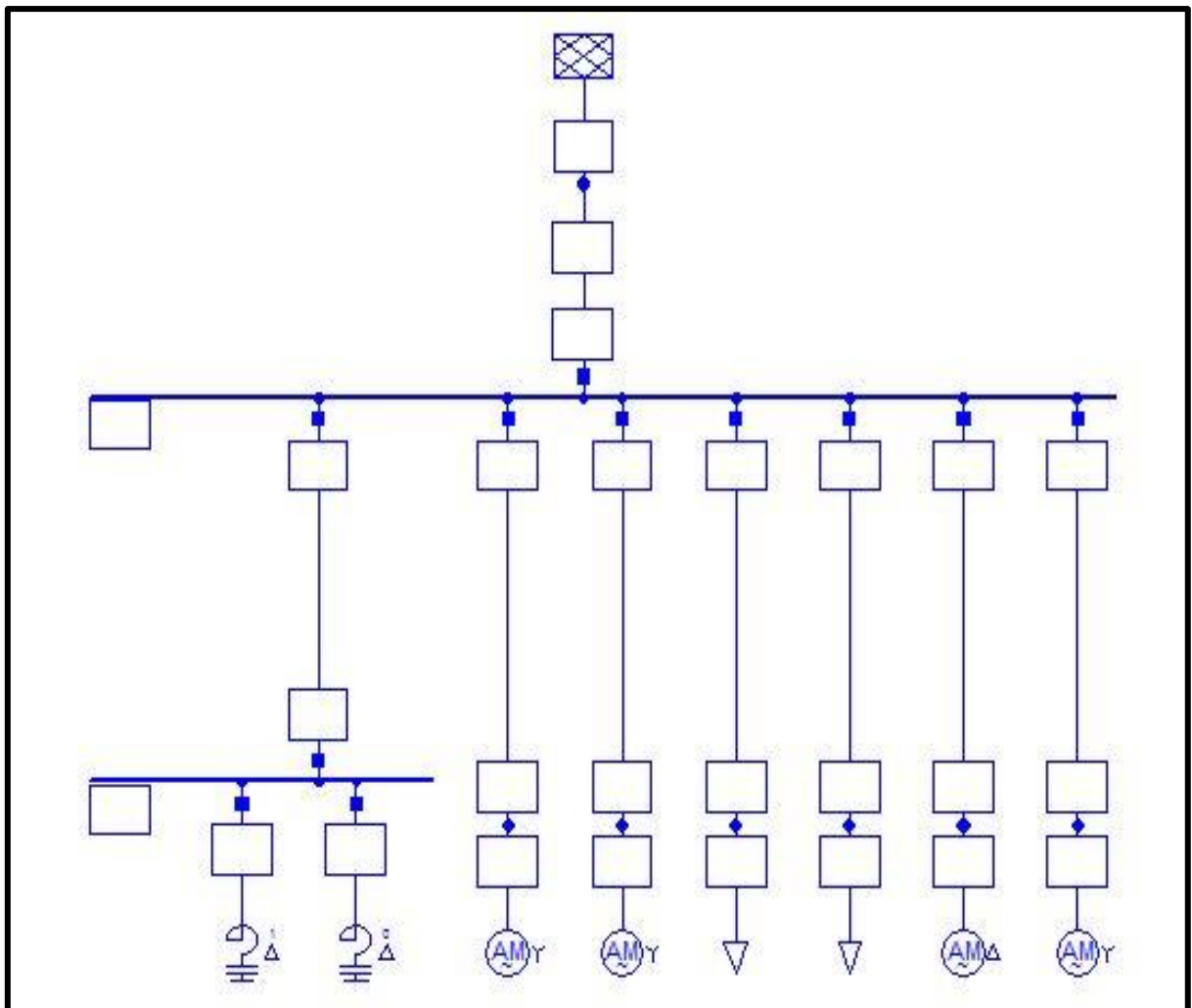
2.3 O PROGRAMA *POWERFACTORY*®

O programa *POWERFACTORY*® é uma ferramenta de engenharia desenvolvida para representação, análise, cálculos, simulações e otimização de variáveis de sistemas elétricos. Dentre as opções disponibilizadas pelo programa, destaca-se o cálculo de fluxo de potência (tensões, potências ativas, potências reativas, ângulos e perdas), mas o programa também se presta para simulações; como análise de curto-circuito e correção de fator de potência. O programa foi desenvolvido na Alemanha a partir da década de 80 e goza de grande reputação internacional.

2.3.1 Desenhando o diagrama unifilar no programa *POWERFACTORY*

O programa *POWERFACTORY*® dispõe de interface gráfica que permite ao usuário efetivamente desenhar o sistema elétrico. Para tal o programa dispõe de vasta biblioteca contendo a representação dos principais componentes elétricos: transformadores de força, transformadores de corrente, linhas aéreas, linhas subterrâneas, capacitores, motores, indutores, resistores, chaves, interruptores, disjuntores, baterias, outros. A biblioteca também contém os principais componentes eletrônicos de potência: retificadores controlados, retificadores não controlados e inversores de frequência. Cada componente gráfico é acompanhado por uma tela com vários campos, onde então se complementa sua especificação. Os dados inseridos através destas telas são, dentre outras: potência nominal, corrente nominal, tensão nominal, relação de transformação. A partir das figuras, das interligações entre elas, e de seus dados, o programa passa a conhecer todas as particularidades do sistema elétrico a ser analisado.

O unifilar de um sistema elétrico, correspondente a um consumidor industrial típico, já inserido no ambiente do programa *POWERFACTORY*®, é exemplificado através da Figura 6. Este tipo de consumidor é caracterizado por não dispor de geração interna, sendo alimentado pela empresa de distribuição elétrica local. Além disto, tem, internamente, pelo menos dois níveis de tensão, com forte predominância de conexão das cargas no nível de menor tensão; sendo elas, quase que integralmente, motores elétricos de indução.

FIGURA 6: INTERFACE GRÁFICA DO PROGRAMA *POWERFACTORY*®

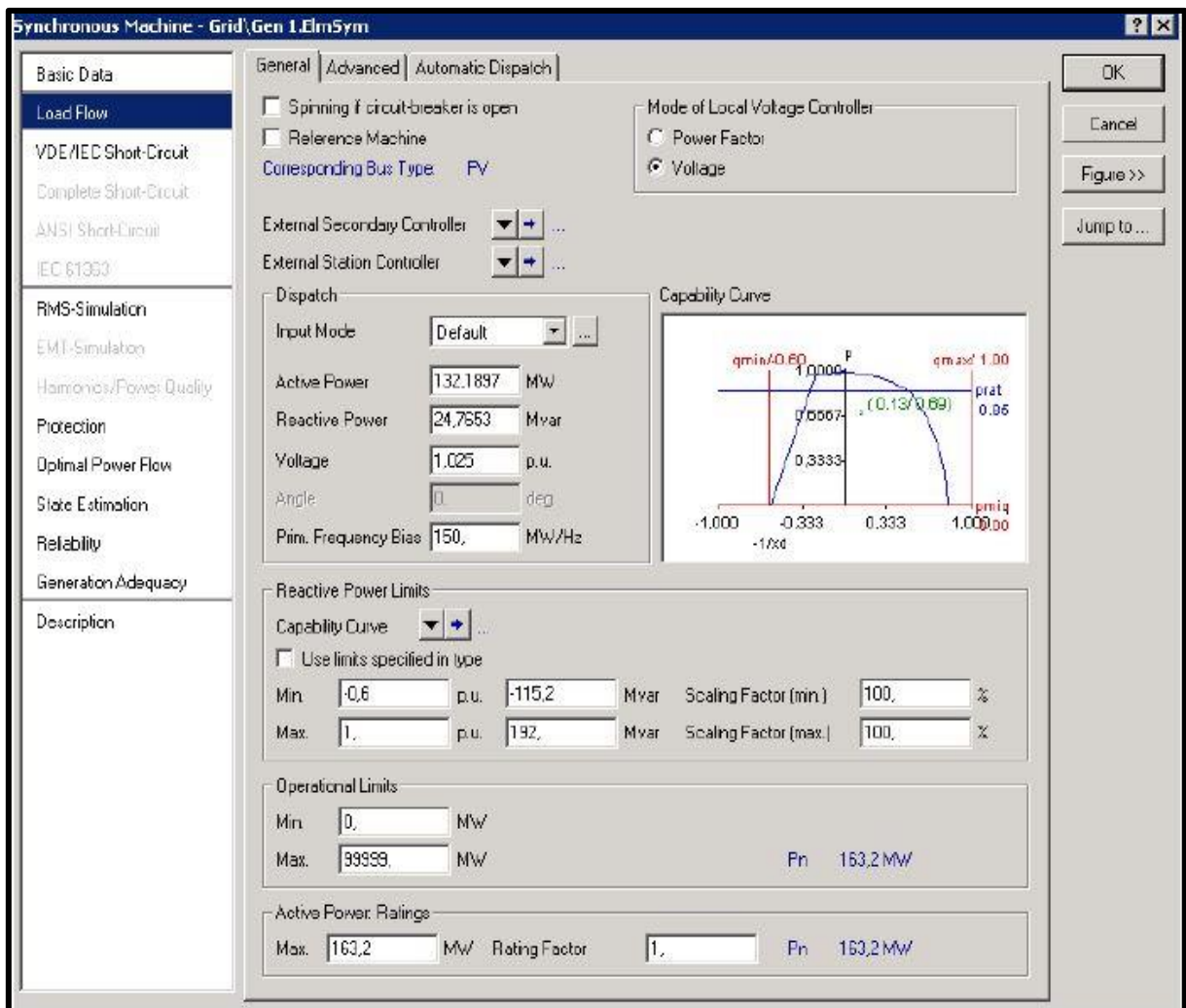
Fonte: Extraída de relatório gerado pelo programa *POWERFACTORY*®

2.3.2 Introduzindo dados no programa *POWERFACTORY*

O programa *POWERFACTORY*® dispõe de diversas interfaces para digitação e entrada de dados. Tais dados permitem que o usuário complete as informações necessárias, em função da aplicação que deseja fazer. As demandas de dados são, por exemplo: impedância de linhas, tipo de linhas, comprimento de linhas, tipo de transformadores, forma de interligação do neutro dos transformadores, potência reativa de capacitores, tipo de filtros, tipo de cargas, potência das cargas, tipo de motores, potência de motores, dentre outras.

Uma, dentre as centenas de interfaces existentes no programa *POWERFACTORY*®, é exemplificada através da Figura 7, em que se introduz os dados de um gerador síncrono (potência ativa, potência reativa e curva de capacidade, dentre outros parâmetros).

FIGURA 7: INTERFACE DO PROGRAMA *POWERFACTORY*® RELATIVA À ENTRADA DE DADOS



Fonte: Extraída de relatório gerado pelo programa *POWERFACTORY*®

2.3.3 Obtendo resultados gerados pelo programa *POWERFACTORY*

O programa *POWERFACTORY*® dispõe de várias formas para saída de resultados, sendo uma, dentre elas, em formato de “relatório”, conforme o mostrado na Figura 8, onde se obtém diversas informações operacionais (custos, perdas ativas,

perdas reativas, intensidade de reativo fornecido por capacitores instalados localmente, dentre outras).

FIGURA 8: EXEMPLO DE RELATÓRIO DO PROGRAMA *POWERFACTORY*®

		DIGSILENT	Project:
		PowerFactory	
		15.1.7	Date: 21/01/2017
Network Representation	Balanced, positive sequence	Constraints	
Time period under consideration	1 year	Upper Voltage Limit	1,10 p.u.
		Lower Voltage Limit	0,90 p.u.
Energy Costs		Limit Reactive Power of all Capacitors	No
Take from External Grid	No	Maximum	100,00 Mvar
Energy Cost	0,05 \$/kWh		
Costs		Reactive power of installed capacitors	
Before Optimization		Before Optimization	0,00 Mvar
Power Losses	53057,60 \$	New Capacitors	1,22 Mvar
Voltage Violations	0,00 \$	After Optimization	1,22 Mvar
Total	53057,60 \$		
After Optimization			
Power Losses	38924,03 \$		
Voltage Violations	0,00 \$		
Costs of new capacitors	0,00 \$		
Total	38924,03 \$		
Saved Costs			
Power Losses	14133,57 \$		
Voltage Violations	0,00 \$		
Total	14133,57 \$		

Fonte: Extraída de relatório gerado pelo programa *POWERFACTORY*®

3 REPRESENTAÇÃO BINÁRIA OU REAL?

3 REPRESENTAÇÃO BINÁRIA OU REAL?

Conforme mencionado e exemplificado no capítulo 2, os indivíduos de Algoritmos Genéticos, quando voltados para otimização de funções numéricas, podem ser representados por números binários ou por números reais.

Uma vez que faz parte deste trabalho produzir programa específico, destinado a estudos de sistemas elétricos industriais, torna-se necessário, em primeiro passo, decidir o formato da representação de seus indivíduos. Para tal foram desenvolvidos quatro diferentes programas: (1) representação binária com busca pelo máximo, (2) representação binária com busca pelo mínimo, (3) representação real com busca pelo máximo e (4) representação real com busca pelo mínimo. Na elaboração destes programas foram utilizadas instruções básicas do *MATLAB*®; significando, então, que não se lançou mão da instrução GA, já disponível na biblioteca de otimização do mencionado ambiente de programação.

Para que os resultados pudessem ser comparados em uma única e idêntica base, as estruturas dos programas foram elaboradas de forma bem simplificada, evitando-se inserir parâmetros e estruturas que tornassem os programas funcionalmente diferentes.

De posse dos quatro programas foram realizadas várias otimizações e, com base nos resultados, foi promovida comparação entre as duas formas de representação. Os critérios usados na comparação foram precisão e tempo de execução. Como ambiente para todas as otimizações foi utilizada a função $f(x) = (x * \text{seno}(10 * \pi * x) + 1)$, intervalo de -1 a 2, cuja representação gráfica está mostrada na Figura 9. O valor da variável “x” que provoca o máximo da função é 1,850450; correspondendo a $f(x) = 2,850274$. O valor da variável “x” que provoca o mínimo da função é 1,954774; correspondendo a $f(x) = -0,950260$.

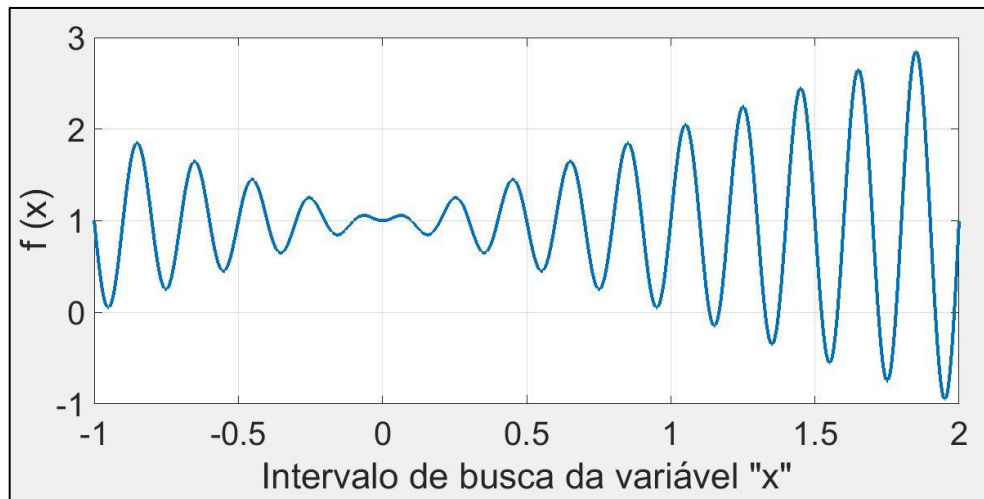
Nos programas com números binários foram utilizados os seguintes parâmetros:

- Faixa de busca superior = 2;
- Faixa de busca inferior = -1;
- Número de indivíduos = 220;
- Número de iterações = 25;
- Número de bits = 22.

Nos programas com números reais foram utilizados os seguintes parâmetros:

- Faixa de busca superior = 2;
- Faixa de busca inferior = -1;
- Número de indivíduos = 220;
- Número de iterações = 25.

FIGURA 9: $F(X) = (X * \text{SENO}(10 * \pi * X) + 1)$ DENTRO DO ESPAÇO DE BUSCA -1 A 2



Fonte: Do autor, gerada por programa *MATLAB*®

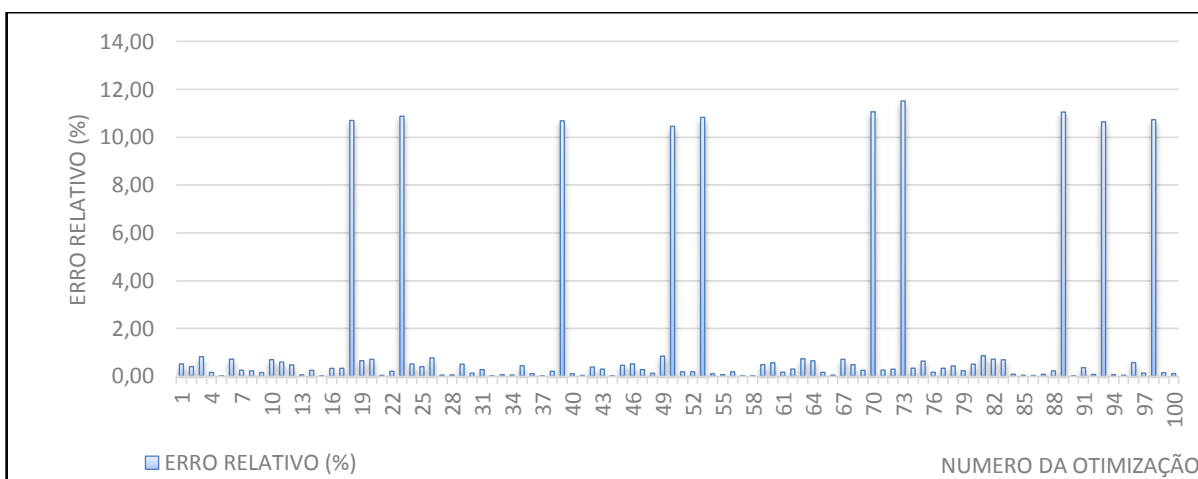
3.1 OTIMIZAÇÕES REALIZADAS PARA AVALIAR AS FORMAS DE REPRESENTAÇÃO DOS INDIVÍDUOS

Em busca da melhor forma de representação, foram realizadas 200 otimizações com números binários; 100 em busca do valor de “x” que correspondesse ao ponto máximo e 100 em busca do valor de “x” que correspondesse ao ponto de mínimo da função. De forma similar, também foram realizadas 200 otimizações com números reais; 100 em busca do valor de “x” que corresponde ao ponto máximo e 100 em busca do valor de “x” que corresponde ao ponto de mínimo da função. Os cálculos do programa *MATLAB*® foram realizados com 15 casas decimais, superando assim a precisão arbitrariamente definida como necessária para avaliação das respostas (6 casas decimais).

3.1.1 Resultados das otimizações em busca do máximo, representação binária

Na Figura 10 estão mostrados os erros relativos incorridos ao longo de todas as 100 otimizações realizadas utilizando representação binária, busca pelo máximo.

FIGURA 10: ERROS RELATIVOS INCORRIDOS NAS OTIMIZAÇÕES REALIZADAS COM REPRESENTAÇÃO BINÁRIA E BUSCA PELO MÁXIMO

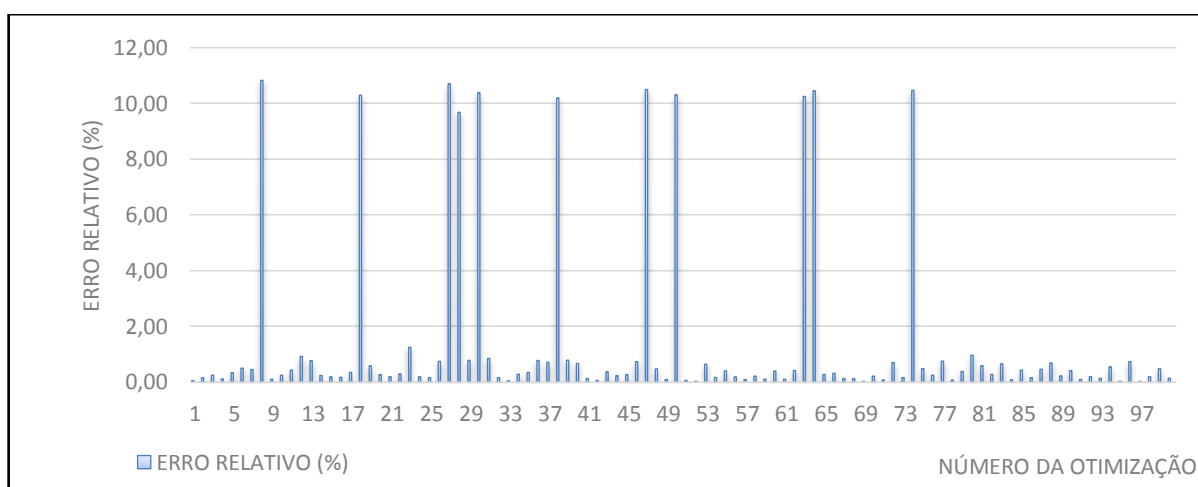


Fonte: Do autor

3.1.2 Resultados das otimizações em busca do mínimo, representação binária

Na Figura 11 estão mostrados os erros relativos incorridos ao longo de todas as 100 otimizações realizadas utilizando representação binária, busca pelo mínimo.

FIGURA 11: ERROS RELATIVOS INCORRIDOS NAS OTIMIZAÇÕES REALIZADAS COM REPRESENTAÇÃO BINÁRIA E BUSCA PELO MÍNIMO

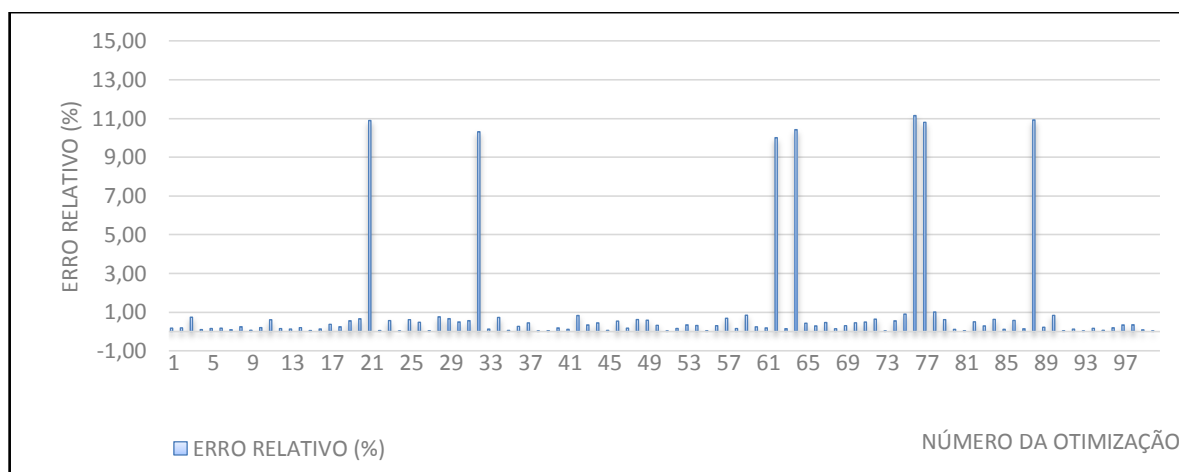


Fonte: Do autor

3.1.3 Resultados das otimizações em busca do máximo, representação real

Na Figura 12 estão mostrados os erros relativos incorridos ao longo de todas as 100 otimizações realizadas utilizando representação real, busca pelo máximo.

FIGURA 12: ERROS RELATIVOS INCORRIDOS NAS OTIMIZAÇÕES REALIZADAS COM REPRESENTAÇÃO REAL E BUSCA PELO MÁXIMO

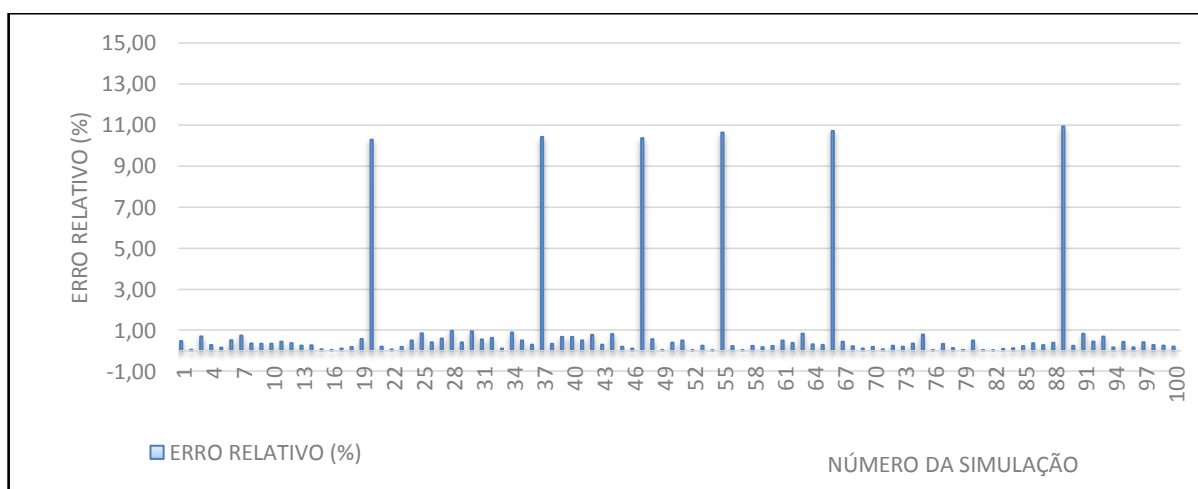


Fonte: Do autor

3.1.4 Resultados das otimizações em busca do mínimo, representação real

Na Figura 13 estão mostrados os erros relativos incorridos ao longo de todas as 100 otimizações realizadas utilizando representação real, busca pelo mínimo.

FIGURA 13: ERROS RELATIVOS INCORRIDOS NAS OTIMIZAÇÕES REALIZADAS COM REPRESENTAÇÃO REAL E BUSCA PELO MÍNIMO



Fonte: Do autor

3.2 RESUMO DAS 400 OTIMIZAÇÕES REALIZADAS COM POPULAÇÃO DE 220 INDIVÍDUOS

A Tabela 17 contém o resumo de todos as 400 otimizações realizadas com base na função $f(x) = (x * \text{seno}(10 * \pi * x) + 1)$, dentro do espaço de busca -1 a 2, precisão de 6 casas decimais. Nela, na primeira coluna, está informado o tipo de representação dos indivíduos, enquanto que na segunda coluna está registrado o propósito da otimização: maximizar ou minimizar. Na terceira e quarta coluna estão registrados a quantidade de eventos, dentre uma centena, cujos resultados produziram erro relativo, em percentual, maior que, respectivamente, 1,0 % e 0,1 %. Na coluna 5 está informado o maior erro relativo, em percentual, que foi gerado ao longo de cada centena de otimizações; enquanto que na coluna 6 está a demanda computacional, em segundos, para a execução da batelada com 100 eventos.

TABELA 17: SÍNTESE DAS OTIMIZAÇÕES REALIZADAS COM POPULAÇÕES DE 220 INDIVÍDUOS, $F(X) = [X * \text{SENO}(10 * \pi * X) + 1]$, ESPAÇO DE BUSCA -1 A 2

Tipo	Propósito	Eventos com erro maior que 1,0 %	Eventos com erro maior que 0,1 %	Maior Erro (%)	Demanda, para 100 otimizações (s)
Binário	Maximizar	10	76	11,52	902
	Mínimizar	12	83	20,72	895
Real	Maximizar	8	80	11,15	913
	Mínimizar	6	86	10,94	923

Fonte: Do autor

3.3 CONCLUSÕES SOBRE AS OTIMIZAÇÕES COM 220 INDIVÍDUOS

3.3.1 Tempo de execução

Conforme resultados contidos na Tabela 17, quando contando com 220 indivíduos a demanda computacional das duas formas de representação é similar, sem ocorrer significativa distinção dentre elas.

3.3.2 Eliminando erros de otimizações realizadas com Algoritmos Genéticos

Conforme foi constatado nas otimizações realizadas com populações de 220 indivíduos, várias, dentre elas, produziram erros expressivos, e isto é consequência de características inerentes aos AG's.

A teoria de Algoritmos Genéticos fornece algumas explicações sobre “como”, para uma dada formulação de problema, seu ponto ótimo é localizado; mas, infelizmente, aplicações práticas nem sempre seguem a teoria, sendo as principais razões para isto (MICHALEWICZ, 1996):

- A codificação do problema geralmente move o AG para operar em um espaço diferente do real espaço do problema;
- Existe limite para o hipotético ilimitado número de iterações;
- Existe limite para o hipotético ilimitado tamanho das populações.

Pelo exposto, fica em evidência que quanto maior é o tamanho das populações, menor é a probabilidade de se produzir resultados inconsistentes. Baseando-se neste conceito, todas as otimizações realizadas na secção 3.1 foram repetidas, mudando-se somente o tamanho das populações; que foi triplicado, passando-se a tomar 660 indivíduos; ao invés de 220. Os novos resultados estão contidos na Tabela 18, cuja organização é idêntica à organização da Tabela 17, e mostram que o significativo aumento do número de indivíduos propiciou a condição para que todos os programas, tanto com números binários quanto com números reais, produzissem resultados excelentes no que tange à precisão, ocorrendo, contudo, geométrico crescimento no tempo de demanda computacional.

TABELA 18: SÍNTESE DAS OTIMIZAÇÕES REALIZADAS COM POPULAÇÕES DE 660 INDIVÍDUOS, $F(X) = [X * \text{SENO}(10 * \pi * X) + 1]$, ESPAÇO DE BUSCA -1 A 2

Tipo	Propósito	Eventos com erro maior que 1,0 %	Eventos com erro maior que 0,1 %	Maior Erro (%)	Demanda, para 100 otimizações (s)
Binário	Maximizar	0	52	0,62	8533
	Mínimizar	0	67	0,68	8409
Real	Maximizar	0	49	0,67	8223
	Mínimizar	0	87	0,62	8290

Fonte: Do autor

3.4 TOMADA DE DECISÃO SOBRE A FORMA DE REPRESENTAÇÃO A SER UTILIZADA NO PROGRAMA PAGIND

Considerando exclusivamente as otimizações e considerações já apresentadas ao longo do capítulo 3, e em especial os resultados contidos nas Tabelas 17 e 18, não houve, no que tange a precisão e tempo de demanda computacional, significativa diferença dentre as duas formas de representação. Portanto, por uma simples questão de preferência acadêmica, o programa objeto da dissertação será desenvolvido fazendo uso de indivíduos binários.

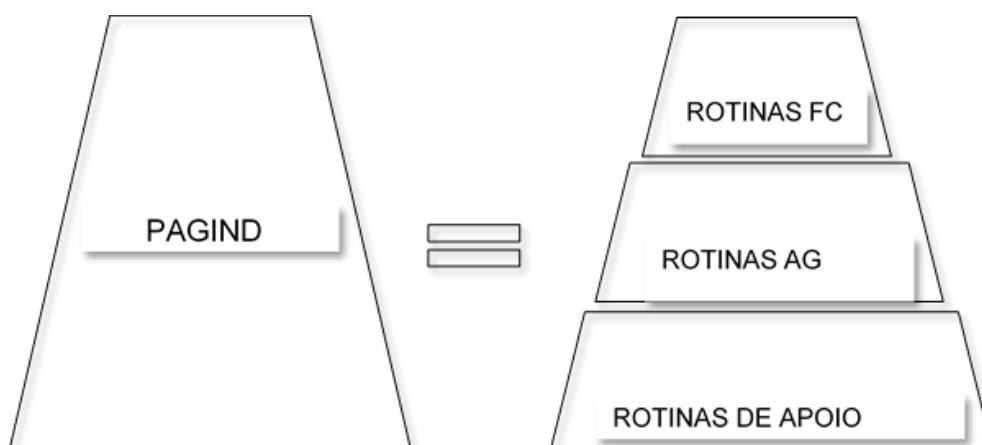
4 DESENVOLVIMENTO DO PROGRAMA OBJETO DA DISSERTAÇÃO

4 DESENVOLVIMENTO DO PROGRAMA OBJETO DA DISSERTAÇÃO

O programa desenvolvido, e objeto da dissertação (PAGIND, Programa Algoritmo Genético Industrial), é constituído, conforme esquematizado na Figura 14, por três diferentes blocos: “ROTINAS DE APOIO”, “ROTINAS AG” e “ROTINAS FC”.

O bloco “ROTINAS AG” contém a estrutura do “Algoritmo Genético Dedicado”, conforme definida no item 2.1.4 e conteúdo da Figura 2; enquanto que o bloco “ROTINAS FC” corresponde ao programa especialista em cálculo de fluxo de carga. O último bloco, referenciado como “ROTINAS DE APOIO”, realiza funções auxiliares (entrada de parâmetros, controle dos laços iterativos, organização de resultados e integração das demais rotinas).

FIGURA 14: ESTRUTURA MACRO DO PROGRAMA OBJETO DA DISSERTAÇÃO (PAGIND)



Fonte: Do autor

Em termos do ambiente *MATLAB*®, o programa PAGIND, como um todo, é constituído por um único “*script*”, que comanda a execução das demais partes; todas elas desenvolvidas como funções *MATLAB*®. Os nomes, tanto para o “*script*” como para as funções, seguem ordenação alfabética, o que auxilia bastante na compreensão do funcionamento do programa.

Todo conteúdo do PAGIND está relacionado na Tabela 19, onde na coluna 1 estão listadas as funções, ou “*script*”; e na coluna 2 está informado a rotina de sua efetiva aplicação.

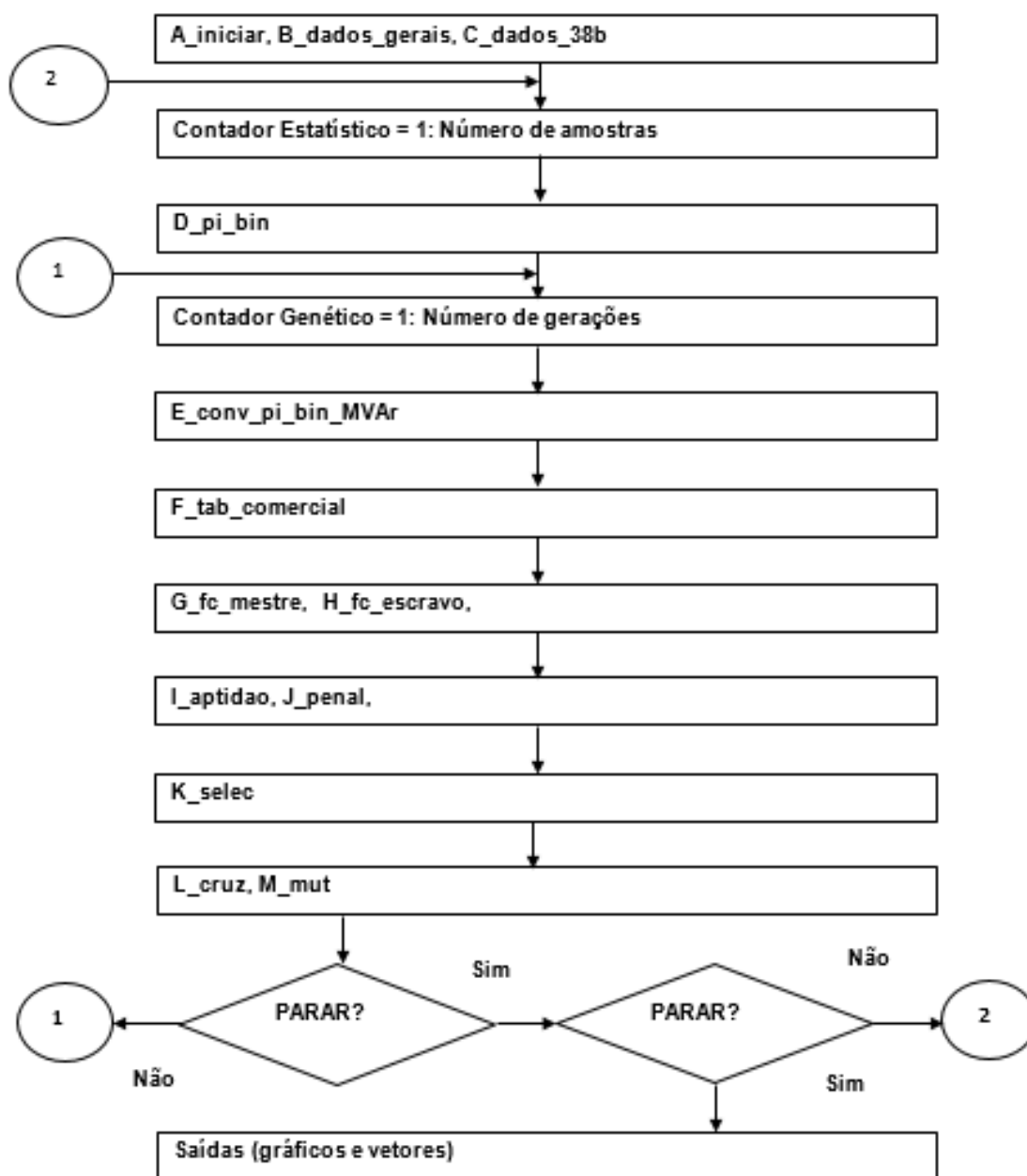
TABELA 19: RELAÇÃO DE FUNÇÕES E “SCRIPT” EM RELAÇÃO À APLICAÇÃO DENTRO DA ESTRUTURA DO PAGIND

“SCRIPT”/FUNÇÃO	APLICAÇÃO
A_iniciar	“ROTINAS DE APOIO”
B_dados_gerais	“ROTINAS DE APOIO”
C_dados_38b	“ROTINAS DE APOIO”
D_pi_bin	“ROTINAS AG”
E_conv_pi_bin_MVAr	“ROTINAS DE APOIO”
F_tab_comercial	“ROTINAS DE APOIO”
G_fc_mestre	“ROTINAS FC”
H_fc_escravo	“ROTINAS FC”
I_aptidao	“ROTINAS AG”
J_penal	“ROTINAS AG”
K_selec	“ROTINAS AG”
L_cruz	“ROTINAS AG”
M_mut	“ROTINAS AG”

Fonte: Do autor

4.1 O FLUXOGRAMA DO PAGIND

O fluxograma do programa PAGIND é conforme mostrado na Figura 15, e para a descrição do funcionamento de cada uma de suas partes está sendo considerado, como referência, um sistema constituído por 38 barras elétricas, população com 90 indivíduos e 5 genes (bits) por barra elétrica; o que redundaria em cada indivíduo representado por vetor nas dimensões 1x190 e população representada por matriz nas dimensões 90x190. Também é considerado que a potência complexa da maior carga, em regime permanente, seja $Scarga = (0,29420 \text{ MW} + j * 0,15879 \text{ MVAr})$, o que é equivalente a motores assíncronos de 4 polos/400 CV.

FIGURA 15: ESTRUTURA DETALHADA DO PROGRAMA OBJETO DA DISSERTAÇÃO (PAGIND) ¹

Fonte: Do autor

O funcionamento do *script* e das diversas funções constituintes do fluxograma mostrado na Figura 15 é detalhado nos itens a seguir.

¹ Apesar de não estar evidenciado no fluxograma (Figura 15), o programa que efetivamente desenvolve os cálculos de fluxo de carga está inserido dentro da função H_fc_escravo.

4.1.1 “Script” “A_iniciar”

Este “script” *MATLAB*® executa, sequencialmente, a chamada de todas as funções e laços iterativos que constituem o programa de otimização, e é nele que estão inseridos todos os parâmetros acessíveis aos usuários. Ele comanda a execução de dois laços: no primeiro, mais interno, se encontra o Algoritmo Genético; enquanto que no segundo, mais externo, dentre outros, está o controle estatístico e a emissão de resultados (gráficos e o melhor arranjo de capacitores).

4.1.2 Função “B_dados_gerais”

Esta função *MATLAB*® é usada para acondicionar dados gerais (custo de energia, custo de demanda, custo para instalação de capacitores, custo para aquisição de capacitores, potência base, tensão base, entre outros) e disponibiliza, em sua saída, vetor contendo todos os dados inseridos na função. A dimensão do vetor de saída é fixa e independe das demais características da otimização.

4.1.3 Função “C_dados_38b”

Trata-se de função *MATLAB*® responsável por conter a configuração elétrica do sistema que será submetida ao processo de otimização (de, para, resistência do ramo elétrico, reatância do ramo elétrico, potência ativa e reativa das cargas, valores iniciais e definição das barras onde não deverão ser instalados capacitores). Dentro desta função é gerada uma matriz que contém 31 valores de capacitores, em MVar, associados a 31 números, em código binário, com 5 dígitos. Os valores das potências são proporcionalmente relacionados com o carregamento do sistema elétrico sob otimização, e são automaticamente reajustados caso ocorra mudança no nível de carga.

A mencionada tabela é disponibilizada na saída da função em questão através da matriz “Faixa_busca”, dimensões 31x7.

4.1.4 Função “D_pi_bin”

Trata-se de função *MATLAB*® responsável por gerar, de forma aleatória, a população inicial; que é codificada através de 5 dígitos. Como o exemplo considera 38 barras elétricas; cada indivíduo demanda, em sua representação, 190 bits (38 indivíduos com 5 dígitos); e como são 90 indivíduos, a função disponibiliza em sua saída matriz “Pi_bin”, dimensões 90x190.

4.1.5 Função “E_conv_pi_bin_MVAr”

Trata-se de função *MATLAB*® responsável por converter os códigos binários da população contida na matriz “Pi_bin” (90x190) para seus equivalentes, em números reais. A conversão é feita com base nos valores contidos na tabela “Faixa_busca”. Como o exemplo considera 90 indivíduos, e 38 barras, a função disponibiliza em sua saída a matriz “Pi_MVAr”, nas dimensões 90x38, e maiores detalhes relativos a esta conversão são apresentados no item 4.2.

4.1.6 Função “F_tab_comercial”

Trata-se de função *MATLAB*® responsável por substituir cada célula da matriz “Pi_MVAr” pelo valor de capacitor comercial que lhe seja mais próximo. A função contém uma lista com capacitores comerciais, cujos valores estão na faixa de 5 a 100 kVAr e, a partir desta lista, é criado, dentro da função, um vetor nominado “Tab_multipla”, com 198 linhas e uma única coluna, contendo diferentes valores comerciais; que vão de 5 kVAr até 900 kVAr. Como o exemplo considera 90 indivíduos, e 38 barras, a função disponibiliza em sua saída a matriz “Pi_MVAr_comercial”, nas dimensões 90x38.

4.1.7 Função “G_fc_mestre”

Trata-se de função *MATLAB*® responsável por receber a população de indivíduos (arranjo com diversos capacitores) e passá-los, um a um, para a função que efetivamente executa o cálculo de fluxo de carga. O termo “mestre” deriva da forma com que a função em questão se relaciona com outra, que a ela fica

subordinada. Como o exemplo considera 90 indivíduos, 38 barras e 37 ramos, a função disponibiliza, em sua saída, 4 matrizes: (1) “Tensão de barras”, dimensões 90x38; (2) “Fluxo nos ramos”, dimensões 90x37; (3) “Perda nos ramos”, dimensões 90x37 e (4) “Perdas por indivíduo”, dimensões 90x1.

4.1.8 Função “G_fc_escravo”

Trata-se de função *MATLAB*® responsável por receber, passo a passo, cada um dos indivíduos (conjunto de capacitores) e executar o cálculo do “fluxo de carga”. Esta função está adaptada para inserir os capacitores nas barras, ao invés de os inserir nos ramos. Para cada indivíduo adentrado na função a matriz admitância é atualizada; o que, por sua vez, redundará em novos resultados (perdas, tensões, ângulos, potências, outras). Esta função trabalha subordinada e comandada pela função descrita no item 4.1.7, vindo daí a rotulação “escravo”.

4.1.9 Função “I_aptidão”

Trata-se de função *MATLAB*® responsável por calcular o valor de aptidão (FA) correspondente a cada indivíduo (arranjo com capacitores), e disponibiliza, em sua saída, vetor contendo a aptidão de todos eles; o que, para o exemplo considerado, tem dimensões 90 x1. O FA é calculado exclusivamente baseado no valor de perdas ativas, repetindo a função já utilizada por (Q. A. SALIH, 2015) e (CHING-TZONG SU, 2000) em seus estudos voltados para redes elétricas.

Especificamente, o FA utilizado para otimização de perdas em ambientes industriais é conforme equação 4.1.

$$FA = Total\ de\ Perda_{indiv\u00edduo} \quad (4.1)$$

Onde:

FA = Valor de aptidão, em kW;

$Total\ de\ Perda_{indiv\u00edduo}$ = Total de perda ativa, presente no sistema elétrico devido a inserção dos capacitores de determinada população de indivíduos, representada por vetor 90x1.

O gráfico mostrado na Figura 16 exemplifica o conceito utilizado no cálculo do FA, onde:

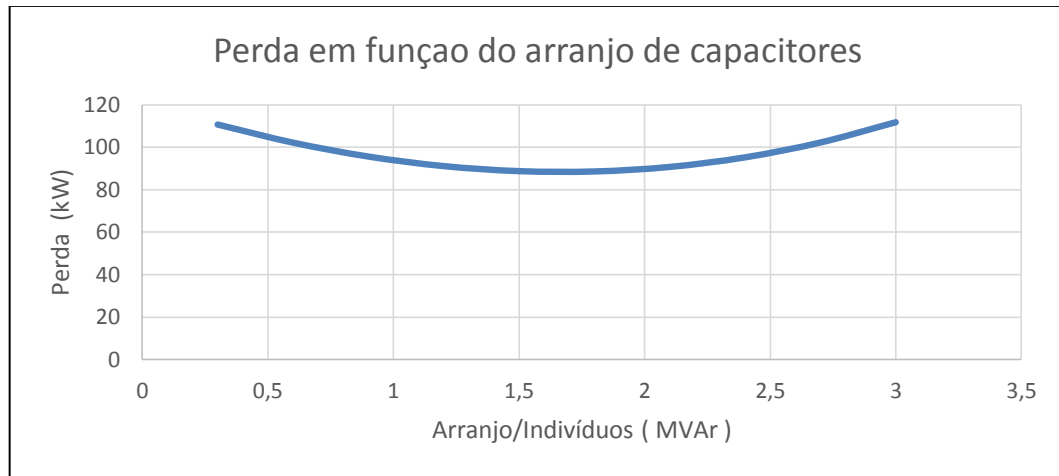
- Sem a inserção dos capacitores (eixo horizontal igual a 0,00 MVar), a perda elétrica assume valor máximo;
- Na medida que a potência capacitiva é aumentada, progressivamente a perda elétrica é reduzida. Contudo, a partir de determinado ponto (no exemplo este ponto é 1,7 MVar), o aumento da potência capacitiva deixa de produzir redução das perdas elétricas, passando a produzir resultado contrário. Esta inversão se deve ao fato das barras elétricas começarem a operar capacitivamente, com consequente aumento da corrente elétrica, o que se transforma em aumento de perdas;
- O melhor arranjo de capacitores, ou “indivíduo ótimo”, é aquele que provoca a inversão da curva, o que, no presente caso, ocorre, como já mencionado anteriormente, em 1,7 MVar.

Destaca-se também que a curva mostrada na Figura 16 procede de simulação, não estando, portanto, fundamentada em nenhum processo de otimização. Para obtê-la foi usado um sistema elétrico de 38 barras, sendo tal sistema também utilizado no capítulo 7.

A simulação foi dividida em 10 etapas e, na primeira delas, um total de 0.3 MVar foi dividido igualmente entre as 38 barras elétricas, cabendo a cada uma delas um capacitor de 7,89 KVar; na segunda etapa um total de 0.6 MVar foi igualmente dividido entre as 38 barras, cabendo a cada uma delas um capacitor de 15,78 KVar; e assim sucessivamente, até atingir um total de 3,0 MVar, com 78,94 KVar inserido em cada barra.

A curva mostrada na Figura 16 pode ser representada pelo polinômio $F(x) = 0,51 * x^3 + 9,9 * x^2 + 37 * x + 121$ e, conseqüentemente, ele representa a evolução das perdas elétricas em função da variação da potência capacitiva inserida nas barras elétricas quando os capacitores, de todas as barras, são iguais.

FIGURA 16: EXEMPLO DE EVOLUÇÃO DA PERDA ELÉTRICA EM FUNÇÃO DA POTÊNCIA CAPACITIVA INSTALADA



Fonte: Do autor

4.1.10 Função “J_penal”

Trata-se de função *MATLAB*® responsável por penalizar (desqualificar) os indivíduos que provocam, no posto de recebimento da instalação industrial (barra elétrica número 1), fator de potência fora dos limites contratuais. Como o exemplo considera 90 indivíduos, e 38 barras, a função disponibiliza em sua saída vetor contendo a aptidão de todos os seus indivíduos (penalizados e não penalizados), o que, para o exemplo dado, tem dimensões 90 x1. A penalização é realizada conforme equação 4.2.

$$FA_{\text{penalizado}} = [1 + K1 * \Delta FP] * FA \quad (4.2)$$

Onde:

$FA_{\text{penalizado}}$ = Vetor 90x1, contendo os valores de aptidão após a aplicação da penalização;

FA = Vetor 90x1, contendo os valores originais de aptidão antes da aplicação da penalização;

$K1$ = Constante de penalização, cujo valor é definido através de simulações;

ΔFP = Vetor correspondente à intensidade do desvio de Fator de Potência provocado por cada indivíduo. Para os indivíduos não infratores o valor é nulo.

Junto à concessionária contratualmente admite-se FP maior que 0.92 indutivo e nunca FP capacitivo. Por isto, a partir das informações de entrada, a função “J_penal” identifica o indivíduo cujo fator de potência, na barra 1, esteja fora do limite contratual, e então calcula a intensidade de desvio provocado por ele.

Conforme Equação 4.3, o valor de desvio (ΔFP) é nulo para os indivíduos não infratores. No caso de infração por FP abaixo de 0,92 indutivo, o desvio é conforme Equação 4.4; e no caso de infração por FP capacitivo, o desvio é calculado conforme Equação 4.5.

$$\Delta FP = 0, \text{ se } FP \geq 0,92 \text{ e } FP \leq 1 \quad (4.3)$$

$$\Delta FP = (0,92 - FP), \text{ se } FP < 0,92 \quad (4.4)$$

$$\Delta FP = [1 - \text{abs}(FP)], \text{ se } FP < 0 \quad (4.5)$$

4.1.11 Função “K_selec”

Trata-se de função *MATLAB*® responsável por realizar a seleção dos indivíduos. Ela se baseia no resultado de torneios promovidos entre eles e funciona da seguinte forma: (1) o número de torneios é igual ao tamanho da população, (2) o número de jogadores, por torneio, é definido aleatoriamente (mas limitado ao número de indivíduos) e (3) os jogadores que participarão de cada torneio também são escolhidos aleatoriamente. A cada torneio o jogador vencedor é o que for mais adaptado; o que, no PAGIND, significa escolha pelo indivíduo que propicia menor perda. Imprimindo característica elitista ao torneio, o indivíduo com melhor aptidão antes do início do torneio é preservado, sendo inserido, sempre, na população selecionada.

Como o exemplo considera 90 indivíduos e 38 barras, a função disponibiliza em sua saída a população binária selecionada, que é representada pela matriz “Pop_selecionada” (90x190).

4.1.12 Função “L_cruz”

Trata-se de função *MATLAB*® responsável por provocar cruzamento entre os indivíduos da população. A escolha dos indivíduos a serem cruzados é feita de forma aleatória, sendo a quantidade definida através do parâmetro “Pc”. Os casais (pares) são formados aleatoriamente, a partir dos indivíduos selecionados para cruzamento, sendo que cada casal gera, obrigatoriamente, dois filhos.

Na implementação da função foram assumidas as seguintes particularidades: (1) ela não permite o cruzamento de pai com pai, (2) não impinge cruzamento a todos indivíduos, (3) permite que o mesmo indivíduo seja acasalado várias vezes e (4) faz com que indivíduos não cruzados passem a fazer parte da população de filhos.

O ponto de início da transferência dos genes é definido aleatoriamente, com base em um número inteiro e, independentemente de qualquer condição, o ponto de início de troca dos genes é sempre anterior ao bit menos significativo.

Como o exemplo considera 90 indivíduos e 38 barras, a função disponibiliza em sua saída a população binária cruzada, que é representada pela matriz “Pop_cruzada” (90x190).

4.1.13 Função “M_mut”

Trata-se de função *MATLAB*® responsável por provocar mutação em parte da população de filhos, enriquecendo-os ou empobrecendo-os geneticamente.

Na implementação desta função foi optado pela mutação “adaptativa”, cujo funcionamento é conforme Equação 4.6.

$$Prob_{perc} = Pm * [1 - \left(\frac{Gerações_{realizadas}}{Gerações_{programados}} \right) * \left(\frac{Taxa_{decaimento}}{100} \right)] \quad (4.6)$$

Onde:

$Prob_{perc}$ = Probabilidade efetivamente imposta à população, em %;

Pm = Probabilidade “Pm”, definida através de parâmetro, em %;

$Gerações_{realizados}$ = Número de ciclos (ou iterações, ou gerações) já realizados no laço genético;

$Gerações_{programados}$ = Número de ciclos (ou iterações, ou gerações) programados para o laço genético;

$Taxa_{decaimento}$ = Taxa de decaimento, definida através de parâmetro, em %.

Apesar de não ser mostrado na Equação 4.6, o programa não permite resultados negativos para a variável calculada ($Prob_{perc}$) e, além disto, força o seu resultado para zero quando o número de gerações realizadas atinge uma fração do total de gerações programadas. De acordo com a quantidade calculada, ou imposta, para a variável " $Prob_{perc}$ ", aleatoriamente são escolhidos os indivíduos que sofrerão mutação. Uma vez selecionado o indivíduo, também de forma aleatória é escolhido o seu gene que sofrerá mutação, o que, no programa desenvolvido, com população binária, corresponde à inversão de seu estado lógico.

No exemplo considerado, a função disponibiliza em sua saída a nova população binária, representada por matriz 90x190.

A Figura 17 mostra a evolução da intensidade de mutação " $Prob_{perc}$ ", consequente do parâmetro " P_m " igual a 5, parâmetro " $Gerações_{programados}$ " igual a 25 e parâmetro " $Taxa_{decaimento}$ " igual a 50.

FIGURA 17: EVOLUÇÃO DA VARIÁVEL "INTENSIDADE DE MUTAÇÃO"



Fonte: Do autor, gerada pelo programa MATLAB®

4.2 CONVERSÃO DE POPULAÇÃO BINÁRIA PARA POPULAÇÃO COMERCIAL

A matriz “PI_Bin”, exemplificada pela Tabela 20, contém a população que será submetida ao processo genético. Esta população poderá ser a primeira; gerada aleatoriamente, antes do início da primeira iteração, ou poderá proceder da saída do bloco que realiza cada mutação, à cada nova iteração. A matriz “PI_Bin” tem seu indivíduos representados por código binário de 5 bits e, por isto, torna-se necessário convertê-los para números reais; condição indispensável para que possam ser injetados no bloco que faz o cálculo de fluxo de carga.

O processo de conversão da população binária (PI_Bin) para seu equivalente, em MVAR, já com valores comerciais, envolve 4 matrizes, conforme exemplificado através da Tabela 20 até Tabela 24.

A matriz “Faixa_Busca” (Tabela 21) contém 31 diferentes valores de potência reativa, em MVAR, relacionados através de 31 itens, escritos em código binário, com 5 bits. O maior valor de potência reativa (item 31) tem seu valor definido pelo valor da potência reativa da maior carga presente no sistema elétrico a ser submetido à minimização de perdas. A potência do capacitor inserido na linha 1 tem valor igual a 1/31 do maior valor de potência (item 31); a potência do capacitor inserido na linha 2 tem valor igual a 2/31 do maior valor de potência (item 31); e assim progressivamente. Se, e somente se, o valor do reativo da maior carga for alterado, também os valores das potências dos 31 capacitores são automaticamente recalculados/alterados.

TABELA 20: MATRIZ “PI_ BIN”, CODIFICADA POR 5 BITS

Indivíduo	Barra número 38					-	-	-	-	Barra número 01				
	Bit 190	Bit 189	Bit 188	Bit 187	Bit 186	-	-	-	-	Bit 005	Bit 004	Bit 003	Bit 002	Bit 001
1	1	0	0	1	0	-	-	-	-	0	1	0	1	1
2	1	1	1	0	1	-	-	-	-	1	0	0	0	1
3	1	0	1	1	0	-	-	-	-	0	1	0	1	0
	-	-	-	-	-	-	-	-	-	-	-	-	-	-
90	1	0	1	1	1	-	-	-	-	0	0	1	1	1

Fonte: Do autor

TABELA 21: MATRIZ “FAIXA_BUSCA”, CODIFICADA COM 5 BITS, CAPACITORES EM MVar

Item	Capacitores (MVar)	Codificação Binária				
1	0,005122	0	0	0	0	1
-	-	-	-	-	-	-
31	0,158790	1	1	1	1	1

Fonte: Do autor, extraída de tabela comercial (ABB, 2011)

O confronto e identidade do código binário da matriz “Faixa_Busca” com o código binário da matriz “PI_BIN”, gera a matriz “PI_MVAR” (Tabela 22). Findo este passo, a população que irá ser injetada no bloco que faz o cálculo de fluxo de carga já está expressa MVar, porém com valores não comerciais e, por isto, ainda carece de transformação adicional.

O confronto dos valores contidos na matriz “PI_MVAR” (Tabela 22) com os valores contidos na matriz “Tabela_comercial” (Tabela 23), gera a matriz “PI_MVAR_comercial” (Tabela 24), já com valores comerciais e, por isto, já pronta para ser injetada no bloco que faz o cálculo de fluxo de carga. A escolha dos valores das células da matriz “PI_MVAR_comercial” (Tabela 24) é feita baseada em intervalos de valores existentes na “Tabela_comercial”, sempre se optando pelo capacitor de menor valor. Como exemplos: para célula com valor igual a 0,005122 MVar (5,122 kVar) é escolhido valor comercial de 5 kVar; para célula com valor igual a 0,010244 MVar (10,24 kVar) é escolhido valor de 10 kVar; para célula com valor igual a 0,153667 MVar (153,66 kVar) é escolhido valor de 140 kVar.

TABELA 22: MATRIZ “PI_MVAR, CAPACITORES EM MVar, VALORES NÃO COMERCIAIS

Indivíduo	Capacitor da Barra número 38 (MVar)	-	-	Capacitor da Barra número 01 (MVar)
1	0,092200006	-	-	0,056344838
2	0,148545483	-	-	0,087078387
-	-	-	-	-
90	0,097322903	-	-	0,037051000

Fonte: Do autor

Registra-se e destaca-se que a maior potência capacitiva disponível para seleção (900 kVAr), cobre, com folga, a máxima potência capacitiva possível de ser determinada pelo programa; o que, por sua vez, corresponde à máxima potência motora tecnicamente possível de ser instalada no nível de baixa tensão de instalações industriais típicas.

TABELA 23: MATRIZ “TABELA_COMERCIAL, VALORES EM kVAr

C1 kVAr	C2 kVAr	-	-	-	-	-	-	-	-	-	-	-	-	C197 kVAr	C198 kVAr
0	5	-	-	100	0	10	-	200	0	15	-	300	-	810	900

Fonte: Do autor

TABELA 24: MATRIZ “PI_MVAR_COMERCIAL”, MOSTRANDO PARTE DE SEUS INDIVÍDUOS, JÁ COM VALORES COMERCIAIS

Indivíduo	Capacitor da Barra núm. 38 (MVar)	-	-	Capacitor da Barra núm. 01 (MVar)
1	0,090000000	-	-	0,050000000
2	0,140000000	-	-	0,080000000
90	0,100000000	-	-	0,035000000

Fonte: Do autor

4.3 PARÂMETROS EXISTENTES NO PAGIND

Na Tabela 25 estão listados os parâmetros acessíveis aos usuários do PAGIND através do *script* que comanda a execução do programa. Todos os parâmetros foram determinados empiricamente, buscando conciliar precisão com baixa demanda computacional. Dentre eles estão os correspondentes à configuração elementar dos Algoritmos Genéticos (Número de indivíduos, Número de gerações, Probabilidade de cruzamento e probabilidade de mutação), os que transformam a configuração genética elementar em uma configuração dedicada (Penalidade de Fator de Potência e Decremento de mutação) e os existentes para impor condições funcionais (Perfil de Carga e Seleção de mutação adaptativa).

TABELA 25: RESUMO DOS PARÂMETROS EXISTENTES NO PAGIND

Parâmetro	Função	Faixa	Típico
Núm. de amostras	Define número de eventos estatísticos	Aberta	1
Perfil de carga	Define intensidade das cargas	0.5 a 2	1
Núm. de indivíduos	Define tamanho das populações	Aberta	220
Núm. de gerações	Define o número de gerações do AG	Aberta	25
K1	Estabelece penalidade de FP	1 a 5	1,2
Prob. de mutação	Define a probabilidade de mutação	0 a 5 (%)	5
Probab. cruzamento	Define a probabilidade de cruzamentos	5 a 95 (%)	50
Decr. mutação	Define decaimento da taxa de mutação	0 a 100(%)	50
Seleciona mutação adaptativa	Bit para controle da mutação 1=adaptativa, 0= não adaptativa	-	1
Ativa perda natural	Bit para controle do estudo de perda 1=ativo, 0= não ativo	-	1

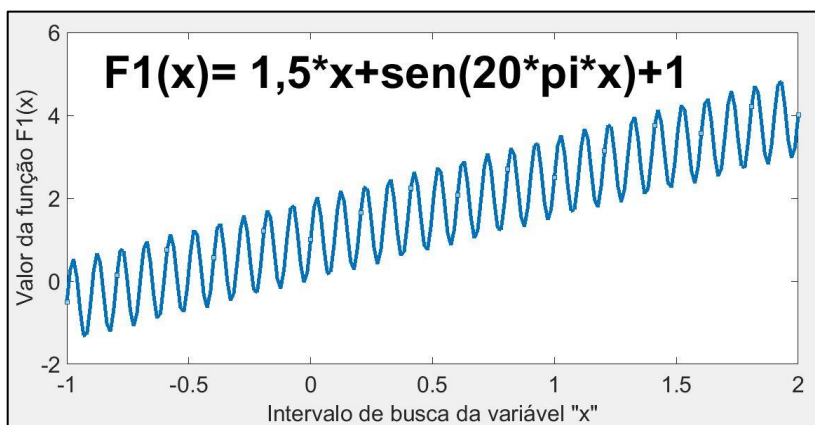
Fonte: Do autor

5 VALIDAÇÃO DO BLOCO “ROTINAS AG”

5 VALIDAÇÃO DO BLOCO “ROTINAS AG”

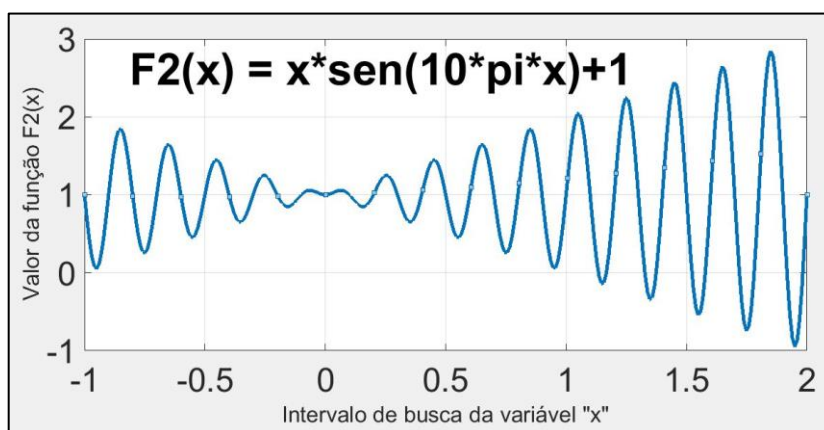
Conforme mostrado na Figura 14, contida no capítulo 4, o programa objeto da dissertação, referenciado como PAGIND, usa, em sua estrutura, dentre outras, o bloco “ROTINAS AG”. Uma vez que se espera do programa PAGIND efetiva performance e confiabilidade, torna-se necessário atestar a qualidade de todos os seus componentes, inclusive os constituintes do bloco “ROTINAS AG”. Para validação da performance do mencionado bloco foram escolhidas diversas funções, cujas representações gráficas estão apresentadas nas Figuras 18 até 22. Para cada função, os limites da faixa de busca estão definidos na própria figura.

FIGURA 18: FUNÇÃO $F1(X) = 1,5 \cdot X + \text{SEN}(20 \cdot \text{PI} \cdot X) + 1$

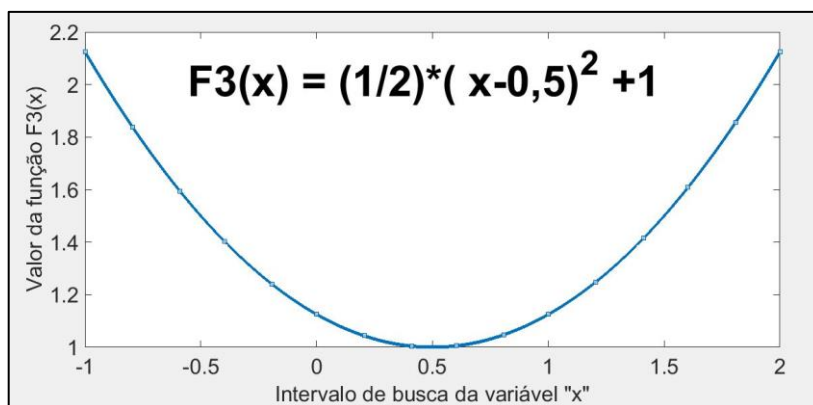


Fonte: Do autor, gerada pelo programa *MATLAB*®

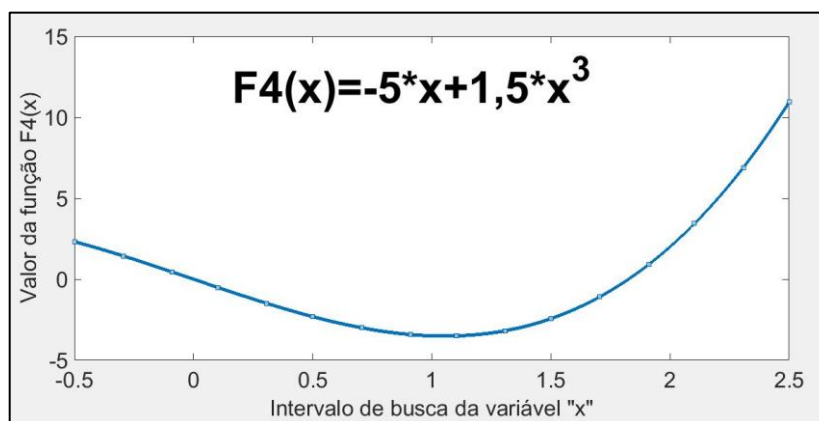
FIGURA 19: FUNÇÃO $F2(X) = X \cdot \text{SEN}(10 \cdot \text{PI} \cdot X) + 1$



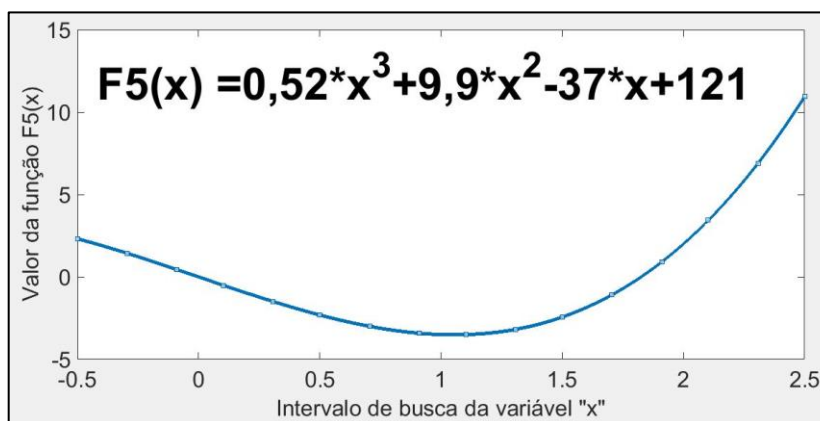
Fonte: Do autor, gerada pelo programa *MATLAB*®

FIGURA 20: FUNÇÃO $F3(X) = (1/2)*(X-0,5)^2 + 1$ 

Fonte: Do autor, gerada pelo programa *MATLAB*®

FIGURA 21: FUNÇÃO $F4(X) = -5*X + 1,5*X^3$ 

Fonte: Do autor, gerada pelo programa *MATLAB*®

FIGURA 22: FUNÇÃO $F5(X) = 0,52*X^3 + 9,9*X^2 - 37*X + 121$ 

Fonte: Do autor, gerada pelo programa *MATLAB*®

Duas, dentre as cinco funções escolhidas para validar o bloco “ROTINAS AG”, são multimodais (Figuras 18 e 19), e foram selecionadas intencionalmente, visto que são ricas em picos, o que representa significativa dificuldade para programas otimizadores; as três últimas funções, contidas nas Figuras 20, 21 e 22, foram utilizadas por terem formato similar ao da função que efetivamente guia a otimização do PAGIND (item 4.1.9, Figura 16).

5.1 VALIDANDO O BLOCO “ROTINAS AG”

Para validar a performance do bloco “ROTINAS AG”, para cada uma dentre as cinco funções tomadas como referência foram realizadas diversas otimizações, em todas elas buscando-se determinar o ponto de mínimo global.

Nas minimizações, excluindo os limites da faixa de busca, demais parâmetros foram mantidos inalterados, e conforme listados a seguir:

- Número de indivíduos = 220, ou 660;
- Número de gerações = 25;
- Número de bits = 22
- Probabilidade de cruzamento (%) = 50;
- Probabilidade de mutação percentual = 5;
- Decremento de mutação percentual = 50.

A mutação utilizada foi tipo “adaptativa”; cuja intensidade decai em função do número de gerações do Algoritmo Genético, e cuja evolução é conforme mostrada na Figura 17 (capítulo 4).

Os cálculos do *MATLAB*® foram realizados com 15 casas decimais, superando a precisão arbitrariamente considerada como necessária (6 casas decimais). Destaca-se também que o número de bits (22) utilizados para constituir a codificação binária de cada indivíduo, de todas as funções, corresponde ao necessário para que eles, quando transformados em números reais, disponham de 6 casas decimais.

5.2 RESUMO DOS RESULTADOS PRODUZIDOS PELO BLOCO “ROTINAS AG” QUANDO REALIZANDO MINIMIZAÇÕES

A estatística dos resultados produzidos pelo bloco “ROTINAS AG” quando realizando minimizações das funções contidas nas Figuras 18 até 22, é conforme conteúdo da Tabela 26, onde:

- Coluna 1: Contém a expressão numérica da função que está sendo otimizada;
- Coluna 2: Contém o tamanho da população utilizada na otimização;
- Colunas 3: Contêm a quantidade de eventos, dentre o universo de 100 otimizações realizadas, em que o erro relativo percentual inserido no resultado foi maior que 0,1 %;
- Coluna 4: Informa o maior erro relativo percentual incorrido dentre os 100 eventos realizados;
- Coluna 5: Registra a demanda de tempo necessária para realizar o conjunto de 100 otimizações.

TABELA 26: ESTATÍSTICA DOS RESULTADOS PRODUZIDOS PELO BLOCO “ROTINAS AG”

Expressão Numérica	Tamanho da população	Eventos com erro > 0,1 %	Maior Erro (%)	Demanda (Segundos)
$F1(x) = 1,5x + \sin(20\pi x) + 1$	220	94	0,63	1105
$F1(x) = 1,5x + \sin(20\pi x) + 1$	660	2	0,44	10751
$F2(x) = x \sin(10\pi x) + 1$	220	18	0,13	1228
$F2(x) = x \sin(10\pi x) + 1$	660	5	0,11	11022
$F3(x) = (1/2) * (x - 0,5)^2 + 1$	220	0	0,03	911
$F4(x) = -5x + 1,5x^3$	220	4	0,78	871
$F5(x) = 0,51x^3 + 9,9x^2 - 37x + 121$	220	0	0,01	963

Fonte: Do autor

5.3 CONCLUSÕES SOBRE OS RESULTADOS PRODUZIDOS PELO BLOCO “ROTINAS AG”

Avaliando-se os dados lançados na Tabela 26 pode ser concluído que o bloco “ROTINAS AG” apresenta a seguinte performance:

- Para as funções com várias descontinuidades, mostrada nas Figura 18 e 19, o bloco “ROTINAS AG” produz, com populações moderadas (menores ou iguais a 220 indivíduos), considerável frequência de erros relativos maiores que 0,1 %; contudo, confirmando conceitos relativos a Algoritmos Genéticos, o aumento da população (de 220 para 660 indivíduos) diminui significativamente os desvios, obstante exponencial crescimento da demanda computacional;
- Para as funções com várias descontinuidades, mostradas nas Figuras 18 e 19, os erros produzidos com populações moderadas (menores ou iguais a 220 indivíduos), não são devidos à ineficiência do programa computacional desenvolvido e referenciado como “ROTINAS AG”; prendendo-se, os erros, exclusivamente ao tamanho da população utilizado para realizar a minimização;
- Para as funções monomodais (Figuras 20 e 21), com 220 indivíduos a demanda computacional é suficientemente rápida (aproximadamente 11 segundos por cada execução com 25 iterações), e o erro relativo das respostas nunca excede a 1 %;
- Para a função $F(x) = 0,51 * x^3 + 9,9 * x^2 + 37 * x^3 + 121$ que, conforme conteúdo do item 4.1.9 e Figura 16, é uma réplica do formato da função de aptidão do programa desenvolvido, as minimizações realizadas com o bloco “ROTINAS AG” produzem, conforme Tabela 26, resultados precisos (maior erro relativo, em 100 minimizações, é de somente 0.01 %) e bastante rápidos (aproximadamente 9 segundos por cada minimização), sendo, ambos os resultados, compatíveis para aplicações em estudos e projetos voltados para sistemas elétricos.

5.4 TOMADA DE DECISÃO SOBRE A EFICIÊNCIA DO BLOCO “ROTINAS AG”

O programa *MATLAB*®, estruturado como Algoritmo Genético e referenciado como “ROTINAS AG”, consegue produzir resultados extremamente precisos e suficientemente rápidos quando minimizando expressões numéricas similares à função que guia a otimização do PAGIND, sendo, por isto, considerado como aprovado para ser utilizado no programa e em suas aplicações voltadas para minimização de perdas em sistemas elétricos industriais.

6 VALIDAÇÃO DO BLOCO “ROTINAS FC”

6 VALIDAÇÃO DO BLOCO “ROTINAS FC”

Conforme mostrado na Figura 14, contida no capítulo 4, o programa objeto da dissertação (PAGIND) usa, em sua estrutura, dentre outros, o bloco referenciado como “ROTINAS FC”. Este bloco corresponde a um programa editado em *MATLAB*®, especialista em cálculo de fluxo de carga, e sob uso recorrente dentre o meio discente da UFPB, sendo de autoria desconhecida.

Uma vez que se espera do programa PAGIND efetiva performance (precisão e velocidade), torna-se necessário atestar a qualidade de todos os seus componentes, inclusive a do bloco “ROTINAS FC”. Para efetuar a validação do bloco “ROTINAS FC” foi tomado como referência o programa *POWERFACTORY*®, visto que tal programa tem reputação internacional e contém, dentre outras, especialidade em cálculo de fluxo de carga.

Tanto o programa validador (*POWERFACTORY*®) quanto o programa em validação (“ROTINAS FC”) são ferramentas determinísticas, sem nenhum comportamento estocástico e, por isto, a comparação dentre eles pode ser feita com base nos resultados produzidos por uma única simulação.

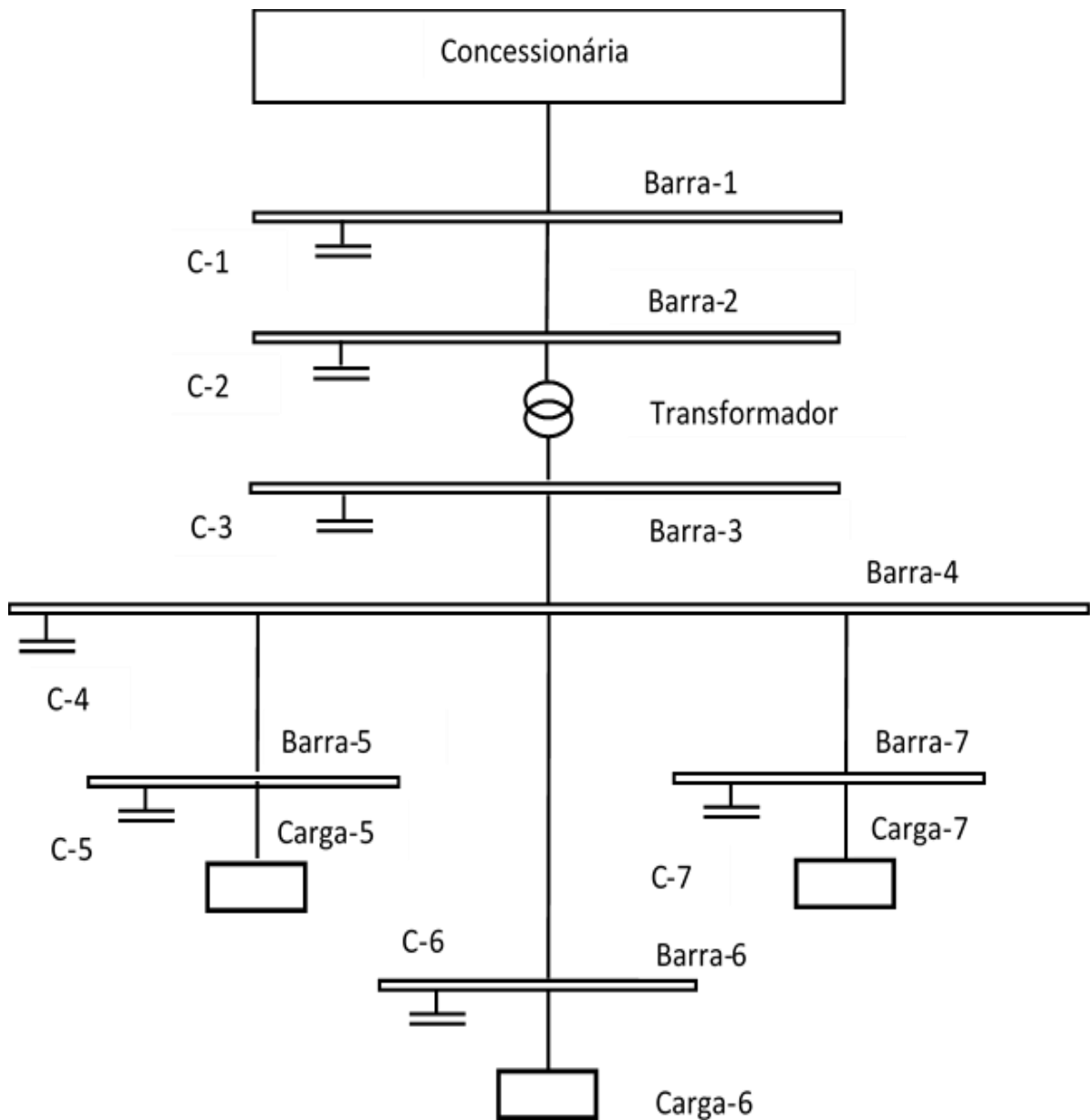
6.1 O SISTEMA ELÉTRICO UTILIZADO PARA VALIDAÇÃO DO BLOCO “ROTINAS FC”

O sistema elétrico escolhido para se fazer a validação do bloco “ROTINAS FC” corresponde a uma instalação tipicamente industrial, constituída por 7 barras, com sua energia sendo suprida externamente, e a Figura 23 contém o unifilar do mencionado sistema.

A escolha por um sistema simples, com número reduzido de barras, repousou na conveniência por concisão; tanto para reportar dados, como também para reportar resultados. Também cabe destaque o fato de que o sistema de 7 barras é, na realidade, parte do sistema elétrico de 38 barras que, no capítulo 7, é utilizado para aplicação do PAGIND.

Os valores dos capacitores “*shunt*” utilizados no sistema elétrico referencial foram escolhidos arbitrariamente, e não procedem de nenhuma base de cálculo ou de otimização.

FIGURA 23: UNIFILAR DO SISTEMA ELÉTRICO UTILIZADO PARA VALIDAÇÃO DO BLOCO “ROTINAS FC”



Fonte: Do autor

6.2 DADOS INSERIDOS NO PROGRAMA POWERFACTORY®

A Tabela 27 contém dados e especificações dos componentes utilizados no sistema elétrico utilizado para validação do bloco “ROTINAS FC”, sendo todos eles inseridos, com as mesmas unidades mostradas, no ambiente do programa POWERFACTORY®.

TABELA 27: DADOS ALIMENTADOS NO PROGRAMA POWERFACTORY®

Dispositivo Elétrico	Função/Característica Elétrica
Cabo elétrico, nível 13,8 kV	Interliga barra 1 com a barra 2, impedância igual a $(0,232500 + j*0,051150)$ Ohms
Transformador	Inserido entre barra 2 e barra 3, impedância de 3,0 %, potência de 1 MVA, perda nula no cobre; primário 13,8KV e secundário 0,38 kV
Cabo elétrico, nível 0,38 kV	Interliga barra 3 com a barra 4, impedância igual a $(0,003000 + j*0,002000)$ Ohms
Cabo elétrico, nível 0,38 kV	Interliga barra 4 com a barra 5, impedância igual a $(0,070500 + j*0,016500)$ Ohms
Cabo elétrico, nível 0,38 kV	Interliga barra 4 com a barra 6, impedância igual a $(0,022500 + j*0,015000)$ Ohms
Cabo elétrico, nível 0,38 kV	Interliga barra 4 com a barra 7, impedância igual a $(0,011250 + j*0,007500)$ Ohms
Carga da barra 5	P=0,147 MW; Q=0,10268 MVar
Carga da barra 6	P=0,2942 MW; Q=0,1588 MVar
Carga da barra 7	P=0,5884 MW e Q=0,31758 MVar
Capacitores	Potências de 0,01; 0,02; 0,03; 0,04; 0,05; 0,06 e 0,07 MVar, inseridos, respectivamente, nas barras elétricas números 1,2, 3, 4, 5, 6 e 7
Potência base	100 MVA

Fonte: Do autor

6.3 DADOS INSERIDOS NO BLOCO “ROTINAS FC”

Na Tabela 28 e na Tabela 29 estão os dados inseridos no bloco “ROTINAS FC”, que procedem, através de conversões, dos dados relacionados na Tabela 27.

O conteúdo da Tabela 28 é conforme descrito a seguir:

- Coluna 1: Contém o número das barras; onde a barra número 1 corresponde ao posto de recebimento de energia (conexão com a concessionária);

- Coluna 2: Contém a codificação do tipo das barras: 1=PQ, 2=PV, 3=Vθ;
- Colunas 3 e 4: Contém os valores relativos às potências das cargas;
- Colunas 5: Contém os valores da potência reativa nominal dos capacitores inseridos nos barramentos.

TABELA 28: DADOS ALIMENTADOS NO BLOCO “ROTINAS FC”, CORRESPONDENTES ÀS BARRAS ELÉTRICAS

Num. Barra	Tipo	Pc (MW)	Qc (MVar)	Cap. (MVar)
1	3	0,0000	0,0000	0,01
2	1	0,0000	0,0000	0,02
3	1	0,0000	0,0000	0,03
4	1	0,0000	0,0000	0,04
5	1	0,1470	0,1027	0,05
6	1	0,2942	0,1588	0,06
7	1	0,5884	0,3176	0,07

Fonte: Do autor

O conteúdo da Tabela 29 é conforme descrito a seguir:

- Coluna 1: Contém a origem dos ramos elétricos (“De”);
- Coluna 2: Contém o destino dos ramos elétricos (“Para”);
- Colunas 3 e 4: Contêm, respectivamente, os valores das componentes resistivas e reativas que compõem as impedâncias existentes nos ramos, valores em p.u., obtidos a partir das impedâncias base de 1,9044 Ohms e 0.0001444 Ohms, respectivamente para o nível de 13,8 kV e para o nível 0,38 kV. A potência nominal do transformador é de 1 MVA, com impedância percentual, em sua própria base, igual a 3 %. A perda no cobre do transformador é considerada nula. Trazendo para a nova base (100 MVA), obtém-se o valor da impedância do transformador, conforme lançado na Tabela 29.

TABELA 29: DADOS ALIMENTADOS NO BLOCO “ROTINAS FC”, CORRESPONDENTES AOS RAMOS ELÉTRICOS

De	Para	Resistência (p.u.) ²	Reatância (p.u.)	Nível de Tensão (kV)
1	2	0,1220856960	0,0268588530	13,8
2	3	0,0000000000	3,0000000000	(Transformador)
3	4	2,0775623270	1,3850415510	0,380
4	5	48,8227146800	11,4265928000	0,380
4	6	15,5817174500	10,3878116300	0,380
4	7	7,7908587260	5,1939058170	0,380

Fonte: do autor

6.4 RESULTADOS GERADOS PELO PROGRAMA VALIDADOR (POWERFACTORY®)

Parte dos diversos relatórios que são gerados pelo programa POWERFACTORY®; relativos ao cálculo de fluxo de carga do sistema elétrico em referência, estão mostrados nas Figuras 24 e 25. A Figura 24 contém, dentre outros, a quantificação das perdas (112,13kW + j*112,77 kVar), enquanto que na Figura 25 estão reportados tensões e ângulos dos barramentos elétricos.

6.5 RESULTADOS GERADOS PELOS PROGRAMAS SOB VALIDAÇÃO (ROTINAS FC)

Em relação aos resultados gerados pelo programa sob validação (ROTINAS FC) quando aplicado ao sistema elétrico tomado como referência, a Tabela 30 contém os resultados que foram gerados em relação às barras elétricas, enquanto a Tabela 31 contém os resultados gerados em relação aos ramos elétricos.

² As perdas do cobre do transformador foram consideradas como nulas, tanto no programa em validação (ROTINAS FC) como no programa validador (POWERFACTORY®)

FIGURA 24: RELATÓRIO EMITIDO PELO PROGRAMA POWERFACTORY® COM RESULTADOS RELATIVOS ÀS PERDAS INTERNAS

Grid: Grid		System Stage: Grid		Study Case: Estudo_sistema_7barras		Annex: / 1	
Grid: Grid		Summary					
No. of Substations	12	No. of Busbars	7	No. of Terminals	159	No. of Lines	9
No. of 2-w Trfs.	1	No. of 3-w Trfs.	0	No. of syn. Machines	0	No. of asyn.Machines	0
No. of Loads	6	No. of Shunts	7	No. of SVS	0		
Generation	=	0,00 kW	0,00 kvar	0,00 kVA			
External Infeed	=	1141,83 kW	456,84 kvar	1229,82 kVA			
Inter Grid Flow	=	0,00 kW	0,00 kvar				
Load P(U)	=	1029,69 kW	579,06 kvar	1181,35 kVA			
Load P(Un)	=	1029,70 kW	579,06 kvar	1181,35 kVA			
Load P(Un-U)	=	0,01 kW	-0,00 kvar				
Motor Load	=	0,00 kW	0,00 kvar	0,00 kVA			
Grid Losses	=	112,13 kW	112,77 kvar				
Line Charging	=		0,00 kvar				
Compensation ind.	=		0,00 kvar				
Compensation cap.	=		-234,99 kvar				

Fonte: Gerada pelo programa Powerfactory®

FIGURA 25: RELATÓRIO EMITIDO PELO PROGRAMA POWERFACTORY® COM RESULTADOS RELATIVOS ÀS BARRAS ELÉTRICAS E RAMOS ELÉTRICOS

Grid: Grid		System Stage: Grid		Study Case: Estudo_sistema_7barras		Annex:		/ 1	
	rtd.V	Bus - voltage		Voltage - Deviation [%]					
	[kV]	[p.u.]	[kV]	[deg]	-10	-5	0	+5	+10
Single Busbar(1)									
BARRA2	13,80	0,998	13,78	0,02					
Single Busbar(2)									
BARRA3	0,38	0,984	0,37	-1,98					
Single Busbar(3)									
BARRA4	0,38	0,954	0,36	-2,35					
Single Busbar(5)									
BARRA5	0,38	0,862	0,33	-1,29					
Single Busbar(6)									
BARRA6	0,38	0,889	0,34	-3,24					
Single Busbar(7)									
BARRA7	0,38	0,887	0,34	-3,03					
Single Busbar									
BARRA1	13,80	1,000	13,80	0,00					

Fonte: Gerada pelo programa Powerfactory®

TABELA 30: RESULTADOS GERADOS PELO BLOCO “ROTINAS FC”, RELATIVOS ÀS BARRAS ELÉTRICAS

Barra	Tensão	Ângulo
1	1,000	0,00
2	0,998	0,02
3	0,984	-1,98
4	0,954	-2,35
5	0,862	-1,29
6	0,889	-3,24
7	0,887	-3,03

Fonte: Do autor, gerada pelo programa *MATLAB*®

TABELA 31: RESULTADOS GERADOS PELO BLOCO “ROTINAS FC”, RELATIVOS AOS RAMOS ELÉTRICOS

Ramo	Perda Ativa kW	Perda Reativa kVAr
1/2	1,85	0,40
2/3	0,00	46,22
3/4	32,57	21,71
4/5	17,05	3,99
4/6	19,53	13,03
4/7	41,13	27,42
Total	112,13	112,77

Fonte: Do autor, gerada pelo programa *MATLAB*®

6.6 COMPARANDO RESULTADOS

Para facilitar a comparação, todos os resultados; tantos os gerados pelo programa *POWERFACTORY*® (Figuras 24 e 25) como os gerados pelo programa “ROTINAS FC” (Tabelas 30 e 31), foram copiados e concentrados nas Tabelas 32 e 33.

6.6.1 Comparando resultados relativos às barras (tensões e ângulos)

No que tange a valores de tensão e de ângulo, comparando os resultados gerados pelo programa *POWERFACTORY*® (Tabela 32, colunas 2 e 3) com os resultados gerados pelo programa “ROTINAS FC” (Tabela 32, colunas 4 e 5), já

arredondados; conclui-se que ambos, programa validador e programa em validação, produzem resultados idênticos.

TABELA 32: RESULTADOS DO PROGRAMA *POWERFACTORY*® E DO BLOCO “ROTINAS FC”

Barra Elétrica	<i>POWERFACTORY</i> ®		ROTINAS FC	
	Tensão (pu)	Ângulo (Graus)	Tensão (pu)	Ângulo
1	1,00	0,00	1,00	0,00
2	0,998	0,02	0,998	0,02
3	0,984	-1,98	0,984	-1,98
4	0,954	-2,35	0,954	-2,35
5	0,862	-1,29	0,862	-1,29
6	0,889	-3,24	0,889	-3,24
7	0,887	-3,03	0,887	-3,03

Fonte: Do autor

6.6.2 Comparando resultados relativos às perdas

No que tange a perdas, comparando os resultados gerados pelo programa *POWERFACTORY*® (Tabela 33, colunas 1 e 2) com os resultados gerados pelo programa “ROTINAS FC” (Tabela 33, colunas 3 e 4), já arredondados; conclui-se que ambos, programa validador e programa em validação, produzem resultados idênticos.

TABELA 33: RESULTADOS GERADOS PELO PROGRAMA *POWERFACTORY*® E PELO BLOCO “ROTINAS FC”

<i>POWERFACTORY</i> ®		<i>MATLAB</i> ®	
Perda ativa (kW)	Perda reativa (kVAr)	Perda ativa (kW)	Perda reativa (kVAr)
112,13	112,77	112,13	112,77

Fonte: Do autor

6.6.3 Comparando tempos de execução

Apesar de não estar lançado em nenhuma tabela, para convergir aos resultados o bloco “ROTINAS FC” requer, aproximadamente 1,2 segundos; enquanto que o programa validador (*POWERFACTORY*®) requer, aproximadamente, 1,00

segundo; concluindo-se então que o bloco “ROTINAS FC” tem demanda computacional equivalente à demanda do programa validador.

6.7 TOMADA DE DECISÃO SOBRE A EFICIÊNCIA DO BLOCO “ROTINAS FC”

Considerando a otimização, comentários e resultados reportados ao longo deste capítulo, conclui-se que o programa que constitui o bloco “ROTINAS FC” consegue produzir resultados equivalentes aos resultados produzidos pelo programa validador (*POWERFACTORY*®), sendo, por isto, aprovado para utilização no programa PAGIND.

**7 APLICAÇÃO DO PROGRAMA OBJETO DA
DISSERTAÇÃO**

7 APLICAÇÃO DO PROGRAMA OBJETO DA DISSERTAÇÃO

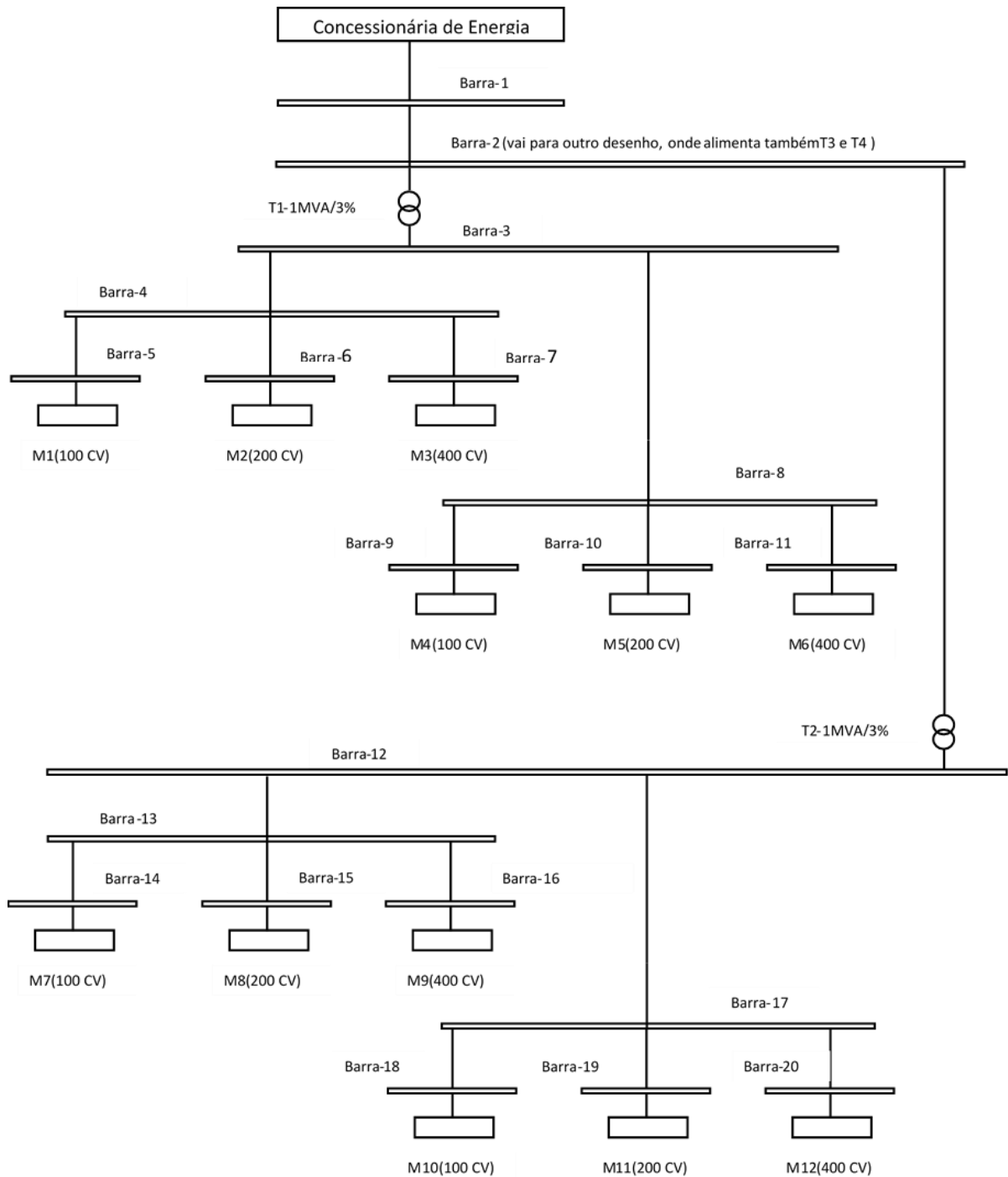
O programa objeto da dissertação (PAGIND) usa, em sua estrutura, conforme mostrado na Figura 14, capítulo 4, dentre outros, o bloco nomeado “ROTINAS DE APOIO”.

Devido à sua própria natureza, e conteúdo, a validação do bloco “ROTINAS DE APOIO” somente pode ser feita imprimindo funcionamento às demais estruturas do PAGIND. Considerando isto, e considerando também que as demais estruturas constituintes do PAGIND foram validadas anteriormente (capítulos 5 e 6), o presente capítulo trata da efetiva aplicação do programa PAGIND.

O projeto, desenvolvimento e funcionamento do PAGIND está voltado para promover minimização de perdas elétricas em sistemas industriais. Adicionalmente, o programa evita, através de penalizações, que a instalação industrial opere, no posto de recebimento, com FP fora dos limites contratuais (nunca abaixo de 0.92 indutivo e nunca capacitivo) e permite que o usuário, por livre escolha, selecione e decida em quais barras elétricas ele não deseja a inserção de capacitor.

7.1 UNIFILAR DO SISTEMA ELÉTRICO 38 BARRAS SUBMETIDO À MINIMIZAÇÃO DE PERDAS

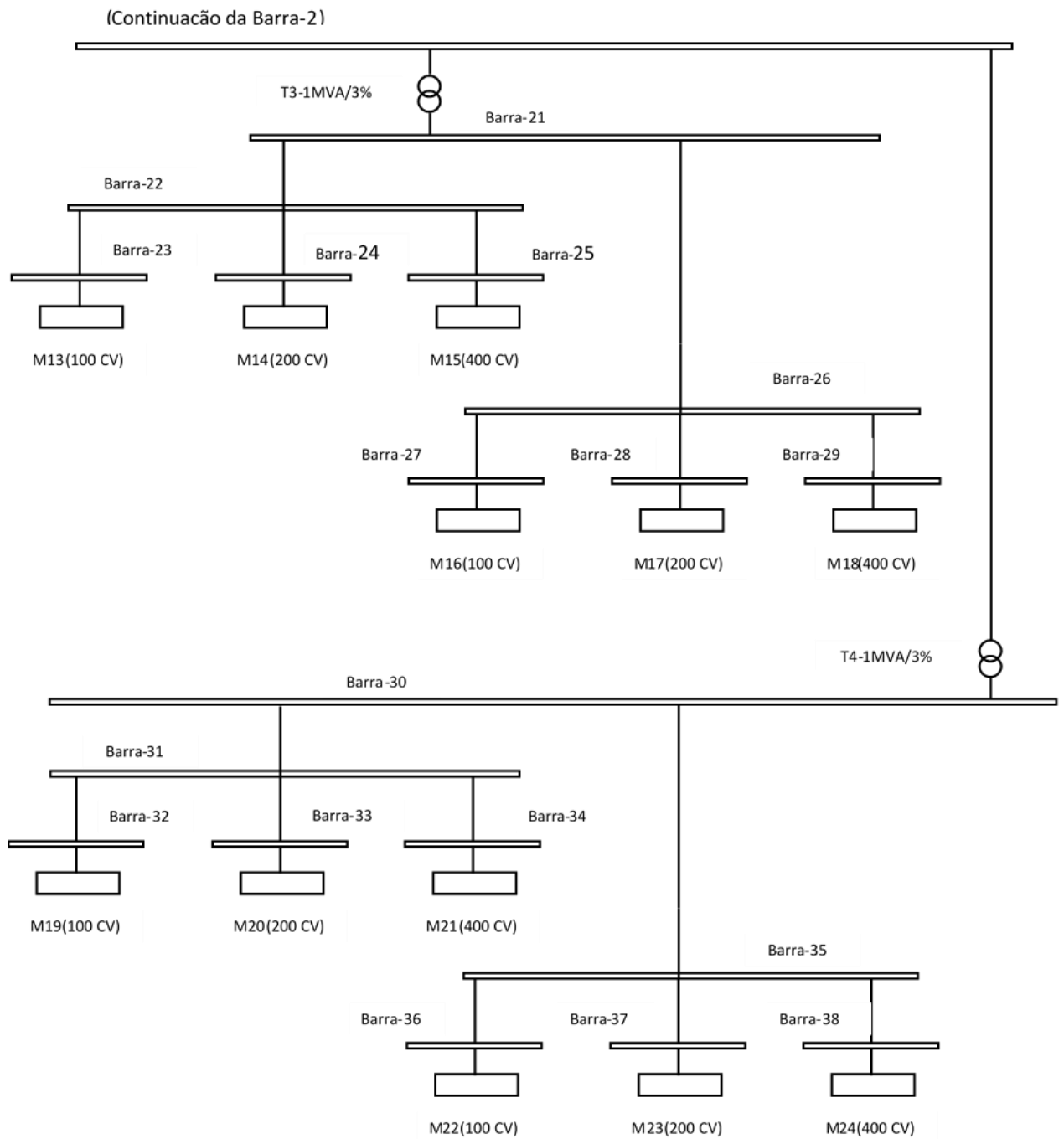
O sistema submetido à minimização de perdas é conforme mostrado nas Figuras 26 e 27; correspondendo a circuito elétrico industrial, 38 barras, conectado à rede de distribuição através do nível de tensão 13,8 kV; com cargas alimentadas em 380 V e dispondo de quatro subestações unitárias, idênticas sob todos os aspectos: transformadores idênticos, topologia idêntica, mesma quantidade de cargas, mesmas cargas, mesmos cabos e mesmos comprimentos de cabos. Cada subestação unitária é constituída por transformador de 1,0 MVA/3,0%, suprindo 9 barras elétricas e 6 motores. Os dados de barras e ramos estão listados, respectivamente, nas Tabelas 34 e 35, onde P_c =potência ativa das cargas, Q_c =potência reativa das cargas, R (p.u.) = componente resistiva da impedância do ramo e X (p.u.) = componente reativa da impedância do ramo.

FIGURA 26: UNIFILAR DO SISTEMA ELÉTRICO DE 38 BARRAS ³


Fonte: Do autor

³ Primeira parte do desenho unifilar do sistema elétrico de 38 barras (a segunda parte é apresentada na figura 27).

FIGURA 27: COMPLEMENTAÇÃO DO UNIFILAR DO SISTEMA ELÉTRICO DE 38 BARRAS ⁴



Fonte: Do autor

⁴ Segunda e ultima parte do desenho unifilar do sistema elétrico de 38 barras (a primeira parte é apresentada na figura 26).

TABELA 34: DADOS DAS BARRAS DO SISTEMA ELÉTRICO SUBMETIDO À MINIMIZAÇÃO

Barra	Pc (MW)	Qc (MVar)	Barra	Pc (MW)	Qc (MVar)	Barra	Pc (MW)	Qc (MVar)
1	0,00000	0,00000	14	0,07355	0,05134	27	0,07355	0,05134
2	0,00000	0,00000	15	0,14710	0,07940	28	0,14710	0,07940
3	0,00000	0,00000	16	0,29420	0,15879	29	0,29420	0,15879
4	0,00000	0,00000	17	0,00000	0,00000	30	0,00000	0,00000
5	0,07355	0,05134	18	0,07355	0,05134	31	0,00000	0,00000
6	0,14710	0,07940	19	0,14710	0,07940	32	0,07355	0,05134
7	0,29420	0,15879	20	0,29420	0,15879	33	0,14710	0,07940
8	0,00000	0,00000	21	0,00000	0,00000	34	0,29420	0,15879
9	0,07355	0,05134	22	0,00000	0,00000	35	0,00000	0,00000
10	0,14710	0,07940	23	0,07355	0,05134	36	0,07355	0,05134
11	0,29420	0,15879	24	0,14710	0,07940	37	0,14710	0,07940
12	0,00000	0,00000	25	0,29420	0,15879	38	0,29420	0,15879
13	0,00000	0,00000	26	0,00000	0,00000			

Fonte: do autor

TABELA 35: DADOS DOS RAMOS DO SISTEMA ELÉTRICO SUBMETIDO À MINIMIZAÇÃO

De	Para	R (pu)	X (pu)	De	Para	R (pu)	X (pu)
1	2	0,122085	0,026858	2	21	0,000000	3,000000
2	3	0,000000	3,000000	21	22	2,077562	1,385041
3	4	2,077562	1,385041	22	23	48,822714	11,426592
4	5	48,822714	11,426592	22	24	15,581717	10,387811
4	6	15,581717	10,387811	22	25	7,790858	5,193905
4	7	7,790858	5,193905	21	26	2,077562	1,385041
3	8	2,077562	1,385041	26	27	48,822714	11,426592
8	9	48,822714	11,426592	26	28	15,581717	10,387811
8	10	15,581717	10,387811	26	29	7,790858	5,193905
8	11	7,790858	5,193905	2	30	0,000000	3,000000
2	12	0,000000	3,000000	30	31	2,077562	1,385041
12	13	2,077562	1,385041	31	32	48,822714	11,426592
13	14	48,822714	11,426592	31	33	15,581717	10,387811
13	15	15,581717	10,387811	31	34	7,790858	5,193905
13	16	7,790858	5,193905	30	35	2,077562	1,385041
12	17	2,077562	1,385041	35	36	48,822714	11,426592
17	18	48,822714	11,426592	35	37	15,581717	10,387811
17	19	15,581717	10,387811	35	38	7,790858	5,193905
17	20	7,790858	5,193905				

Fonte: do autor

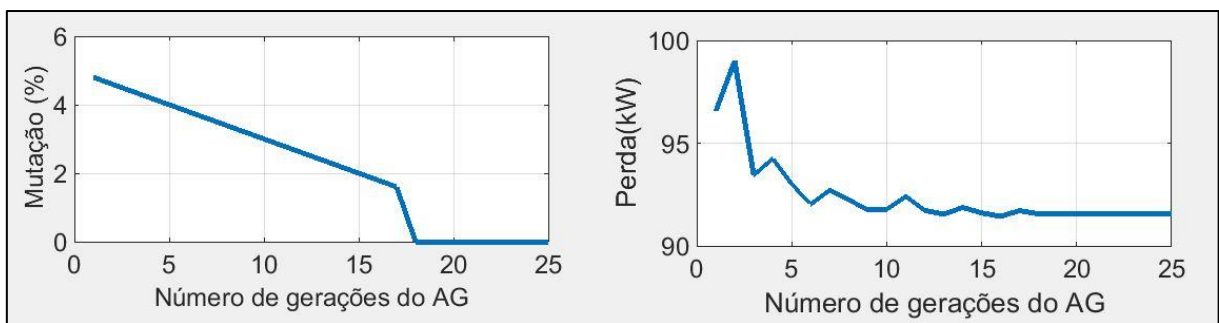
7.2 PARÂMETROS AJUSTADOS NO PROGRAMA PAGIND

Em todas as minimizações realizadas os ajustes implantados foram:

- Número de amostras = 100;
- Número de indivíduos = 220;
- Número de gerações = 25;
- Penalidade por infração de fator de potência, $K1=1,2$;
- Probabilidade de cruzamento (%) = 50;
- Probabilidade de mutação percentual = 5;
- Decremento de mutação percentual = 100.

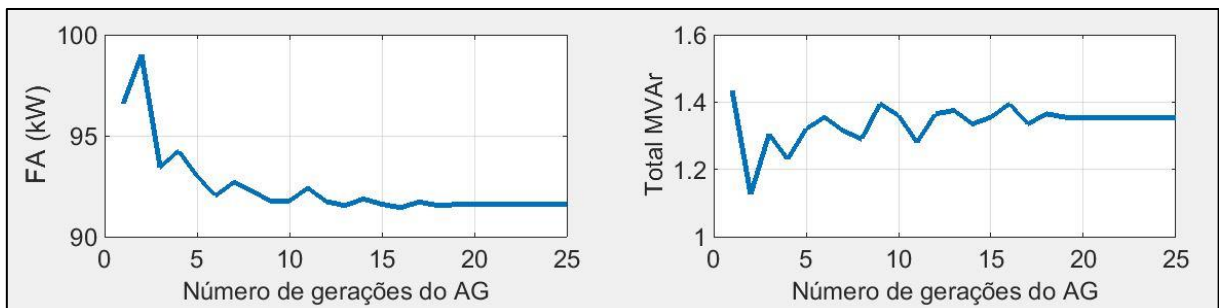
Um ciclo de funcionamento completo do PAGIND, com 25 gerações, pode ser caracterizado por suas principais variáveis, conforme conteúdo das Figuras 28 e 29; onde destaca-se que a mutação utilizada é tipo “adaptativa”.

FIGURA 28: CURVA DE MUTAÇÃO ADAPTATIVA E PERDAS ELÉTRICAS



Fonte: Do autor, gerada pelo programa *MATLAB*®

FIGURA 29: VALOR DE APTID O E POT NCIA TOTAL DE CAPACITORES



Fonte: Do autor, gerada pelo programa *MATLAB*®

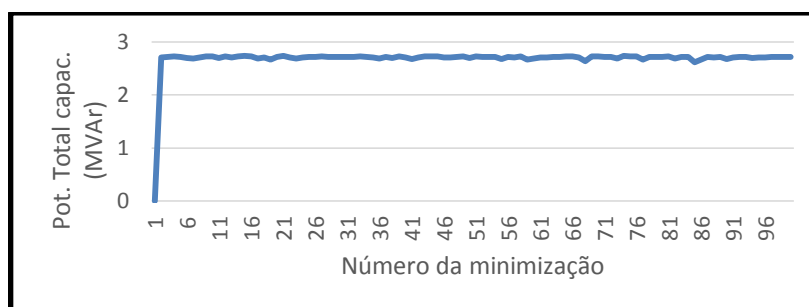
7.3 MINIMIZAÇÕES REALIZADAS À PLENA CARGA

Neste item o programa desenvolvido (PAGIND) foi utilizado para fazer 100 minimizações, estando inibidas, em todas elas, instalação de capacitores na barra número 1. Na primeira minimização todos os capacitores foram forçados para zero e, a partir da segunda minimização o programa funcionou livremente, sem nenhuma variável forçada, realizando as iterações em busca do melhor arranjo de capacitores. Cada minimização, com 220 indivíduos e 25 iterações, demandou, aproximadamente, 130 segundos.

7.3.1 Resultados das minimizações, carregamento nominal

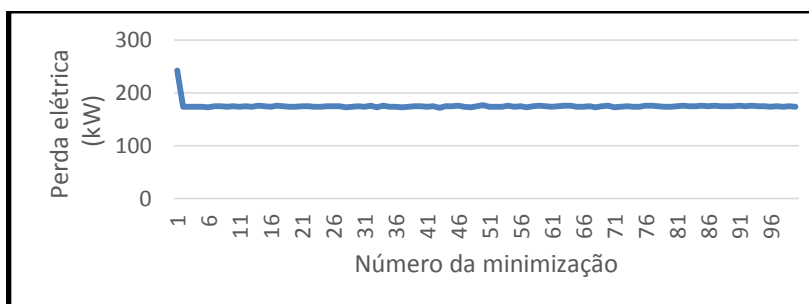
Nas Figuras 30 e 31 estão apresentados os resultados produzidos ao longo das minimizações realizadas, destacando-se, novamente, que no primeiro evento a potência de todos os capacitores, de todas as barras, é forçada para zero.

FIGURA 30: VALOR TOTAL DA POTÊNCIA CAPACITIVA CALCULADA À CADA MINIMIZAÇÃO



Fonte: Do autor

FIGURA 31: PERDA TOTAL REMANESCENTE APÓS CADA MINIMIZAÇÃO



Fonte: Do autor

Obstante a realização de batelada com 100 minimizações, por racionalidade a Tabela 36 registra somente os resultados do primeiro e segundo evento. Nela, as colunas identificadas como “Cap.” correspondem ao valor da potência reativa capacitiva calculada pelo programa, em MVar, e inserida em cada barra; e as colunas identificadas como “Tens.” contêm os valores operativos de tensão, nas barras, em p.u. Na penúltima linha, tanto para a primeira minimização quanto para a segunda minimização, são mostrados os valores médios das tensões dos barramentos e o FP’s da barra-1; e na última linha estão lançados os valores correspondes às perdas totais remanescentes nos ramos elétricos.

TABELA 36: RESULTADOS PRODUZIDOS PELO PAGIND SEM CAPACITORES E COM CAPACITORES

Fluxo de carga sem capacitores						Otimização, com capacitores					
Barra	Cap. (MVar)	Tens. (p.u.)	Barra	Cap. (MVar)	Tens. (p.u.)	Barra	Cap. (MVar)	Tens. (p.u.)	Barra	Cap. (MVar)	Tens. (p.u.)
1	0,000	1,000	21	0,000	0,977	1	0,000	1	21	0,120	0,998
2	0,000	0,997	22	0,000	0,961	2	0,035	0,997	22	0,000	0,986
3	0,000	0,977	23	0,000	0,916	3	0,050	0,997	23	0,050	0,947
4	0,000	0,961	24	0,000	0,928	4	0,120	0,986	24	0,120	0,966
5	0,000	0,916	25	0,000	0,928	5	0,060	0,949	25	0,100	0,959
6	0,000	0,928	26	0,000	0,961	6	0,060	0,96	26	0,050	0,987
7	0,000	0,928	27	0,000	0,916	7	0,070	0,957	27	0,050	0,948
8	0,000	0,961	28	0,000	0,928	8	0,060	0,986	28	0,070	0,961
9	0,000	0,916	29	0,000	0,928	9	0,080	0,951	29	0,140	0,961
10	0,000	0,928	30	0,000	0,977	10	0,060	0,960	30	0,120	0,999
11	0,000	0,928	31	0,000	0,961	11	0,120	0,960	31	0,040	0,988
12	0,000	0,977	32	0,000	0,916	12	0,030	0,994	32	0,060	0,951
13	0,000	0,961	33	0,000	0,928	13	0,005	0,981	33	0,070	0,963
14	0,000	0,916	34	0,000	0,928	14	0,050	0,942	34	0,140	0,963
15	0,000	0,928	35	0,000	0,961	15	0,090	0,958	35	0,090	0,988
16	0,000	0,928	36	0,000	0,916	16	0,100	0,954	36	0,050	0,949
17	0,000	0,961	37	0,000	0,928	17	0,045	0,982	37	0,050	0,961
18	0,000	0,916	38	0,000	0,928	18	0,070	0,945	38	0,120	0,962
19	0,000	0,928				19	0,050	0,954			
20	0,000	0,928				20	0,120	0,955			
Média Tensão=0,942 p.u; FP barra1=0,854						Média Tensão=0,969 p.u; FP barra1=1,000					
Perda total 242,27 + j*333,51						Perda total 174,22 + j*239,59					

Fonte: Do autor

7.3.2 Estatísticas relativas às otimizações com carregamento nominal

Os resultados produzidos pelo programa desenvolvido (PAGIND), e inseridos nas Figuras 30 e 31, e Tabela 36, permitem a seguinte síntese:

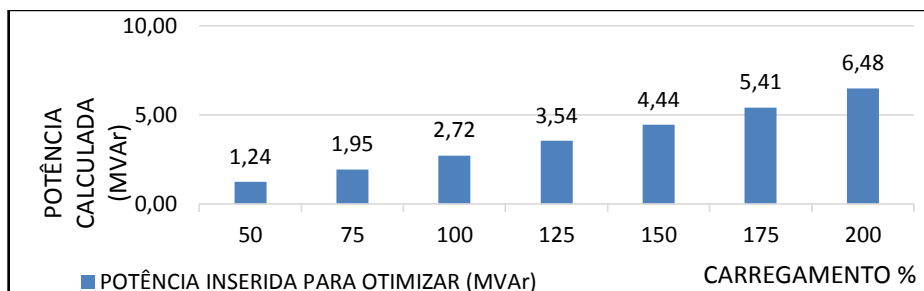
- Os capacitores calculados promoveram redução de 28 % na perda com energia ativa;
- Os capacitores calculados promoveram restauração de 2,8 % na média da tensão de todas as barras elétricas;
- Foi perfeitamente observada a restrição de Fator de potência na barra elétrica número 1, que sai de 0,854 para 1,000;
- Foi perfeitamente observada a seleção de não instalação de capacitor na barra elétrica número 1;
- Para algumas barras elétricas o programa determinou a instalação de potência nula, o que equivale a não instalar nenhum capacitor na barra.

7.4 VARIANDO O CARREGAMENTO

Neste item, o mesmo sistema elétrico, já caracterizado anteriormente neste capítulo e mostrado nas Figuras 26 e 27, foi submetido a variação de carga, com excursão em 7 degraus: 50, 75, 100, 125, 150, 175 e 200 %. Para cada carregamento foram realizadas 100 minimizações, e cada um dos valores que estão inseridos nas Figuras 32 e 33 são a média aritmética dos resultados obtidos nas minimizações; excluindo a primeira, visto que ela é executada com todas as potências dos capacitores forçadas para valor nulo.

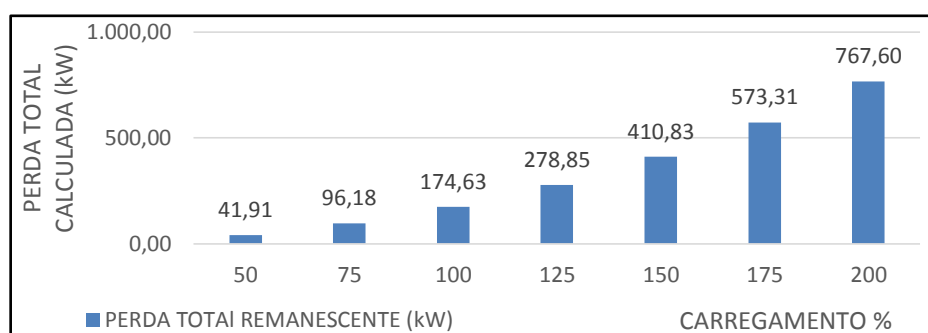
Conforme esperado, a evolução da potência total dos capacitores calculados traça, praticamente, uma linha reta; enquanto a evolução das perdas segue uma relação quadrática, e o formato destas duas progressões serve para confirmar o perfeito funcionamento do programa desenvolvido; tanto no que tange à sua capacidade de otimização/minimização de perdas elétricas de circuitos elétricos na topologia industrial, tanto quanto à sua capacidade de obedecer aos fundamentos que regem o funcionamento de circuitos elétricos.

FIGURA 32: VALOR TOTAL DA POTÊNCIA CAPACITIVA CALCULADA COM DIFERENTES CARREGAMENTOS



Fonte: Do autor

FIGURA 33: PERDA ELÉTRICA (kW) CALCULADA COM DIFERENTES CARREGAMENTOS



Fonte: Do autor

7.5 PERFORMANCE DO PROGRAMA DESENVOLVIDO

Todas as minimizações realizadas pelo PAGIND, e reportadas ao longo deste capítulo, serviram para estabelecer a performance do programa desenvolvido, cuja caracterização está sintetizada nos itens que se seguem.

7.5.1 Pontos Positivos do programa desenvolvido

- Excelente precisão no cálculo de perdas;
- Boa repetibilidade no cálculo da potência total do arranjo de capacitores;
- Imediata aplicabilidade, determinando somente capacitores comerciais;
- Eficiente em impor restrição para o FP da barra de conexão com a concessionária de energia;
- Fácil programação das barras onde não deverão ser instalados capacitores;

- Total versatilidade, sendo capaz de avaliar sistemas elétricos industriais em diferentes níveis de carregamento.

7.5.2 Limite do programa desenvolvido

Apesar dos bons resultados relacionados anteriormente, o programa desenvolvido tem, como limitação, o fato de não conseguir impor 100 % de repetibilidade no valor do capacitor calculado para cada barra.

Este limite prende-se à característica da própria estrutura do Algoritmo Genético, que opera guiado por um único valor, não conseguindo, desta forma, particularizar o conteúdo de cada ramo elétrico.

8 CONCLUSÕES

8 CONCLUSÕES

Conforme estabelecido no capítulo 1, o objetivo desta dissertação era desenvolver programa específico, estruturado como Algoritmo Genético, capaz de calcular os capacitores *shunt* (potência e locais de implantação) necessários para reduzir perdas elétricas incidentes em sistemas elétricos industriais.

Como conclusão podemos estabelecer que, conforme conteúdo dos capítulos 2 até 6, efetivamente foi desenvolvido o programa proposto, estruturado como Algoritmo Genético, e editado com instruções *MATLAB®*, cujas estruturas, inclusive a especialista em cálculo de fluxo de carga, foram avaliadas e qualificadas.

Como indispensável observação, o capítulo 5 deixa evidente que quando se trata de expressões numéricas ricas em descontinuidades, a otimização através de Algoritmo Genético requer cuidado especial no que tange à determinação dos parâmetros utilizados, em especial o relativo ao tamanho da população; e que a não observância deste detalhe pode provocar resultados inconsistentes e não confiáveis. Contudo, o mesmo capítulo 5 deixa evidente que quando se trata de funções monomodais; o que é o caso da expressão que calcula o FA do programa desenvolvido, a otimização funciona perfeitamente, com excelente exatidão, mesmo com tamanho convencional de populações (220 indivíduos).

Ainda como conclusão, e conforme conteúdo do capítulo 7, o programa desenvolvido foi efetivamente e exaustivamente aplicado, mostrando-se preciso e estável, sem restrição para aplicação em estudos de sistema elétricos na topologia industrial, demonstrando, subsidiariamente, que o Algoritmo Genético é uma ferramenta muito precisa e confiável quando devidamente condicionado.

Nas otimizações e simulações realizadas, sempre se mostrou bastante eficaz utilizar cruzamento tipo simples (*one point crossover*), envolvendo somente 50 % da população, aliando esta opção à mutação sempre tipo adaptativa, com valor inicial sempre bem alto, e igual a 5 %. O número de iterações/gerações foi sempre mantido igual a 25, e a quantidade de 220 indivíduos produziu sempre, para funções monomodais, resultados suficientemente precisos para aplicação em estudos elétricos.

Especificamente, quando aplicado ao sistema elétrico de 38 barras com carregamento de 100 %, o programa promoveu redução de 28 % na perda de energia

e restaurou, em média, 2.8 % o nível de tensão dos barramentos. O programa, quando submetido a variação de carga, de 50 a 200 %, com intervalos de 25 %, não perdeu precisão e manteve plena funcionalidade, mostrando repetibilidade e coerência com as leis que regem os circuitos elétricos.

Adicionalmente, as otimizações realizadas mostraram que o programa especialista em cálculo de fluxo de carga, usualmente utilizado para cálculo de redes elétricas (transmissão & distribuição), também se aplica, sem perda de performance, e sem nenhuma restrição, para cálculos de circuitos elétricos na topologia industrial.

Em centenas de otimizações realizadas ao longo do desenvolvimento desta dissertação foi observado que, quando otimizando funções multimodais, o algoritmo genético converge, erroneamente, de forma eventual, para um dentre os dois picos mais próximos ao ponto efetivamente ótimo; observando-se maior incidência para o mais próximo. Esta observação abre espaço para um trabalho futuro, correspondente ao desenvolvimento de um algoritmo genético modificado, que seja auto ajustável, e que consiga evitar as armadilhas representadas pelos “falsos pontos ótimos”, e parece ser factível a ideia de um algoritmo que execute, sempre, duas otimizações, a primeira excursionando na faixa de busca original, determinando uma primeira resposta, que então seria utilizada para definir uma nova faixa de busca, extremamente menor que a anterior, sendo esta nova e diminuta faixa de busca utilizada para fazer a segunda e última minimização, chegando-se então à resposta final, presumivelmente sempre exata.

REFERÊNCIAS

REFERÊNCIAS

- ABB. **Low Voltage Capacitors | Sizing Capacitors 19 ISXP981001D020**. [S.l.]. 2011.
- ABUL'WAFA, A. R. Optimal capacitor allocation in radial distribution systems for loss reduction: A two stage method. **Electric Power Systems Research 95 (2013) Electric Power Systems Research**, 2013. 168-174,.
- CHING-TZONG SU, C.-S. L. Feeder reconfiguration and capacitor setting for loss reduction of distribution systems, 2000. 97-102.
- DAS, D. Reative power compensation for radial distribution networks using genetic algoritm. **Electrical Power and Enegy Systems**, 7 april 1999. 573-581.
- IEEE. **IEEE recommended practice for energy management in industrial and commercial facilities (IEEE std 739-1995**. [S.l.]: IEEE, 1995.
- KHALIL BANAN, M. B. B. S. J. M. Induction Motor Efficiency Estimation using Genetic Algorithm. **International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering Vol:1, No:3, 2007**, 2007.
- KISEOK SUNG, J. A. M. G. H. B. **A Genetic Algorithm to Optimise Signal Phasing in Area Traffic Control**. [S.l.], p. 404-408. 1997.
- KOZA, J. R. **GENETIC PROGRAMMING: A PARADIGM FOR GENETICALLY BREEDING POPULATIONS OF COMPUTER PROGRAMS TO**. Computer Science Department Stanford University Stanford, CA 94305. [S.l.], p. 1-131. 1990.
- MATTEO ANEDDA, A. M. M. **64-APSK Constellation and Mapping Optimization for Satellite Broadcasting using Genetic Algorithms**. University of Cagliari, Piazza d'Armi, 09123, Cagliari, Italy. [S.l.], p. 1-9. 2015.
- MICHALEWICZ, Z. **Genetic Alorthms + Data Structures = Evolution Programs**. Third, revised and extended edition. ed. New York: Springer, v. 1, 1996.
- MOHAMMAD H. MORADI, A. Z. . Y. M. . M. A. An efficient hybrid method for solving the optimal sitting and sizing problem of DG and shunt capacitor banks simultaneously based. **Electrical Power and Energy Systems 54 (2014) 101–111 journal homepage: www.elsevier.com/locate/ijepes**, 2014. 101-111.
- MONTICELLI, A. J. **Fluxo de carga em redes de energia elétrica**. São Paulo: Edgard Blucher, 1983.
- MUSIRIN, W. M. D. . H. M. . R. A. . A. H. A. B. . I. Simultaneous Network Reconfiguration and DG Sizing Using Evolutionary Programming and Genetic

- Algorithm. **RESEARCH ARTICLE - ELECTRICAL ENGINEERING**, Arab J Sci Eng (2014) 39:6327–6338 DOI 10.1007/s13369-014-1299-9 2014. 6328/6338.
- NORAINI MOHD RAZALI, J. G. **Genetic Algorithm performance with different selection strategies in solving TSP**. Proceedings of the World Congress on Engineering 2011 Vol II. London: [s.n.]. 2011.
- PEDRO LOPEZ GARCIA, E. O. E. O. A. D. M. A. P. **A Hybrid Method for Short-Term Traffic Congestion Forecasting Using Genetic Algorithms and Cross Entropy**. IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 17, NO. 2, FEBRUARY 2016. [S.I.]: [s.n.]. 2016. p. 557-569.
- Q. A. SALIH, D. M. S. A. I. S. S. **Optimal Distributed Generators Location For Power Losses Improvement Using Sensitivity Based Method**. IEEE International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA). Putrajaya, Malaysia: [s.n.]. 2015. p. 4.
- S.N.DEEPA, S. N. S. . **Introduction to Genetic Algorithms**. Berlin: Springer (ISBN 978-3-540-73189-4 Springer Berlin), 2008.
- SCHOENAUERY, Z. M. A. M. **Evolutionary Algorithms for Constrained Parameter Optimization Problems**. [S.I.], p. 37. 1996.
- TANOMARU, J. **Motivação, Fundamentos e Aplicações de Algoritmos Genéticos**. II CONGRESSO BRASILEIRO DE REDES NEURAIIS III ESCOLA DE REDES NEURAIIS. Curitiba. 1995.
- THOMAS BACK, D. B. F. A. Z. M. **Evolutionary Computation 1 Basic Algorithms and operators**. New York: Taylor & Francis, v. ISBN 10:0-7503-0664-5, 2000.

APÊNDICE 1

APÊNDICE 1

SCRIPT “A_iniciar”

```
% 1-Este script contém todas as etapas necessárias para
executar o Algoritmo
% Genético que faz minimização de perdas elétricas incidentes
% em plantas industriais.
% 2-A minimização é feita através da inserção de bancos fixos
de
% capacitores, em paralelo com as barras elétricas,
% 3-Os capacitores são representados por números binários, que
fazem
% indexação a uma tabela comercial, com valores em MVAR.
% 4-Fator de carga da instalação: Variável através de
parâmetro.
%%
%%
clc;                clear all;                format long;    tic;
%%
%%
% Principais parametros
N_amostras=1;        % Determina o numero de amostras do
loop estatístico
Perfil_carga=1.0;    % [0.5 a 2], determina intensidade de
carregamento
N_indiv=220;         % [sem limite], determina numero de
indivíduos
N_geracoes=25;      % [sem limite], numero de iterações do
loop do AG

%%
%%
%%
% Parametros complementares
K1=1.2;              % [1 a 5], penalidade para desvio de
FP_bus1_indiv
Pc_perc=50;          % [0 a 100], quantidade perc.
acasalamentos na população,
Pm_perc=5;           % [0 a 5], quantidade perc. mutação,
Tx_Dec_mut_perc=50;  % [0 a 100], veloc. decaimento da
mutação, em %
Ativa_mut_adap=1;    % [0 ou 1], 1=ativa a mutação
adaptativa
Ativa_perda_natural=1 % [0 ou 1], 1=força todos indivíduos
para 0 MVar
%%
%%
```

```

% Loop para Controle Estatístico
%%
    Predispondo Variaveis
FA_search=zeros(N_geracoes,1);
Perdas_search=zeros(N_geracoes,1);
Tensao_search=zeros(N_geracoes,1);
Genes_a_mutar_perc_search=zeros(N_geracoes,1);
Soma_MVAR_search=zeros(N_geracoes,1);
Penal_FP_search=zeros(N_geracoes,1);
spc=111*ones(N_amostras,1);

%1-A          Abrindo dados gerais
[Gerais_dados]=B_dados_gerais;

%1-B          Chamando função com dados específicos do
Sist. eletrico
[barras,ramos,N_barras,dados_prim_pop,N_ramos,Fx_busca]=C_dados_20b(N_indiv,Perfil_carga);

% Formando sinais do controle estatístico
FA_amostr=zeros(N_amostras,1);
Delta_perda_KW_amostr=zeros(N_amostras,1);
Tens_amostr=zeros(N_amostras,N_barras);
FP_amostr=zeros(N_amostras,N_barras);
Cap_amostr=zeros(N_amostras,N_barras);
Soma_Cap_amostr=zeros(N_amostras,1);
Perd_KW_ram_amostr=zeros(N_amostras,N_ramos);
Perd_KVAR_ram_amostr=zeros(N_amostras,N_ramos);

for Sample=1:N_amostras; % LOOP PARA ESTUDOS ESTATÍSTICOS
%%
%%
%2-          Chamando função que gera a primeira população
[PI_bin]=D_pi_bin(dados_prim_pop);
%%
%%
%3-          Chamando o Loop que executa o AG ( Algoritmo
Genético )
for search=1:N_geracoes;
%%
%%
%3-1          Chamando a função que converte a população
inicial, em
% binários para valores, em MVAR
[PI_MVAR]=E_conv_pi_bin_Mvar(PI_bin,Fx_busca,barras);
%%
%%
%3-2          Chama função que adapta valores, em MVAR, para
valores de
% tabela comercial

```

```

[PI_MVAR_comercial,PI_MVAR_comer_col]=F_tab_comercial(PI_MVAR)
;

if Ativa_perda_natural==1 & Sample==1 % Estabelece as perdas naturais
    PI_MVAR_comercial=zeros(N_barras,N_indiv);
    PI_MVAR_comer_col=transpose(PI_MVAR_comercial);
end
% Fim do estudo das perdas naturais

%%
%%
%3-3:          Chama função que gera o fluxo de carga %%
[iter,V_bus_indiv,pkm_indiv,qkm_indiv,Perdas_KW,Perdas_KVAR,Perdas_KW_ramos,Perdas_KVAR_ramos,Ang_deg]=G_fc_mestre(barras,ramos,PI_MVAR_comercial,N_indiv,N_barras,N_ramos);

%%
%%
%3-4:          Chama função que gerar o FA (valor de aptidão)
%%
[Func_adap]=I_aptidao(Perdas_KW,Gerais_dados,PI_MVAR_comercial);
%%
%%
%3-5:          Espaço reserva
%%
%%
%%
% 3-6:          Chama função Penalidade por FP_bus1_indiv capacitivo %%
[Func_adap_penal_FP,delta_FP,FP_bus1_indiv]=J_penal_FP(Func_adap,K1,pkm_indiv,qkm_indiv,PI_MVAR_comercial,barras,V_bus_indiv);
%%
%%
%3-7:          Mudanças de nomes, face a não utilização da normalização %%
Func_adap_penal_norm=Func_adap_penal_FP;
%%
%%
%3-8:          Chama Selecao tipo games
%%
[Pop_selecionada,delta_dec,]=K_selec(Func_adap_penal_norm,PI_bin);
%%
%%
%3-9:          Chama Funcao Cruzamento
%%

```

```

[Pop_filho,desvio_36]=L_cruz(Pc_perc,Pop_selecionada);
%3-10:      Fazendo troca de nomenclaturas
%%
Pop_bal=Pop_filho;
%%
%%
%3-11:      Chama Função Mutação
[Pop_com_mutacao,Genes_prog_perc]=M_mut(Pop_bal,search,Tx_Dec_
mut_perc,Pm_perc,N_geracoes,Ativa_mut_adap);
%%
%%
%3-12:      Melhor população
%%
PI_bin=Pop_com_mutacao;
%%
%%
%3-13:      Formando sinais de saída
Genes_a_mutar_perc_search(search,1)=Genes_prog_perc;
Soma_MVAR_search(search,1)=sum(PI_MVAR_comer_col(N_indiv,:),2)
;
Penal_FP_search(search,1)=delta_FP(N_indiv,1);
FA_search(search,1)=Func_adap(N_indiv,1);
Perdas_search(search,1)=Perdas_KW(N_indiv,1);
Tensao_search=V_bus_indiv(N_indiv,:);
FP_search=FP_bus1_indiv(N_indiv,1);
Cap_search=PI_MVAR_comer_col(N_indiv,:);
Perd_KW_ramos_search=Perdas_KW_ramos(N_indiv,:);
Perd_KVAR_ramos_search=Perdas_KVAR_ramos(N_indiv,:);
%  ENCERRA O LOOP DO ALGORITMO GENETICO-----
-----
end

% Preparando exportação de resultados
FA_amostr(Sample,1)=FA_search(N_geracoes,1);
Tens_amostr(Sample,:)=Tensao_search;
FP_amostr(Sample,1)=FP_search;
Cap_amostr(Sample,:)=Cap_search;
Soma_Cap_amostr=sum(Cap_amostr,2);
Perd_KW_ram_amostr(Sample,:)=Perd_KW_ramos_search;
Perd_KVAR_ram_amostr(Sample,:)=Perd_KVAR_ramos_search;
Soma_perd_Kw_amostr=sum(Perd_KW_ram_amostr,2);
Soma_perd_Kvar_amostr=sum(Perd_KVAR_ram_amostr,2);
Media_Tens_amostr=mean(Tens_amostr,2)
FP_buss_um=FP_amostr(:,1)
% %  ENCERRA O LOOP DE ESTUDOS ESTATISTICOS-----
-----
% %%
%
end

```

[illegible]

FUNÇÃO “B_dados_gerais”

```
% Funcao dados gerais
% Esta função serve como banco de dados (custos, quantidades,
outros), para
% todas as demais funções do Algoritmo genético que faz
minimização de
% perdas em ambientes industriais.
% Para facilitar a propagação dos dados, todos eles são
"empacotados" em
% um unico vetor, que é disponibilizado na saída da função
```

```
function [Gerais_dados]=B_dados_gerais;
Impostos=1.3;
D_atv_pt_R_KW=32.96;
D_reat_fpt_R_KW=11.12;
E_reat_pt_R_KW=0.31011;
UFDR_fpt_R_KW=11.12;
Anos_inv=25;
Hr_mes_fpt=660;
FP_ref=0.92;
KVAR_15KV_reais=(2400/40)*Impostos;
KVAR_BT_reais=(1100/60)*Impostos;
KVAR_medio_Reais=(KVAR_15KV_reais+KVAR_BT_reais)/2;
C_aquis_KVAR_Reais=KVAR_medio_Reais;
C_rep_KVAR_Reais=0.2*C_aquis_KVAR_Reais;
D_atv_fpt_R_KW=11.12;
E_atv_pt_R_KW=0.31011;
E_reat_fpt_R_KW=0.20289;
UFER_pt_R_KW=0.31011;
Mes_ano=12;
FP_ref_pt=0.92;
FP_nat=0.86;
D_reat_pt_R_KW=32.96;
E_atv_fpt_R_KW=0.20289;
UFDR_pt_R_KW=32.96;
UFER_fpt_R_KW=0.20289;
Hr_mes_pt=60;
FP_ref_fpt=0.92;
Tx_rep_anual=0.03;

% Saída ( Vetor "Dados_FO_FA", contendo todos os dados
Gerais_dados=[Impostos;
D_atv_pt_R_KW;
D_reat_fpt_R_KW;
E_reat_pt_R_KW;
UFDR_fpt_R_KW;
Anos_inv;
Hr_mes_fpt;
```

[illegible]

FUNÇÃO “C_dados_38b”

```

% Funcao com dados do sistema eletrico
% 1- Os dados do sistema eletrico inseridos dentro desta
função
% são os convencionalmente demandados para calculo de fluxo de
potencia,
% 2- A função limita o parametro "Perfil_carga", [0.5 a 2] p.u,
controlando
% a real intensidade das cargas,
% 3- A função cria tabela, com 31 valores, todos vinculados à
potencia
% da maior carga,
% A potencia base considerada é de 100 MVA
function
[barras,ramos,N_barras,dados_prim_pop,N_ramos,Fx_busca]=C_dado
s_20b(N_indiv,Perfil_carga);
% Limitando faixa para o parametro "Perfil_carga"
if Perfil_carga<=0.5; K=0.5;
elseif Perfil_carga>=2; K=2;
elseif Perfil_carga>0.5 & Perfil_carga<2; K=Perfil_carga;
end
% Dados do circuito eletrico
% Número Tipo V(pu) Ângulo(g) Pg(MW) Qg(MVar) Pc(MW)
Qc(MVar) gshk(MW) barras selecionadas
barras = [
    131.00  0.001.00  1.00  0  0  0.00  0
    211.00  0.000.00  0.00  0  0  0.00  1
    311.00  0.000.00  0.000  0  0.00  1
    411.00  0.000.00  0.000  0  0.00  1
    511.00  0.000.00  0.000.07355*K  0.05134*K  0.00  1
    611.00  0.000.00  0.000.14710*K  0.07940*K  0.00  1
    711.00  0.000.00  0.000.29420*K  0.15879*K  0.00  1
    811.00  0.000.00  0.000  0  0.00  1
    911.00  0.000.00  0.000.07355*K  0.05134*K  0.00  1
    1011.00  0.000.00  0.000.14710*K  0.07940*K  0.00  1
    1111.00  0.000.00  0.000.29420*K  0.15879*K  0.00  1
    1211.00  0.000.00  0.000  0  0.00  1
    1311.00  0.000.00  0.000  0  0.00  1
    1411.00  0.000.00  0.000.07355*K  0.05134*K  0.00  1
    1511.00  0.000.00  0.000.14710*K  0.07940*K  0.00  1
    1611.00  0.000.00  0.000.29420*K  0.15879*K  0.00  1
    1711.00  0.000.00  0.000  0  0.00  1
    1811.00  0.000.00  0.000.07355*K  0.05134*K  0.00  1
    1911.00  0.000.00  0.000.14710*K  0.07940*K  0.00  1
    2011.00  0.000.00  0.000.29420*K  0.15879*K  0.00  1
    2111.00  0.000.00  0.000  0  0.00  1
    2211.00  0.000.00  0.000  0  0.00  1
    2311.00  0.000.00  0.000.07355*K  0.05134*K  0.00  1

```

2411.00	0.000.00	0.000.14710*K	0.07940*K	0.00	1
2511.00	0.000.00	0.000.29420*K	0.15879*K	0.00	1
2611.00	0.000.00	0.000	0	0.00	1
2711.00	0.000.00	0.000.07355*K	0.05134*K	0.00	1
2811.00	0.000.00	0.000.14710*K	0.07940*K	0.00	1
2911.00	0.000.00	0.000.29420*K	0.15879*K	0.00	1
3011.00	0.000.00	0.000	0	0.00	1
3111.00	0.000.00	0.000	0	0.00	1
3211.00	0.000.00	0.000.07355*K	0.05134*K	0.00	1
3311.00	0.000.00	0.000.14710*K	0.07940*K	0.00	1
3411.00	0.000.00	0.000.29420*K	0.15879*K	0.00	1
3511.00	0.000.00	0.000	0	0.00	1
3611.00	0.000.00	0.000.07355*K	0.05134*K	0.00	1
3711.00	0.000.00	0.000.14710*K	0.07940*K	0.00	1
3811.00	0.000.00	0.000.29420*K	0.15879*K	0.00	1];

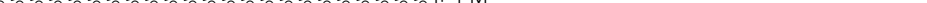
% %Dados dos ramos;

%De	Para	r(pu)	x(pu)	bchr(pu)	tap	phase(g)
spare						
ramos = [						
12	0.122085696/2			0.026858853/2	0.00000	0.000000
0.000000						
23	0.0			3	0.00000	0.000000
0.000000						
34	2.077562327			1.385041551	0.00000	0.000000
0.000000						
45	48.822714680			11.426592800	0.00000	0.000000
0.000000						
46	15.581717450			10.387811630	0.00000	0.000000
0.000000						
47	7.790858726			5.193905817	0.00000	0.000000
0.000000						
38	2.077562327			1.385041551	0.00000	0.000000
0.000000						
89	48.822714680			11.426592800	0.00000	0.000000
0.000000						
810	15.581717450			10.387811630	0.00000	0.000000
0.000000						
811	7.790858726			5.193905817	0.00000	0.000000
0.000000						
212	0.0			3	0.00000	0.000000
0.000000						
1213	2.077562327			1.385041551	0.00000	0.000000
0.000000						
1314	48.822714680			11.426592800	0.00000	0.000000
0.000000						
1315	15.581717450			10.387811630	0.00000	0.000000
0.000000						
1316	7.790858726			5.193905817	0.00000	0.000000
0.000000						

12172.077562327	1.385041551	0.00000	0.000000
0.000000			
1718 48.822714680	11.426592800	0.00000	0.000000
0.000000			
1719 15.581717450	10.387811630	0.00000	0.000000
0.000000			
17207.790858726	5.193905817	0.00000	0.000000
0.000000			
221 0.0	3	0.00000	0.000000
0.000000			
2122 2.077562327	1.385041551	0.00000	0.000000
0.000000			
2223 48.822714680	11.426592800	0.00000	0.000000
0.000000			
2224 15.581717450	10.387811630	0.00000	0.000000
0.000000			
2225 7.790858726	5.193905817	0.00000	0.000000
0.000000			
21262.077562327	1.385041551	0.00000	0.000000
0.000000			
2627 48.822714680	11.426592800	0.00000	0.000000
0.000000			
2628 15.581717450	10.387811630	0.00000	0.000000
0.000000			
26297.790858726	5.193905817	0.00000	0.000000
0.000000			
230 0.0	3	0.00000	0.000000
0.000000			
3031 2.077562327	1.385041551	0.00000	0.000000
0.000000			
3132 48.822714680	11.426592800	0.00000	0.000000
0.000000			
3133 15.581717450	10.387811630	0.00000	0.000000
0.000000			
3134 7.790858726	5.193905817	0.00000	0.000000
0.000000			
30352.077562327	1.385041551	0.00000	0.000000
0.000000			
3536 48.822714680	11.426592800	0.00000	0.000000
0.000000			
3537 15.581717450	10.387811630	0.00000	0.000000
0.000000			
35387.790858726	5.193905817	0.00000	0.000000
0.000000];			

```
% Calculo das tabelas
range=max(barras(:,8))-0;
step=range/31;
```

```
Fx_busca=[
```



[illegible]

FUNÇÃO “E_conv_pi_bin_MVAR”

```

% Função MATLAB que converte população binaria para seu
equivalente,
% representado por numeros reais, com unidade em MVAR,
% 1- Converte a população binaria, criada na função
"D_pi_bin", para
% população equivalente, em MVAR,
% 2- A conversão é feita com base em uma tabela, com 31
valores, todos
% eles dependentes da maior carga e do carregamento do sistema
eletrico,
% 3- O maior valor da tabela é proporcional ao maior valor de
% carregamento presente em qualquer barra do sistema eletrico,
% 4- Pelo exposto, a conversão da população binaria presente
na entrada da
% presente função produz, em sua saída, uma população, em
Mvar, sempre
% proporcional ao maior carregamento presente no sistema
eletrico,

function [PI_MVAR]=E_conv_pi_bin_Mvar(PI_bin,Fx_busca,barras);
% Preparando sinais
Igual_PI_bin=PI_bin;
[nlinha,nbit]=size(Igual_PI_bin);
ncel=nbit*nlinha/5;
Col_cel_PI_bin=zeros(ncel,5); % vetor coluna que ira receber
todos os valores
% existentes em todas as celulas.
pos=1;passo=5;
% Insere no vetor coluna os capacitores da população original,
% ainda em formato binario, facilitando a comparação com
% o codigo binario da tabela não comercial. A comparação é
efetuada linha à linha
for linha=1:nlinha;
    for celula=1:ncel/nlinha;
        pi=1+(celula-1)*passo;
        pf=5+(celula-1)*passo;
        Col_cel_PI_bin(pos,1:5)=Igual_PI_bin(linha,[pi:pf]);
        pos=pos+1;
    end
end
PIn_MVAR=zeros(ncel,1);
Bin_tabela=Fx_busca(:,[3:7]);
MVAR_tab=Fx_busca(:,[2]);
% Insere no vetor coluna os capacitores da pop. original,ja em
MVAR,
for pont_tab=1:31;
    for pont_L=1:ncel;

```


[illegible]

[illegible]

FUNÇÃO “H_FC_Escravo”

```

% Funcao cálculo Fluxo de Potência - Método de Newton Raphson
function
[V_bus,pperdas,qperdas,pkm,qkm,pcalc,qcalc,pnom,qnom,Ang_deg]=
H_fc_escravo(barras,ramos,PI_MVAR_Col);
format long;
tol=10.^-10;
itmax=40;
nome_da_rede='IEEE30';
%
=====
=====
% Parametros Gerais e Configuração
%
=====
=====
baseMVA      = 100;
graus_to_rad = pi/180;
rad_to_graus = 180/pi;
IdReadTD     = 0; % 1: Com transformadores desfasadores na
leitura
                % 0: Sim transformadores desfasadores na
leitura
IdReadCSB    = 0; % 1: Com condutancia shunt de barra na leitura
                % 0: Sim condutancia shunt de barra na leitura
IdOutBusD    = 0; % Relatorio de dados de barra
IdOutBraD    = 0; % Relatorio de dados de ramos
IdFlatStar   = 0; % Definição do Estado Inicial da Rede
IdOutJac     = 1; % Relatorio da Jacobiana
IdOutYGB     = 0; % Matriz de admitancia
%
=====
=====
% Interface
%
=====
=====
fprintf('\n> -----\n')
fprintf('> Loadflow \n')
fprintf('> Fluxo de carga - método de Newton\n')
fprintf('> -----\n')
%
=====
=====
% Leitura dos Dados do Sistema
%
=====
=====

```

```

% eval(NameFile);
fprintf('\n> Processando dados da rede ...\n')
%
=====
=====
% Dimensões dos Vetores
%
=====
=====
[nb, colunas] = size(barras);
[nr, colunas] = size(ramos);
%
=====
=====
% Dados de Barra
%
=====
=====
for k = 1 : nb,
    numext(k) = barras(k,1);           % Número da barra
    tipo(k)   = barras(k,2);           % Tipo da barra
    v(k)      = barras(k,3);           % Tensão da barra
    ang(k)    = barras(k,4)*graus_to_rad; % Angulo da barra
    pg(k)     = barras(k,5)/baseMVA;    % Potência ativa
gerada
    qg(k)     = barras(k,6)/baseMVA;    % Potência reativa
gerada
    pc(k)     = barras(k,7)/baseMVA;    % Potência ativa da
carga
    qc(k)     = barras(k,8)/baseMVA;    % Potência reativa
da carga

%
=====
=====
%
=====
=====
% Modificação feita pelo PT
Bsh_b_pu=PI_MVAR_Col/baseMVA;
%%%%%%%%%%%%%%
%
=====
=====
%
=====
=====
if IdReadCSB == 1,

```

```

        gshk(k) = barras(k,9)/baseMVA;           % Condutancia shunt
da barra
%        bshk(k) = barras(k,10)/baseMVA;        % Suceptancia
shunt da barra
        else
            gshk(k) = 0.0;
            bshk(k) = barras(k,9)/baseMVA;
        end
        pnom(k)    = pg(k) - pc(k);              % Potência ativa
especificada
        qnom(k)    = qg(k) - qc(k);              % Potência reativa
especificada
        numint(barras(k,1)) = k;                 % Posição da barra
no vetor
    end
%
=====
=====
% Dados dos Circuitos
%
=====
=====
for l = 1 : nr,
    de(l)    = numint(ramos(l,1)); % Barra inicial
    para(l)  = numint(ramos(l,2)); % Barra final
    r(l)     = ramos(l,3);         % Resistencia
    x(l)     = ramos(l,4);         % Indutancia
    bshl(l)  = ramos(l,5);         % Shunt da linha
    tap(l)   = ramos(l,6);         % Tap do transformador
    if tap(l) == 0,
        tap(l) = 1;
    end
    if IdReadTD == 1,
        fi(l) = - ramos(l,7)*graus_to_rad; % Transformador
Desfasador
    else
        fi(l) = 0.0;
    end
end
if IdOutJac == 1,
    fid = fopen('Jacob.dat','w');
end
if IdOutYGB == 1,
    pid = fopen('Admit.dat','w');
end
%
=====
=====
% Iniciando Contagem de Tempo Computacional

```

```

%
=====
=====
tic;
fprintf('\n> Executando cálculo de fluxo de carga ...\n')
%
=====
=====
% Matriz de Admitancia
%
=====
=====
Y = spalloc(nb,nb,nb+2*nr);
%
=====
=====
% Elementos Shunt da Barra na Diagonal da Matriz Admitancia
%
=====
=====
for k = 1 : nb,
    Y(k,k) = gshk(k) + i*bshk(k);
end
for l = 1 : nr,
    k = de(l);
    m = para(l);
%
=====
=====
% PI do Transformador
%
=====
=====
    akk(l) = 1/(tap(l)*tap(l));
    amm(l) = 1.0;
    akm(l) = 1/tap(l);
    y(l) = 1/(r(l) + i*x(l));
    Y(k,k) = Y(k,k) + akk(l)*y(l) + i*bshl(l);
    Y(m,m) = Y(m,m) + amm(l)*y(l) + i*bshl(l);
    Y(k,m) = Y(k,m) - akm(l)*y(l);
    Y(m,k) = Y(m,k) - akm(l)*y(l);
end
%
=====
=====
% Matriz Conductancia e Susceptancia
%
=====
=====
G = real(Y);

```

```

B = imag(Y);
if IdOutYGB == 1,
    for k = 1 : nb,
        if ((G(k,k) ~= 0) | (B(k,k) ~= 0)),
            fprintf(pid, '%4d %4d %50.40f\n', k, k, G(k,k), B(k,k));
        end
    end
    for l = 1 : nr,
        k = de(l);
        m = para(l);
        if ((G(k,m) ~= 0) | (B(k,m) ~= 0)),
            fprintf(pid, '%4d %4d %50.40f\n', k, m, G(k,m), B(k,m));
        end
    end
end
%
=====
====
% % Modificação feita pelo PT
%
=====
====%
M_nova=zeros(nb,nb);
for pont_L=1:nb;
    for pont_C=1:nb;
        if pont_L==pont_C;
            M_nova(pont_L,pont_C)=Bsh_b_pu(pont_L,1);    %%%%%%%%%
        end
    end
end
B=B+M_nova;
%
=====
====
%
=====
====
% Definição do Estado Inicial da Rede
% -> v    = 1 pu    para barras PQ
% -> ang = 0 graus para barras PQ e PV
%
=====
====
if IdFlatStar == 1,
    for k = 1 : nb,
        if tipo(k) ~= 3,
            ang(k) = 0.0 ;
            if tipo(k) < 2,

```

```

                v(k)= 1.0;
            end
        end
    end
end
%
=====
=====
% Contador de Iterações
%
=====
=====
iter = 0;
%
=====
=====
% Processo Iterativo
%
=====
=====
while (1),
%
=====
=====
% Cálculo das Potências Nodais
%
=====
=====
    for k = 1 : nb,
        pcalc(k) = G(k,k)*v(k)*v(k);
        qcalc(k) = -B(k,k)*v(k)*v(k);
    end
    for l = 1 : nr,
        k = de(l);
        m = para(l);
        ab = ang(k) - ang(m) + fi(l);
        gkm = akm(l)*real(y(l));
        bkm = akm(l)*imag(y(l));
        pcalc(k) = pcalc(k) + v(k)*v(m)*(-gkm*cos(ab) -
bkm*sin(ab));
        pcalc(m) = pcalc(m) + v(k)*v(m)*(-
gkm*cos(ab)+bkm*sin(ab));
        qcalc(k) = qcalc(k) + v(k)*v(m)*(-
gkm*sin(ab)+bkm*cos(ab));
        qcalc(m) = qcalc(m) - v(k)*v(m)*(-gkm*sin(ab)-
bkm*cos(ab));
    end
%
=====
=====

```

```

% Cálculo dos Mismatches de Potência
%
=====
=====
DP = zeros(nb,1);
DQ = zeros(nb,1);
maxDP = 0 ;
maxDQ = 0 ;
busDP = 0 ;
busDQ = 0 ;
for k = 1 : nb,
    if tipo(k) ~= 3,
        DP(k) = pnom(k) - pcalc(k);
        if abs(DP(k)) > abs(maxDP),
            maxDP = DP(k);
            busDP = numext(k);
        end
    end
    if tipo(k) <= 1,
        DQ(k) = qnom(k) - qcalc(k) ;
        if abs(DQ(k)) > abs(maxDQ),
            maxDQ = DQ(k) ;
            busDQ = numext(k) ;
        end
    end
end
%
=====
=====
% Relatorio de Erros
%
=====
=====
fprintf('\n> Iteração - %d\n',iter)
fprintf(' * Máximo mismatch P = %06.4f (barra
%04d)\n',baseMVA*maxDP,busDP)
fprintf(' * Máximo mismatch Q = %06.4f (barra
%04d)\n',baseMVA*maxDQ,busDQ)
if ( ( abs(maxDP) > tol ) | ( abs(maxDQ) > tol ) ),
%
=====
=====
% Montar e Inverter a Matriz Jacobiana
%
=====
=====
H = spalloc(nb,nb,nb+2*nr); M=H; N=H; L=H;
for k = 1 : nb,
    H(k,k) = -qcalc(k) - v(k)*v(k)*B(k,k);
    N(k,k) = (pcalc(k) + v(k)*v(k)*G(k,k))/v(k);

```

```

M(k,k) = pcalc(k) - v(k)*v(k)*G(k,k);
L(k,k) = (qcalc(k) - v(k)*v(k)*B(k,k))/v(k);
if tipo(k) == 3,
    H(k,k) = 10^10;
end
if tipo(k) >= 2,
    L(k,k) = 10^10;
end
end
for l = 1 : nr,
    k = de(l);
    m = para(l);
    ab = ang(k) - ang(m) + fi(l);
    H(k,m) = v(k)*v(m)*( G(k,m)*sin(ab) -
B(k,m)*cos(ab));
    H(m,k) = v(k)*v(m)*(-G(k,m)*sin(ab) -
B(k,m)*cos(ab));
    N(k,m) = v(k)*(G(k,m)*cos(ab)+B(k,m)*sin(ab));
    N(m,k) = v(m)*(G(k,m)*cos(ab)-B(k,m)*sin(ab));
    M(k,m) = -
v(k)*v(m)*(G(k,m)*cos(ab)+B(k,m)*sin(ab));
    M(m,k) = - v(k)*v(m)*(G(k,m)*cos(ab) -
B(k,m)*sin(ab));
    L(k,m) = v(k)*(G(k,m)*sin(ab)-B(k,m)*cos(ab));
    L(m,k) = - v(m)*(G(k,m)*sin(ab)+B(k,m)*cos(ab));
end

%
=====
=====
% Matriz Jacobiana
%
=====
=====
J = [ H N ; M L ];
%
=====
=====
% Imprimiendo Jacobiana
%
=====
=====
%
    if IdOutJac == 1,
%
        fprintf(fid,'ITERACAO',iter,' \n');
%
        for k = 1 : 2*nb,
%
            for w = 1 : 2*nb,
%
                if (J(k,w) ~= 0),
%
                    if (J(k,w) == 10^10),
%
                        fprintf(fid,'%4d %4d
%15s\n',k,w,'INFINITO');
%
                    else

```

```

%                                fprintf(fid, '%4d %4d
%15.10f\n', k, w, J(k, w));
%                                end
%                                end
%                                end
%                                end
%                                end
%                                end
%
=====
=====
% Vetor de Erros
%
=====
=====
% DS = [ DP; DQ ];
%
=====
=====
% Vetor de correção de estado
%
=====
=====
% DV = J\DS;
%
=====
=====
% Atualizar as variáveis de estado
%
=====
=====
% for k = 1 : nb,
%     ang(k) = ang(k) + DV(k);
%     v(k)    = v(k)    + DV(k+nb);
% end
%
=====
=====
% Incrementar contador de iterações
%
=====
=====
% iter = iter + 1;
% if iter > itmax,
%     fprintf('\n> Número máximo de iterações permitido
foi excedido.\n')
%     fprintf('> Execução interrompida.\n\n')
%     return
% end
else
break

```

```

        end
    end
    %
    =====
    =====
    % Final do While
    %
    =====
    =====
    fprintf('\n> Fim do cálculo de fluxo de carga. Preparando
    relatório de saída ...\n')
    %
    =====
    =====
    % Terminando Contagem de Tempo Computacional
    %
    =====
    =====
    tempo = toc ;
    fprintf('\n\n> Tempo computacional = %7.4f segundos.',tempo)
    fprintf('\n\n> Fim da simulação.\n\n')
    %
    =====
    =====
    % Cálculo dos Fluxos de Potência nos Ramos
    %
    =====
    =====
    for l = 1 : nr,
        k = de(l);
        m = para(l);
        gkm = real(y(l));
        bkm = imag(y(l));
        ab = ang(k) - ang(m) + fi(l);
        vkm = v(k)*v(m);
        pkm(l) = akk(l)*v(k)*v(k)*gkm -
        akm(l)*vkm*(gkm*cos(ab)+bkm*sin(ab));
        pmk(l) = amm(l)*v(m)*v(m)*gkm - akm(l)*vkm*(gkm*cos(ab) -
        bkm*sin(ab));
        qkm(l) = -akk(l)*v(k)*v(k)*(bkm+bshl(l)) +
        akm(l)*vkm*(bkm*cos(ab)-gkm*sin(ab));
        qmk(l) = -amm(l)*v(m)*v(m)*(bkm+bshl(l)) +
        akm(l)*vkm*(bkm*cos(ab)+gkm*sin(ab));
        pperdas(l) = pkm(l) + pmk(l);
        qperdas(l) = qkm(l) + qmk(l);
    end
    if IdOutJac == 1,
        fclose(fid);
    end
    if IdOutYGB == 1,

```

```

        fclose(pid);
end
% %
=====
=====
% % Relatório final
% %
=====
=====
% fprintf('\n\n> Relatório final\n\n')
% fprintf('> Rede: %s\n',nome_da_rede)
% fprintf('  Convergiu em %d iterações\n\n',iter)
% %if IdOutBusD == 1,
%     fprintf('> Estado da rede\n\n')
%     fprintf('  Barra Tipo          Mag    Fase          P          Q
Qsh\n')
%     for k = 1:nb
%         fprintf('%7d %4d %9.3f %6.2f %9.2f %7.2f %9.2f
\n',numext(k),tipo(k),v(k),...
%
(ang(k)*rad_to_graus),baseMVA*(pcalc(k)+pc(k)),baseMVA*(qcalc(
k)+qc(k)),baseMVA*bshk(k)*v(k)^2)
%     end
% %end
% %if IdOutBraD == 1,
%     fprintf('\n> Fluxos de potência\n\n')
%     fprintf('          De Para          Pkm          Qkm          Pmk          Qmk
Ploss    Qloss\n')
%     for l = 1:nr
%         fprintf('%7d %4d %9.2f %7.2f %9.2f %7.2f %9.2f
%7.2f\n',de(l),para(l),baseMVA*pkm(l),...
%
baseMVA*qkm(l),baseMVA*pmk(l),baseMVA*qmk(l),baseMVA*pperdas(l
),baseMVA*qpperdas(l))
%     end
% %end
%
% %
=====
=====
% % Dados necesarios para o modelo Y_modificado
% %
=====
=====
% % Dados necesarios para o modelo Y_modificado
% %
=====
=====
% % Dados necesarios para o modelo Y_modificado

```

```
% %  
=====
```

=====

```
% % Dados necesarios para o modelo Y_modificado  
% %  
=====
```

=====

```
Ang_deg=rad2deg(ang);  
V_bus=v;  
Return
```

[illegible]

FUNÇÃO “J_pen”

```

% Esta funcao gera penalidade para o individuo que provoque
FP_bus1_indiv capacitivo
% na barra 1

function
[Func_adap_penal_FP,delta_FP,FP_bus1_indiv]=J_penal_FP(Func_ad
ap_penal,K11,pkm_indiv,qkm_indiv,PI_MVAR_comercial,barras,V_bu
s_indiv);

% dimensiona sistema
[N_indiv, N_ramos]=size(pkm_indiv);
% testa faixa de K11
if K11<=1;      K=1;
elseif K11>=5; K=5;
elseif K11>1 & K11<5; K=K11;
end

% prepara variáveis
Func_adap_penal_FP=zeros(N_indiv,1);
Flag_penal_bus1_indiv=zeros(N_indiv,1);
MVAR_col=transpose(PI_MVAR_comercial);
Cap_bus1=1000*MVAR_col(:,1).*V_bus_indiv(:,1).*V_bus_indiv(:,1)
)
Pkm_bus1=100*1000*pkm_indiv(:,1)
Qkm_bus1=100*1000*qkm_indiv(:,1)-Cap_bus1

% calcula FP na barra 1
FP_bus1_indiv=abs(Pkm_bus1(:,1)./sqrt(Pkm_bus1(:,1).^2+Qkm_bus
1(:,1).^2));

for pont_C=1:N_indiv;
    if (Pkm_bus1(pont_C,1)<0 &
Qkm_bus1(pont_C,1)>0) | (Pkm_bus1(pont_C,1)>0 &
Qkm_bus1(pont_C,1)<0);
        FP_bus1_indiv(pont_C,1)=-FP_bus1_indiv(pont_C,1);
        Flag_penal_bus1_indiv(pont_C,1)=1;
        Func_adap_penal_FP(pont_C,1)=
K.*Func_adap_penal(pont_C,1);
    end
end
for pont_C=1:N_indiv;
    if Flag_penal_bus1_indiv(pont_C,1)==0;
        Func_adap_penal_FP(pont_C,1)=
Func_adap_penal(pont_C,1);
    end
end
delta_FP=Func_adap_penal_FP-Func_adap_penal;

```

end



FUNÇÃO "K_selec"

```

% Função Game binario, executando seleção em busca de maximos
% 1-Aceita FA's(valores de aptidão) negativos, positivos e
nulos
% 2-Está ajustada para fazer otimização em busca de maximos,
% nova selecao torneio, busca pelo maximo

function
[Pop_selecionada,delta_dec]=K_selec(Func_adap_penal_norm,PI_bin);

% Inverte a FA, permitindo que a seleção busque,
efetivamente, o minimo
Func_adap_penal_norm=-Func_adap_penal_norm;

Min_FA=min(Func_adap_penal_norm);
[N_Indiv,N_colunas]=size(PI_bin);
Masc_jogad=10*N_Indiv*ones(N_Indiv,N_Indiv);
% as instruções acima criam mascara de jogadores, onde
inicialmente os
% valores das celulas são muito maiores que o numero de
jogadores existentes;
% Em outras palavras cria individuos falsos; nao
existentes na população inicial

Masc_FAS=(Min_FA*ones(N_Indiv,N_Indiv)-1/3);
% a instrução acima cria mascara de FAs, onde inicialmente
os
% valores das celulas são menores que o menor FA
existentes na lista de
% aptidões iniciais. Em outras palavras cria FAS
garantidamente perdedores,
% visto que a minimização busca pelo máximo

for linha=1:N_Indiv;
    No_athet=randi([2,N_Indiv],1);
    for coluna=1:No_athet;
        Masc_jogad(linha,coluna)=randi([1,N_Indiv],1);
    end
end
% O laço "for" acima, cria a tabela de jogadores, por
competição. Caso o
% numero de jogadores seja menor que o numero de
indivíduos, a celula
% permanece com o numero muito alto, correspondente a u
jogador
% inexistente.

```

```

    for linha=1:N_Indiv;
        for coluna=1:N_Indiv;
            if Masc_jogad(linha,coluna)<=N_Indiv;
                Masc_FAS(linha,coluna)=
Func_adap_penal_norm(Masc_jogad(linha,coluna),1);
            end
        end
    end
    % O laço "for" acima, cria vetor com o FA do jogador
vencedor de cada
    % competição

    Max_FAS=max(Masc_FAS,[],2);
    for linha_2=1:N_Indiv;
        for linha_1=1:N_Indiv;
            if Max_FAS(linha_2,1)==Func_adap_penal_norm(linha_1,1);
                Indiv_selec(linha_2,1)=linha_1;
            end
        end
    end
    % O laço "for" acima, cria vetor com o numero da linha, da
população
    % inicial,que corresponde ao individuo/jogador vencedor de
cada
    % competição

    for linha=1:N_Indiv;
        Pop_selec(linha,:)=PI_bin(Indiv_selec(linha,1),:);
    end
    % O laço "for" acima, cria matriz com a população
    % selecionada, mas ainda sem inserir o conceito de
elitismo

    Pop_selecionada=Pop_selec;
    [x,list_top_menor]=sort(Func_adap_penal_norm);
    Orig_best_gay=list_top_menor(N_Indiv,1);
    delta_dec=Func_adap_penal_norm-Max_FAS;
    Pop_selecionada(1,:)=PI_bin(Orig_best_gay,:);
    % As instruções acima inserem na população selecionada,
    % o melhor individuo da população inicial, realizando
elitismo
end

```

FUNÇÃO "L_cruz"

```

% Funcao cruzamento de população binária
% 1-Pior situação ocorre se população de pais tem somente uma
linha. Neste
% caso nao se faz cruzamento, mas não se gera nenhum erro
% 2-Entrada do parametro "Pos", limitada dentro da faixa de 0
a 100 %.
% 3- Quando em o parametro "Pos=0", a troca é feita à partir
do bit 1,
% com total influencia da mae. Quando o parametro "Pos=100", a
troca
% é feita no bists da ultima coluna,
% com total influencia da mae
% 3-Entradas do parametro "Pc_perc", limitada no numero de
linhas
% da população dos pais (Pop_selecionada)
% 4-A Função forma os casais,e troca os bits deles, sendo que
determinado
% individuo pode acasalar varias vezes, ou nenhuma vez.
% 5-Quando o individuo não é acasalado ele permanece a
população de filhos
% 6-Nenhuma troca é feita caso o individuo seja selecionado
para acasalar
% consigo mesmo,
% 7-A formação dos casais é aleatória
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function
[Pop_filho,desvio_36]=L_cruz(Pc_perc,Pop_selecionada);
% trocando nomes e preparando sinais
Pais=Pop_selecionada;
[N_linhas,N_bits_indiv]=size(Pais);
Filhos=zeros(N_linhas,N_bits_indiv);
% Impondo limites ao numero de cruzamentos
if Pc_perc>=0 & Pc_perc<100;   N_cruz=Pc_perc;
elseif Pc_perc<0;   N_cruz=0;
elseif Pc_perc>=100;   N_cruz=100;
end
Qtd_cruz=round(N_cruz*N_linhas/100);% Definindo a quantidade
de cruzamentos
% definindo posição de avanco

for   ciclos=1:Qtd_cruz;   %N_cruz;   % Forma casais e faz a
troca dos bits
    Rndi_L1=randi(N_linhas,1);
    Rndi_L2=randi(N_linhas,1);
    if Rndi_L1~=Rndi_L2;

```

[illegible]

FUNÇÃO "M_mut"

```

% Função Mutação de populacao binaria

% A função varre toda a matriz que contem a população de
individuos e para
% cada bit desta matriz/população gera valor randmico.
% Se o valor randomico de cada um dos bits é menor que a
quantidade
% de bits à mutar, o valor do bit é entao invertido.
% Variaveis:
% Pm_perc: [0 a 5], em percentual, define a taxa de mutação,
% Tx_Dec_mut_perc: [0 a 100], em percentual, define a
velocidade de
% decaimento da mutação,
% Ativa_mut_adap: [ 1 ou 0 ], se "1" impõe mutação adaptativa,
% A mutação adaptativa, quando ativada, parte da taxa "Pm_perc"
em direção a
% valor nulo. A taxa de decaimento é influenciada pelo numero
de iterações
% ja realizadas e pelo parametro "Tx_Dec_mut_perc". Contudo, a
partir da
% metade do numero de iterações programada, a mutação é
zerada,
% A mutação adaptativa, quando desativada, mantem ao longo de
todo o
% funcionamento, a taxa de mutação igual ao 'Pm_perc'
% Matriz de saida:"Pop_com_mutacao", ja com os bits mutados
% Matriz de saida:"Intens_mut", serve para identificar
especificamente,
% atraves de artificio, os bits que foram efetivamente mutados

%=====
=====
function
[Pop_com_mutacao,Genes_prog_perc]=M_mut(Pop_bal,search,Tx_Dec_
mut_perc,Pm_perc,N_geracoes,Ativa_mut_adap);
% % Definindo limite para parametro que determina quantidade
de bits a mutar
if Pm_perc<=0;
Prob_mut_perc=0.0; end
if Pm_perc>=5;
Prob_mut_perc=5; end
if Pm_perc>0 & Pm_perc<5;
Prob_mut_perc=Pm_perc; end

% % Definindo decremento
if Tx_Dec_mut_perc<=0;
Decai_perc=0.0; end

```

```
if Tx_Dec_mut_perc>=100;  
Decai_perc=100; end  
if Tx_Dec_mut_perc>0 & Tx_Dec_mut_perc<100;  
Decai_perc=Tx_Dec_mut_perc; end  
  
% Expressaõ  
if Ativa_mut_adap==1;  
Genes=Prob_mut_perc-  
Prob_mut_perc*(search/N_geracoes)*Decai_perc/100;  
end  
if Ativa_mut_adap==0;  
Genes=Prob_mut_perc;  
end  
if Genes<=0;  
Genes=0;  
end  
if (search/N_geracoes)>=0.6 & Ativa_mut_adap==1;  
Genes=0;  
end  
%  
%  
% trocando nomes e preparando variaveis  
Genes_prog_perc=Genes;  
Populacao=Pop_bal;  
[N_linhas,N_colunas]=size(Populacao);  
Spot=zeros(N_linhas,N_colunas);  
Pop_sem_mutacao=Populacao;  
%  
% %  
%  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
N_mut=(Genes_prog_perc./100);  
for pont_L=1:N_linhas;  
    for pont_C=1:N_colunas;  
        Rnd=rand;  
        if Rnd<N_mut;  
            Spot(pont_L,pont_C)=Populacao(pont_L,pont_C);  
            if Spot(pont_L,pont_C)==1;  
                Populacao(pont_L,pont_C)=0;  
            end  
            if Spot(pont_L,pont_C)==0;  
                Populacao(pont_L,pont_C)=1;  
            end  
        end  
    end  
end
```

[illegible]