

UNIVERSIDADE FEDERAL FLUMINENSE
ESCOLA DE ENGENHARIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

AYRES NISHIO DA SILVA JUNIOR

ALGORITMO EVOLUTIVO NA ALOCAÇÃO DE PLANOS DE MEDIÇÃO

NITERÓI

2018

AYRES NISHIO DA SILVA JUNIOR

ALGORITMO EVOLUTIVO NA ALOCAÇÃO DE PLANOS DE MEDIÇÃO

Trabalho de Conclusão de Curso apresentado ao Corpo Docente do Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal Fluminense, como parte dos requisitos necessários à obtenção do título de Engenheiro Eletricista.

Orientador:

Prof. Dr. Marcio Andre Ribeiro Guimaraens

Niterói, RJ

2018

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTES
TRABALHOS, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA
FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

FICHA CATALOGRÁFICA GERADA EM:

<http://www.bibliotecas.uff.br/bee/ficha-catalografica>

Ficha catalográfica automática - SDC/BEE

U47t Último sobrenome do autor, Nome do autor
Título do Trabalho / Nome do autor Último sobrenome do
autor; Nome do orientador Último sobrenome do orientador,
orientador. Niterói, 2018.
100 f.

Trabalho de Conclusão de Curso (Graduação em Engenharia
Elétrica)-Universidade Federal Fluminense, Escola de
Engenharia, Niterói, 2018.

1. Primeiro assunto. 2. Segundo assunto. 3. Produção
intelectual. I. Título II. Último sobrenome do
orientador, Nome do orientador, orientador. III. Universidade
Federal Fluminense. Escola de Engenharia. Departamento de
Engenharia Elétrica.

CDD -

AYRES NISHIO DA SILVA JUNIOR

ALGORITMO EVOLUTIVO NA ALOCAÇÃO DE PLANOS DE MEDIÇÃO

Trabalho de Conclusão de Curso apresentado ao Corpo Docente do Departamento de Engenharia Elétrica da Escola de Engenharia da Universidade Federal Fluminense, como parte dos requisitos necessários à obtenção do título de Engenheiro Eletricista.

Aprovado em 26 de junho de 2018, com nota _____ (_____), pela banca examinadora.

BANCA EXAMINADORA

Prof. Dr. Marcio André Ribeiro Guimaraens – Orientador
UFF

Prof. Dr. Vitor Hugo Ferreira –
UFF – Universidade Federal Fluminense

Prof Dr. Rainer Zanghi, –
UFF – Universidade Federal Fluminense

Niterói
2018

*Dedicatória: A meus pais Ana Cristina e Ayres
por todo amor, apoio e conselhos valiosos.*

AGRADECIMENTOS

Agradeço primeiramente aos meus pais por todo amor, apoio e determinação para me ajudar em todas as minhas conquistas. Não existem palavras que expressem minha gratidão.

Ao meu orientador Marcio Guimaraens por todo suporte e inspiração que contribuíram para a realização deste trabalho e talvez uma futura carreira acadêmica.

Aos meus amigos da UFF que me ajudaram nessa caminhada, entre eles: Jacson Moura, pelo suporte e madrugadas estudando, suas críticas construtivas sempre ajudaram a me manter na linha e serão sempre consideradas; Vinicius Biajoni pelas prazerosas conversas a caminho do CEPEL e pela convivência que ajudou no meu amadurecimento pessoal; Igor Costa pelo auxílio em todas as matérias difíceis e trabalhos em grupo realizados no decorrer do curso.

A toda *galere* pelos momentos de descontração que dão prazer a vida.

RESUMO

Com a existência de sistemas de potência de grande porte, surge a necessidade do uso de um estimador de estados para o processamento das informações em tempo real, a fim de possibilitar a operação e a supervisão desses sistemas. Contudo, para que o estimador de estados funcione de forma correta é preciso que unidades de medidores forneçam medidas suficientes, para que o sistema se torne observável. Este trabalho tem como objetivo desenvolver algoritmos evolutivos, que retornem o mínimo de UTR's (unidades terminais remotas) possível para tornar o sistema observável de forma também a fornecer o máximo de informações possíveis do mesmo.

O SimSEP (Simulador do Sistema Elétrico de Potência) tem por objetivo simular um sistema elétrico de potência ao longo de um dia. O método proposto por este trabalho será adicionado ao simulador com o objetivo de fornecer um plano de observação do sistema em simulação.

Palavras-Chave: Alocação de medidores; Algoritmo Evolutivo; Otimização.

ABSTRACT

With the existence of large power systems, the need arises to use a state estimator to process the information in real time, in order to enable the operation and supervision of these systems. However, for the state estimator to work properly, meter units must provide sufficient measures for the system to become observable. This work aims to develop evolutionary algorithms, which return the minimum of RTUs (remote terminal units) possible to make the system observable so as to provide as much information as possible.

The SimSEP (Simulator of the Electric Power System) aims to simulate an electrical power system over a day. The method proposed by this work will be added to the simulator in order to provide a simulation system observation plan.

LISTA DE FIGURAS

Figura 2.1 Exemplo estimação de estados.....	4
Figura 2.2-Exemplos de estimação com erros de medição	6
Figura 2.3-Cálculo de fluxos para $M12$ e $M32$,.....	6
Figura 2.4 - Exemplo de Depuração de erros	10
Figura 2.5 - Ótimo global(G) e local(L) para casos de minimização	13
Figura 2.6- Representação binária dos cromossomos	17
Figura 2.7-Representação Real de cromossomos	17
Figura 2.8-Fluxograma de um Algoritmo genético	18
Figura 2.9- Exemplo Roleta	20
Figura 2.10-Cruzamento com um único ponto.....	20
Figura 2.11-Cruzamento com dois pontos.....	21
Figura 2.12-Exemplo de cruzamento uniforme.....	21
Figura 2.13-Exemplo de mutação em representação binária.....	21
Figura 2.14-Alocação ótima para o sistema de 14 barras do IEEE	23
Figura 2.15-Transformação topológica	28
Figura 2.16-Representação da solução	32
Figura 2.17-Alocação de medidas	33
Figura 2.18- Resultado da alocação de medidas.....	35
Figura 2.19-Fluxograma simplificado do SimSEP.....	37
Figura 3.1-Criação da população inicial.....	40
Figura 3.2-Matriz da população e da observabilidade.....	40
Figura 3.3-Matriz de reprodutores $m5s$	42
Figura 3.4-Exemplo de cruzamento.....	42
Figura 3.5-Exemplo de mutação.....	43

Figura 3.6-Fluxograma de V0	44
Figura 3.7-Matriz de rankings	46
Figura 3.8-Taxa de diversidade	47
Figura 3.9-Fluxograma da versão quatro.....	49
Figura 3.10-Hierarquia orientada objeto	50
Figura 3.11-Mutantes gerados	53
Figura 3.12-Representações das soluções	54
Figura 3.13-Busca local.....	55
Figura 3.14 -Fluxograma da última versão.....	56
Figura 3.15-Fluxo de processos da Função <i>Spartan</i>	57
Figura 5.1-Mínimo de UTR para o sistema de 118 barras	65

LISTA DE TABELAS

Tabela 2.1-Resultados obtidos sem considerar perda de PMU	30
Tabela 2.2-Resultados obtidos considerando perda de PMU	31
Tabela 3.1- Tabela de convenção de medidores.....	58
Tabela 4.1Parâmetros de entrada do algoritmo evolutivo	59
Tabela 4.2-Resultados Versão 0	59
Tabela 4.3-Resultados Versão 1	60
Tabela 4.4-Resultados Versão 2	60
Tabela 4.5-Resultados Versão 4	61
Tabela 4.6-Resultados Versão 5	61
Tabela 4.7-Resultados Versão 6	62
Tabela 4.8-Resultados Versão 7	62
Tabela 4.9-Melhores Arranjos Encontrados	63
Tabela 4.10-Comparação de nº de medidas para 57 barras	63
Tabela 4.11-Comparação de nº de medidas para 118 barras	64
Tabela 4.12-Resultados Obtidos Considerando perda de Medidores.....	64

LISTA DE ABREVIATURAS E SIGLAS

Cdf	Common Data Format
EE	Estimador de Estados
IEEE	Instituto de Engenheiro Eletricistas e Eletrônicos
Matlab	MATrix LABoratory
ONS	Operador Nacional Sistema Elétrico
PMU	Phasorial Measurement Unit
SCADA	<i>Supervisory Control and Data Acquisiton</i>
UTR	Unidade Terminal Remota

SUMÁRIO

1	INTRODUÇÃO.....	1
1.1	OBJETIVO	2
1.2	ESTRUTURA DO TRABALHO	2
2	FUNDAMENTAÇÃO TEÓRICA	3
2.1	ESTIMAÇÃO DE ESTADOS	3
2.1.1	Etapas	Erro! Indicador não definido.
2.2	OTIMIZAÇÃO	12
2.2.1	MÉTODOS DIRETOS	13
2.2.2	Heurística.....	15
2.3	ALOCAÇÃO DE MEDIDORES.....	22
2.3.1	OBSERVAÇÃO DE SISTEMAS POR PMU’S	22
2.3.2	OBSERVAÇÃO DE SISTEMAS POR MEDIDAS SCADA.....	23
2.3.3	LEVANTAMENTO HISTÓRICO	24
2.4	PROGRAMA SimSEP	36
3	METODOLOGIA PROPOSTA	38
3.1	VERSÕES DO ALGORITMO	39
3.1.1	VERSÃO 0:	39
3.1.2	VERSÃO 1:	44
3.1.3	VERSÃO 2	45
3.1.4	VERSÃO 3	46
3.1.5	VERSÃO 4	46
3.1.6	VERSÕES ALTERNATIVAS	49
3.1.7	VERSÃO 5	51
3.1.8	VERSÃO 6	52
3.1.9	VERSÃO 7	54
3.2	REGISTRAR ESQUEMA DE MEDIÇÃO	56
3.3	MEDIDORES DE RETAGUARDA	58
4	ANÁLISE DE RESULTADOS	59
4.1	VERSÃO 0	59
4.2	VERSÃO 1	60
4.3	VERSÃO 2	60
4.4	VERSÃO 4	61

4.5	VERSÃO 5	61
4.6	VERSÃO 6	62
4.7	VERSÃO 7	62
4.8	CONSIDERANDO PERDA DE MEDIDORES	64
5	CONCLUSÕES	64
6	Bibliografia.....	67

1 INTRODUÇÃO

O sistema de produção e transmissão de energia elétrica do Brasil é um sistema hidro-termo-eólico de grande porte, com predominância de usinas hidrelétricas e com múltiplos proprietários. O Sistema Interligado Nacional (SIN) é constituído por quatro subsistemas: Sul, Sudeste/Centro-Oeste, Nordeste e a maior parte da região Norte. (ONS, 2018).

Um sistema desse porte, interligado e em constante crescimento, requer uma supervisão e uma operação extremamente segura; assim surgindo, a necessidade da obtenção de informações confiáveis do sistema para o auxílio nas tomadas de decisão. O processamento dessas informações deve ser feito em tempo real e é atribuído ao estimador de estados.

O estimador de estados (EE) tem a função retornar informações confiáveis do sistema a partir do processamento de medidas redundantes, fornecidas pelo sistema de supervisão conhecido como SCADA a partir de unidades medidoras denominadas UTR's (unidades terminal remota). Com estas medidas, o estimador é capaz de retornar o valor de módulo e ângulo das tensões de todas as barras do sistema. Uma vez que esses valores são conhecidos, qualquer outra informação desejada pode ser calculada (correntes fluxos e injeções de potência).

Contudo, o EE só irá operar de forma correta se o sistema possuir unidades medidoras bem posicionadas, que forneçam medidas suficientemente redundantes, de forma a possibilitar que a estimação das variáveis seja realizada. É desejado também o maior número de medidas redundantes possível, pois estas permitirão que o estimador encontre com mais facilidade erros grosseiros de medição.

Quanto maior o número de unidades medidoras instaladas, maior será a quantidade de medidas presentes no sistema. Porém, instalar um número grande de medidores em um sistema não é uma opção financeiramente viável, pois a quantidade de medidores é diretamente proporcional ao custo do projeto. Sendo assim, um problema de otimização é criado, onde procuramos não apenas encontrar o menor número de unidades medidoras que garanta a observabilidade do sistema, como também o posicionamento destas unidades que forneça o maior número de informações possíveis do sistema.

1.1 OBJETIVO

O objetivo deste trabalho é desenvolver um algoritmo “evolutivo”, que resolva o problema de alocação ótima de medidores em um sistema elétrico de potência. Este tipo de algoritmo recebe este nome por possuir a estrutura de um algoritmo genético (população inicial, seleção, cruzamento e mutação) e também possuir algumas medidas que agilizam seu processo de convergência para a solução desejada. O algoritmo deve ser capaz de resolver o problema para qualquer sistema e foi testado nos de 14, 30, 57 e 118 barras do IEEE.

1.2 ESTRUTURA DO TRABALHO

O Capítulo 2 se inicia com o tópico sobre estimação de estados, uma sucinta explicação sobre estimação de estados. No tópico seguinte é abordado o tema de otimização e são citados os métodos diretos e heurísticos que podem solucionar o problema. Em seguida no terceiro tópico é abordado o problema de alocação de medidores, como a observação de sistemas pode ser realizada por unidades medidoras e um levantamento histórico em que são analisados alguns artigos que abordam o tema. Por fim, O ultimo tópico deste artigo encarrega-se de explicar o SimSEP, o simulador que o algoritmo foco deste trabalho será aplicado.

No capítulo 3, é apresentado o algoritmo evolutivo, como foi realizado seu desenvolvimento até a versão atual do programa, como é gerado o arquivo que será utilizado no simulador e a funcionalidade extra, que permite ao algoritmo gerar um esquema de medição considerando possíveis perdas de unidades de medição.

O Capítulo 4 apresenta os resultados obtidos nas simulações dos sistemas de teste do IEEE no algoritmo apresentado e uma comparação com a literatura já existente.

Por fim, o Capítulo 5 apresenta as principais conclusões alcançadas ao analisar os resultados, bem como propostas para melhorar o método proposto.

2 FUNDAMENTAÇÃO TEÓRICA

Para assegurar a Operação de um sistema de potência é necessário o monitoramento das condições de operação do sistema. Esta função normalmente é atribuída ao estimador de estado. A partir dos dados coletados de subestações monitoradas e controladas remotamente, o estimador pode fornecer um valor próximo, calculado para todas as grandezas elétricas e todos os parâmetros de rede do sistema de potência, como também detectar e filtrar tantos erros grosseiros de medição (XU e ABUR, 2005).

Este capítulo apresentará a fundamentação teórica utilizada na implementação do método proposto de alocação de unidades medidoras. Primeiramente, será abordado o tema de estimação de estados, serão citadas suas principais etapas e como as medidas fornecidas pelas unidades afetam o processo. Em seguida, como procuramos o menor número de unidades medidoras que garanta a observabilidade do sistema, serão abordados problemas de otimização e seus principais métodos de solução. O problema de alocação de medidores também é descrito nesse capítulo, assim como suas abordagens em artigos passados. Por fim será explicado a finalidade do SimSEP e como o método proposto será adicionado á este.

2.1 ESTIMAÇÃO DE ESTADOS

O desempenho seguro do monitoramento e da operação de sistemas de potencia depende da disponibilidade de informações confiáveis do ponto de operação atual do sistema. Os sistemas SCADA são úteis para possibilitar a operação de redes por fornecerem, em tempo real, diversos dados adquiridos remotamente, entre eles: dados de chaves (aberto/fechado) de chaves e/ou disjuntores, que definem a refletem a configuração atual da rede elétrica e valores de medidas de grandezas elétricas normalmente fornecidas por unidades de medição. Apesar disso, a utilização de medidas oriundas diretamente do sistema SCADA pode ser considerada incompleta, pois não são fornecidas ao operador, informações sobre barras não monitoras (Salgado, 2002).

Além disso, os valores fornecidos pelo sistema estão sujeitos a erros, que podem ser tanto de pequena magnitude, estes são comuns e são referentes à margem de erro, presente em qualquer medição, quanto erros grosseiros, estes são originados pelo funcionamento inadequado de elementos que compõem o processo de medição. Sendo assim, torna-se necessário a utilização de um estimador de estados que deverá funcionar como um filtro capaz de detectar, identificar e remover inconsistências em dados redundantes de medição. Como é

dito por (Guimaraens, 2015) “Entende-se como redundância para a estimação de estados o excedente de medidas disponíveis em relação ao número de variáveis de estado a estimar”.

Citando novamente (Guimaraens, 2015) “Entende-se por estado operativo do sistema aquele plenamente caracterizado por grandezas elétricas referentes a uma determinada configuração de rede, ou seja, tensões de cada barra da rede elétrica (módulo e ângulo). Uma vez conhecido o estado, grandezas elétricas dele dependentes podem ser determinadas (correntes, fluxos e injeções de potência)”. Sendo assim, cabe também ao estimador de estados, processar todas as medidas fornecidas pelo sistema SCADA de forma a estimar os valores das tensões complexas em todas as barras, incluindo as não monitoradas.

Pode-se concluir então, que as funções do estimador de estado são:

- Filtrar erros de pequena magnitude
- Detectar medidas portadoras de erros grosseiros
- Estimar grandezas que não foram medidas em função de perda dos canais de comunicação.

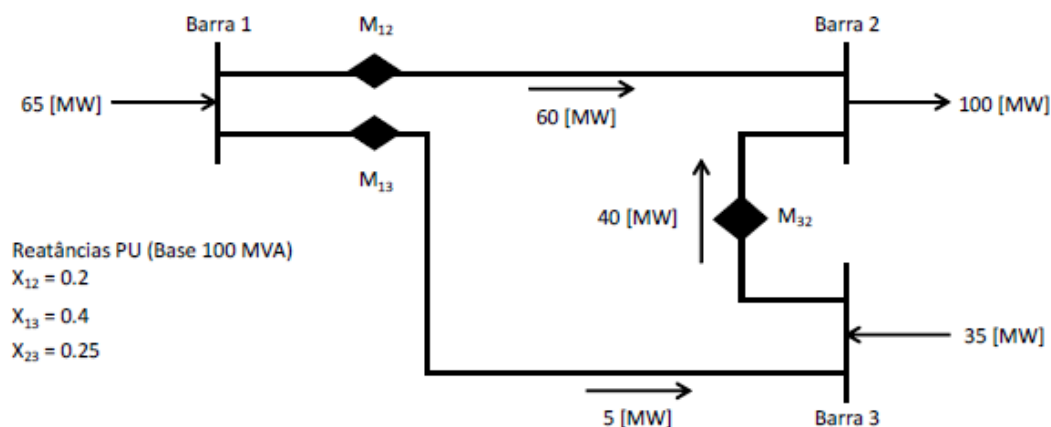
2.1.1 ETAPAS

Usualmente como é descrito por (Guimaraens, 2015) o processo de estimação de estados pode ser dividido em três etapas descritas a seguir:

Filtragem

Nesta etapa estima-se o estado da rede utilizando o conjunto de medidas e configuração da rede. Para que isso seja possível às chamadas variáveis de estado que definem o sistema devem ser calculadas (tensão e ângulo para casos de fluxo CA). Para auxiliar no entendimento do processo, considera-se o exemplo ilustrativo do (Wood, 1984) apresentado na **Erro! Fonte de referência não encontrada. Erro! Fonte de referência não encontrada...**

Figura 2.1 Exemplo estimação de estados



Fonte: (Wood, 1984)

Para simplificação dos cálculos o modo CC será utilizado, neste modelo, como é dito em (Monticelli, 1983) valores de potência reativa e magnitude de tensão são desconsiderados, sendo assim as únicas variáveis que devem ser calculadas são os ângulos de tensão das barras do sistema.

Como pode ser visto em (2.1) e (2.2) somente duas das três medições disponíveis são necessárias para o calcula das variáveis de estado envolvidas.

$$f_{13} = \frac{1}{x_{13}}(\theta_1 - \theta_3) = M_{13} = 5[MW] = 0,05pu \quad (2.1)$$

$$f_{32} = \frac{1}{x_{32}}(\theta_3 - \theta_2) = M_{32} = 40[MW] = 0,40pu \quad (2.2)$$

- M_{ij} Representa a medida de fluxo de potência ativa da barra i para a barra j.
- f_{ij} Representa o fluxo de potência ativa calculada da barra i para a barra j

Considerando a barra 3 como referência angular ($\theta_3 = 0$), os valores de θ_1 e θ_2 , podem ser obtidos como mostra (2.3).

$$\begin{cases} \frac{1}{x_{13}}(\theta_1 - \theta_3) = 0,05pu \\ \frac{1}{x_{32}}(\theta_1 - \theta_3) = 0,40pu \end{cases} \rightarrow \begin{cases} \theta_1 = 0,2 [rad] \\ \theta_2 = -0,1 [rad] \end{cases} \quad (2.3)$$

No caso apresentado, apesar de existirem três medidas disponíveis somente duas foram necessárias para o cálculo das variáveis de estado. Sendo assim, a medida que não foi utilizada na estimação pode ser considerada redundante.

Porém, como foi citado anteriormente, as medidas que geralmente são fornecidas ao estimador são portadoras de algum tipo de erro. Refazendo o exemplo anterior, porem utilizando medidas com erro:

$$M_{12} = 62[MW] = 0,62pu$$

$$M_{13} = 6[MW] = 0,06pu$$

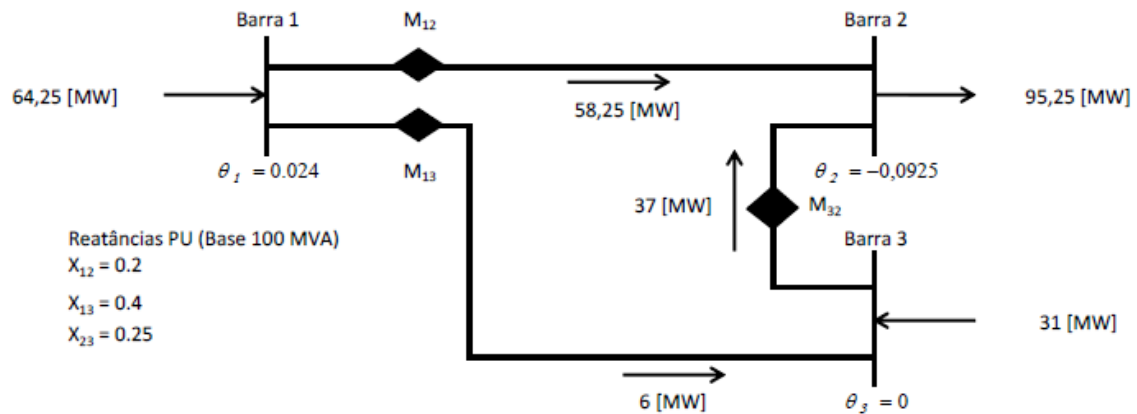
$$M_{32} = 37[MW] = 0,37pu$$

Resolvendo de forma análoga a (2.3)

$$\begin{cases} \frac{1}{x_{13}}(\theta_1 - \theta_3) = 0,06pu \\ \frac{1}{x_{32}}(\theta_1 - \theta_3) = 0,37pu \end{cases} \rightarrow \begin{cases} \theta_1 = 0,024 [rad] \\ \theta_2 = -0,0925 [rad] \end{cases}$$

Sendo assim, com todas as variáveis de estado, os fluxos podem ser calculados e são apresentados na Figura 2.1. Como pode ser visto o fluxo estimado é coerente com as medidas M_{12} e M_{32} , porém, o fluxo da linha 1-2 não se iguala à medida registrada 62MW.

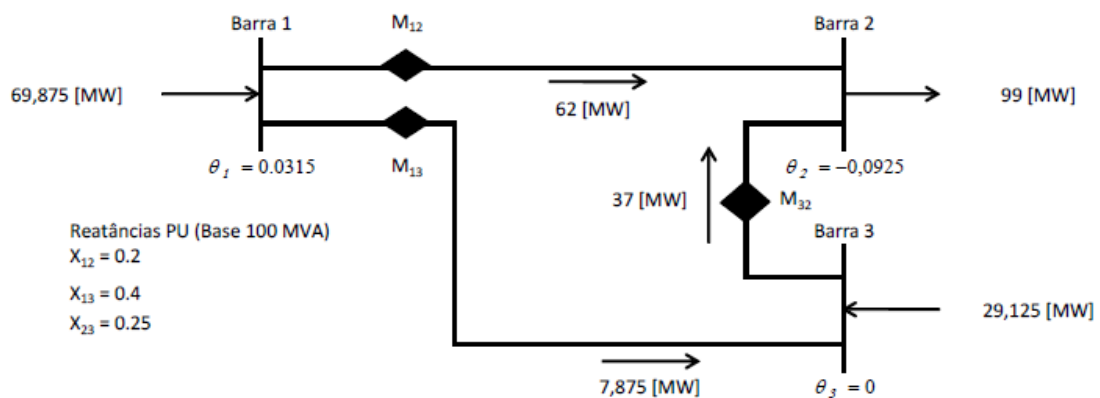
Figura 2.2-Exemplos de estimação com erros de medição



Fonte: (Wood, 1984)

Se o mesmo processo for realizado porem considerando M_{12} e M_{32} , ao invés de M_{13} e M_{32} , os resultados obtidos são expostos na Figura 2.3. Como pode ser visto, reconsiderar os medidores fez com que o fluxo estimado M_{13} se igualasse ao medido, porém, o valor de M_{12} deixou de coincidir com o medido anteriormente.

Figura 2.3-Cálculo de fluxos para M_{12} e M_{32} ,



Fonte: (Wood, 1984)

Como pode ser visto apesar do sistema possuir medidas redundantes que podem ser desconsideradas no cálculo das variáveis de estado, todas as medidas possuem algum tipo de informação importante para a realização do mesmo e por isso não devem ser descartadas.

O estimador de estado então precisa de um procedimento que use as informações disponíveis de todos os três medidores para produzir a melhor estimativa dos ângulos reais,

fluxos de linha e carga e gerações de barramento. Para isso utiliza-se então o método dos mínimos quadrados ponderados, de forma que, que medidas com maior exatidão exerçam maior influência na obtenção na estimativa das variáveis.

Considerando todas as medidas portadoras de erro

$$\varepsilon = z - h(\underline{x}) \quad (2.4)$$

- z – É o vetor de medidas do sistema, formado por medidas fornecidas normalmente de UTR's, tais como: fluxos de potência ativa e reativa, injeções de potência ativa e reativa; e módulos de tensão nas barras. Possui dimensão $(N_m \times 1)$, onde N_m é o número de medidas do sistema.
- $h(.)$ – É o vetor que relaciona o estado verdadeiro com as medidas isentas de erros, através de funções não lineares.
- x – Vetor de estado, representando o módulo e ângulo das tensões nas barras, com dimensão $(N_s \times 1)$, onde n é o número de variáveis de estado, sendo $N_s = 2nb - 1$ e nb o número de barras do sistema.
- ε – Vetor de erros associados a medição, descrito como variável aleatória de distribuição de probabilidades Normal, valor esperado zero e matriz de covariância R .

$$J(x) = \sum_i^m [z - h_i(x)]^2 * \alpha$$

Sendo:

- α_i – Peso atribuído a i -ésima medida.
- $J(x)$ – Número de medidas independentes

Formulação Matricial

$$J(x) = [z - h(x)]^t W [z - h(x)] \quad (2.5)$$

Onde:

- $W = R^{-1}$ – Matriz de ponderação
- $R = \begin{bmatrix} 1/\alpha_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1/\alpha_{N_m} \end{bmatrix}$

O objetivo a ser alcançado consiste em se obter uma estimativa para x que minimize $J(x)$, através de:

$$\left. \frac{\partial J(x)}{\partial x} \right|_{x=\hat{x}} = 0 \quad (2.6)$$

Aplicando as condições (2.5) e (2.6), obtém-se a seguinte equação:

$$g(x) = H^t W [z - h(\hat{x})] \quad (2.7)$$

Onde $H = \frac{\partial h(x)}{\partial x}$ denomina-se Matriz Jacobiana.

Como $h(x)$ é formado por funções não lineares, não existe solução direta para a expressão, utiliza-se então o método de Newton Rapshon para encontrar o estado estimado em (2.7), obtendo-se o seguinte processo iterativo:

$$x^{(k+1)} = x^{(k)} + [(H^t W H)^{-1} H^t W]^{(k)} [z - h(x^{(k)})] \quad (2.8)$$

Onde k é o contador de iterações

Define-se a matriz de ganho do processo como:

$$G = (H^t W H) \quad (2.9)$$

Reescrevendo (2.9) então obtemos:

$$x^{(k+1)} = x^{(k)} + [G^{-1} H^t W]^{(k)} [z - h(x^{(k)})]$$

A convergência do processo iterativo é avaliada através do maior componente do módulo do vetor desvio $|\Delta x^{(k)}| = |x^{(k+1)} - x^{(k)}|$ que deve ser menor que uma determinada tolerância.

O problema pode ser simplificado ao considerar fluxo CC uma relação linear entre estado e medida:

$$z = Hx + \varepsilon$$

Onde H é a matriz Jacobiano obtida através da linearização das equações de fluxo de potência.

A função objetivo passa a ter a seguinte forma:

$$J(x) = \sum_i^{Nm} \varepsilon_i^2 \alpha_i = \sum_i^{Nm} (z_i - Hx_i)^2 \alpha_i$$

Matricialmente:

$$J(x) = [z - Hx]^t W [z - Hx]$$

O estado estimado é obtido através de:

$$\hat{x} = [G^{-1} H^t W] z$$

Portanto o vetor de medidas filtradas vem de:

$$\hat{z} = H \cdot \hat{x}$$

Análise de observabilidade

Antes de se estimar o estado de um determinado sistema, deve-se avaliar se o mesmo é observável ou não. Como é dito por (Salgado, 2002) “Um sistema de potência é observável com respeito a um plano de medição M se as variáveis de estado da rede podem ser determinadas através do processamento das medidas em M por um estimador de estados”. Em outras palavras, é preciso avaliar se o conjunto de dados disponíveis possibilita a estimação de estados em toda a rede.

$$\Delta x = \{G^{-1}H^tR^{-1} \begin{bmatrix} z_1 - f_1(\underline{x}) \\ \vdots \\ z_{Nm} - f_{Nm}(\underline{x}) \end{bmatrix} \quad (2.10)$$

$$G = \{[H^t]R^{-1}H\}$$

Como pode ser visto em (2.10) observabilidade de um sistema pode ser analisada se a matriz de ganho G for inversível, isto é, e possuir determinante não nulo. Supondo que a matriz R seja diagonal, A matriz Jacobiana H deve ter posto completo (Todas suas N_s colunas devem ser linearmente independentes). Isto implica que, do conjunto de N_m pelo menos N_s devem ser linearmente independentes. Pode-se concluir então que para um sistema ser considerado observável, ele deve possuir, pelo menos, um número de medidas não redundantes iguais ao número de variáveis de estado.

Análise de resíduos

Uma vez que o estado tenha sido estimado, é preciso realizar uma análise da consistência dos resultados obtidos. Este processo é denominado análise de resíduos e seu principal objetivo é detectar Medidas com grau de imprecisão muito maior do que o suposto no modelo de medição (Wood, 1984). Estas são denominadas medidas portadoras de erro grosseiro.

O processo é realizado através da análise do vetor de resíduos como é demonstrado por (Guimaraens, 2015) conforme:

$$r = z - h(x)$$

Onde:

- z – Valor medido
- $h(x)$ – Valor estimado
- r – resíduo da estimação

O vetor de resíduos pode ser interpretado como uma variável aleatória Normal, de valor esperado zero e matriz de covariância U , obtida por:

$$U = R - H(H^tWH)^{-1}H^t$$

O vetor de resíduos é normalizado e verifica-se a seguinte condição:

$$r_N(i) = \frac{|r(i)|}{\sqrt{U(i,i)}} \leq \lambda$$

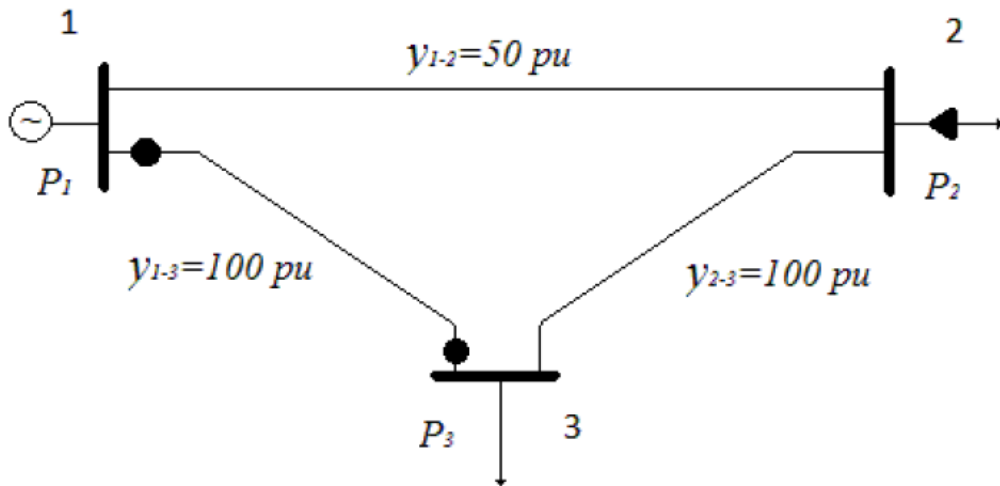
Onde o índice i indica a i -ésima componente do vetor dos resíduos. E λ representa o limite de detecção. Medidas que possuírem resíduos normalizados maiores que o limite de detecção indicam a presença de erros grosseiros. Uma medida portadora de erro grosseiro compromete a estimação de componentes de estado a ela associados, pois consequentemente, influencia a estimativa de outras medidas, fazendo com que diversos resíduos elevados sejam detectado. Tal situação é descrita por (Guimaraens, 2015) como espalhamento de ED.

Para a identificação de erros grosseiros, a análise de resíduos deve ser efetuada e, em caso de violação do limite, a medida que possuir maior resíduo deve ser considerada como suspeita e removida do esquema de medição. A estimação deve ser realizada novamente até que não ocorra nenhuma violação do limite.

Medidas que possuírem resíduo nulo são consideradas medidas críticas, e sua remoção irá comprometer a observabilidade do sistema.

Para exemplificar o processo de estimação de estados e a depuração de erros grosseiros, considere o sistema de três barras da Figura 2.4

Figura 2.4 - Exemplo de Depuração de erros



Fonte: (Guimaraens, 2015)

Com o sistema de medição: $P_2 = -4,07pu$, $P_{13} = 2,04pu$ e $P_{31} = -1,90pu$

E suas respectivas variâncias: $\sigma_2^2 = 0,04pu$, $\sigma_{13}^2 = 0,04pu$, $\sigma_{31}^2 = 0,04pu$.

A Equação pode ser resolvida seguindo o modelo linear:

$$\hat{z} = H \cdot \hat{x} \quad \begin{bmatrix} P_2 \\ P_{13} \\ P_{31} \end{bmatrix} = H \cdot \begin{bmatrix} \theta_2 \\ \theta_3 \end{bmatrix} \quad \begin{bmatrix} P_2 \\ P_{13} \\ P_{31} \end{bmatrix} = \begin{bmatrix} 150 & -100 \\ 0 & -100 \\ 0 & 100 \end{bmatrix} \cdot \begin{bmatrix} \theta_2 \\ \theta_3 \end{bmatrix}$$

A matriz de covariância de medidas será:

$$R = \begin{bmatrix} \sigma_2^2 & 0 & 0 \\ 0 & \sigma_2^2 & 0 \\ 0 & 0 & \sigma_2^2 \end{bmatrix} = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \cdot 10^3$$

A matriz de ganho será:

$$G^{-1} = \{[H^t]R^{-1}H\}^{-1} = \begin{bmatrix} 2,222 & 0,667 \\ 0,667 & 1,000 \end{bmatrix} \cdot 10^7$$

$$\hat{z} = HG^{-1}H^tR^{-1} \cdot z$$

$$\begin{aligned} &= \begin{bmatrix} 150 & -100 \\ 0 & -100 \\ 0 & 100 \end{bmatrix} \cdot \begin{bmatrix} 2,222 & 0,667 \\ 0,667 & 1,000 \end{bmatrix} \cdot \begin{bmatrix} 150 & 0 & 0 \\ -100 & -100 & 100 \end{bmatrix} \cdot \begin{bmatrix} 250 & 0 & 0 \\ 0 & 500 & 0 \\ 0 & 0 & 500 \end{bmatrix} \cdot \begin{bmatrix} -4,07 \\ 2,04 \\ -1,90 \end{bmatrix} \\ &= \begin{bmatrix} -4,07 \\ 2,04 \\ -1,90 \end{bmatrix} \end{aligned}$$

Agora podem-se calcular os resíduos:

$$r = z - \hat{z} = \begin{bmatrix} -4,07 \\ 2,04 \\ -1,90 \end{bmatrix} - \begin{bmatrix} -4,07 \\ -1,97 \\ -1,97 \end{bmatrix}$$

A matriz de covariância U pode então ser calculada:

$$\begin{aligned} U &= R - H(H^tWH)^{-1}H^t \\ &= \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix} \cdot 10^{-3} \\ &\quad - \begin{bmatrix} 150 & -100 \\ 0 & -100 \\ 0 & 100 \end{bmatrix} \cdot \begin{bmatrix} 2,222 & 0,667 \\ 0,667 & 1,000 \end{bmatrix} \cdot \begin{bmatrix} 150 & 0 & 0 \\ -100 & -100 & -100 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \end{aligned}$$

Por fim os resíduos normalizados podem ser calculados:

$$r_N(i) = \frac{|r(i)|}{\sqrt{U(i,i)}} = \begin{bmatrix} * \\ 2,21 \\ 2,21 \end{bmatrix}$$

Logo, percebe-se que o conjunto possui uma medida crítica P_2 que não pode ser retirada do sistema de medição e isto pode ser comprovado, pois a linha da matriz de

covariância referente a esta medida, é nula. Outra informação que pode ser extraída deste processo é a presença de um conjunto crítico(P_{13}, P_{31}), que pode ser constatado pela presença de resíduos idênticos.

Recapitulando. Após a análise das etapas que compõem o estimador de estado é possível concluir:

- Pode-se concluir então que para um sistema ser considerado observável, e a estimação possa ser devidamente realizada, este deve possuir, pelo menos, um número de medidas não redundantes iguais ao número de variáveis de estado do sistema.
- As medidas redundantes proporcionam estimativas mais precisas dos estados do sistema e auxiliam a detecção de erros grosseiros.

2.2 OTIMIZAÇÃO

Como visto em Kendirchuen e Ongsakul (2006), Ramesh *et al.* (2007) e Gamm *et al.* (2008), o problema de alocação de unidades medidoras pode ser formatado como um problema de otimização linear com restrições. Sendo assim, de acordo com Maculam e Fampa (2004) este tipo de problema pode ser formalizado como apresentado em (2.19) e (2.20), para casos de maximização:

$$\text{maximizar } f(x) = \sum_{j=1}^k c_j \cdot x_j \quad (2.1)$$

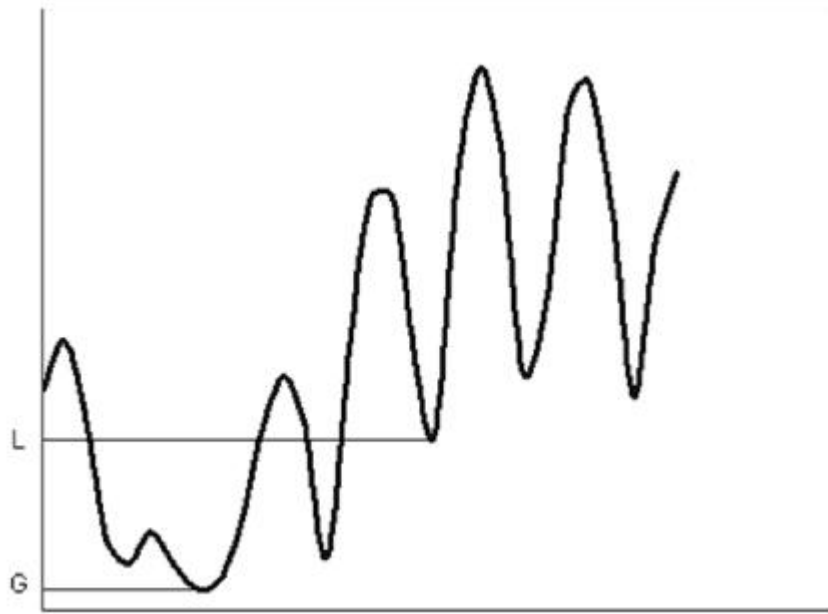
Sujeito a restrições de igualdade e desigualdade:

$$\begin{cases} \sum_{j=1}^k a_{ij} \cdot x_j \leq b_i \\ x_i \geq 0 \end{cases} \quad (2.12)$$

Onde a_{ij} , c_j e b_i são dados e x_i são variáveis de decisão. A função linear a ser otimizada $f(x)$ pode ser denominada função objetivo, econômica ou critério. Sendo assim, para casos de maximização, procuramos dentro de um espaço de solução delimitados pelas restrições, aquela que possuir o maior valor possível. Também é viável adaptar a função objetivo (2.21) e suas respectivas restrições (2.22) para casos de minimização em que o menor valor possível.

O melhor resultado possível dentro do conjunto de soluções restringido é chamado de ótimo global, máximo global para casos de maximização e mínimo global para casos de minimização. Agora, caso o resultado seja o melhor apenas em sua vizinhança, este é chamado de ótimo local. A Figura 2.5 exemplifica uma função objetivo de um caso de minimização que possui um mínimo local em L e um mínimo global em G.

Figura 2.5 - Ótimo global(G) e local(L) para casos de minimização



(Zanghi, 2016)

2.2.1 MÉTODOS DIRETOS

Como visto em Xu e Abur (2005), o problema de alocação de medidores pode ser abordado como um problema de otimização da forma $Ax=b$; e sendo assim, pode ser escrito como por um conjunto de equações lineares, ou seja, um sistema linear. Estas são entendidas como equações da forma:

$$a_1x_1 + a_2x_2 + \dots + a_nx_n = b \quad (2.11)$$

Sendo a_1, a_2, \dots, a_n e b números reais.

Então, um sistema linear terá a seguinte forma:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases} \quad (2.12)$$

Sistemas lineares são classificados pelo seu número de soluções, podendo então possuir: uma única solução, mais de uma solução ou nenhuma solução. Para encontrar estas

soluções, são utilizados métodos numéricos, que são divididos em dois grupos: métodos iterativos, que fornecem uma sequência convergente de aproximações da solução a partir de um chute inicial e métodos diretos, que fornecem uma solução exata do sistema com um número finito de soluções. Entre os métodos diretos existentes, serão citados: Decomposição de Gauss e por matrizes LU.

A eliminação de Gauss consiste em obter um sistema triangular superior equivalente a $\bar{A}.\bar{x} = \bar{b}$ usando operações elementares que preservem o determinante (não alterem a solução do sistema). Para isso utiliza-se da matriz aumentada $[A|b]$ representada em (2.13).

$$\left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ a_{31}^{(1)} & a_{32}^{(1)} & a_{33}^{(1)} & \cdots & a_{3n}^{(1)} & b_3^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & a_{n3}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array} \right] \quad (2.13)$$

O primeiro passo é anular todos os elementos abaixo do pivô $a_{1,1}^{(1)}$. Para isso é realizada a seguinte operação (2.14) para cada uma das linhas L da matriz, até que (2.15) seja obtido.

$$L_i^{(2)} \leftarrow L_i^{(1)} + m_{i1} L_1^{(1)} \text{ com } m_{i1} = -a_{i1}^{(1)} / a_{11}^{(1)} \quad (2.14)$$

$$\left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} & b_3^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array} \right] \quad (2.15)$$

O mesmo processo é repetido para zerar os valores abaixo dos pivôs $a_{22}^{(2)}, a_{33}^{(3)}, \dots e a_{nn}^{(n)}$. Quando (2.16) é obtido, basta utilizar um algoritmo de substituição regressiva para encontrar a solução do sistema.

$$\left[\begin{array}{cccc|c} a_{11}^{(1)} & a_{12}^{(1)} & a_{13}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ 0 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ 0 & 0 & a_{33}^{(3)} & \cdots & a_{3n}^{(3)} & b_3^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn}^{(n)} & b_n^{(n)} \end{array} \right] \quad (2.2)$$

A decomposição por matrizes LU consiste em escrever o sistema na forma matricial $\bar{A} \cdot \bar{x} = \bar{b}$ e decompor a matriz \bar{A} em duas \bar{L} e \bar{U} , sendo, respectivamente, uma triangular inferior com diagonal unitária e a outra triangular superior.

Exemplificando em um sistema 3x3 na forma matricial $\bar{A} \cdot \bar{x} = \bar{b}$:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (2.17)$$

A matriz A pode ser decomposta da seguinte forma LU:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{12} & 1 & 0 \\ l_{13} & l_{23} & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \quad (2.18)$$

Após a decomposição das matrizes, o sistema pode ser resolvido de acordo com (2.19) e (2.20).

$$Ly = b \quad (2.19)$$

$$Ux = y \quad (2.20)$$

2.2.2 HEURÍSTICA

Outra forma de solucionar o problema problemas de otimização é a implementação um método que realize uma busca dentro do espaço de soluções por uma resposta satisfatória para o problema. Segundo Zangui(2006), na implementação de métodos exatos para solucionar problemas de otimização, todas as soluções do problema devem ser enumeradas e avaliadas. O método deve ser então capaz de indicar qual a melhor solução entre todas as visitadas.

Para casos em que os processos de enumeração e de avaliação do espaço de soluções demandam um custo computacional elevado, torna-se necessária a aplicação de métodos não exatos que reduzem o número de soluções visitadas para a diminuição deste custo. Para este caso, existem métodos classificados como meta-heurísticos.

De acordo com Gendreau e Potvin (2010), as Meta-heurísticas são definidas como “métodos de solução que orquestram uma interação entre procedimentos de melhoria local e estratégias de nível mais alto para criar um processo capaz de escapar de ótimos locais e realizar uma busca robusta do espaço de soluções”. Existem diversos métodos meta-heurísticos, dentre eles são citados: Recozinhamento Simulado (Simulated Annealing), Busca Tabu, Colônia de Formigas e Algoritmos Genéticos. Este último é de maior importância para este trabalho e será explicado mais a fundo em 2.2.2.1.

2.2.2.1 ALGORITMOS GENÉTICOS

Segundo Meza (2006, pag. 174) “Algoritmos genéticos são uma família de modelos computacionais inspirados na evolução, os quais modelam uma solução para um problema específico, em uma estrutura de dados como cromossomo. Neles se aplicam operadores genéticos, que recombina estas estruturas preservando informações importantes durante o processo de busca pela melhor solução.” Nesta seção do trabalho será apresentada a fundamentação teórica para utilização desse tipo de algoritmo.

2.2.2.1.1 ESPAÇO DE SOLUÇÕES

Inicialmente para se utilizar um AG é necessário estabelecer a função objetivo, que será nosso principal objeto de otimização. A partir dela, podemos definir nosso Espaço de Soluções, um conjunto que conterá todas as soluções possíveis para nossa função. Cada uma das possíveis soluções pode ser "marcada" pelo seu valor (ou adequação) para o problema. Os algoritmos genéticos permitem que possamos encontrar as soluções mais adequadas para resolver o problema a partir de uma pesquisa feita em amostras deste conjunto, sem a necessidade de que ele seja todo percorrido.

A possibilidade de se conseguir chegar à melhor solução possível, a partir de apenas da análise de soluções, é dita como um ponto forte desse tipo de algoritmo, pois não é necessário o entendimento do funcionamento da função objetivo. Contudo, como é dito por (Arcos, 2017) justamente pela dependência da probabilidade e de um conjunto limitado de variáveis analisadas, a melhor solução para o problema, dita solução ótima, não é assegurada.

2.2.2.1.2 REPRESENTAÇÃO DE SOLUÇÕES

Como dito por Ramesh et AL.(2007), “Algoritmos genéticos são implementados como uma simulação computacional em que uma população de representações abstratas (chamadas cromossomos ou genótipos) de soluções candidatas (chamados indivíduos, criaturas ou fenótipos) de uma problema de otimização evolui para uma solução melhor”.

Sendo assim, é preciso escolher a maneira mais adequada de se codificar os cromossomos e esta codificação depende do problema em questão, **devendo** conseguir representar todas as informações importantes das soluções a serem analisadas. Algumas destas formas de codificação são:

- **Binária**

A codificação binária consiste em representar cada cromossomo como um vetor binário, em que cada elemento deste vetor denota a presença (1) ou ausência (0) de uma determinada característica. A Figura 2.6 apresenta um exemplo desse tipo de representação.

Figura 2.6- Representação binária dos cromossomos

Cromossomo A	7	5	1	2	3	4	7	9
Cromossomo B	9	8	5	7	7	7	6	2

Fonte: autoria própria

Este tipo de representação pode gerar cromossomos de grande tamanho na representação de problemas multivariáveis, podendo demandar um grande tempo computacional durante as operações genéticas, (Meza, 2006)

Exemplo de Problema: Problema da Mochila (Obitko, 1998)

O problema: É dada uma lista de coisas com preços e tamanhos. É fornecido o valor da capacidade da mochila. Escolha as coisas de forma a maximizar o valor **daquelas** que cabem dentro da mochila, sem ultrapassar sua capacidade.

Codificação: Cada bit é usado para dizer se a coisa correspondente está ou não na mochila.

- **Real**

Também é possível representar cada cromossomo como um vetor de números reais, em que cada elemento apresenta o valor de uma determinada característica. “Esta representação possui uma interpretação mais natural para o ser humano, **possuindo** cromossomos compactos e com maior precisão” (Meza, 2006). A Figura 2.7 apresenta um exemplo de representação real dos cromossomos.

Figura 2.7-Representação Real de cromossomos

Cromossomo A	7	5	1	2	3	4	7	9
Cromossomo B	9	8	5	7	7	7	6	2

Fonte: autoria própria

Exemplo de Problema: Problema do Caixeiro Viajante (Travelling Salesman Problem-TSP) (Obitko, 1998)

O problema: São dadas cidades e as distâncias entre elas. O caixeiro viajante tem que visitar todas elas, sem viajar mais do que o necessário. A solução do problema consiste

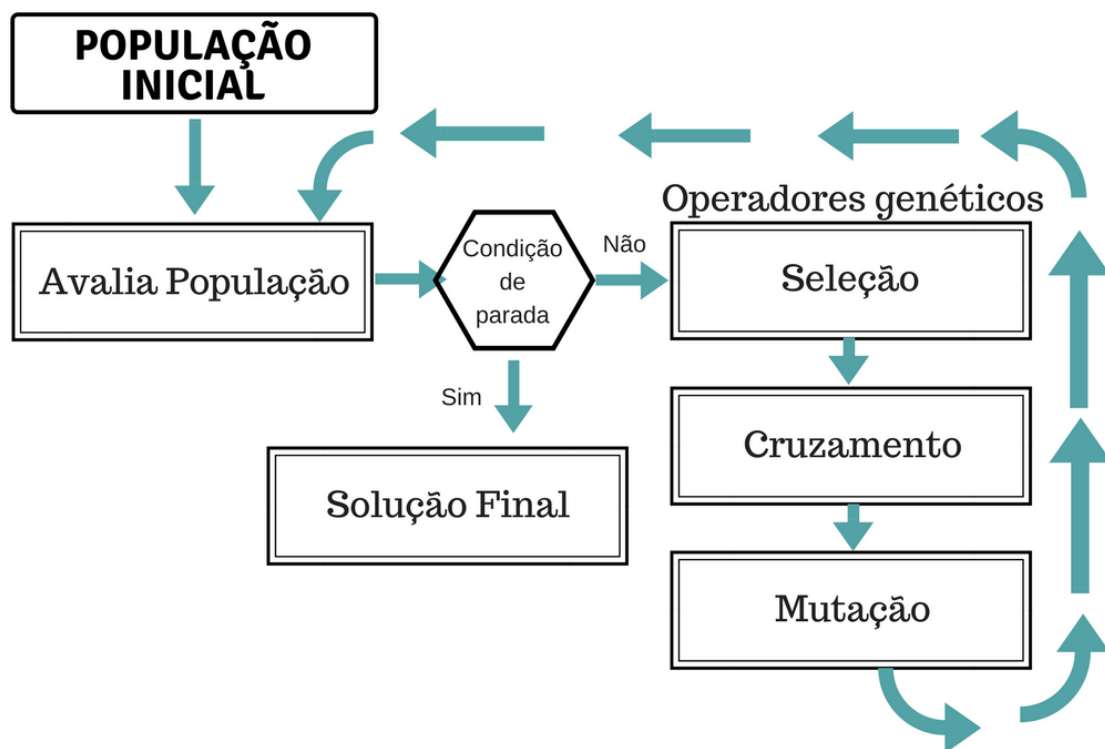
em encontrar a sequência de cidades em que as viagens devem ser feitas de forma que a distância percorrida seja a mínima possível.

Codificação: Os cromossomos descrevem a ordem em que o caixeiro visitará as cidades.

2.2.2.2 PRINCÍPIO BÁSICO DE FUNCIONAMENTO

Nesta seção será analisado cada um dos processos do funcionamento de um algoritmo genético, apresentados na Figura 2.8.

Figura 2.8-Fluxograma de um Algoritmo genético



Fonte: autoria própria

Inicialmente é preciso estabelecer nossa população inicial, selecionando um conjunto de indivíduos portadores do código genético capaz de representar todos os valores da solução, relevantes para o nosso problema de otimização da função objetivo. Em outras palavras, nossa população deve ser formada de elementos do conjunto solução da nossa função de otimização.

Uma população muito pequena pode ter problema de não conseguir englobar o número de soluções necessárias para chegar à solução desejada, enquanto uma população inicial muito grande pode gerar um custo computacional muito elevado.

- **Avaliação da População**

A próxima etapa de avaliação da população é a parte do algoritmo em que avaliamos a aptidão dos elementos de nossa população. Em seguida, checamos a condição de parada do algoritmo. Até que esta seja realizada, o algoritmo entrará em *loop* realizando as operações genéticas, que serão aprofundadas no seguinte tópico. É estabelecido que a cada vez que tais operações são realizadas, uma geração é concluída e, normalmente, o número de gerações é estabelecido como o fator de parada.

- **Operadores genéticos**

Os próximos processos do algoritmo, as chamadas operações genéticas (seleção, cruzamento e mutação), são utilizados para manipular o código genético de cada indivíduo; com a finalidade de gerar uma população nova e possivelmente mais apta que a anterior.

Nesta fase, pode-se utilizar o chamado elitismo, que consiste em marcar os melhores indivíduos (a chamada elite) e automaticamente, levá-los até a próxima geração. Desta forma, nenhuma informação contida nestes elementos será perdida no decorrer das gerações.

- a) Seleção**

Nesta parte do algoritmo, os elementos da população mais aptos serão selecionados para serem pais de uma nova população. Existem diversos métodos para selecioná-los e entre eles estão Torneio e Roleta, que serão aprofundados neste trabalho.

- a) Torneio*

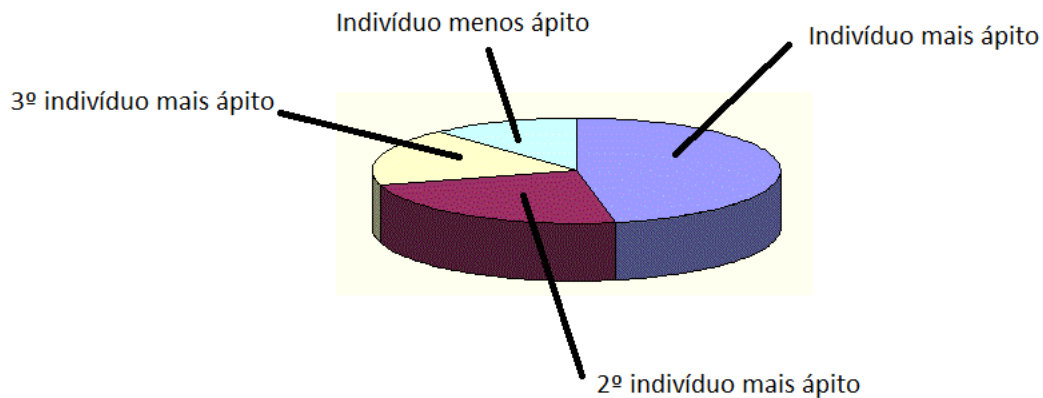
Este método se baseia em selecionar aleatoriamente indivíduos da população e identificar o melhor desta amostra. Isto é repetido até que todos os reprodutores sejam definidos.

- b) Roleta*

Em alguns casos é preciso que o melhor indivíduo seja selecionado, porém não de forma exclusiva, para que não se perca a diversidade da população e esta não convirja para um ótimo local. Um indivíduo menos apto pode possibilitar um cruzamento ou uma mutação futura, que irá gerar um elemento ainda melhor que os já presentes em gerações anteriores.

Neste método de seleção, os indivíduos são selecionados de forma aleatória, porém ponderados com pesos relacionados à suas respectivas aptidões. A Figura 2.9 pode ajudar a interpretar este método, representando uma roleta em que a área é proporcional à aptidão do indivíduo. Caso uma bolinha seja jogada na roleta, ela terá mais chances de parar em um indivíduo mais apto.

Figura 2.9- Exemplo Roleta



Fonte: adaptação (Obitko, 1998)

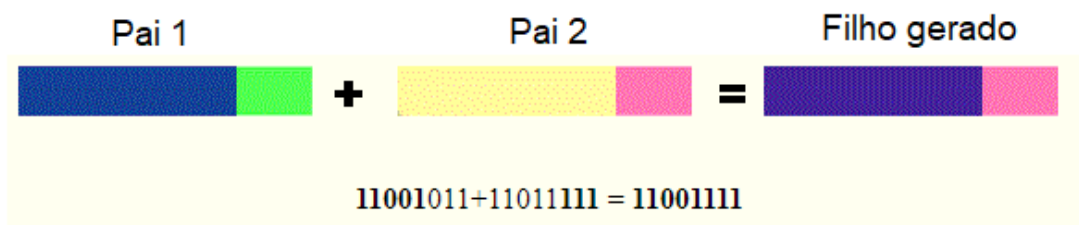
b) Cruzamento

O cruzamento tem a finalidade de realizar o intercâmbio do material genético entre indivíduos, certificando de que as principais características sejam passadas entre gerações. Por causa disso, é importante assegurar-se de que os indivíduos mais aptos se reproduzam com mais frequência. Duas técnicas de cruzamento aprofundadas neste trabalho são ponto de cruzamento e cruzamento uniforme.

a) Ponto de cruzamento

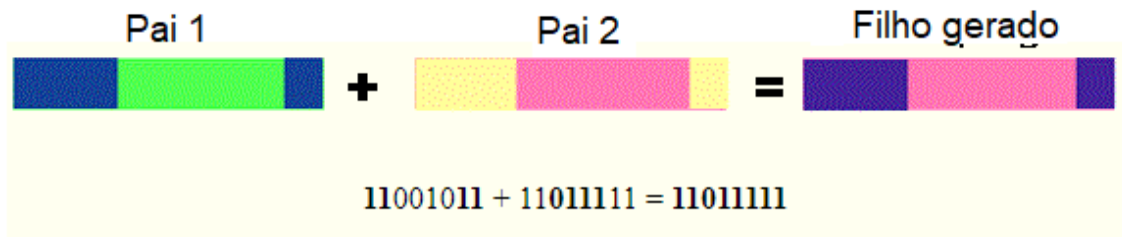
Neste caso, um ponto de cruzamento é escolhido aleatoriamente. No caso de uma representação binária, o filho que será gerado deste cruzamento terá a parte antes do ponto escolhido herdada do primeiro pai e a parte após o ponto, do segundo pai. Esta operação pode ser realizada com um ponto, como mostra a Figura 2.10 ou com mais de um como mostra a Figura 2.11.

Figura 2.10-Cruzamento com um único ponto



Fonte: autoria própria

Figura 2.11-Cruzamento com dois pontos

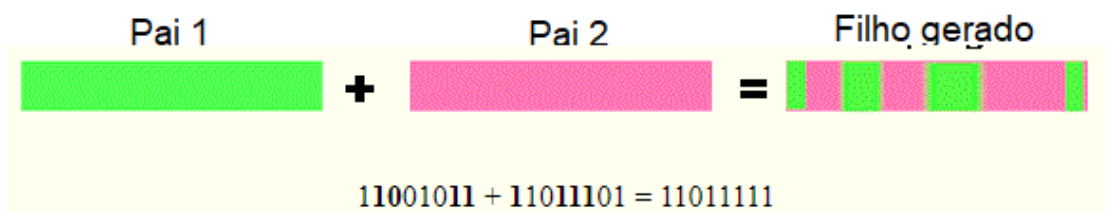


Fonte: autoria própria

b) Cruzamento uniforme

Neste caso, a informação herdada pelo filho é escolhida de forma aleatória, podendo vir do primeiro ou do segundo pai. A Figura 8 representa um exemplo desta operação para uma representação uniforme.

Figura 2.12-Exemplo de cruzamento uniforme

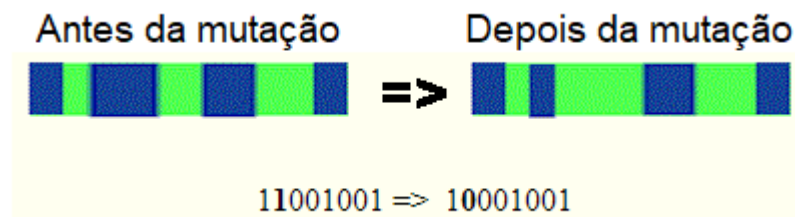


Fonte: autoria própria

c) Mutação

A mutação tem a intenção de prevenir que todas as soluções do problema da população caiam em um ponto ótimo local. Esta é aplicada em alguns elementos providos pelo cruzamento e tem como objetivo alterar uma ou mais informações herdadas, para adicionar diversidade à geração seguinte. No caso da codificação, os valores de alguns bits são invertidos, como representado na Figura 2.13.

Figura 2.13-Exemplo de mutação em representação binária



Fonte: autoria própria

- **Solução Final**

Finalmente, após a condição de parada ser alcançada, espera-se que a melhor solução possível esteja presente na população final alcançada. Este elemento será selecionado dentro desta população de acordo com sua aptidão em relação ao problema estabelecido.

2.3 ALOCAÇÃO DE MEDIDORES

Os dados processados pelo estimador de estado, como dito anteriormente, são providos normalmente por UTR's que são unidades de medição de subestações capazes de medir valores tanto de magnitude de tensão e corrente nas malhas, quanto fluxo e injeção de potência ativa e reativa.

Contudo, para diminuir gastos é preciso evitar a instalação de UTR's. Sendo assim, um esquema de medição ótimo deve ser planejado, que posicione o mínimo de unidades remotas possível e que forneça todas as medidas necessárias da rede para torná-la observável para que o estimador de estados opere devidamente.

2.3.1 OBSERVAÇÃO DE SISTEMAS POR PMU'S

Apesar não serem o foco do trabalho, Synchronized Phase Measurement Unit (PMU) são unidades de medição citadas em diversos artigos e será utilizada o exemplo de (Xur & Abur, 2005) para exemplificar como alocar estas unidades de forma a observar os estados de todas as barras. PMU's provém dois tipos de medição: tensão fasorial na barra e corrente fasorial no ramo. Será considerado que cada unidade possuirá canais suficientes, capazes de medir estas grandezas para todas as filiais que incidem na barra em que a mesma foi instalada. A Figura 2.1 Exemplo estimação de estados apresenta a alocação ótima para o sistema de 14 barras do IEEE.

Desse modo, é possível afirmar que UTR's podem garantir a observabilidade total da rede, se forem instaladas nos mesmos pontos que seriam instalados PMU's.

2.3.3 LEVANTAMENTO HISTÓRICO

Alocação de medidores é um tópico bastante abordado em problemas de engenharia, nesta seção serão analisados alguns artigos que abordaram este tema no decorrer dos anos.

a) MÉTODO PARA ALOCAÇÃO ÓTIMA DE PMU's

Xu e Abur (2005) propõem, em seu relatório, um método numérico baseado em programação inteira para resolver o problema de alocação de PMU's. A seguinte formulação (2.1) para uma rede de n barras é proposta para o problema.

$$\begin{cases} \min \sum_{i=1}^n w_i \cdot x_i \\ f(X) \geq 1 \end{cases} \quad (2.1)$$

Onde:

- x_i é um vetor binário de decisão em que é definido: 1 se um PMU for alocado na barra i , 0 caso contrário.
- w_i é o custo de instalação do PMU na barra i .
- $f(x)$ uma função vetor em que as entradas são não nulas, caso a tensão na barra correspondente seja calculável. A condição é que esta função possua todos os valores, contidos no vetor, maiores do que um.

Resolvendo o somatório, obtemos o custo total de instalação dos PMU's selecionados. Com a adição das restrições a solução final garante uma rede completamente observável, minimizando o custo de instalação.

Com isso, é apresentada no relatório a construção dessa formulação para três casos, onde a rede: 1-não possua medidores prévios, 2- já possua medidores de fluxo e 3- já possua tanto medidores de fluxo quanto de injeção.

- **Caso1:**

Neste caso, para se formular as restrições à matriz de conectividade, A deve ser estabelecida da seguinte forma (2.2).

$$A_{k,m} = \begin{cases} 1 & \text{se: } k = m \text{ ou se } k \text{ e } m \text{ estão conectadas} \\ 0 & \text{em outros casos} \end{cases} \quad (2.2)$$

A matriz A pode ser diretamente obtida ao transformar a matriz de admitância da rede em uma matriz binária. (2.1) representa um exemplo dessa matriz para o sistema de 14 barras do IEEE.

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad (2.3)$$

Sendo assim, as restrições podem ser estabelecidas:

$$f(X) = A \cdot X = \begin{cases} f_1 = x_1 + x_2 + x_5 & \geq 1 \\ f_2 = x_1 + x_2 + x_3 + x_4 + x_5 & \geq 1 \\ f_3 = x_2 + x_3 + x_4 & \geq 1 \\ f_4 = x_2 + x_3 + x_4 + x_5 + x_7 + x_9 & \geq 1 \\ f_5 = x_1 + x_2 + x_4 + x_5 + x_6 & \geq 1 \\ f_6 = x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_7 = x_4 + x_7 + x_8 + x_9 & \geq 1 \\ f_8 = x_7 + x_8 & \geq 1 \\ f_9 = x_4 + x_7 + x_9 + x_{10} + x_{14} & \geq 1 \\ f_{10} = x_9 + x_{10} + x_{11} & \geq 1 \\ f_{11} = x_6 + x_{10} + x_{11} & \geq 1 \\ f_{12} = x_6 + x_{12} + x_{13} & \geq 1 \\ f_{13} = x_6 + x_{12} + x_{13} + x_{14} & \geq 1 \\ f_{14} = x_9 + x_{13} + x_{14} & \geq 1 \end{cases} \quad (2.4)$$

Esta representação mostra que se pelo menos um medidor for posto em uma das barras {1,2,5}, f_1 é satisfeita, tornando a barra 1 observável.

- **Caso 2:**

Este caso é exemplificado no artigo [O](#), considerando a existência de uma medida de fluxo entre as barras 5 e 6 do sistema de 14 barras. Neste caso, a existência dessa medida levará a uma modificação nas restrições referentes a estas barras. A existência da medida de fluxo entre as barras possibilita o cálculo da tensão em uma das barras, caso a mesma tensão na outra barra já seja conhecida. Sendo assim, as restrições referentes a cada uma das barras pode ser unida, formando uma nova restrição como é mostrado em

$$\begin{cases} f_5 = x_1 + x_2 + x_4 + x_5 + x_6 & \geq 1 \\ f_6 = x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_{5_new} = f_5 + f_6 = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \end{cases} \quad (2.5):$$

$$\begin{cases} f_5 = x_1 + x_2 + x_4 + x_5 + x_6 & \geq 1 \\ f_6 = x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_{5_new} = f_5 + f_6 = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \end{cases} \quad (2.5)$$

Logo, caso um fasor seja observável, o outro também [O](#) será. Aplicando essa modificação no sistema de 14 barras obtemos (2.6):

$$f(X) = \begin{cases} f_1 = x_1 + x_2 + x_5 & \geq 1 \\ f_2 = x_1 + x_2 + x_3 + x_4 + x_5 & \geq 1 \\ f_3 = x_2 + x_3 + x_4 & \geq 1 \\ f_4 = x_2 + x_3 + x_4 + x_5 + x_7 + x_9 & \geq 1 \\ f_{5_new} = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_7 = x_4 + x_7 + x_8 + x_9 & \geq 1 \\ f_8 = x_7 + x_8 & \geq 1 \\ f_9 = x_4 + x_7 + x_9 + x_{10} + x_{14} & \geq 1 \\ f_{10} = x_9 + x_{10} + x_{11} & \geq 1 \\ f_{11} = x_6 + x_{10} + x_{11} & \geq 1 \\ f_{12} = x_6 + x_{12} + x_{13} & \geq 1 \\ f_{13} = x_6 + x_{12} + x_{13} + x_{14} & \geq 1 \\ f_{14} = x_9 + x_{13} + x_{14} & \geq 1 \end{cases} \quad (2.6)$$

- **Caso 3:**

Neste caso, considera-se que já existam medidas de injeção e de fluxo na rede, porém estas não são suficientes para tornar o sistema observável. O artigo também explica como este caso é tratado, utilizando um exemplo no sistema de 14 barras.

Considerando uma injeção zero na sétima barra, o artigo afirma que se três das tensões das barras {4, 7, 8, 9} são conhecidas, a quarta pode ser calculada pela aplicação lei de Kirshoff, na barra 7. Duas formas de se construir essa formulação são apresentadas, uma não linear e outra topológica.

1. Método Não Linear para representar sistemas possuidores de medidas de fluxo.

Este método consiste em modificar as restrições das barras vizinhas de forma a gerar uma nova restrição não linear. Sendo assim, a formulação das barras vizinhas {4, 8, 9} são escritas das seguintes formas:

$$\begin{aligned} f_4 &= x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + f_7 \cdot f_8 \cdot f_9 & \geq 1 \\ f_8 &= x_7 + x_8 + f_4 \cdot f_7 \cdot f & \geq 1 \\ f_9 &= x_4 + x_7 + x_9 + x_{10} + x_{14} + f_4 \cdot f_7 \cdot f_8 & \geq 1 \end{aligned} \quad (2.7)$$

Por se tratar de uma representação binária, trata-se '+' como 'ou' e '.' como 'e'. Dessa forma, substituindo e simplificando o sistema (7), obtemos (8).

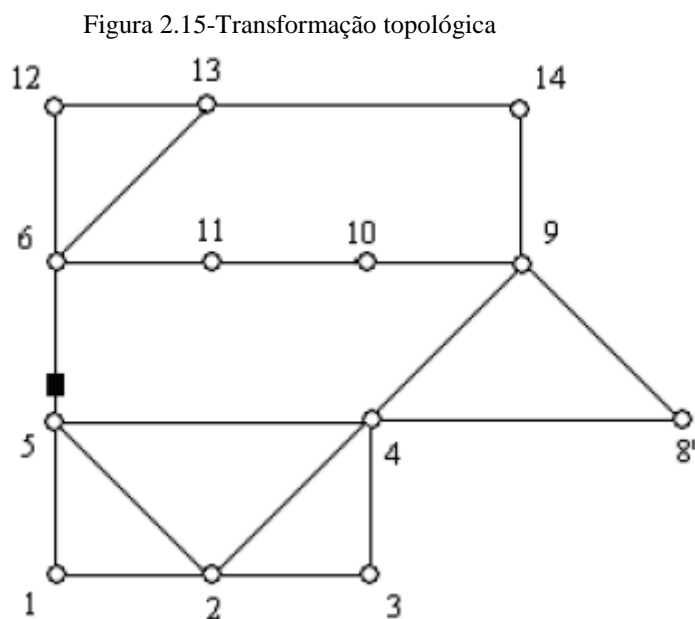
$$f(X) = \begin{cases} f_1 = x_1 + x_2 + x_5 & \geq 1 \\ f_2 = x_1 + x_2 + x_3 + x_4 + x_5 & \geq 1 \\ f_3 = x_2 + x_3 + x_4 & \geq 1 \\ f_4 = x_2 + x_3 + x_4 + x_5 + x_7 + x_9 + x_8 \cdot x_{10} + x_8 \cdot x_{14} & \geq 1 \\ f_{5_new} = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_8 = x_4 + x_7 + x_8 + x_9 & \geq 1 \\ f_9 = x_4 + x_7 + x_9 + x_{10} + x_{14} + x_2 \cdot x_8 + x_3 \cdot x_8 + x_5 \cdot x_8 & \geq 1 \\ f_{10} = x_9 + x_{10} + x_{11} & \geq 1 \\ f_{11} = x_6 + x_{10} + x_{11} & \geq 1 \\ f_{12} = x_6 + x_{12} + x_{13} & \geq 1 \\ f_{13} = x_6 + x_{12} + x_{13} + x_{14} & \geq 1 \\ f_{14} = x_9 + x_{13} + x_{14} & \geq 1 \end{cases} \quad (2.8)$$

Assim a restrição 7 é removida do sistema, pois os efeitos da injeção já estão sendo considerados indiretamente pelos termos que possuem multiplicação adicionados às barras vizinhas. O relatório, por fim, comenta que este método por possuir uma parte não linear é

complicado e custoso computacionalmente. Sendo assim, o próximo método é mais aconselhado para sistemas com um maior número de injeções.

2. Método Topológico representar sistemas possuidores de medidas de fluxo.

Este método consiste de uma transformação topológica do sistema, unindo a barra que possui um valor de injeção nula ou conhecida com uma de suas vizinhas. Esta ideia é baseada no fato de que se as tensões em todas as barras vizinhas forem conhecidas, a tensão na barra que possui injeção pode ser calculada pela lei de Kirshoff. Este método é exemplificado unindo a barra com a injeção conhecida 7 com a barra vizinha 8; criando assim uma nova barra 8'. A modificação topológica é mostrada na Figura 2.15 e a alteração nas restrições é mostrada em (2.9).



$$f(X) = \begin{cases} f_1 = x_1 + x_2 + x_5 & \geq 1 \\ f_2 = x_1 + x_2 + x_3 + x_4 + x_5 & \geq 1 \\ f_3 = x_2 + x_3 + x_4 & \geq 1 \\ f_4 = x_2 + x_3 + x_4 + x_5 + x_{8'} + x_9 & \geq 1 \\ f_{5_new} = x_1 + x_2 + x_4 + x_5 + x_6 + x_{11} + x_{12} + x_{13} & \geq 1 \\ f_{8'} = x_4 + x_{8'} + x_9 & \geq 1 \\ f_9 = x_4 + x_{8'} + x_9 + x_{10} + x_{14} & \geq 1 \\ f_{10} = x_9 + x_{10} + x_{11} & \geq 1 \\ f_{11} = x_6 + x_{10} + x_{11} & \geq 1 \\ f_{12} = x_6 + x_{12} + x_{13} & \geq 1 \\ f_{13} = x_6 + x_{12} + x_{13} + x_{14} & \geq 1 \\ f_{14} = x_9 + x_{13} + x_{14} & \geq 1 \end{cases} \quad (2.9)$$

Caso a nova barra fictícia, criada pela união de duas barras, seja selecionada para alocação de PMU, o medidor poderá ser posto em qualquer uma ou em ambas as barras que a compõem. Para tomar esta decisão é preciso realizar uma análise topológica a fim de checar observabilidade do sistema. O artigo conclui que este método é mais rápido e não irá adicionar nenhuma operação não linear às restrições.

- **Considerando perdas de PMU's**

Também é estudada pelo relatório a aplicação de medidores de retaguarda. Apesar dos PMU's serem bastante confiáveis, estes são susceptíveis a falhas como qualquer outro equipamento. Sendo assim, mesmo que haja a perda única de qualquer um dos PMU's presentes, outro medidor deve garantir a total observabilidade do sistema.

Após o cálculo dos medidores mínimos que garantem observabilidade, todas as barras que já os possuem são desconsideradas das funções de restrição. Ao realizar os cálculos novamente, são obtidas as novas barras que devem ser alocados os PMU's de retaguarda.

- **Resultados**

O artigo realizou os testes sob as condições citadas anteriormente para os sistemas dos IEEE de 14, 30, 57 e 118 barras. Os resultados são mostrados na Tabela 2.1 e na Tabela 2.2.

Tabela 2.1-Resultados obtidos sem considerar perda de PMU

	Sem medições prévias		Condiderando injeção				
			Não-Linear		Topologico		Barras com injeção
Sistema IEEE	Nº de Medidores	Barras com medidores	Nº de Medidores	Barras com medidores	Nº de Medidores	Barras com medidores	
14	4	2,6,7,9	3	2,6,9	3	2,6,9	7
30	10	2, 4, 6, 9, 10, 12, 15,18, 25, 27	7	3, 5, 10, 12, 18, 23, 27	8	2, 3, 6, 10, 12, 18, 23, 27	6, 9, 11, 25, 28
57	17	1, 4, 7, 9, 15, 20, 24, 25, 27, 32, 36, 38, 39, 41, 46, 50, 53	13	1, 6, 9, 15, 20, 25, 27, 32, 38, 47, 50, 53, 56	12	1, 5, 9, 14, 15, 20, 25, 28, 32, 50, 53, 56	4, 7, 11, 21, 22, 24, 26, 34, 36, 37, 39, 40, 45, 46, 48
118	32	2, 5, 9, 11, 12, 17, 21, 24, 25, 28, 34, 37, 40, 45, 49, 52, 56, 62, 63, 68, 73, 75, 77, 80, 85, 86, 90, 94, 101, 105, 110, 114	29	2, 8, 11, 12, 15, 19, 21, 27, 31, 32, 34, 40, 45, 49, 52, 56, 62, 65, 72, 110 94, 101, 105,80, 85, 86, 90,75, 77,	28	2, 8, 11, 12, 17, 21, 25, 28, 33, 34, 40, 45, 49, 52, 56, 62, 72, 75, 77, 80, 85, 86, 90, 94, 101, 105, 110, 114	5, 9, 30, 37, 38, 63, 64, 68, 71, 81

Tabela 2.2-Resultados obtidos considerando perda de PMU

Sistema IEEE	Sem medições prévias		Condiderando injeção				
	Nº de Medidores	Barras com medidores	Não-Linear		Topologico		Barras com injeção
			Nº de Medidores	Barras com medidores	Nº de Medidores	Barras com medidores	
14	9	1, 2, 3, 6, 7, 8, 9, 10, 13	7	1, 2, 4, 6, 9, 10, 13	7	3, 5, 10, 12, 18, 23, 27	7
30	22	2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 21, 23, 25, 26, 27, 29	17	2, 3, 4, 5, 6, 10, 12, 13, 15, 17, 18, 19, 21, 23, 24, 27, 29	17	1, 2, 3, 4, 7, 8, 9, 13, 14, 15, 16, 19, 20, 21, 23, 24, 28, 29	6, 9, 11, 25, 28
57	35	1, 2, 4, 6, 7, 9, 11, 12, 13, 15, 19, 20, 22, 24, 25, 26, 27, 29, 30, 32, 33, 34, 36, 37, 38, 39, 41, 44, 46, 47, 50, 51, 53, 54, 56	30	2, 3, 4, 5, 6, 10, 1, 2, 6, 7, 9, 10, 12, 14, 15, 18, 20, 21, 24, 25, 27, 29, 31, 32, 33, 34, 37, 38, 41, 44, 47, 49, 50, 53, 54, 56	26	1, 2, 3, 4, 7, 8, 9, 1, 2, 4, 5, 9, 12, 13, 14, 15, 18, 20, 23, 25, 28, 29, 30, 32, 33, 35, 38, 41, 50, 51, 53, 54, 56	4, 7, 11, 21, 22, 24, 26, 34, 36, 37, 39, 40, 45, 46, 48
118	72	1, 2, 4, 5, 6, 9, 10, 11, 12, 15, 16, 17, 19, 21, 22, 24, 25, 27, 28, 29, 30, 32, 34, 35, 37, 39, 40, 41, 43, 45, 46, 49, 50, 51, 52, 54, 56, 59, 62, 63, 64, 66, 68, 70, 71, 73, 75, 76, 77, 78, 80, 81, 83, 85, 86, 87, 89, 90, 92, 94, 96, 100, 101, 105, 106, 108, 110, 111, 112, 114, 116, 117	65	1, 2, 6, 8, 9, 11, 12, 13, 14, 15, 17, 19, 20, 21, 23, 24, 27, 28, 31, 32, 34, 35, 37, 40, 41, 43, 45, 46, 49, 51, 52, 54, 56, 57, 59, 62, 65, 67, 68, 70, 72, 75, 76, 77, 78, 80, 83, 85, 86, 87, 89, 90, 92, 94, 96, 100, 101, 105, 106, 108, 110, 111, 112, 114, 117	65	1, 2, 6, 8, 9, 11, 12, 13, 14, 15, 17, 19, 20, 21, 23, 24, 27, 28, 31, 32, 34, 35, 37, 40, 41, 43, 45, 46, 49, 51, 52, 54, 56, 57, 59, 62, 65, 67, 68, 70, 72, 75, 76, 77, 78, 80, 83, 85, 86, 87, 89, 90, 92, 94, 96, 100, 101, 105, 106, 108, 110, 111, 112, 114, 117	5, 9, 30, 37, 38, 63, 64, 68, 71, 81

b) ALGORITMO HÍBRIDO (GA/SA) PARA ALOCAÇÃO DE MEDIDAS

Kendirchuen e Ongsakul (2006) propuseram em seu artigo o uso de um algoritmo híbrido entre genético e de pareamento simulado (GA/SA). Neste, procura-se o melhor posicionamento de medidores de fluxo e de injeção em uma rede de barras, sendo o mínimo global de medidas já conhecido como $N-1$, sendo N o número de barras. O artigo apresenta a seguinte definição “Um sistema é dito observável se $H\theta=0$ e $A\theta=0$, sendo H a matriz de

medidas, A matriz de conexões entre barras e θ a matriz contendo as variáveis de estado do sistema. Sabendo disso, a função objetivo e suas restrições são mostradas em (2.8).

$$\begin{cases} fit = \sum medidas \\ \sum medidas > N - 1 \\ H\theta = 0 \\ A\theta = 0 \end{cases} \quad (2.8)$$

O objetivo da otimização é minimizar o custo da instalação, alocando o menor número de medidas possível. O número de medidores não pode ser menor que N-1 e as restrições de observabilidade também devem ser atendidas.

A representação das soluções possui o seguinte formato: N bits que representam a existência de medidas em cada uma das barras e M bits; sendo M o número de linhas entre as barras que representam a existência de medidas entre elas. A Figura 2.16 apresenta um exemplo desta representação para um sistema de seis barras.

Figura 2.16-Representação da solução

N=6					M=10									
0	1	1	0	1	0	0	1	0	1	0	0	1	0	1

Fonte: autoria própria

O algoritmo genético consiste gerar uma população com diversas soluções para o problema e combiná-las por um número de gerações até que a melhor solução seja alcançada. Para um sistema de 14 barras foi obtida a combinação de medidas como mostra a Figura 2.17, sendo 8 medidas de injeção e 5 medidas de fluxo.

d) CRITÉRIOS DE ALOCAÇÃO DE PMU's EM SISTEMAS DE POTÊNCIA

Gamm *et al.* (2008) propuseram em seu artigo o uso de um algoritmo genético que leva em consideração alguns critérios para a melhor alocação de PMU's. Tais critérios são:

- **Melhoria na detecção de medições ruins.**

Estimadores de estado retornam a resposta certa se não houver a presença de erros de medidas. Logo, detectar medições ruins se torna um dos principais problemas durante a estimação. Deve haver um número significativo de medidas redundantes para que assim seja possível obter variáveis de estado, que reflitam propriamente o estado do sistema.

Medidas que quando removidas tornam o sistema não observável são chamadas de medidas críticas. Essas medidas podem ser descobertas analisando a observabilidade do sistema. Erros de medição desse tipo de medida não podem ser detectados por métodos convencionais. Quanto menor for a redundância do esquema de medição, maior o número de medidas críticas. Dessa forma, um sistema aquisição de dados confiável não deve possuir medidas críticas.

- **Adicionar o maior número de Medidas.**

Uma unidade PMU instalada pode fornecer a medida de tensão fasorial da barra em que foi instalada e de suas adjacentes. Quanto maior for o número de ligações e menor for o número de medidas SCADA que uma barra possuir, maior será o número de medições obtidas pela alocação de um PMU em tal barra.

Assim sendo, como pode ser visto em (2.9), o número de novas medições, a serem adicionadas pela adesão de k PMU's a um sistema, é definido pelo somatório de todas as medidas adicionadas pela unidade à barra i , menos a quantidade de medidas que já eram presentes naquela barra.

$$n_{def} = \sum_{i=1}^k (K_{PMUi} - K_{SCADA}) \quad (2.9)$$

A aplicação do algoritmo genético do artigo consiste de uma junção da alocação de medidas SCADA e medidas de PMU. As medidas SCADA, inicialmente, possibilitam que o sistema seja completamente observável, porém com pouca redundância, permitindo a existência de algumas medidas críticas. Para diminuir o número dessas medidas, PMU's são adicionados às barras como uso de um algoritmo genético com a função objetivo (2.10).

$$Fit = \frac{n_{def}}{n_{mc} + n_{cc} + \sum_i^k c_i \cdot n_{PMUi} + const} \quad (2.10)$$

Onde:

n_{mc} = nº de medidas críticas;

n_{cc} = nº de conjuntos críticos;

n_{PMU_i} = nº de PMU's;

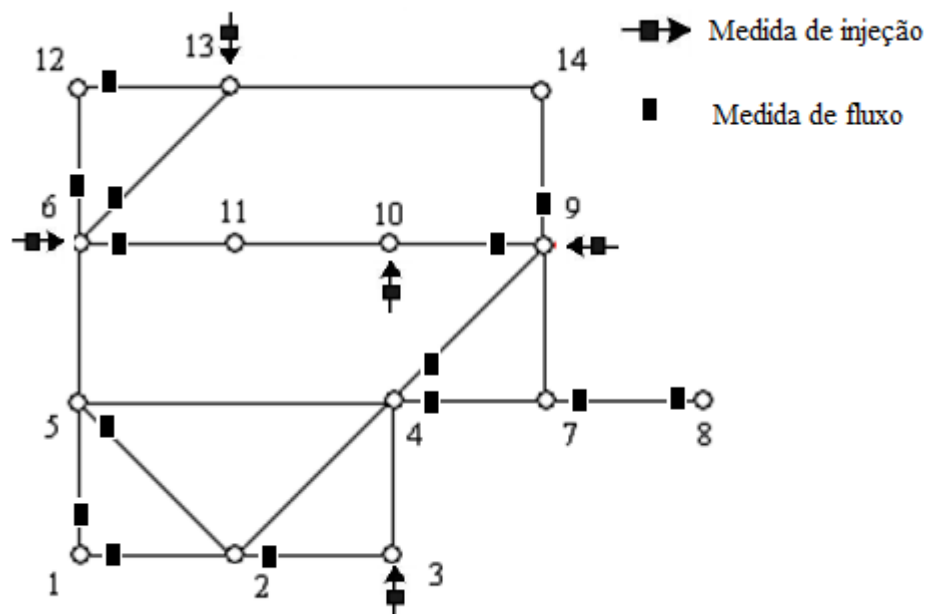
c_i = custo de instalação de pmu's;

n_{defi} = nº de novas medições adicionadas pela alocação de PMU's

O objetivo do algoritmo é maximizar esta função objetivo. A população inicial é definida aleatoriamente e as barras em que os PMU's serão alocados são representadas por bit's. O valor de Fit é calculado para cada solução e combinado por gerações até encontrar a melhor das soluções para o problema.

Na aplicação do método para um sistema de 14 barras com 19 medidas, representado na Figura 2.18, concluiu-se então, que para todos os critérios serem estabelecidos, é preciso a alocação de PMU's nas barras {2, 4, 6, 7, 9, 10}.

Figura 2.18- Resultado da alocação de medidas



Fonte: adaptação (Xur & Abur, 2005)

e) MÉTODO NÃO ITERATIVO POR ABORDAGEM GRÁFICA

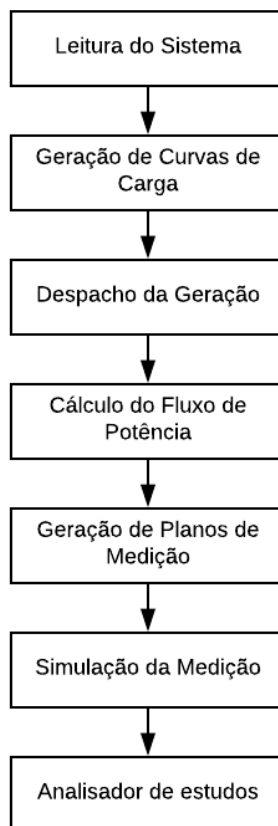
Motiyani e Chudasama (2011) apresentam uma abordagem gráfica para resolver o problema de alocação de medidores. O método proposto é não iterativo e consiste em alocar medidores, dando preferência às barras com maior número de ligações. Este foi aplicado aos

sistemas de 14 e 30 barras, obtendo assim, uma alocação de 44 e 90 medidas respectivamente, tornando o sistema observável e fornecendo redundâncias para casos de erros de medição e falhas de medidores.

2.4 PROGRAMA SimSEP

O Simulador de Sistemas Elétricos de Potência – SimSEP tem por objetivo simular um sistema elétrico de potência ao longo de um dia e assim simular a ocorrência de faltas no sistema; testar modelagens de circuitos de usinas geradoras, medidores, e diversos outros equipamentos; analisar métodos numéricos e heurísticas nas diversas etapas de funcionamento de um sistema elétrico de potencia, tais como previsão de curvas de cargas, análise de planos de medição, localização/identificação de faltas, detecção/identificação de erros grosseiros de medição, análise de incertezas de equipamentos por exemplo. Desenvolvido no Matlab, o programa simula um sistema elétrico desde sua geração até o consumo, levando em consideração sua topologia. Suas simulações geradas podem possuir um fim tanto didático, quanto para fins de simulação de sistemas e de algoritmos. Figura 2.19 mostra o fluxograma simplificado dos processos do simulador.

Figura 2.19-Fluxograma simplificado do SimSEP



Fonte: autoria própria

Para inicializar o SimSEP, é preciso fornecer uma entrada no formato *Common Data Format* (CDF) que irá fornecer todas as informações necessárias da topologia da rede a ser estudada. Em seguida, as curvas de carga são geradas de acordo com as informações obtidas pelo CDF da entrada. O próximo passo para o programa será calcular o fluxo de potência da rede, pelo método de Newthom Rhapson desacoplado, para diversos instantes de tempo de acordo com o intervalo desejado. A próxima fase consiste em gerar um plano de medidores ao sistema, para que estes simulem as observações efetuadas pelas medições do sistema real simulado no fluxo de potencias. Assim, a partir do plano dos medidores, com suas respectivas simulações das medidas e levando em consideração os respectivos erros e incertezas das medições, uma estimação de estados é realizada na rede simulada.

O método de alocação ótima proposto por este trabalho fará parte do módulo “GERADOR DE PLANO DE MEDIÇÃO” (quarta etapa do sistema) e fornecerá um plano de observação do sistema em simulação.

3 METODOLOGIA PROPOSTA

O programa 300 de Esparta é um algoritmo evolutivo, baseado no conceito da sobrevivência do mais apto e com a mesma estrutura de um algoritmo genético; porém nele são implementados alguns métodos que agilizam a sua convergência para uma solução desejada, sem dispensar a necessidade de analisar várias regiões do espaço de soluções. Inicialmente foi desenvolvido pelo professor Marcio Guimaraens e tem como finalidade encontrar uma alocação ótima de UTR's para uma dada topologia. Levando em consideração o que foi explicado anteriormente sobre observação de sistemas no tópico 2.3.2: a alocação de uma UTR em uma barra, garante a observabilidade da mesma e de todas que são conectadas a ela. Sendo assim, os estados observados por estas unidades de medidas podem ser determinadas com a seguinte equação:

$$b = A \cdot x \quad (3.1)$$

Onde:

- A: A matriz A é uma matriz n x n, sendo n o número de barras da topologia de rede, onde os medidores serão alocados. É uma matriz binária em que os elementos unitários representam a existência da conexão entre barras, exemplo: se o elemento $A_{2,3}=1$, isso significa que as barras dois e três são conectadas entre si..
- x: É um vetor binário que informa a posição do medidor a partir da posição dos valores '1'.
- b: É a matriz que representa quantas medidas são possibilitadas em cada barra pelo arranjo de medidas.

Se todos os elementos da matriz b forem maiores que zero, pode-se garantir que todas as barras da topologia em questão estão sendo observadas por pelo menos um medidor alocado. Então, se esta condição é satisfeita, a rede é dita observável. Sabendo disso, o problema em questão pode ser considerado como multiobjetivo, com as funções (3.2) e (3.3) a serem otimizadas, levando em consideração a restrição (3.4):

$$\min \left(\sum_{i=1}^{n^{\circ} \text{ de barras}} x_i \right) \quad (3.2)$$

$$\max \left(\sum_{i=1}^{n^{\circ} \text{ de barras}} b_i \right) \quad (3.3)$$

$$\bar{b} \geq 1 \quad (3.4)$$

Como o custo de medição do sistema está diretamente relacionado ao número de UTR's instaladas, em (3.2), desejamos o menor número de medidores possíveis no arranjo. Já em (3.3), procuramos maximizar o número de estados observáveis, pois como visto por Ramesh et al. (2007), quanto maior for a redundância de medições na rede, mais fácil será detectar medidas com presença de erros grosseiros e mais segura se encontrará a rede em casos de defeitos nos medidores. Sendo assim, a cada geração o programa procura primeiramente soluções com menor número de medidores e em seguida, entre as já encontradas, a que possibilitar o maior número de vezes que o estado é observado por uma medida.

Inicialmente já se encontravam prontas as versões V0, V1, V2 e V3. As versões seguintes V4, V5, V6 e V7 foram implementadas durante a realização deste trabalho. As versões antigas foram inicialmente implementadas no SciLab e por praticidade foram traduzidas para o Matlab, mesma linguagem em que as novas foram criadas.

3.1 VERSÕES DO ALGORITMO

Nesta sessão serão explicados mais a fundo os avanços em cada uma das versões do programa.

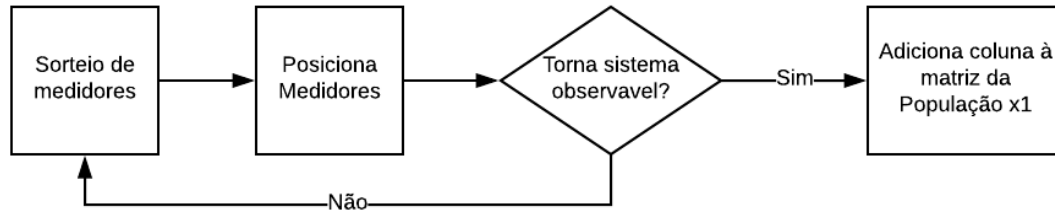
3.1.1 VERSÃO 0:

A versão inicial do programa foi desenvolvida pelo Professor Márcio Guimaraens. Esta versão consiste de um algoritmo genético híbrido com as seguintes etapas: Criação da população inicial, Seleção, Cruzamento e Mutação, conforme se verá abaixo:

- **Criação da População inicial**

Nesta etapa, a população inicial é gerada como mostra a Figura 3.1. Primeiramente, o número de unidades UTR's a serem alocados é sorteado e em seguida, também são escolhidas aleatoriamente as barras em que as unidades serão alocados (informadas como 1 no vetor x). Após esta alocação, é avaliado se o arranjo é uma solução válida que permita que o sistema seja totalmente observável. Se a restrição (3.4) é satisfeita, a solução é válida; sendo armazenada como uma coluna da matriz $x1$, que representa a população. Este processo é repetido até que toda a matriz da população seja preenchida.



Figura 3.1-Criação da população inicial



Fonte: autoria própria

Ao fim desta etapa, terminamos com uma matriz x1 referente à população e uma matriz b referente à observabilidade de cada solução, como é mostrada na **Erro! Fonte de referência não encontrada.**

Figura 3.2-Matriz da população e da observabilidade

		x1: Matriz da população												b: Matriz de quantidade de medidas																
nº de barras	{	1	0	0	0	0	0	1		0	1	0	0	0	1	0		2	2	2	1	1	2		2	1	2	1	1	2
		1	1	1	1	0	1	0		1	0	1	0	1	0	1		3	3	2	2	2	3		3	2	3	2	2	3
		0	0	0	0	0	1	1	...	0	1	0	1	1	1	1		2	2	1	2	1	2	...	1	2	1	2	1	2
		1	1	0	1	1	0	0		1	0	1	0	0	0	0		4	5	4	4	4	5		4	4	4	4	3	4
		0	1	1	0	1	1	1		0	1	0	1	1	0	1		4	4	3	3	2	2		3	3	3	2	2	2
		1	1	1	1	0	0	1		1	1	0	1	0	0	1		2	3	3	2	4	3		3	2	2	3	5	3
		1	1	1	1	1	1	1	...	1	1	1	0	1	1	1		3	3	2	3	3	2	...	2	3	2	4	2	2
		0	0	0	0	0	0	0		0	0	1	1	1	0	0		1	1	1	1	1	1		1	1	1	2	1	2
		1	1	1	1	1	1	1		1	1	1	1	0	0	1		4	3	3	3	4	2		3	4	2	3	3	3
		0	0	0	0	0	0	0		0	0	0	1	1	1	0		2	1	1	1	2	2		1	1	1	2	3	2
		1	0	0	0	1	1	0		0	0	1	1	1	0	1		2	1	1	1	1	1		1	1	1	1	3	2
		0	1	1	1	1	0	0	...	1	0	1	1	0	0	1		1	2	2	2	2	1		2	2	1	2	3	1
		0	0	0	0	1	1	1		0	0	1	1	1	1	0		2	2	3	2	3	1		3	3	1	2	4	2
		1	0	1	0	1	0	1		1	0	0	1	1	0	0		2	1	2	1	3	2		3	2	1	2	3	2
																														
		Tamanho da população												Tamanho da população																

Após a população inicial ser estabelecida, o algoritmo chega à etapa chamada de Era Genética, composta pela Seleção, Cruzamento, Mutação e Sucessão. Estes três processos são repetidos por um número preestabelecido de gerações.

- **Seleção**

Nesta etapa, a melhor solução da geração é encontrada e armazenada em um vetor *evolution*, que contém os melhores valores de cada geração. Como cada uma das colunas de x1 representa uma solução presente na população, o número de UTR's instaladas referentes a cada solução será a soma das colunas de x1. Sendo assim, vetor som é criado para armazenar cada um destes valores como mostra (3.5):

$$som(col) = \sum_{i=1}^{nbarras} x1(i, col) \quad (3.5)$$

Da mesma forma, o número total de medidas que o a solução possibilita na rede é encontrado a partir da soma de todos os valores do vetor b. O vetor *somb* é criado como mostra (3.6):

$$somb(col) = \sum_{i=1}^{nbarras} b(i, col) \quad (3.6)$$

Com estes vetores estabelecidos para encontrar a melhor solução, primeiramente é preciso percorrer o vetor *som* atrás do índice do elemento com menor número de UTR's; em seguida, encontrar o equivalente a este valor na matriz *x1*. Agora que o valor mínimo de medidores presente na população é conhecido, o próximo passo é descobrir qual o maior número estados observados que esta quantidade de UTR's permite. Para isso, é preciso percorrer novamente o vetor *som*, encontrar cada elemento que possui a menor quantidade de UTR's, comparar a quantidade de medidas possibilitada por cada um destes e armazenar sempre que uma solução melhor for encontrada.

Nesta etapa do programa, também são selecionados os reprodutores que serão utilizados no cruzamento. Estes são encontrados da seguinte forma: enquanto o número de reprodutores não for alcançado, o vetor *som* é percorrido até se encontrar o índice do elemento com menor número de UTR's. Assim que este é encontrado, o valor equivalente a este índice em *x1* é armazenado no vetor de reprodutores *m5s*, uma variável que representa o número de reprodutores encontrados é incrementada, a busca se reinicializa para percorrer o vetor *som* novamente e o valor que já foi encontrado na busca anterior é penalizado, sendo incrementado de forma “absurda” para que não seja selecionado novamente.

Como pode ocorrer o caso, em que na busca anterior, todos os reprodutores sejam ocupados por soluções com o menor número de UTR's, antes mesmo de se percorrer toda a população, não é garantido que as soluções com maior número de medidas sejam preservadas. Para resolver este problema, o programa percorre mais uma vez o vetor *som*, procurando menor valor registrado no vetor *m5s* e comparando o valor da quantidade de medidas referente, armazenada no vetor *somb*. Desta forma é garantido que o vetor *m5s* contenha as soluções com menor número de UTR's e maior número de medições. Ao final da etapa, possuímos uma matriz como mostra a Figura 12.

Figura 3.3-Matriz de reprodutores *m5s*

ms5: matriz de reprodutores

	0	0	0	0	0	0	0	1	0	1
	1	1	1	1	0	1	1	0	1	0
	0	0	0	0	0	0	1	1	0	1
	0	1	0	1	1	1	0	0	1	0
	0	1	1	0	1	1	1	1	0	1
	0	1	1	1	0	0	0	1	1	1
	1	1	1	1	1	0	1	1	1	1
	0	0	0	0	0	1	0	0	0	0
	0	1	1	1	1	1	1	1	1	1
	1	0	0	0	0	1	0	0	0	0
	0	0	0	0	1	0	1	0	0	0
	0	1	1	1	1	1	0	0	1	0
	1	0	0	0	1	0	1	1	0	0
	0	0	1	0	1	0	0	1	1	0

nº de barras

nº de reprodutores

Fonte: autoria própria

A forma como esta seleção foi implementada se mostrou muito custosa por precisar percorrer a matriz da população e as matrizes de somas diversas vezes. A seleção foi aprimorada na versão 5 e será explicada no tópico referente a esta.

- **Cruzamento**

Nesta Etapa é criada a matriz de filhos com o objetivo de gerar melhores soluções para compor a próxima geração. Todos os elementos da matriz *m5s* são combinados dois a dois, gerando uma matriz de dimensão $(1 + Rep) * Rep * 2$, sendo o *Rep* o número de reprodutores. Para realizar estas combinações, um ponto de cruzamento é estabelecido no meio das soluções, desta forma ela é dividida em dois (A1/A2 e B1/B2), podendo assim gerar as combinações A1B1; A2B2; A1B2; A2B1. A Figura 3.4 apresenta um exemplo desse cruzamento.

Figura 3.4-Exemplo de cruzamento

				A1B1	A2B2	A1B2	A2B1
	Sol_A	Sol_B		filho1	filho2	filho3	filho4
1	0	1		0	1	0	1
	1	0		1	1	1	1
2	1	0		1	0	0	1
	1	0		0	0	0	0

- **Mutação**

Nesta etapa, as mutações são criadas para aumentar a diversidade da população. Os mutantes são adicionados no final da matriz filhos (aumentando o número de colunas) e são gerados da seguinte forma: um elemento do vetor de melhores da população ($ms5$) é sorteado e é gerado um mutante para cada barra que possua uma UTR, com a exceção de cada um dos gerados, possui uma barra a menos. A Figura 3.5 explica como é feita esta etapa.

Figura 3.5-Exemplo de mutação

solução		mut.1	mut.2	mut.3
0		0	0	0
1		0	1	1
1		1	0	1
0		0	0	0
0		0	0	0
1		1	1	0

Fonte: autoria própria

- **Sucessão**

Após a matriz de filhos ser completada com mutantes, a matriz de observabilidade $bfilhos$ deve ser gerada de forma análoga a de b (calculada anteriormente para a população). As matrizes $somf$ e $somabf$ devem ser calculadas também de forma equivalente a som e $somb$ respectivamente. Com estes valores calculados, as colunas de $bfilhos$ são analisadas e são excluídos todos os filhos que não satisfizerem (3.4).

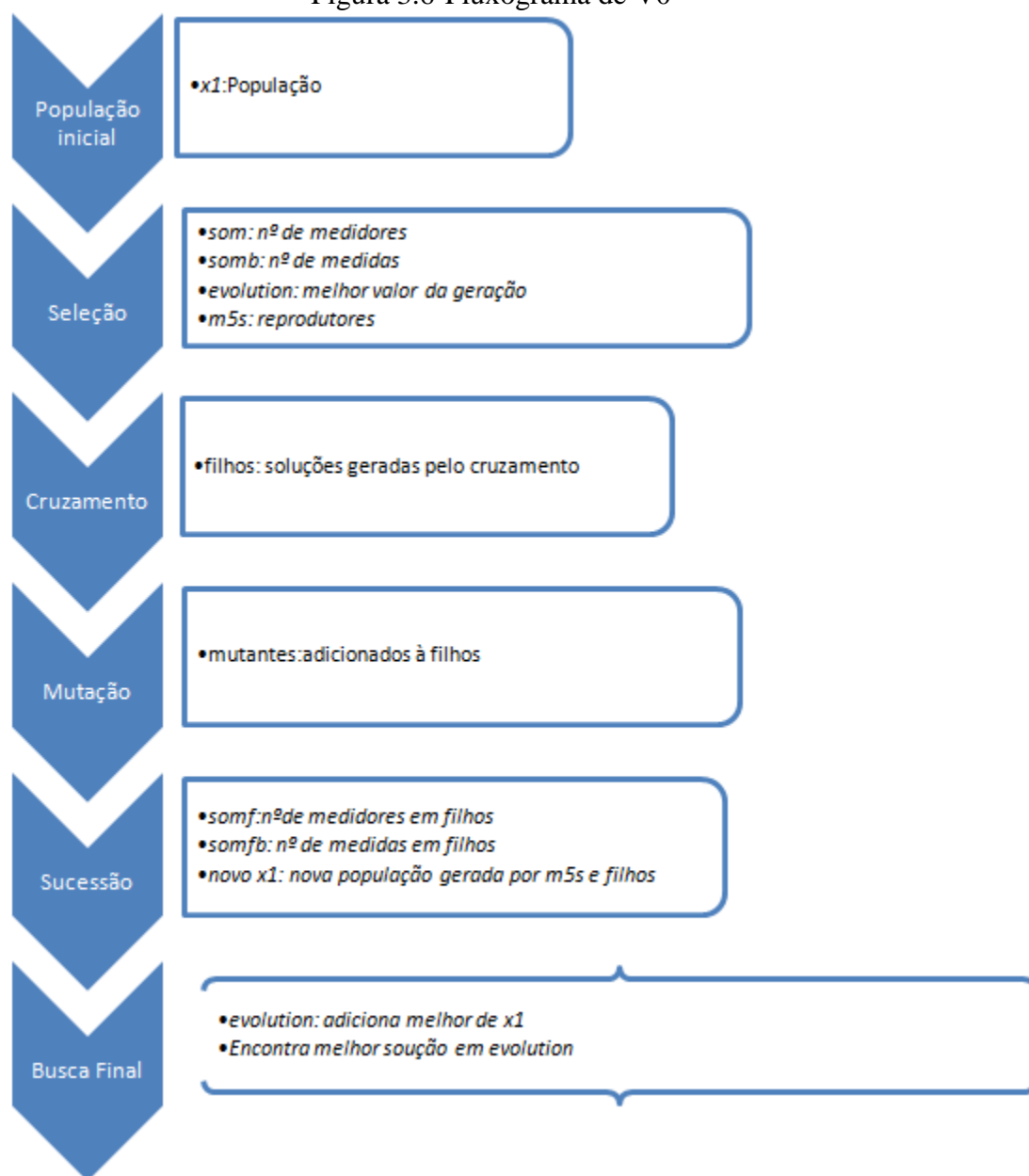
Em seguida, é preciso modificar a matriz de população para a próxima geração. Os filhos que apresentaram resultados válidos são adicionados à nova população da seguinte forma: até que o tamanho da população menos cinco seja alcançado, o vetor de $somf$ é percorrido até encontrar o índice do elemento com menor número de medidores, a coluna da matriz de filhos referente a este índice é adicionada à matriz $x1$, o valor de $somf$ referente a este índice é penalizado, de forma que não possa ser escolhido novamente e a busca recomeçada. Os cinco elementos restantes da população são ocupados pelos cinco melhores elementos do vetor $ms5$ (a chamada elite).

- **Busca final**

Após a era genética ser repetida até o número de gerações ser alcançado, acontece a busca local final. Nesta etapa, o melhor valor de $x1$ é encontrado da mesma forma que é feita na etapa de seleção e em seguida, o melhor valor da matriz *evolution*, que armazenou todos os

melhores valores de cada geração é encontrado e este será o melhor resultado alcançado pelo algoritmo. A Figura 3.6 apresenta um fluxograma resumido das etapas e suas variáveis.

Figura 3.6-Fluxograma de V0



Fonte: autoria própria

3.1.2 VERSÃO 1:

A versão um do programa teve como objetivo aprimorar a criação da população inicial. Nela foram adicionadas as margens de sorteio inferior (*p1*) e superior (*p2*) para aprimorar o algoritmo.

- **Margens de sorteio de barras *p1* e *p2***

Pode ocorrer o caso em que, durante a população inicial, o número sorteado de UTR's seja muito pequeno, quando isso acontece, há chances de não existir nenhuma combinação possível de medidores que possa tornar o sistema observável. Também é possível acontecer do valor sorteado ser muito próximo ao número de barras. A ocorrência deste caso pode atrasar a convergência do algoritmo, já que procuramos o menor valor.

Para ajudar o algoritmo, então foram implementados as margens p_1 e p_2 . Estes variáveis percentuais têm a função de delimitar o intervalo de sorteio do número de UTR's na população inicial. Exemplo: para o sistema de 30 barras, caso seja definido $p_1=0.3$ e $p_2=0.7$, só serão geradas soluções para a população inicial que possuam um número de UTR's entre 9 e 21.

3.1.3 VERSÃO 2

A versão dois também teve como objetivo aprimorar a criação da população inicial e nela foi implementada a matriz de ranking de barras.

- **Ranking das barras**

Analisando o método realizado por Motiyani e Chudasama (2011) e os resultados obtidos por Xu e Abur (2005), foi possível perceber que a barra com maior número de ligações sempre estava presente na melhor solução encontrada; o que faz total sentido, pois uma UTR nesta barra possibilitará um número maior de estados observados do que em qualquer outra barra.

Sabendo disso, foi implementada a matriz *rankB*. Esta matriz possui em primeira coluna as conexões que cada uma das barras da topologia possui. Estes valores são obtidos a partir da soma de todos os valores das linhas da matriz A. A matriz de *ranks* também possui uma segunda coluna formada pelas somas das conexões das barras anteriores. Esta soma permite que um sorteio por roleta seja realizado, de forma que as barras com maior número de ligações tenham mais chances de serem selecionadas para alocação de medidores na geração população inicial. A Figura 3.7 mostra a matriz de *ranks* de uma topologia de 14 barras.

Figura 3.7-Matriz de rankings

$A =$	1	1	0	0	1	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	0	1	1	1	0	0	0	0	0	0	0	0	0	0
	0	1	1	1	1	0	1	0	1	0	0	0	0	0
	1	1	0	1	1	1	0	0	0	0	0	0	0	0
	0	0	0	0	1	1	0	0	0	0	1	1	1	0
	0	0	0	1	0	0	1	1	1	0	0	0	0	0
	0	0	0	0	0	0	1	1	0	0	0	0	0	0
	0	0	0	1	0	0	1	0	1	1	0	0	0	1
	0	0	0	0	0	0	0	0	1	1	1	0	0	0
	0	0	0	0	0	1	0	0	0	1	1	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	1	1	0
	0	0	0	0	0	1	0	0	0	0	0	1	1	1
	0	0	0	0	0	0	0	0	1	0	0	0	1	1
	0	0	0	0	0	0	0	0	1	0	0	0	1	1

RankB	
Conexões	Soma
2	0
4	2
2	6
5	8
4	13
4	17
3	21
1	24
4	25
2	29
2	32

Fonte: autoria própria

3.1.4 VERSÃO 3

A versão três tinha como objetivo registrar taxas de diversidade e de mutação da população no decorrer do código. Contudo, ela não foi finalizada, por isso não será aprofundada neste trabalho.

3.1.5 VERSÃO 4

Esta versão e as próximas não se encontravam prontas. A versão 4 teve como objetivo implementar o registro das taxas de diversidade e de mutação, como também adicionar a função da Grande Migração.

- **Taxa de Diversidade e de Mutação**

Para possibilitar que a analise as alterações no decorrer das populações e assim implementar formas de se aprimorar o programa, foram adicionadas ao algoritmo o cálculo das taxas de diversidade e de mutação.

a) *Taxa de diversidade*

A taxa de diversidade é calculada da seguinte forma: cada elemento da População é comparado com um elemento de uma lista de elementos diferentes; caso este elemento não esteja nesta lista, a variável que armazena o número de elementos diferentes é incrementada e este é adicionado à lista. Ao fim dessa contagem, a taxa de diversidade é calculada a partir da Equação:

$$Taxa\ de\ diversidade = \frac{(n^{\circ}\ de\ diferentes)}{(Tamanho\ da\ População)} \quad (3.6)$$

Como cada elemento da população é composto por um vetor, a comparação, caso feita de forma muito rigorosa, entre todos os seus componentes e os da lista de diferentes, é computacionalmente muito custosa. Dessa forma, foi necessário encontrar outras formas de se codificar estes elementos para facilitar esta comparação.

Uma das formas escolhidas foi criar um id para cada elemento formado a partir da soma dos índices das barras que contém medidor, como é apresentada na Figura 3.8.

Figura 3.8-Taxa de diversidade

Barra	Medidor	nº de medidas
1	0	19
2	1	
3	0	
4	0	
5	0	
6	1	
7	1	
8	0	
9	1	
10	0	
11	0	
12	0	
13	0	
14	0	

1ª Representação
id=1x2+1x6+1x7+1x9=24

Fonte: autoria própria

b) Taxa de mutação

Para avaliar a eficiência da mutação no algoritmo genético, foi adicionado ao código o cálculo da taxa de mutação. Para realizar esta operação, os mutantes gerados que são adicionados à população de filhos são marcados em um vetor auxiliar chamado *mut*, e em seguida, são contabilizados quantos destes mutantes vão para a próxima geração. O cálculo da taxa de mutação pode então ser representado por:

$$taxa\ de\ mutação = \frac{n^{\circ}\ de\ mutantes\ na\ população\ que\ avançam}{tamanho\ da\ população}$$

- **Grande migração**

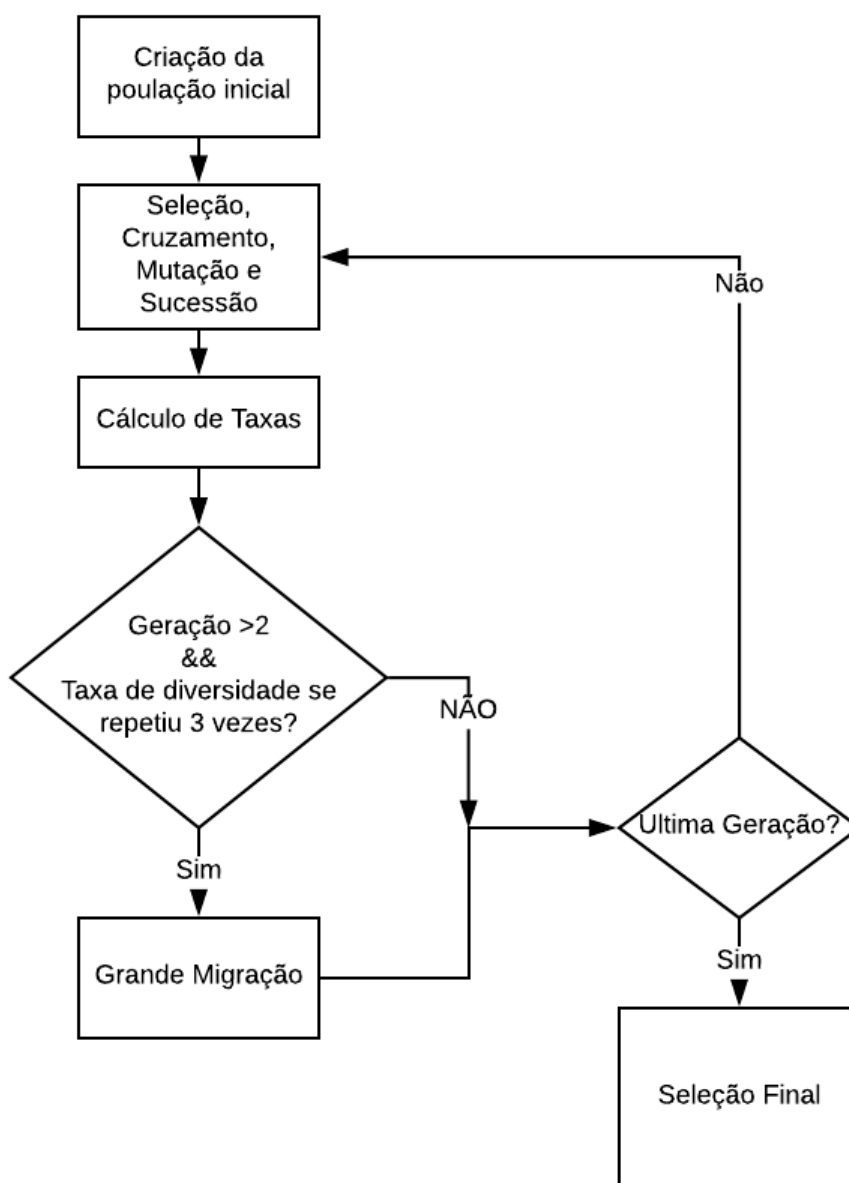
As versões anteriores do programa apresentavam problemas de convergência prematura para um ponto ótimo local não desejado. Esse tipo de problema em que existem diversos ótimos locais, pode ser resolvido com a adição de uma técnica de diversificação da população, como explicado por Zangui(2006). Sendo assim, na quarta versão do programa foi implementado um algoritmo de reinicialização da população chamado de Grande Extinção.

Inspirado no conceito da biologia migração, que denota o movimento de informação genética de uma população a outra, este algoritmo tem como objetivo aumentar a diversidade no processo evolutivo através da inclusão de novos indivíduos, selecionados pela sua qualidade (aptidão) e diversidade, assim que ela se encontrar presa em um ponto.

A cada geração do algoritmo, a taxa de diversidade da população é calculada e a partir da terceira geração, o código compara este valor com os três últimos. Caso a diversidade não se altere por três gerações consecutivas, é concluído que a população convergiu para um ponto ótimo local que dificilmente irá sair dele apenas utilizando o cruzamento e uma simples mutação. Quando isso ocorre, a Grande Migração é chamada, toda a população, menos a elite, é sorteada novamente, de forma análoga à V0. E assim, a diversidade é incrementada bruscamente. É esperado que em uma próxima geração o cruzamento de um indivíduo selecionado para a reprodução com um gerado durante a migração, gere uma solução mais apta para o problema.

A grande migração é implementada após mutação. A Figura 3.9 mostra a inclusão desta função e do cálculo das taxas no fluxograma do algoritmo.

Figura 3.9-Fluxograma da versão quatro



Fonte: autoria própria

3.1.6 VERSÕES ALTERNATIVAS

Com a finalidade de deixar o código mais legível e diminuir o tempo do programa, algumas versões alternativas foram experimentadas. Apesar destas não serem bem sucedidas, elas foram inspiradoras para a confecção de futuras versões.

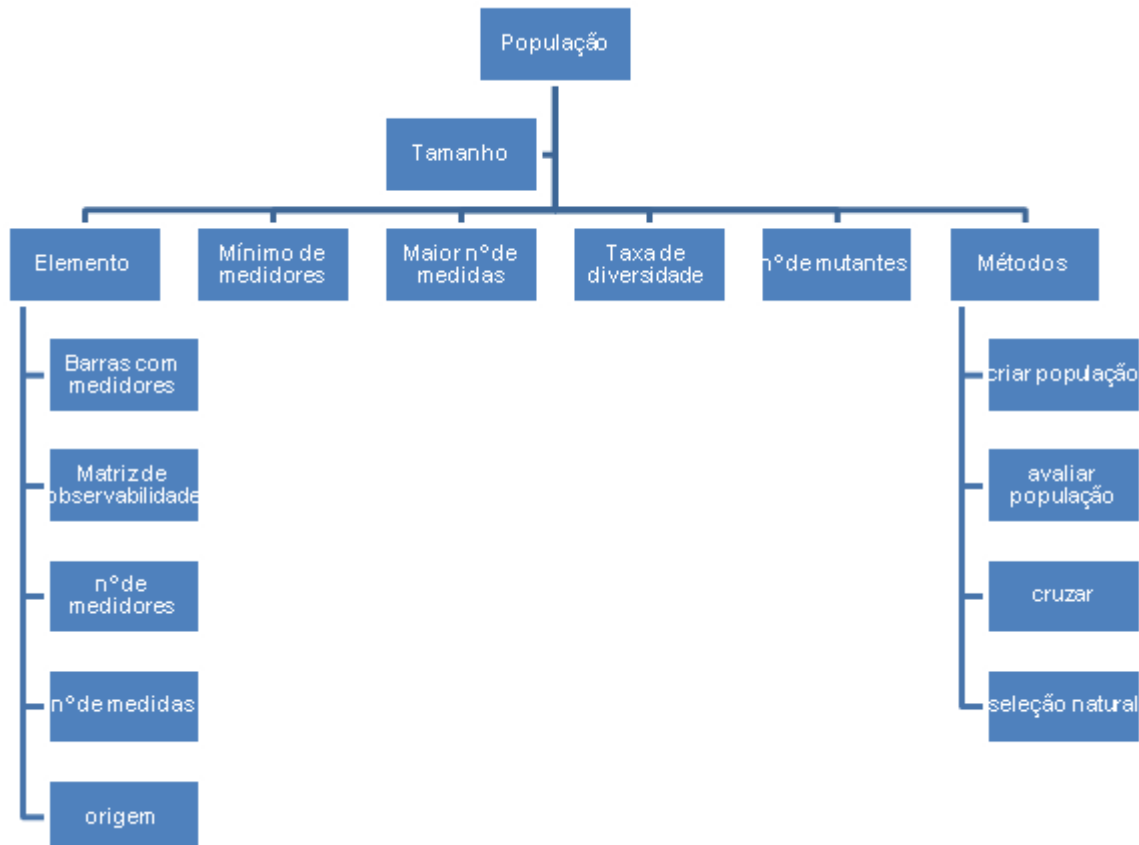
- **Versão Orientada Objeto**

Enquanto os ensaios da versão anterior do programa eram realizados, foi criada uma versão do algoritmo orientada a objeto. A finalidade desta formulação era melhorar a organização do problema.

A ideia era representar a população como uma classe, que é composta pelos parâmetros que representariam: tamanho da população, elementos que a compunham, taxa de diversidade, número de mutantes presentes, menor número de medidores e maior número de medidas. Todos os parâmetros, com a exceção de elemento, eram variáveis inteiras ou reais (as taxas). Os Elementos pertencentes a população seriam representados por um vetor de objetos também de uma classe de elementos que definiam cada solução: as barras com medidores, o número de medidores, a matriz de observabilidade, o número de medidas e a origem da solução no código.

A classe população também possuía todos os métodos necessários para realizar os operadores genéticos: criar população, avaliar população, cruzamento, mutação, seleção natural e cálculo de taxas. A Figura 3.10 apresenta a hierarquia desta classe.

Figura 3.10-Hierarquia orientada objeto



Fonte: autoria própria

Esta implementação levou a um código mais limpo, permitindo que fossem criados objetos da classe para a população inicial e população de filhos e evitando repetições de funções. Contudo, *Matlab* não realiza operações com orientação objeto de forma eficiente, demorando cerca de sete vezes mais que as implementações anteriores.

- **Versão por *Structs***

Simplificando a versão orientada objeto, esta versão do programa tentou substituir o uso da matriz de solução *x1* e dos vetores auxiliares *som* e *somb*, por um vetor de *Structs*. *Structs* são estruturas de dados heterogêneas, disponíveis para a utilização no Matlab, que podem armazenar diversos dados de tipos diferentes (de forma similar a um objeto). Nesse vetor, cada elemento seria uma possível solução contendo: as barras com medidores, o número de medidores, a matriz de observabilidade, o número de medidas e a origem no código.

Contudo, acessar este tipo de estrutura para realizar operações com o vetor das barras se mostrou mais custoso, tanto para calcular a matriz de observabilidade, quanto para realizar o cruzamento.

3.1.7 VERSÃO 5

A quinta versão do programa teve como objetivo aprimorar a forma como a seleção dos elementos da matriz de população era realizada nas versões anteriores, assim como implementar uma forma de monitorar a evolução da melhor solução com o avanço das gerações.

- **Melhor Seleção**

A forma de seleção implementada anteriormente se mostrou muito custosa e confusa. Para aprimorar esta etapa, a seguinte reforma foi realizada: primeiramente, as soluções que possuírem o menor número de unidades de medidas devem ser alocadas no início, sendo assim, todo o vetor *somb* é percorrido e todos os elementos que possuírem o menor valor são realocados entre os primeiros do vetor e à medida que isso ocorre, os valores equivalentes de *x1* e *somb* também são realocados. Em seguida, é preciso ordenar os valores com menor número de medidores de acordo com o número de medidas que são possibilitadas; para isso, a matriz *x1* e o vetor *som* são ordenados simultaneamente, de acordo com seu equivalente em *somb*. Para realizar esta ordenação, uma função auxiliar foi criada de forma a ordenar as três estruturas simultaneamente, de forma mais rápida possível por um *quicksort* (técnica de ordenação recursiva de complexidade $\log n$).

Também na seleção foi implementada uma técnica para auxiliar na convergência para a melhor solução. São preestabelecidas no início do programa o número de reprodutores (Rep). As posições entre o quinto elemento e o elemento “Rep” da população agora passam a ser ocupadas por soluções que contiverem a barra com maior número de ligações, permitindo que soluções que possuam a barra mais importante sejam selecionadas para o cruzamento, no decorrer das gerações.

Esta implementação permitiu que não fosse mais necessário armazenar a elite em um vetor auxiliar para a reprodução n , já que as melhores soluções podem ser acessadas diretamente nos primeiros elementos da matriz da população.

- **Avanço por geração**

Para poder analisar o avanço do melhor resultado de cada uma das gerações, foi adicionada ao programa a possibilidade de registrar este valor em um arquivo de texto, todas as vezes que um melhor resultado era encontrado.

3.1.8 VERSÃO 6

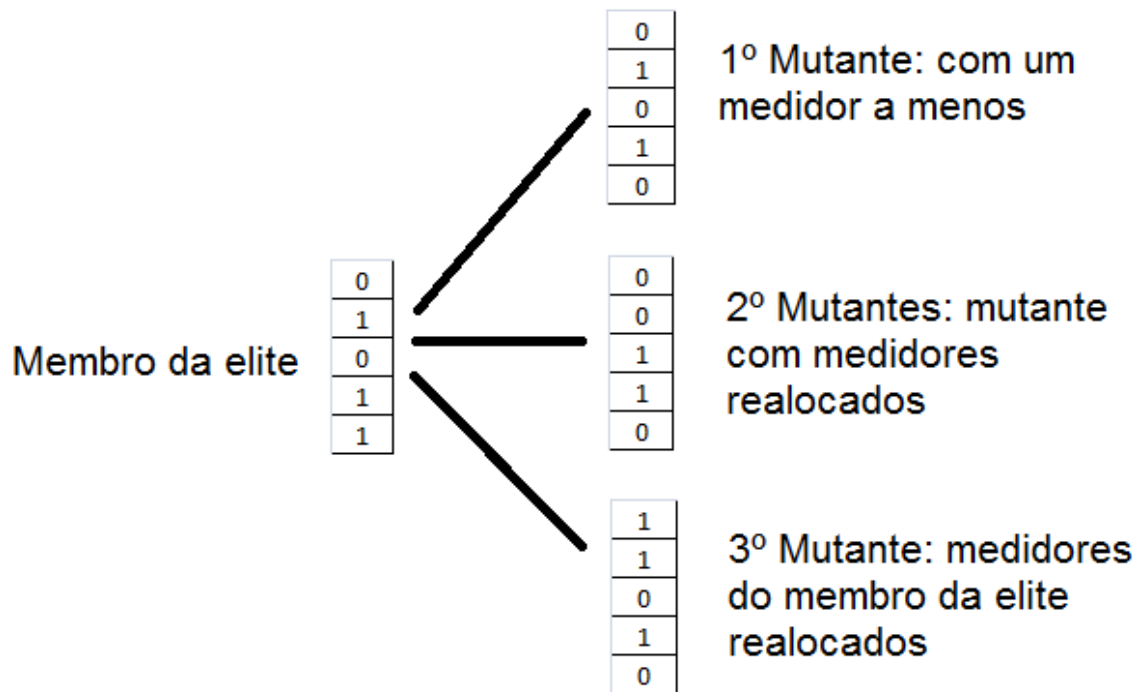
A sexta versão teve como foco aprimorar a mutação e a taxa de diversidade do algoritmo, como também algumas melhorias na apresentação do relatório do programa.

- **Aprimoramento da mutação**

A mutação aplicada nas versões anteriores, apesar de ajudar na convergência das soluções ao sempre remover um dos medidores, não estava cumprindo sua função original de aumentar a diversidade, de acordo com a teoria dos algoritmos genéticos. Para aprimorar esta etapa, mais dois grupos de mutantes são adicionados à matriz de filhos.

O primeiro grupo é formado por novas soluções geradas pela realocação dos medidores dos mutantes gerados anteriormente. Já o segundo grupo é formado da mesma forma, só que por uma realocação dos membros da elite da população. Na Figura 3.11, os mutantes são representados.

Figura 3.11-Mutantes gerados



Fonte: autoria própria

Em gerações mais avançadas, quando há poucos medidores presentes nas soluções e a redundância de medidas é menor, simplesmente remover medidores quase sempre irá gerar mutantes inválidos que não serão selecionados para gerações futuras. Estes aprimoramentos do código permitiram assim, que mais soluções válidas fossem vasculhadas.

- **Melhorias na Taxa de diversidade**

O método utilizado nas versões anteriores para representar cada solução presente na população não conseguiu cumprir sua função de forma eficiente; por esse motivo, a taxa de diversidade não variou muito entre as gerações, causando um número muito grande de migrações, não permitindo que os operadores genéticos atuassem devidamente.

Outras formas de representar cada solução foram criadas, uma delas foi a de identificar a solução pelo número de medidores presentes na mesma, como representada na Figura 3.12. Esta forma não apresentou avanço e o mesmo problema continuou a acontecer.

Uma terceira representação foi experimentada, também representada na Figura 3.12. Neste caso a identificação de cada solução é composta da seguinte forma: os dígitos menos significativos informam a quantidade de UTR's instaladas da solução e os mais significativos, a quantidades de estados observados a partir desta alocação das unidades. Esta representação

diminuiu o número de migrações que ocorriam no decorrer do programa e permitiu a atuação dos operadores genéticos.

Figura 3.12-Representações das soluções

Barra	Medidor	nº de medidas
1	0	19
2	1	
3	0	
4	0	
5	0	
6	1	
7	1	
8	0	
9	1	
10	0	
11	0	
12	0	
13	0	
14	0	

1ª Representação
id=1x2+1x6+1x7+1x9=24

2ª Representação
id=1+1+1+1=4

3ª Representação:
id=4+100x19=1904

Fonte: autoria própria

3.1.9 VERSÃO 7

A Versão final sete teve como objetivo implementar uma busca local ao final da era genética do código. Uma vez que, mesmo utilizando um grande número de gerações, o resultado sempre chegava a um número próximo de medidores, o objetivo da busca local final era de pesquisar por uma solução melhor e próxima, no espaço de soluções, do valor alcançado pela parte genética do código.

- **Busca Local**

O algoritmo da busca teve como base um algoritmo já criado anteriormente pelo professor Márcio, que lista todas as combinações binárias possíveis com um dado número de dígitos. Este código lista os valores deslocando os dígitos unitários mais elevados.

Este código foi adaptado, como representa a Figura 3.13, de forma a listar as respostas a partir do último resultado obtido pela era genética (com o último medidor removido para procurar um resultado melhor com menos medidores) e fixando o número de dígitos unitários, exemplo: todas as combinações possíveis de 30 dígitos, sendo com 4 casas unitárias. A restrição (3.4) é testada para cada um dos resultados listados. E assim, a busca

final é realizada até que o tempo predeterminado se conclua ou até um resultado válido ser encontrado.

Figura 3.13-Busca local

30 dígitos sendo quatro unitários
--

```

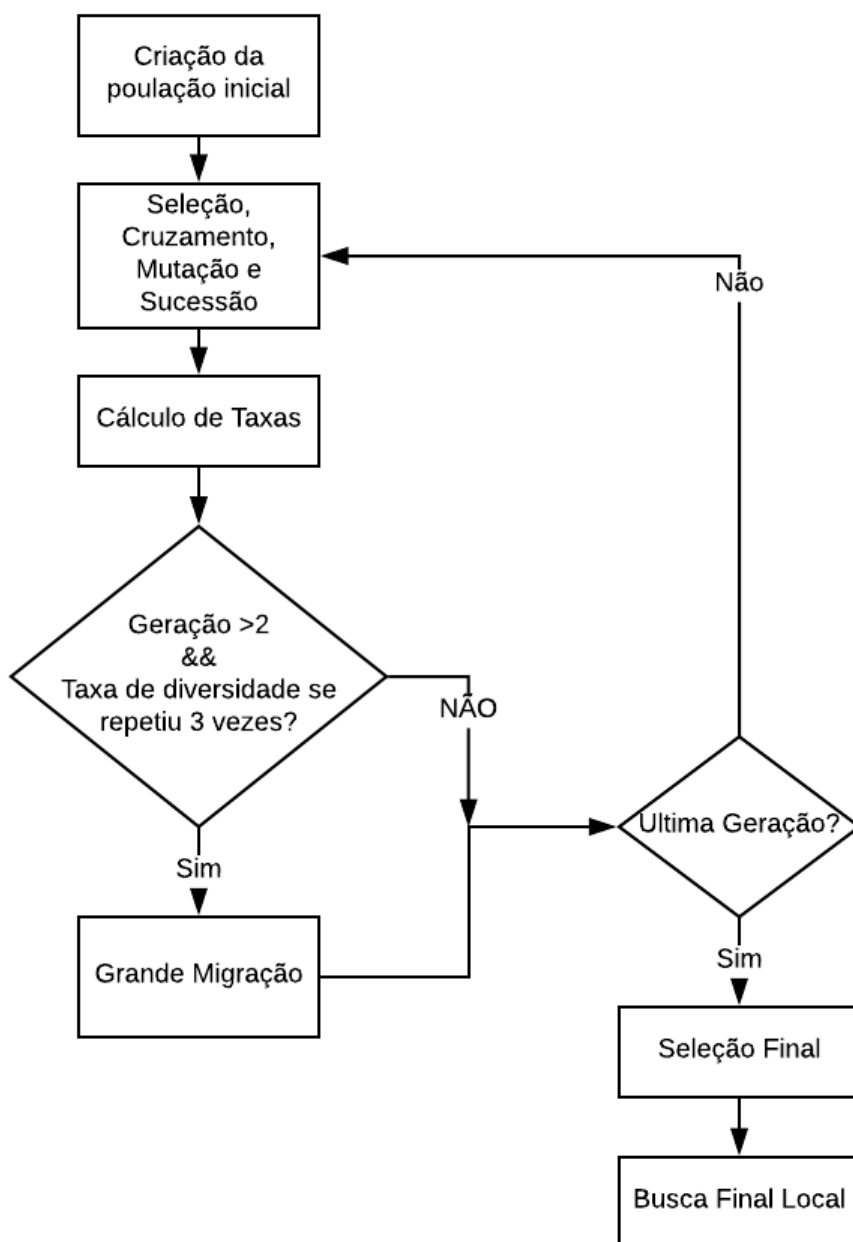
000100010000001100000000000000
000100010000001010000000000000
000100010000001001000000000000
000100010000001000100000000000
000100010000001000010000000000
000100010000001000001000000000
000100010000001000000100000000
000100010000001000000010000000
000100010000001000000001000000
000100010000001000000000100000
000100010000001000000000010000
000100010000001000000000001000
000100010000001000000000000100
000100010000001000000000000010
000100010000001000000000000001
000100010000001100000000000000
000100010000001010000000000000
000100010000001001000000000000
000100010000001000100000000000
000100010000001000010000000000
000100010000001000001000000000
000100010000001000000100000000
000100010000001000000010000000
000100010000001000000001000000
000100010000001000000000100000
000100010000001000000000010000
000100010000001000000000001000

```

Fonte: autoria própria

A busca local apesar de ajudar a avançar o número mínimo de medidores, não acrescenta em nada em relação à busca do máximo de medições. A **Erro! Fonte de referência não encontrada.** apresenta a última atualização do fluxograma.

Figura 3.14 -Fluxograma da última versão



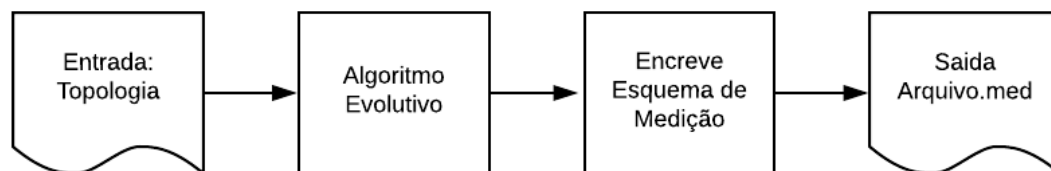
Fonte: autoria própria

3.2 REGISTRAR ESQUEMA DE MEDIÇÃO

Com a última versão finalizada do programa, o próximo passo foi transformar o código em uma função adaptada para o SimSEP. A partir de uma topologia de entrada, a função deve retornar um arquivo de formato nomeado PLANMED, que contém um esquema

de medição para a solução gerada pelo algoritmo evolutivo. A função recebeu o nome de *Spartan* e o fluxo dos processos envolvidos apresentados na Figura 3.15.

Figura 3.15-Fluxo de processos da Função *Spartan*



Fonte: autoria própria

O esquema de medição é registrado em um arquivo do formato PLANMED com a seguinte configuração de colunas:

1. nº do medidor
2. Nº da barra de origem
3. Nº da barra PARA
4. Nº do circuito
5. Tipo de medidor
6. Medida associada ao medidor
7. Utiliza ou não o medidor
8. Exatidão do medidor
9. Fundo de escala do medidor
10. Desvio-padrão
11. Valor de referencia
12. Valor medido

Após o algoritmo evolutivo a função de escrita registra os valores das barras de origem e de destino no caso de medida de fluxo, no caso de medida escalar como tensão a barra é registrada apenas como destino. O tipo de medidor deverá é registrado de acordo com a tabela :

Tabela 3.1- Tabela de convenção de medidores.

MEDIDORES	TIPO
Fluxo de Potência Ativa	1
Injeção de Potência Ativa	2
Medidas de Ângulo	3
Fluxo de Potência Reativa	4
Injeção de Potência Reativa	5
Tensão	6
Corrente de Ramo Real	7
Corrente de Ramo Imaginário	8
Injeção de Corrente Real	9
Injeção de Corrente Imaginária	10

Fonte: (Baptista, 2017)

No caso de UTR's são utilizados fluxo e injeção de potencia ativa (1 e 4), fluxo e injeção de potência reativa (2 e 5) e tensão (6). A exatidão dos medidores e fundo de escala relacionados a cada medidor também são adicionadas, $acc=0,025$ e $fs=1,1$ para medidas de tensão e $acc=0,020$ e $fs=1,0$ para os outros tipos de medida.

Também foi cogitada a inclusão do calculo e registro do desvio padrão dos medidores, mas de acordo com (Moraes, 2009), é preciso do valor de medição para realizar este cálculo. Sendo assim esta função só poderá ser implementada por outra função do SimSEP.

3.3 MEDIDORES DE RETAGUARDA

Também foi adicionada ao programa a funcionalidade de criar um esquema de medição que considere a possibilidade de perdas de uma RTU (Esquema de Contingencia n-1). Caso ocorra algum tipo de defeito e um medidor precise ser removido do esquema, deve sempre haver um medidor de retaguarda que deve manter o sistema observável mesmo com que ocorra a perda.

O programa pode ser facilmente adaptado para considerar perdas. Alterando a restrição (3.4) para $\bar{b} \geq 2$, podemos garantir que todas as barras do sistema em questão estão sendo observadas por pelo menos duas barras. Sendo assim com a adesão dessa funcionalidade, o programa passa a ter como entrada também um valor binário que indica se deve ser considerada a perda de medidores.

4 ANÁLISE DE RESULTADOS

Nesta seção serão analisados os resultados obtidos para os sistemas de barras (14,30, 57 e 118) em todas as versões obtidas no decorrer do desenvolvimento do programa. Foi considerado que o melhor resultado possível foi obtido por Xur e Abur (2005) e para comprovar a funcionalidade do programa, estes são comparados todos os resultados obtidos pelo programa.

Para analisar os resultados de forma justa, os mesmos parâmetros de entrada do algoritmo evolutivo foram pré-estabelecidos de forma igual para todas as versões, como mostra a Tabela 4.1.

Tabela 4.1 Parâmetros de entrada do algoritmo evolutivo

Nº de iterações	100
Tamanho da população	300
Nº de gerações	4000
Nº de reprodutores	10

O número de gerações foi estabelecido como um número bastante elevado, pois a convergência no caso de 118 barras se mostrou muito espaçada entre as gerações, por possuir um conjunto muito extenso de (2^{118}) possíveis soluções.

4.1 VERSÃO 0

A versão inicial do programa, apesar de não assegurar o melhor resultado global com o maior número de medidas, alcançou o mínimo de medidores dos sistemas de 14 e de 30 barras. Mesmo que o melhor resultado para o sistema de 17 tenha sido obtido, este só foi alcançado 1% das vezes, sendo assim esta versão não nos garante bons resultados para este e casos mais complexos. Os resultados dessa versão podem ser visualizados na Tabela 4.2.

Tabela 4.2-Resultados Versão 0

Versão 0					
Número de barras	Mínimo de UTR's encontrado:	Taxa de acerto de mínimo de UTR's	Maior nº de estados observados com mínimo de UTR's	Taxa de acerto de mínimo de UTR's e máximo de estados observados	Tempo - 100 iterações (segundos)
14	4	93%	19	31%	6333
30	10	50%	51	2%	5928
57	17	1%	70	1%	10715
118	35	1%	162	1%	13874

4.2 VERSÃO 1

Como visto na Tabela 4.3, a primeira versão do programa, apesar de ter praticamente assegurado o menor número de medidores no caso de 14 barras, não apresentou um avanço considerável na solução do problema. Os resultados desta versão podem ser melhorados ao reduzir as margens delimitadas nos parâmetros iniciais (p_1 e p_2), porém esta redução pode direcionar a convergência do algoritmo para uma solução não desejada, além de aumentar do tempo de geração da população inicial.

Alguns testes com as margens reduzidas ($p_1=0,25$ e $p_2=0,65$) foram iniciados, porém não finalizados, pois o tempo total de conclusão do programa era estimado para dez dias. Como o tempo de geração da população inicial representa, nesta versão, cerca de 90% do tempo total do programa, foi optado pelo aumento desta margem.

Tabela 4.3-Resultados Versão 1

Versão 1					
Número de barras	Mínimo de UTR's encontrado:	Taxa de acerto de mínimo de UTR's	Maior nº de estados observados com mínimo de UTR's	Taxa de acerto de mínimo de UTR's e máximo de estados observados	Tempo - 100 iterações (segundos)
14	4	99%	19	0,34	5664,292
30	10	65%	52	0,01	6683,304
57	18	11%	74	0,01	10134,552
118	35	2%	169	0,01	13873,038

4.3 VERSÃO 2

Como pode ser visto na Tabela 4.4, a segunda versão que aumentou as chances do sorteio de barras com maior número de ligações na população inicial, conseguiu praticamente resolver os casos de 14 e 30 barras, sendo que para o primeiro foi assegurado o melhor resultado possível em 99% das vezes.

Tabela 4.4-Resultados Versão 2

Versão 2					
Número de barras	Mínimo de UTR's encontrado:	Taxa de acerto de mínimo de UTR's	Maior nº de estados observados com mínimo de UTR's	Taxa de acerto de mínimo de UTR's e máximo de estados observados	Tempo - 100 iterações (segundos)
14	4	99%	19	0,95	6702,089
30	10	99%	52	0,23	7751,794
57	17	6%	70	0,01	11312,642
118	34	9%	179	0,01	13873,038

4.4 VERSÃO 4

A adesão da migração ao algoritmo na quarta versão do programa, apesar de ter aumentado as chances de alguns em alguns dos casos, não apresentou melhorias muito significativas, como pode ser visto na Tabela 4.5. O valor mínimo de 118 foi alcançado nesta versão, porém como só foi atingido uma vez das cem interações, esse resultado não pode ser garantido. Houver um aumento considerável no tempo total do programa, isso é causado pelo fato de ocorrerem muitas grandes esticções no decorrer das gerações.

Tabela 4.5-Resultados Versão 4

Versão 4					
Número de barras	Mímimo de UTR's encontrado:	Taxa de acerto de mínimo de UTR's	Maior nº de estados observádos com mínimo de UTR's	Taxa de acerto de mínimo de UTR's e máximo de estados observados	Tempo - 100 iterações (segundos)
14	4	100%	19	1	12724,383
30	10	100%	52	0,17	12733,086
57	17	6%	71	0,02	20373,113
118	32	1%	159	0,01	108638,91

4.5 VERSÃO 5

A versão 5 também, em que foi alterada a seleção para não apresentou muitos avanços em relação às versões anteriores.

Tabela 4.6-Resultados Versão 5

Versão 5					
Número de barras	Mímimo de UTR's encontrado:	Taxa de acerto de mínimo de UTR's	Maior nº de estados observádos com mínimo de UTR's	Taxa de acerto de mínimo de UTR's e máximo de estados observados	Tempo - 100 iterações (segundos)
14	4	100%	19	100%	13914
30	10	100%	52	14%	10965
57	17	7%	71	2%	19429
118	33	1%	166	1%	102211

4.6 VERSÃO 6

Como mostra a Tabela 4.7, a sexta versão do programa apresentou um grande avanço na solução do problema. O as resoluções dos casos de 14 e 30 com os melhores resultados possíveis são praticamente garantidas. O programa também passou a ser mais bem sucedido no caso de 57 barras.

Com a alteração do método utilizado para o cálculo das taxas de diversidade, o número de migrações foi reduzido drasticamente para os casos de 57 e 118, em que a melhor solução é alcançada em gerações mais elevadas. Com essa redução é possível concluir que o uso muito frequente de migrações pode impedir que os operadores genéticos contribuam com o avanço das soluções no decorrer das gerações.

Tabela 4.7-Resultados Versão 6

Versão 6					
Número de barras	Mínimo de UTR's encontrado:	Taxa de acerto de mínimo de UTR's	Maior nº de estados observados com mínimo de UTR's	Taxa de acerto de mínimo de UTR's e máximo de estados observados	Tempo - 100 iterações (segundos)
14	4	100%	19	1	15251,834
30	10	100%	52	0,97	15680,953
57	17	72%	72	0,3	23490,81
118	33	11%	176	0,01	88820,926

4.7 VERSÃO 7

Como na versão anterior a menor solução alcançada foi de 33 medidores, surgiu a ideia de da implementação de uma busca local ao fim do algoritmo. A Tabela 4.8 mostra a melhoria do código ao alcançar a solução mínima do caso de 57 e 118 barras. A busca não foi aplicada nos casos de 14 e 30, pois estes já são resolvidos na era genética.

A adesão da busca local gerou um aumento no grande no tempo total do programa. Este aumento ocorreu principalmente nos casos em que a solução mínima era alcançada sem a necessidade da busca. Exemplo: se uma solução é encontrada durante a era genética com 17 medidores para o caso de 57 barras, a busca local tenta encontrar uma solução possível com 16 medidores, sendo assim, ela irá gastar no mínimo 1200 segundos nesta etapa para chegar à conclusão que a solução, com este número de medidores, não existe.

Tabela 4.8-Resultados Versão 7

Versão 7					
Número de barras	Mínimo de UTR's encontrado:	Taxa de acerto de mínimo de UTR's	Maior nº de estados observados com mínimo de UTR's	Taxa de acerto de mínimo de UTR's e máximo de estados observados	Tempo - 100 iterações (segundos)
14	4	100%	19	1	16065,993
30	10	100%	52	0,97	26356,246
57	17	94%	72	0,38	198608,63
118	32	9%	161	0,01	113867,74

Os melhores esquemas registrados em todas as iterações do algoritmo são apresentados na Tabela 4.9. As barras que devem ser instaladas medidores coincidiram para os sistemas mais simples, isto é esperado por estes possuírem um espaço menor de soluções possíveis.

Tabela 4.9-Melhores Arranjos Encontrados

Nº de barras	Barras com medidores	Total de medidas
14	2 6 7 9	19
30	2 4 6 9 10 12 15 20 25 27	52
57	1 4 6 9 15 20 24 25 28 32 36 38 39 41 47 50 53	72
118	3 5 9 11 12 17 21 23 29 30 34 37 42 45 49 53 56 62 63 68 71 75 77 80 85 86 90 94 101 105 110	161

Como os casos de 57 e o de 118 barras apresentaram diferentes posicionamentos de medidores, em relação aos obtidos por Xur e Abur (2005), foram comparados então, todas as medidas de todos os casos obtidos pelo programa com o obtido pela literatura, como mostra a Tabela 4.11 para o caso de 57 barras e a Tabela 4.11 para o caso de 118 barras. Pode-se concluir então que todos os resultados obtidos pelo programa possibilitaram um número maior ou igual ao número de medidas fornecido anteriormente pelo relatório.

Tabela 4.10-Comparação de nº de medidas para 57 barras

Programa	Nº de medidas	Barras com Medidores
ABUR	71	1 4 7 9 15 20 24 25 27 32 36 38 39 41 46 50 53
300 de Esparta	72	1 4 6 9 15 20 24 25 28 32 36 38 39 41 47 50 53

Tabela 4.11-Comparação de nº de medidas para 118 barras

Programa	Nº de medidas	Barras com Medidores
ABUR	157	3 5 9 11 12 17 21 24 25 28 34 37 40 45 49 52 56 62 63 68 73 75 77 80 85 86 90 94 101 105 110 114
300 de Esparta	157	3 5 10 11 12 17 21 25 29 34 37 41 45 49 52 56 62 64 68 71 72 75 77 80 85 86 90 94 101 105 110 114
	159	1 7 9 11 12 17 20 23 28 30 34 37 40 45 49 52 56 62 64 68 71 75 77 80 85 86 90 94 101 105 110 115
	160	2 5 9 12 15 17 21 25 28 34 37 40 45 49 52 56 62 64 70 71 75 77 80 85 86 90 94 101 105 110 114 116
	161	3 5 9 11 12 17 21 23 29 30 34 37 42 45 49 53 56 62 63 68 71 75 77 80 85 86 90 94 101 105 110 115
	151	1 7 9 11 12 17 21 24 25 28 34 37 42 45 49 52 56 62 64 73 75 77 80 85 86 91 94 101 105 110 114 116
	158	1 5 10 12 13 17 21 23 25 28 34 37 40 45 49 52 56 62 64 68 71 75 77 80 85 86 91 94 101 105 110 115
	158	2 5 10 12 15 17 20 23 25 29 34 37 42 45 49 53 56 62 63 68 71 75 77 80 85 87 90 94 101 105 110 115
	160	2 5 9 12 15 17 21 25 28 34 37 40 45 49 53 56 62 64 68 70 71 78 85 87 91 92 96 100 105 110 114 118
	160	2 5 9 11 12 17 20 23 28 30 34 37 42 45 49 53 56 62 63 68 71 75 77 80 85 86 90 94 101 105 110 115

4.8 CONSIDERANDO PERDA DE MEDIDORES

Também foram realizados testes da sétima versão do programa que consideravam perdas de unidades medidoras. Os resultados obtidos foram registrados na Tabela 4.12. É interessante observar que todos os resultados para os casos de 30, 57 e 118 barras apresentaram resultados melhores que os apresentados na Tabela 2.2.

Tabela 4.12-Resultados Obtidos Considerando perda de Medidores

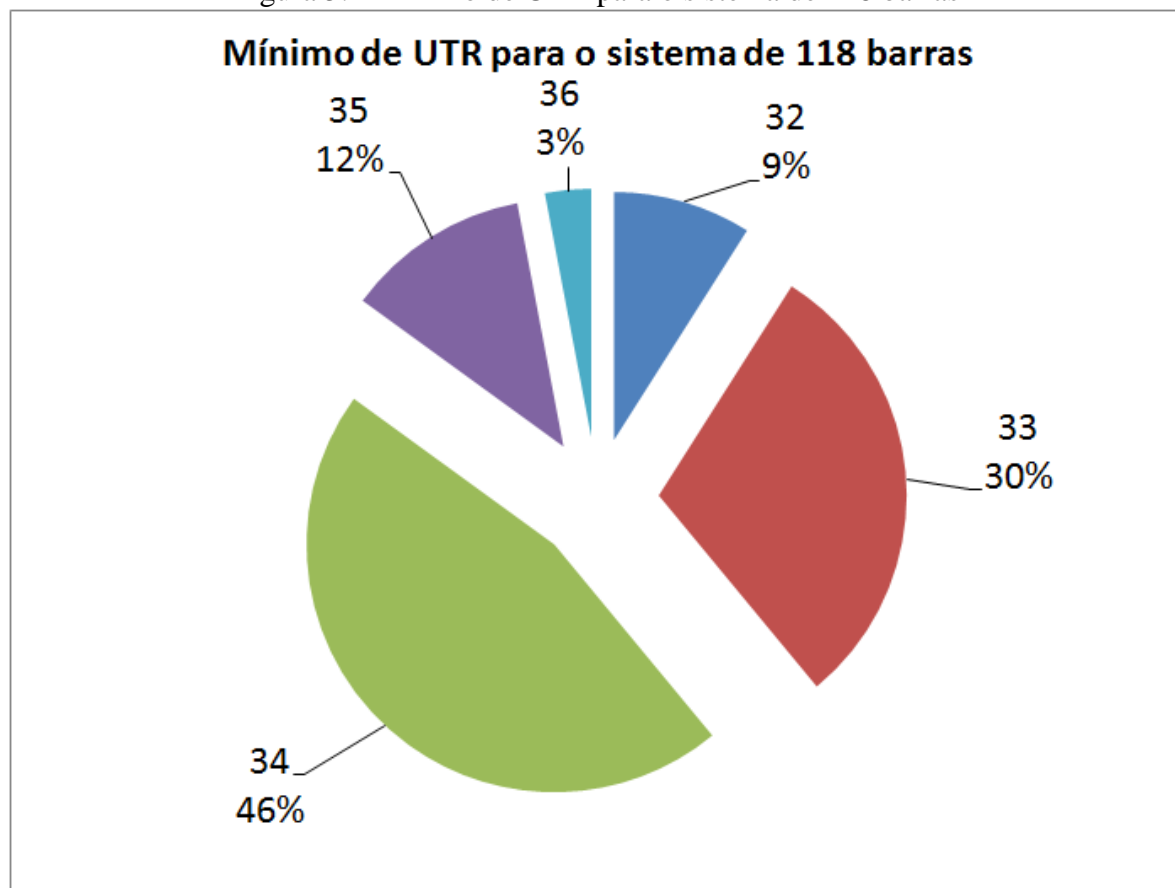
Número de barras	Mínimo de medidores encontrado:	Taxa de acerto de mínimo de medidores	Maior nº de medidas encontradas	Taxa de acerto de mínimo de	Barras
14	9	100%	39	100%	2 4 5 6 7 8 9 10 13
30	21	100%	81	80%	1 2 4 6 7 9 10 11 12 13 15 16 18 19 22 24 25 26 27 28 30
57	33	98%	130	98%	1 3 4 6 9 11 12 15 19 20 22 24 25 27 28 29 31 32 33 35 36 37 38 41 45 46 47 50 51 53 54 56 57
118	68	56%	309	3%	1 3 5 6 9 10 11 12 15 17 19 21 22 24 25 27 29 30 31 32 34 36 37 40 42 43 45 46 49 50 51 52 54 56 59 62 64 65 66 68 70 71 73 75 77 79 80 83 85 86 87 89 91 92 94 96 100 101 105 106 109 110 111 112 114 116 117 118

5 CONCLUSÕES

É possível afirmar que programa 300 de Esparta consegue resolver os casos de 14, 30 57 e 118 barras do IEEE, pois foi possível alcançar todos os valores obtidos anteriormente pela literatura, quatro medidores para o sistema de 14 barras, dez para o 30 barras, dezessete para o 57 barras e trinta e dois para o 118 barras. Também é possível concluir que os arranjos retornados pelo programa são melhores que os obtidos anteriormente pela literatura, pois estes mostraram possibilitar, para os casos mais complexos, um maior número de medidas para o sistema: 72 medidas para 57 barras (anteriormente 71) e 161 para o 118 barras (anteriormente 158).

O caso de 118 barras possui um número extremamente elevado de possíveis soluções (na escala de 2^{118}). Apesar dessa grande dificuldade, o programa se mostrou apto a encontrar a solução mínima. Uma precisão maior do programa é o principal objetivo de melhoria, pois o mínimo foi alcançado 9% das vezes como mostra a Figura 5.1-Mínimo de UTR para o sistema de 118 barras.

Figura 5.1-Mínimo de UTR para o sistema de 118 barras



Fonte: autoria própria

O programa, em sua última versão, para o caso de 118 barras, precisou de 31 horas para completar 100 iterações com 4000 gerações cada uma. Cada uma das iterações demorou cerca de 1138 segundos, sendo que 50% desse tempo foi gasto para gerar a população inicial, 34% foi gasto nos operadores genéticos (seleção, reprodução, mutação e migração) e 16% do tempo foi gasto durante a busca final. Um possível aprimoramento do programa seria o equilíbrio nos tempos de cada uma das etapas, este pode ser alcançado a partir de uma reformulação nos parâmetros de entrada (geração, número de gerações, número de reprodutores etc.), ou até mesmo por uma otimização do algoritmo na etapa de geração da população inicial.

É visada também a aplicação do programa no caso de 300 barras do IEEE. A simulação deste caso não foi possível por dificuldades de obtenção da matriz de topologia (A) a partir do CDF referente à mesma.

Um aprimoramento futuro para o programa é a adesão de métodos que permitam a sua aplicação em casos que já existam medidas de fluxo e de injeção no sistema a ser analisado. Estes métodos já foram implementados e são baseados nos casos apresentados por Xur e Abur (2005), porém, os resultados obtidos não puderam ser averiguados por falta de tempo. Estes métodos são promissores e possivelmente publicáveis após esta avaliação.

A principal conclusão que se pode obter deste trabalho é: até o ponto atual a aplicação de técnicas para antecipar a convergência de um algoritmo genético pode ser bem sucedida para casos em que o espaço de solução seja muito extenso. Uma comparação com algum método puramente genético ainda é visada para a comprovação deste fato.

6 Bibliografia

- Arcos, R. B. (2017). *ELABORAÇÃO DE HEURÍSTICA EVOLUTIVA PARA DESPACHO NA GERAÇÃO*. Trabalho de conclusão de curso-Universidade Federal Fluminense, Niterói.
- Baptista, J. (2017). *SIMULADOR DE SISTEMAS ELÉTRICOS DE POTÊNCIA - SimSEP*. Trabalho de conclusão de curso-Universidade Federal Fluminense, Niterói.
- Gamm, A., Kolosok, I. N., Glazunova, A. M., & Korkina E, S. (2008). PMU placement criteria for EPS state estimation., (p. 5). Nanjing.
- Gendreau, M., & Potvin, J.-Y. (2010). *Handbook of Metaheuristics 2nd edition* (Vol. 1446). New York: Series in Oper. Res. & Manag. Science.
- Kerdchuen, T., & Ongsakul, W. (2006). Optimal Measurement Placement for Power. *International Conference on Power System Technology*, (p. 5).
- Meza, E. B. (2006). *Depuração de parâmetros de redes elétricas via estimação de estado e algoritmos genéticos*. Tese (Doutorado em Computação) - Universidade Federal Fluminense, Niterói.
- Motiyani, R. J., & Chudasama, A. R. (2011). *A Non Iterative Method Based on Graph Theoretic*. Conference on Industrial Electronics and Applications, Gujarat.
- Ramesh, L., Chowdhury, S. P., Chowdhury, S., & Natarajan, A. A. (2007). *POWER SYSTEM OPTIMAL METER PLACEMENT A COMPARITIVE NUMERIC AND GENETIC APPROACH*. International Conference on Information and Communication Technology in Electrical Sciences, Tamil Nadu.
- (2009). *Sicrofasores em sistema de potência: aplicações na estimação de estados*. Tese(Doutorado em Ciencias)-Universidade Federal Fluminense, Niterói.
- Xur, B., & Abur, A. (2005). *Optimal Placement of Phasor Measurement Units*. Texas A&M University, Texas.
- Zanghi, R. (2016). *Meta-heurísticas aplicadas ao agendamento de intervenções em redes elétricas*. Tese (Doutorado em Computação) - Universidade Federal Fluminense, Niterói.