INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA
Área Departamental de Engenharia Eletrotécnica Energia e Automação

Reconfiguração topológica de redes para redução de perdas

EVENI PEREIRA COSME

(Licenciada em Engenharia Eletrotécnica)

Dissertação de natureza científica para obtenção do grau de Mestre em Engenharia Eletrotécnica

Orientador:
Doutor Francisco Alexandre Ganho da Silva Reis

Júri:
Presidente: Doutor Luís Manuel dos Santos Redondo

Vogais:
Doutor João Hermínio Ninitas Lagarto

Maio de 2023

Página deixada propositadamente em branco.

Dissertação realizada sob orientação de

Doutor Francisco Alexandre Ganho da Silva Reis

Professor Adjunto do Departamento de Engenharia Eletrotécnica Energia e Automação do

Instituto Superior de Engenharia de Lisboa

Página deixada propositadamente em branco.

O que adquire conhecimento ama a sua alma;

o que cultiva a inteligência achará o bem.

Provérbios 19:8

Página deixada propositadamente em branco.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus porque sem Ele, nada disso seria possível.

Agradeço ao meu orientador de dissertação, o Professor Doutor Francisco Alexandre Ganho da Silva Reis, por me manter motivada durante o processo, por sempre estar disponível para me ajudar e por ter confiado em mim para a elaborar este trabalho.

Ao Instituto Superior de Engenharia de Lisboa, por disponibilizar vagas para alunos internacionais e aulas em regime pós-laboral e a todo o corpo docente do Departamento de Engenharia Eletrotécnica e Automação que esteve presente no meu percurso acadêmico do mestrado.

Ao meu amigo Pedro da Geometric Talks pela oportunidade de realização do estágio IEFP e aos meus novos colegas de profissão.

Ao meu marido, Breno, por me apoiar mesmo em decisões difíceis, por sempre estar ao meu lado e por sempre acreditar que sou capaz.

Por fim, agradeço aos meus pais, ao meu irmão, aos meus amigos e familiares, pois eles, direta e indiretamente, ajudaram na realização deste sonho.

Página deixada propositadamente em branco.

RESUMO

O objetivo da presente dissertação foi desenvolver algoritmos de reconfiguração topológica de redes de distribuição de energia elétrica para minimização de perdas. Para o efeito são apresentados dois algoritmos.

O primeiro algoritmo apresentado é baseado num modelo heurístico em que a procura da melhor topologia da rede é feita tendo em consideração a identificação de caminhos topológicos conexos que minimizam as perdas. Estes caminhos são realizados iterativamente entre os barramentos adjacentes de acordo com as regras estabelecidas no algoritmo.

Tendo em consideração a natureza combinatória do problema foi desenvolvido um segundo algoritmo que teve como base um modelo evolutivo, algoritmo genético. O objetivo é alargar o espaço de buscas de soluções. Posteriormente os AG's foram combinados com um modelo heurístico, *initial searching point*, para obter soluções ainda mais adaptadas ao problema e com bom desempenho.

Os dois algoritmos foram desenvolvidos em Python. A biblioteca Pandapower foi utilizada para modelar a rede em estudo e para os cálculos de trânsito de energia.

Estes algoritmos foram aplicados à rede IEEE 33 barramentos e ambos tiveram redução de perdas superiores a 30%. A redução de perdas resultante da aplicação do primeiro algoritmo foi de 31% e, do segundo algoritmo, a redução de perdas foi de 35%.

Palavras-chave: Reconfiguração de redes de distribuição, Perdas, Algoritmos Genéticos, Pandapower.

ABSTRACT

The objective of the present dissertation is to develop algorithms for the topological reconfiguration of electrical energy distribution networks to minimize losses. For this purpose, two algorithms are presented.

The first one is based on a heuristic model in which the search for the best network topology takes into account the identification of connected topological paths that minimize losses. These paths are obtained iteratively between adjacent buses according to the rules established in the algorithm.

Due to the combinatorial nature of the problem, a second algorithm was developed. This one is based on an evolutionary model, the genetic algorithm. Its objective is to expand the space for searching for solutions. Subsequently, the GA was combined with a heuristic model, the initial searching point. The solutions obtained were even more adapted to the problem and showed a better performance.

Both algorithms use Python as their coding language. And the library used to model the network under study and to calculate the energy transit was Pandapower.

They were applied to the IEEE 33 bus network, and both achieved a loss reduction of over 30%. The first one resulted in a reduction of 31% and the second one a reduction of 35%.

Keywords: Reconfiguration of electrical energy distribution networks, Energy loss, Genetic Algorithm, Pandapower.

LISTA DE ABREVIAÇÕES

ACO – *Ant Colony Optimization*

AG – Algoritmo Genético

ERSE – Entidade Reguladora dos Serviços Energéticos

IEEE – *Institute of Electrical and Electronics Engineers*

ISP – *Initial Searching Point*

MIT – *Massachussetts Institute of Technology*

PSF – *Python Software Foundation*

PSO – *Particle Swarm Optimization*

RARI – Regulamento de Acesso às Redes e às Interligações do Setor Elétrico

LISTA DE SÍMBOLOS

D: Dimensão do problema;

G_{max}: Quantidade máxima de gerações;

Fit_{max}: Maior valor de *fitness function*;

$Fit_{méd}$: Menor valor de *fitness function*;

Fit_{min}: Valor médio de *fitness function*;

I: Corrente injetada [kA];

$|\boldsymbol{I}_{i,j}|$: Módulo da corrente elétrica entre o barramento "i" e "j" [pu];

$|\boldsymbol{I}_{i,j\ max}|$: Valor máximo admissível da corrente elétrica entre o barramento "i" e o "j" [pu];

$|\boldsymbol{I}_{ramoj}|$: Módulo da corrente elétrica do ramo "j" [pu];

$|\boldsymbol{I}_{ramoj\ max}|$: Valor máximo admissível da corrente elétrica do ramo "j" [pu];

n: Número de ramos da rede;

N: Tamanho da população;

P_1: Potência ativa ligada ao barramento 1 [kW];

P_{C1}: Potência ativa da carga ligada ao barramento "1" [kW];

P_{G1}: Potência ativa de geração ligada ao barramento "1" [kW];

P_j: Potência ativa ligada ao barramento "j" [pu];

$P_{ramoj(z)}$: Potência ativa do ramo "j" associada à topologia da rede [pu];

$p_{total(z)}$: Potência ativa total de perdas associada à topologia da rede [pu];

Q_1: Potência reativa ligada ao barramento 1 [kVar];

Q_{C1}: Potência reativa da carga ligada ao barramento "1" [kVar];

Q_{G1}: Potência reativa de geração ligada ao barramento "1" [kVar];

Q_j: Potência reativa ligada ao barramento "j" [pu];

$Q_{ramoj(z)}$: Potência reativa do ramo "j" associada à topologia da rede [pu];

$r_{i,j}$: Resistência do condutor entre o barramento "i" e o "j" [pu];

$r_{ramoj(z)}$: Resistência do ramo "j" associada à topologia da rede [pu];

$\mathbf{S}_1$ – Potência injetada 1 [kVA];

S_{maxd}: Número aleatório entre 1 e 36;

$|\boldsymbol{V}_i|$: Módulo de tensão no barramento "i" [pu];

$\boldsymbol{V}_{min}$ e $\boldsymbol{V}_{max}$: Limites admissíveis de tensão no barramento "i" [pu];

$V_{ramoj(z)}$: Tensão do ramo "j" associada à topologia da rede [pu]:

X_{keep}: Taxa de seleção;

X_{mut}: Taxa de mutação;

$\mathbf{Y}_{i,i}$: Elementos da diagonal principal da matriz de admitância nodal;

$\mathbf{Y}_{i,j:}$ Elementos fora da diagonal principal da matriz de admitância nodal;

z: Topologia da rede.

ÍNDICE

Índice de Figuras

Índice de Tabelas

CAPÍTULO

1

1. INTRODUÇÃO

1.1 Enquadramento e motivação

A presente dissertação enquadra-se no tema de distribuição de energia elétrica e, referente a este tema, a problemática a ser estudada são as perdas na rede de distribuição. Este assunto afeta diretamente na eficiência e qualidade do serviço oferecido pelas concessionárias de energia.

Esta problemática tem motivado as empresas do setor a investirem em incentivos para o desenvolvimento de estratégias de atuação capazes de reduzir este parâmetro nas redes elétricas, já que a redução das perdas significa um aumento nos lucros.

1.2 Objetivo

O objetivo da presente dissertação é desenvolver algoritmos de reconfiguração topológica de redes de distribuição de energia elétrica para minimização de perdas.

1.3 Metodologia

A metodologia adotada consiste nas etapas a seguir:

- Levantamento bibliográfico de metodologias para redução de perdas;
- Desenvolvimento de algoritmos para redução de perdas em linguagem de programação Python associada à biblioteca Pandapower;
- Aplicação dos algoritmos elaborados à rede IEEE 33 barramentos;
- Análise dos resultados.

1.4 Estrutura

Este trabalho está estruturado de modo a apresentar sequencialmente as etapas realizadas para a obtenção dos objetivos previstos.

No capítulo 2 é realizado um estudo a respeito das perdas em redes de distribuição e de uma estratégia para deteção de ilhas em redes de energia elétrica.

No capítulo 3 é desenvolvido um algoritmo heurístico de redução de perdas com metodologia de reconfiguração iterativa da rede e é realizada sua aplicação à rede IEEE 33 barramentos e apresentação dos resultados.

Seguidamente no capítulo 4 foi desenvolvido um algoritmo evolutivo aplicado ao problema de minimização de perdas por meio da metodologia de reconfiguração iterativa. Assim como o algoritmo anterior, este é aplicado à rede IEEE 33 barramentos e é realizada uma análise dos resultados.

É realizada, no capítulo 5, uma abordagem sobre as ferramentas utilizadas para a elaboração dos dois algoritmos apresentados nos capítulos 3 e 4.

Por fim, no capítulo 6 são apresentadas as conclusões finais e propostas para trabalhos futuros.

CAPÍTULO 2

2. PERDAS EM REDES DE DISTRIBUIÇÃO

2.1 Introdução

De acordo com a Entidade Reguladora dos Serviços Energéticos – ERSE no Regulamento de Acesso às Redes e às Interligações do Setor Elétrico – RARI [1], as perdas são definidas como a diferença entre a energia que entra num sistema elétrico e a energia que sai desse sistema elétrico no mesmo intervalo de tempo.

Elas impactam diretamente na eficiência e na qualidade do serviço oferecido pelas concessionárias de energia, que, por sua vez, têm adotado estratégias para melhorar os índices. A E-Redes, em seu Relatório e Contas E-Redes 2021 [2] contabilizou em 2021, 9,47% de perdas totais. A meta definida na Estratégia de Sustentabilidade 2021 – 2025 é que as perdas totais sejam inferiores à 9,0% até 2025.

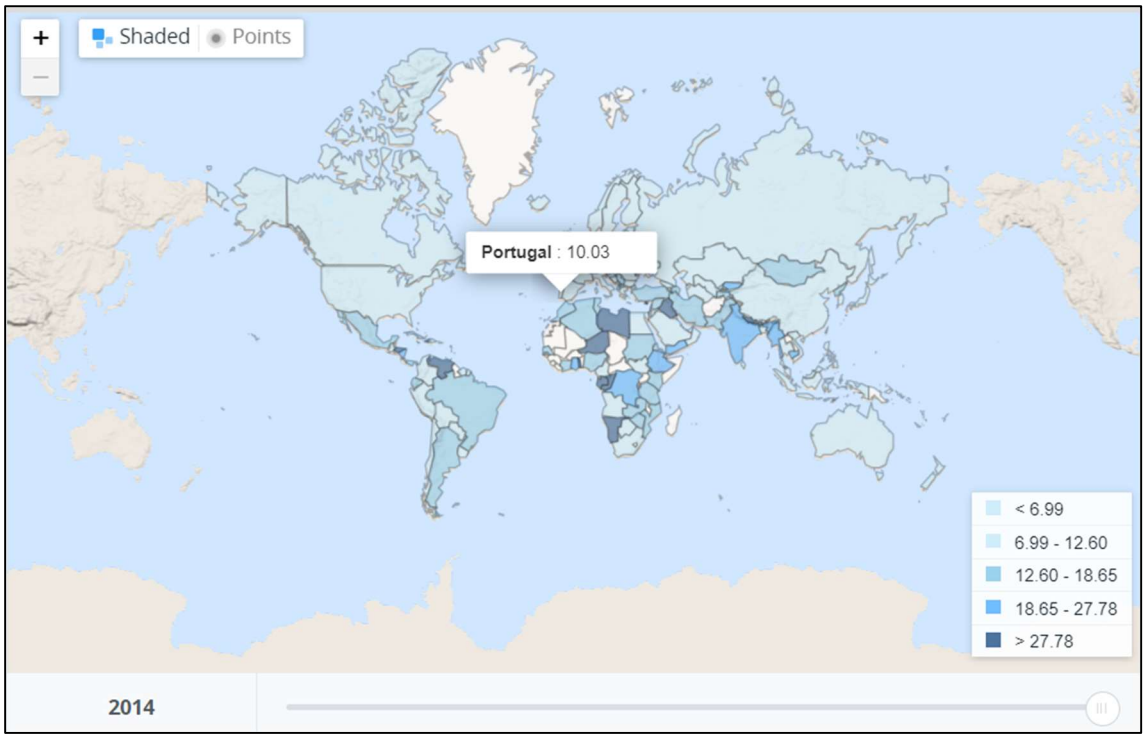


Figura 1: Perdas em redes de distribuição por países [27]

Tipicamente as perdas em redes de distribuição variam entre 8% a 12%. A Figura 1, contém um mapa com as perdas em redes de distribuição por países para o ano de 2014, em que aqueles que apresentam tonalidades mais claras são os que possuem menores perdas. No caso de Portugal, as perdas eram de 10,03% no ano de 2014.

As perdas totais representam a soma das perdas técnicas e não técnicas (ou comerciais). As perdas técnicas estão associadas à energia dissipada nos elementos da linha e estão relacionadas à topologia da rede. Já as perdas não técnicas são as provenientes de ligações clandestinas, contas não pagas pelos consumidores, adulteração dos equipamentos de medição de consumo.

2.2 Topologias de redes de distribuição

As redes de distribuição podem ser exploradas em anel fechado, anel aberto, radial e dupla derivação [3]. Destes, os esquemas mais utilizados são anel aberto e radial.

As redes em anel aberto, exemplo na Figura 2, estão caracterizadas por possuírem duas linhas de alimentação, o que assegura a possibilidade de dois caminhos distintos de alimentação do consumidor. Sendo que, em condições normais, esta é alimentada por apenas uma das linhas.

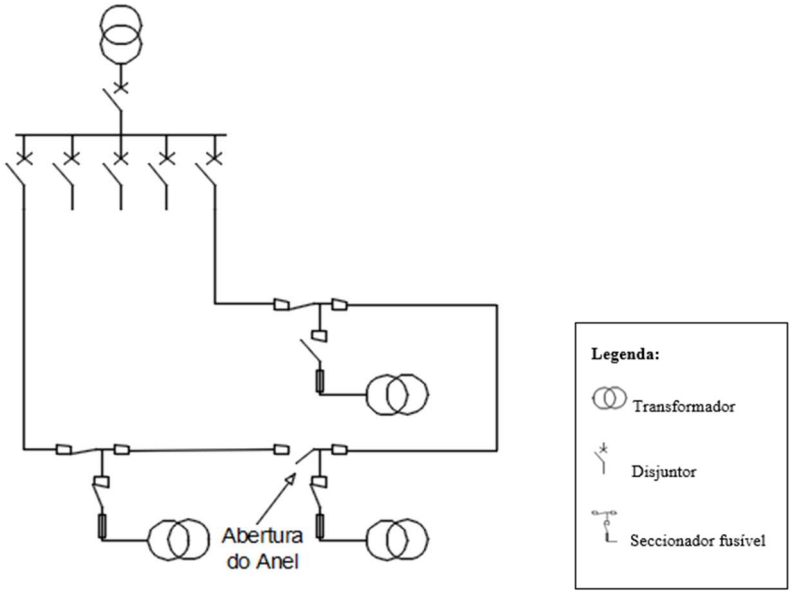


Figura 2: Exemplo rede anel aberto (adaptado de [3])

Já nas redes radiais, exemplo na Figura 3, existe somente um caminho para distribuição da energia ao consumidor. Em situações de defeito na linha, não há a possibilidade de caminho alternativo para a alimentação deste.

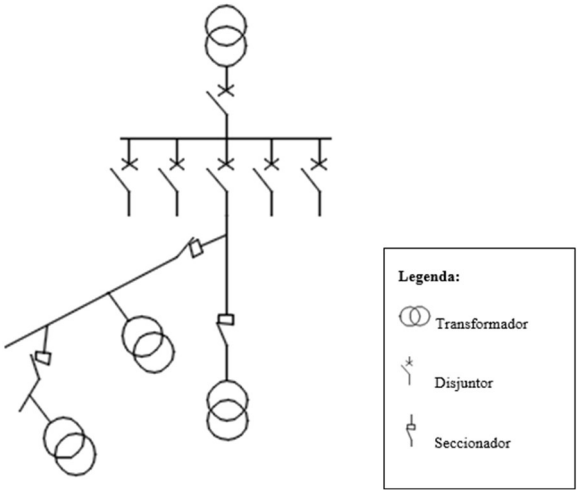


Figura 3: Exemplo rede radial (adaptado de [3])

2.2.1 Deteção de ilhas em redes de energia elétrica

Ilhas em redes de energia elétrica ocorrem quando parte dos barramentos da rede estão eletricamente desconectados do restante desta.

Uma metodologia que faz a deteção de ilhas e assegura que a rede é radial ou anel aberto foi proposta em [4]. O método consiste em calcular a matriz de admitâncias nodais da rede e depois a matriz de impedâncias.

Quando os elementos $\mathbf{Z}_{i,j}$ da matriz de impedâncias nodais possuem valor diferente de zero significa que existe ligação física entre os barramentos i e j. Portanto, se os barramentos i e j estão eletricamente desconectados, $\mathbf{Z}_{i,j} = 0$. Desta forma, se a matriz de impedâncias não possuir nenhum valor igual a zero, implica que todos os nós da rede estão conectados, o que anula a possibilidade de ilhas na rede.

A rede apresentada na Figura 4 por 5 barramentos e 6 linhas. No caso presente, a linha L4 e a linha L6 se encontram com os seccionadores abertos. Essa rede será utilizada para explicação do método proposto em [4].

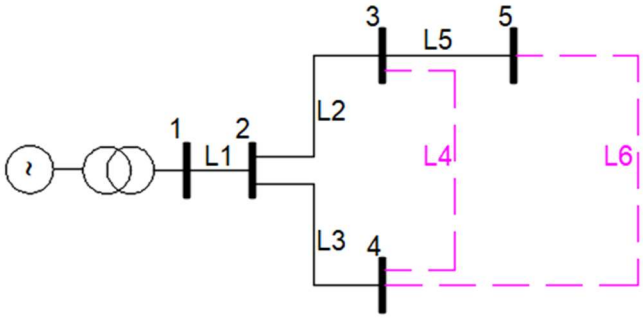


Figura 4: Rede 5 barramentos e 6 linhas

1. Matriz de admitância nodal

A matriz de admitância nodal é uma matriz quadrada, ou seja, a quantidade de linhas é igual a quantidade de colunas e a quantidade de linhas é igual ao número de barramentos presentes na rede em questão. Neste caso, a dimensão da matriz de admitância nodal é 5x5.

Ela é preenchida da seguinte forma: Na diagonal principal, $\mathbf{Y}_{i,i}$, é inserido a soma das admitâncias das linhas que estão fisicamente conectadas ao barramento. Para simplificar o preenchimento, considera-se que todas as linhas têm admitância igual a 1. No exemplo apresentado, o elemento $\mathbf{Y}_{1,1}$ é igual a 1, o elemento $\mathbf{Y}_{2,2}$ é igual a 3 e assim sucessivamente.

Os elementos fora da diagonal principal, $\mathbf{Y}_{i,j}$, foram preenchidos de acordo com os estados dos seccionadores das linhas que interligam os barramentos. Em casos em que o seccionador da linha que interliga 2 barramentos estiver fechado, é inserido o valor -1 na posição da matriz e em casos em que o seccionador da linha que conecta barramentos estiver aberto, o valor a ser inserido é zero. No exemplo, como o seccionador da linha que interliga o barramento 1 ao barramento 2 está fechado, o elemento $\mathbf{Y}_{1,2} = -1$. Já entre o barramento 1 e o barramento 3, o seccionador da linha está aberto, por isso, $\mathbf{Y}_{1,3} = 0$.

Uma característica importante da matriz de admitâncias é ela é simétrica, ou seja, os termos $\mathbf{Y}_{i,j}$ são iguais aos termos $\mathbf{Y}_{j,i}$.

Seguindo as regras descritas, a matriz de admitâncias nodais foi totalmente preenchida conforme apresentada na equação (2.1).

$$\boldsymbol{Y} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & -1 & 0 \\ 0 & -1 & 3 & 0 & -1 \\ 0 & -1 & 0 & 3 & 0 \\ 0 & 0 & -1 & 0 & 2 \end{bmatrix} \quad (2.1)$$

2. Matriz de impedâncias

A matriz de impedâncias equivale à inversa da matriz de admitâncias nodais.

$$\boldsymbol{Z} = \boldsymbol{Y}^{-1} \quad (2.2)$$

Resultando na matriz abaixo.

$$\boldsymbol{Z} = j\begin{bmatrix} 1{,}789 & 0{,}789 & 0{,}315 & 0{,}263 & 0{,}157 \\ 0{,}789 & 0{,}789 & 0{,}315 & 0{,}263 & 0{,}157 \\ 0{,}315 & 0{,}315 & 0{,}526 & 0{,}105 & 0{,}263 \\ 0{,}263 & 0{,}263 & 0{,}105 & 0{,}421 & 0{,}052 \\ 0{,}157 & 0{,}157 & 0{,}263 & 0{,}052 & 0{,}631 \end{bmatrix} \quad (2.3)$$

Como a matriz de impedâncias não teve valores iguais a zero, assume-se que não foram detetadas redes isoladas e que a rede é radial ou anel aberto.

2.2.1.1 Exemplo de rede com ilha

Na Figura 5, tem-se um exemplo de rede em que o seccionador das linhas L1, L4, L5 e L6 estão fechados e os seccionadores das linhas L2 e L3 estão abertos.

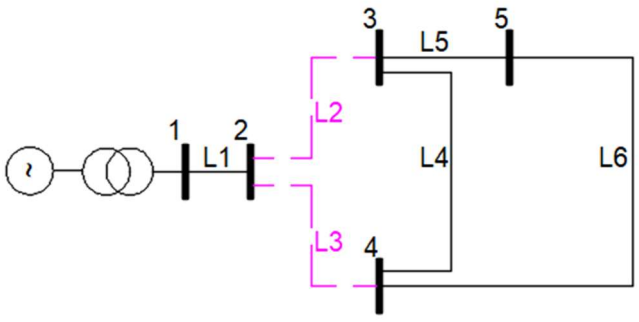


Figura 5: Rede 5 barramentos e 6 linhas com ilha, i.e., barramentos (1,2) não conexos com barramentos (3,4,5)

A fim de verificar o método de deteção de redes com ilha, o passo a passo descrito na seção anterior foi realizado para a rede da Figura 5 e as matrizes resultantes estão representadas abaixo.

$$Y = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 3 & 0 & 0 & 0 \\ 0 & 0 & 3 & -1 & -1 \\ 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & 2 \end{bmatrix} \tag{2.4}$$

$$Z = \begin{bmatrix} 1{,}500 & 0{,}500 & 0 & 0 & 0 \\ 0{,}500 & 0{,}500 & 0 & 0 & 0 \\ 0 & 0 & 0{,}625 & 0{,}375 & 0{,}500 \\ 0 & 0 & 0{,}375 & 0{,}625 & 0{,}500 \\ 0 & 0 & 0{,}500 & 0{,}500 & 1{,}000 \end{bmatrix} \tag{2.5}$$

Como era de se esperar, a matriz de impedâncias possui valores iguais a zero, configurando-se assim, uma rede com ilha.

2.3 Trânsito de energia

A solução de um sistema de energia elétrica em regime estacionário é designada por trânsito de energia e compreende a rede, os geradores e as cargas [5].

O objetivo do trânsito de energia é o cálculo das tensões nodais, e isto é feito pelos seguintes passos:

- Formulação de um modelo matemático para descrever o sistema real;
- Especificação dos tipos de barramentos e grandezas;
- Solução numérica das equações do modelo matemático;
- Cálculo das potências que transitam em todos os ramos – linhas e transformadores.

As equações que modelam o trânsito de energia são não-lineares. Para se encontrar as soluções destas equações, são utilizados métodos iterativos, dentre eles está o método de Gauss-Seidel que será abordado na secção 2.3.2.

2.3.1 Equações do trânsito de energia

A Figura 6 representa um sistema com 2 barramentos ligados por uma linha.

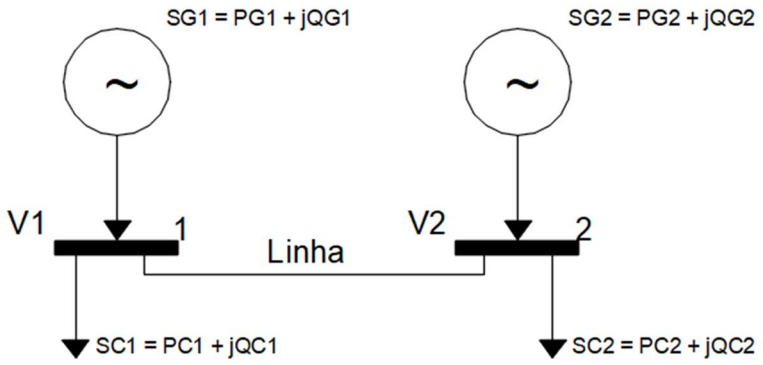


Figura 6: Sistema com 2 barramentos

Nesta figura, cada barramento é alimentado por gerador que está representado pela sua potência complexa $\mathbf{S}_{Gi}$ e estão ligadas cargas que absorvem as potências complexas $\mathbf{S}_{Ci.}$ Os barramentos estão conectados por uma linha de transporte cuja admitância é **Y**i.

Potência injetada **S** é definida como a diferença entre a potência gerada e consumida em cada barramento. Para o sistema da Figura 6, as equações da potência injetada são:

$$\boldsymbol{S}_1 = P_1 + jQ_1 = P_{G1} - P_{C1} + j(Q_{G1} - Q_{C1}) \tag{2.6}$$

$$\boldsymbol{S}_2 = P_2 + jQ_2 = P_{G2} - P_{C2} + j(Q_{G2} - Q_{C2}) \tag{2.7}$$

A corrente injetada **I** no barramento em função da potência injetada e tensão nodal é dada pela equação (2.8).

$$\boldsymbol{I} = \frac{\boldsymbol{S}^*}{\boldsymbol{V}^*} = \frac{P - jQ}{\boldsymbol{V}^*} \tag{2.8}$$

De forma matricial,

$$[\boldsymbol{I}] = [\boldsymbol{Y}][\boldsymbol{V}] \tag{2.9}$$

2.3.2 Método iterativo de Gauss-Seidel aplicado ao trânsito de energia

O método iterativo de Gauss-Seidel inicia-se com a identificação dos barramentos de acordo com suas características: barramento de referência, barramento do tipo PV e barramento do tipo PQ:

a) Barramento de referência: Neste barramento, o módulo e argumento da tensão estão especificados. São calculados a potência ativa e reativa. Normalmente, só existe um barramento de referência.
b) Barramentos do tipo PQ: As potências ativas e reativas são especificadas. São calculados o módulo e argumento da tensão.
c) Barramentos do tipo PV: São os barramentos de geração. Os valores da potência ativa e módulo da tensão são especificados. São calculados a potência reativa e o argumento da tensão.

Normalmente, admite-se que o nó de referência tem o módulo da tensão igual a 1 p.u. e argumento 0°.

Depois da identificação dos tipos de barramentos e da definição da tensão e argumento do barramento de referência, é calculada a matriz de admitâncias nodais conforme demonstrado na seção 2.2.1.

Com isto, é possível escrever a equação (2.10), forma matricial da corrente injetada para a rede de 2 barramentos da Figura 6, assumindo-se que o nó 1 é o de referência.

$$\begin{bmatrix} \frac{\boldsymbol{S}_1^*}{1\angle 0^\circ} \\ \frac{\boldsymbol{S}_2^*}{\boldsymbol{V}_2^*} \end{bmatrix} = j \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{bmatrix} \begin{bmatrix} 1\angle 0^\circ \\ \boldsymbol{V}_2 \end{bmatrix} \tag{2.10}$$

Resolvendo-se a equação matricial, é possível transformar as matrizes da equação (2.10) em um sistema com duas equações. Assumindo-se que o valor da potência injetada no barramento 2 é conhecido, a equação (2.11), possui apenas a variável $\mathbf{V}_2$.

$$\frac{\boldsymbol{S}_2^*}{\boldsymbol{V}_2^*} = jy_{21} \times 1\angle 0^\circ + jy_{22} \times \boldsymbol{V}_2 \tag{2.11}$$

Da equação (2.11), $\mathbf{V}_2$ é colocado em evidência:

$$\boldsymbol{V}_2^{(k+1)} = \frac{1}{-jy_{22}} \left[\frac{\boldsymbol{S}_2^*}{\boldsymbol{V}_2^{*(k)}} - jy_{21} \times 1\angle 0^\circ \right] \tag{2.12}$$

Neste momento é iniciado o contador de iterações, k. k = 0.

Para resolver a iteração 1 é preciso inicializar o valor de $\mathbf{V}_2^{(0)}$. Considerou-se:

$$\boldsymbol{V}_2^{(0)} = 1\angle 0° \tag{2.13}$$

Desta forma é possível resolver a iteração 1 da equação (2.12). Depois de solucionada calcula-se o erro entre as iterações consecutivas que a expressão está representada na equação (2.14).

$$\varepsilon^{(k+1)} = \left|\left|\boldsymbol{V}_2^{(k+1)}\right| - \left|\boldsymbol{V}_2^{(k)}\right|\right| \tag{2.14}$$

Os cálculos do sistema iterativo terminam quando o erro é igual ou inferior ao mínimo estipulado. Normalmente esse valor é 0,0001 pu, ou seja, 10^{-4} pu.

Quando o sistema iterativo termina, o valor de $\mathbf{V}_2$ é substituído na equação (2.10) e o valor de $\mathbf{S}_1^*$ é calculado.

O método de Gauss-Seidel possui uma convergência relativamente lenta, e, dependendo da complexidade da rede, necessita de dezenas ou centenas de iterações até atingir o resultado. Por este motivo, usa-se este método em redes de tamanho pequeno ou médio [5].

2.4 Formulação matemática do problema de reconfiguração topológica

O problema de reconfiguração topológica de redes pode ser formulado matematicamente através da equação (2.15):

$$Minimizar_{\ z\in\Omega}\ p_{total}(z) = \sum_{j=1}^{n} r_{ramoj(z)}\ \frac{P_{ramoj(z)}^2 + Q_{ramoj(z)}^2}{\left|\boldsymbol{V}_{\boldsymbol{ramoj(z)}}\right|^2} \tag{2.15}$$

s.a.

$$\boldsymbol{V}_{min} \leq |\boldsymbol{V}_i| \leq \boldsymbol{V}_{max} \tag{2.16}$$

$$\left|\boldsymbol{I}_{ramo}\ \right| \leq \left|\boldsymbol{I}_{ramoj\,max}\right| \tag{2.17}$$

Em que:

$|\boldsymbol{V}_i|$: Módulo de tensão no barramento "i" [pu];

$\boldsymbol{V}_{min}$ e $\boldsymbol{V}_{max}$: Limites admissíveis de tensão no barramento "i" [pu];

$|\boldsymbol{I}_{ramoj}|$: Módulo da corrente elétrica do ramo "j" [pu];

$|\boldsymbol{I}_{ram\quad max}|$: Valor máximo admissível da corrente elétrica do ramo "j" [pu];

n: Número de ramos da rede;

z: Topologia da rede;

$p_{total(z)}$: Potência ativa total de perdas associada à topologia da rede [pu];

$P_{ramoj(z)}$: Potência ativa do ramo "j" associada à topologia da rede [pu];

$Q_{ramoj(z)}$: Potência reativa do ramo "j" associada à topologia da rede [pu];

$r_{ramoj(z)}$: Resistência do ramo "j" associada à topologia da rede [pu];

$V_{ramoj(z)}$: Tensão do ramo "j" associada à topologia da rede [pu].

Reconfiguração topológica de redes para redução de perdas é um problema combinatório e de natureza não linear com função objetiva quadrática. As variáveis de decisão são os estados dos seccionadores das linhas e o problema está sujeito as restrições dos limites admissíveis de tensão e corrente, equações (2.16) e (2.17). O valor das potência ativa total de perdas é determinado pela solução do trânsito de potências fornecido pelas variáveis de decisão.

2.5 Redução de perdas através de reconfiguração topológica de redes

Na literatura, a metodologia de redução de perdas através de reconfiguração foi primeiramente abordada em 1975 por Merlin; Back, referenciado em [6] e, desde então, muitas estratégias de redução de perdas seguindo a metodologia de reconfiguração foram desenvolvidas.

As novas configurações de redes são obtidas fundamentalmente através da abertura e fechamento das chaves de seccionamento, com o objetivo de encontrar novos caminhos para alimentação das cargas.

As estratégias de redução de perdas por reconfiguração de redes podem ser classificadas em dois grupos [7]:

1. Métodos heurísticos e otimização matemática

   O uso de métodos heurísticos foi justificado pela necessidade de redução do espaço de busca do problema de reconfiguração e as técnicas de otimização matemática incluem programação linear, programação dinâmica. No capítulo 3 será apresentado um algoritmo de redução de perdas [8] que tem como base o método de otimização matemática para reconfiguração iterativa da rede e que utiliza o método de Gauss-Seidel para a solução do trânsito de potências.

2. Abordagens baseadas em computação evolutiva

   São métodos de busca guiados por uma estratégia que tenta evitar os mínimos locais do espaço de busca. Entre os métodos desenvolvidos neste âmbito, destacam-se os Algoritmos Genéticos – AG [9], Colônias de Formigas - *Ant Colony Optimization* - ACO [10] e Otimização por Nuvens de Partículas - *Particle Swarm Optimization* – PSO [11].

   No capítulo 4 serão descritos algoritmos que aplicam Algoritmos Genéticos ao problema de reconfiguração de rede para minimização de perdas.

CAPÍTULO 3

3. ALGORITMO DE RECONFIGURAÇÃO ITERATIVA DA REDE - MÉTODO HEURÍSTICO

3.1 Introdução

O algoritmo heurístico de reconfiguração iterativa da rede abordado neste capítulo teve como base o artigo "*A New Heuristic Approach for Optimal Network Reconfiguration in Distribution Systems*" [8].

O objetivo deste algoritmo é identificar a topologia que representa a menor perda possível da rede em estudo, obedecendo à restrição de que a topologia da rede deve se manter radial ou anel aberto e respeitar os limites de operação da rede.

3.2 Descrição do algoritmo

O algoritmo é inicializado com a leitura das informações da rede a ser otimizada. A configuração inicial da rede deve indicar quais os seccionadores estão abertos, pois este dado determina a quantidade total de ciclos do algoritmo e também é o ponto de partida deste.

Cada ciclo é composto por duas etapas, na primeira etapa, de acordo com os critérios a serem apresentados a seguir, um dos seccionadores abertos é fechado, outro é aberto em seu lugar e as perdas são calculadas.

Na segunda etapa do ciclo, o último seccionador que foi aberto é fechado, o seccionador da linha adjacente é aberto e as perdas são calculadas. A segunda etapa do ciclo repete-se até que as perdas da configuração atual da rede sejam superiores às da configuração anterior.

O fluxograma que está representado na Figura 7 exibe as etapas do método de reconfiguração iterativa do artigo de referência.

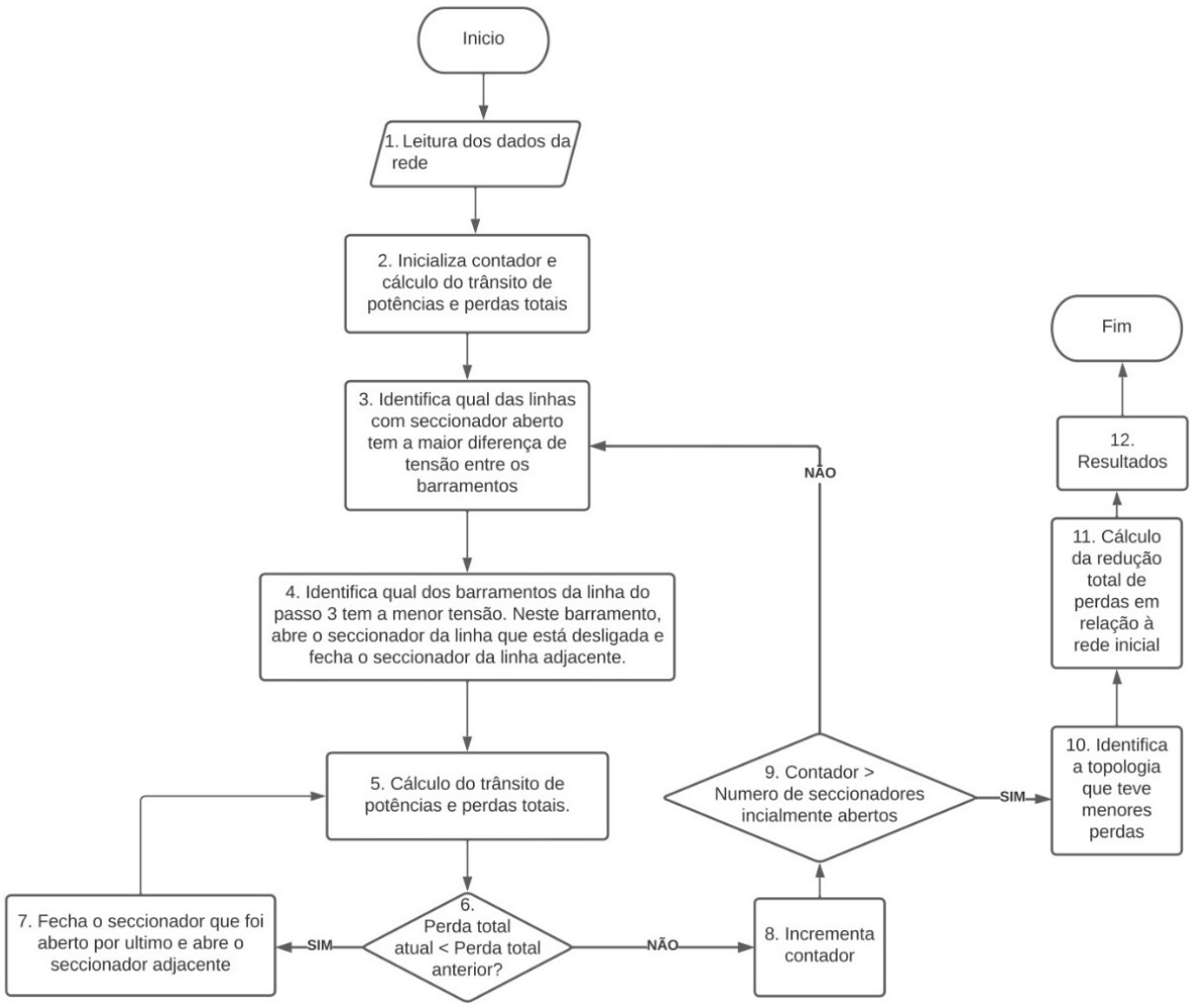


Figura 7: Fluxograma de funcionamento do algoritmo de reconfiguração - método heurístico (adaptado de [26])

Do fluxograma, a metodologia do algoritmo é constituída pelas seguintes etapas:

1. Leitura dos dados da rede:

   Nesta etapa, é identificada a quantidade de linhas com seccionadores abertos.

2. Inicializar contador e cálculo do trânsito de potências e perdas totais:

   O contador é incrementado a cada ciclo completo. O trânsito de potências é calculado. As perdas ativas totais são calculadas a partir das equações apresentadas na equação (2.15).

3. Identificar qual das linhas com seccionador aberto tem a maior diferença de tensão entre os barramentos:

   A linha que tiver o maior módulo da diferença de tensão entre os barramentos interligados pelas linhas com seccionadores abertos é selecionada para o próximo passo.

   O módulo da diferença de tensão da linha entre os barramentos está representado na equação (3.10).

$$\boldsymbol{V}_{i,i} = \left|\boldsymbol{V}_j - \boldsymbol{V}_i\right| \tag{3.10}$$

4. Identificar qual dos barramentos da linha do passo 3 tem a menor tensão:

   Neste barramento, fecha o seccionador da linha que está desligada e abre o seccionador da linha adjacente: Antes de fazer esta manobra, é necessário verificar se a linha adjacente pode ter o seccionador aberto. Se a rede continuar radial ou anel aberto após esta alteração, a manobra é efetuada, caso o contrário, é selecionada outra linha adjacente.

5. Cálculo do trânsito de potências e perdas totais:

   Os valores calculados e a topologia são armazenados para a identificação da configuração da rede que apresenta a menor perda.

6. Perda total atual < perda total anterior:

   É verificada se a perda ativa total da topologia atual é inferior à perda ativa calculada na topologia anterior. Caso seja verdadeiro, o algoritmo passa para o ponto 7, caso contrário, passa para o ponto 8.

7. Fechar o último seccionador que foi aberto e abrir o seccionador adjacente:

   Antes de efetuar essa manobra deve ser verificado novamente se a rede continua radial. Após o ponto 7, o algoritmo retorna ao ponto 5.

8. Incrementar contador:

   Após o ponto 6, caso a perda total da topologia atual seja superior a perda total da topologia anterior, desconsidera-se a última configuração, retornando para a anterior.

   Houve um ciclo completo e o contador é incrementado.

9. Contador > Número de seccionadores inicialmente abertos:

   A contagem de ciclos completos é realizada para verificar se a condição de término do algoritmo já foi atingida. Caso seja verdadeiro, continua para o passo 10 e caso contrário, retorna para o passo 3.

10. Identificar a topologia que teve menores perdas:

    Neste passo, as informações de todas as topologias testadas e suas respetivas perdas totais calculadas, previamente armazenados, é consultado e a topologia com a menor perda é selecionada.

11. Cálculo da redução total de perdas em relação à rede inicial:

    Com os dados obtidos no passo 10, é feita uma relação entre as perdas da rede na configuração inicial e a rede com a reconfiguração que assegurou a melhor minimização das perdas.

12. Resultados:

    É feita a apresentação dos resultados, com a indicação de qual ciclo que apresenta a melhor reconfiguração, o valor da redução percentual e quais seccionadores devem ser abertos.

3.3 Caracterização da rede IEEE 33 barramentos

A rede de teste utilizada para avaliar o desempenho dos algoritmos de reconfiguração de redes desenvolvidos neste trabalho é a IEEE 33 barramentos. Esta rede tem topologia radial possui as seguintes características:

- 37 linhas: L1 a L37;
- 5 linhas com seccionadores abertos: L33, L34, L35, L36 e L37;
- Tensão: 12,66kV;
- Potência de base: 100kVA;
- Carga total: 3,91 MW e 2,40 Mvar.

O diagrama unifilar do sistema de distribuição radial IEEE 33 barramentos é mostrado na Figura 8.

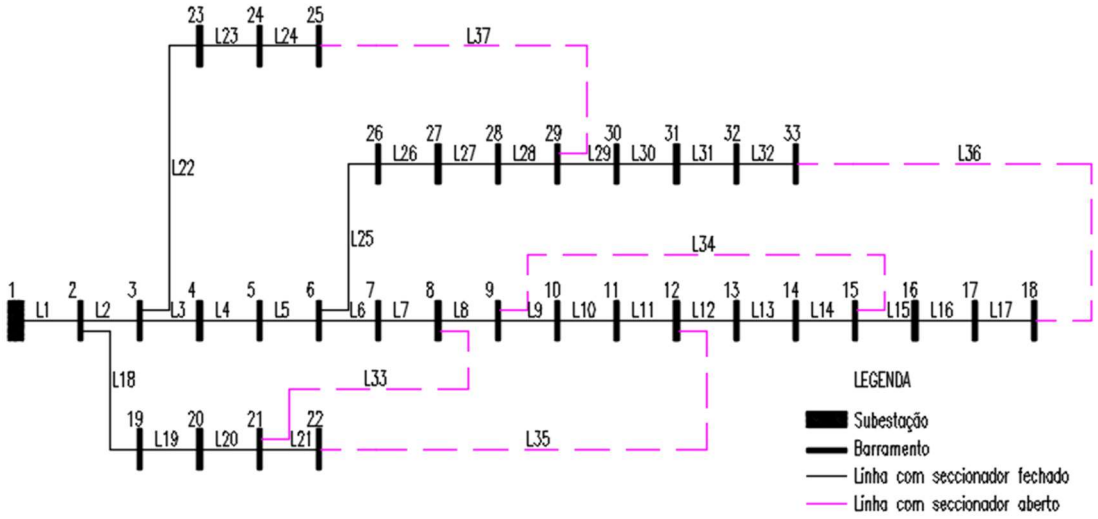


Figura 8: Diagrama unifilar da rede IEEE 33 Barramentos

A rede é alimentada a partir da subestação localizada no barramento 1 e inclui possibilidades de reconfiguração representadas pelos ramos L33, L34, L35, L36 e L37. As características da rede IEEE 33 podem ser encontradas no anexo A1.

3.4 Aplicação do algoritmo de reconfiguração iterativa à rede IEEE 33 barramentos

Para a aplicação do algoritmo de reconfiguração iterativa descrito na secção 3.3 à rede IEEE 33 barramentos, a seguinte consideração foi efetuada:

- Todas as linhas da rede, com exceção da linha L1, possuem seccionadores para possível reconfiguração.

O *script* do código desenvolvido para a aplicação do método está apresentado no anexo A2.

Os passos a realizar para a aplicação do algoritmo é descrito a seguir:

a) Leitura dos dados:

Nesta etapa ocorre a importação do arquivo *Excel* "IEEE_33_Bus.xlsx" para o script do código.

Foi identificado que a rede IEEE 33 barramentos possui 5 linhas com seccionadores abertos, isto indica que o algoritmo terá de completar 5 ciclos completos.

b) Inicialização do contador e cálculo do trânsito de potências:

O contador de ciclos é inicializado em 1.

As perdas calculadas para a configuração inicial da rede IEEE 33 barramentos, Figura 8, são de 2,0029 pu.

c) Primeira etapa do ciclo

As linhas com seccionadores abertos são: L33, L34, L35, L36 e L37.

O módulo da diferença de tensão entre os barramentos das linhas com seccionadores abertos está apresentado nas equações (3.11) a (3.15).

$$\boldsymbol{V}_{L33} = |\boldsymbol{V}_{21} - \boldsymbol{V}_{8}| = 0{,}0443 \text{ pu} \quad (3.11)$$

$$\boldsymbol{V}_{L3} = |\boldsymbol{V}_{15} - \boldsymbol{V}_{9}| = 0{,}0178 \text{ pu} \quad (3.12)$$

$$\boldsymbol{V}_{L3} = |\boldsymbol{V}_{22} - \boldsymbol{V}_{12}| = 0{,}0580 \text{ pu} \quad (3.13)$$

$$\boldsymbol{V}_{L36} = |\boldsymbol{V}_{33} - \boldsymbol{V}_{18}| = 0{,}0035 \text{ pu} \quad (3.14)$$

$$\boldsymbol{V}_{L37} = |\boldsymbol{V}_{29} - \boldsymbol{V}_{25}| = 0{,}0372 \text{ pu} \quad (3.15)$$

A linha com maior módulo da diferença de tensão entre os barramentos é a L35.

d) Segunda etapa do ciclo

- Iteração 1:

Foi calculado que o barramento 12 possui tensão inferior ao barramento 22, logo é o escolhido para ser estudado na segunda etapa deste ciclo.

As opções de linhas a terem os seccionadores desligados é a L12 e L11, contudo, se a L12 for desligada, a rede deixa de ser radial. Por este motivo, a linha L11 teve o seccionador desligado e a linha L35 teve o seccionador ligado. A nova configuração da rede está apresentada na Figura 9.

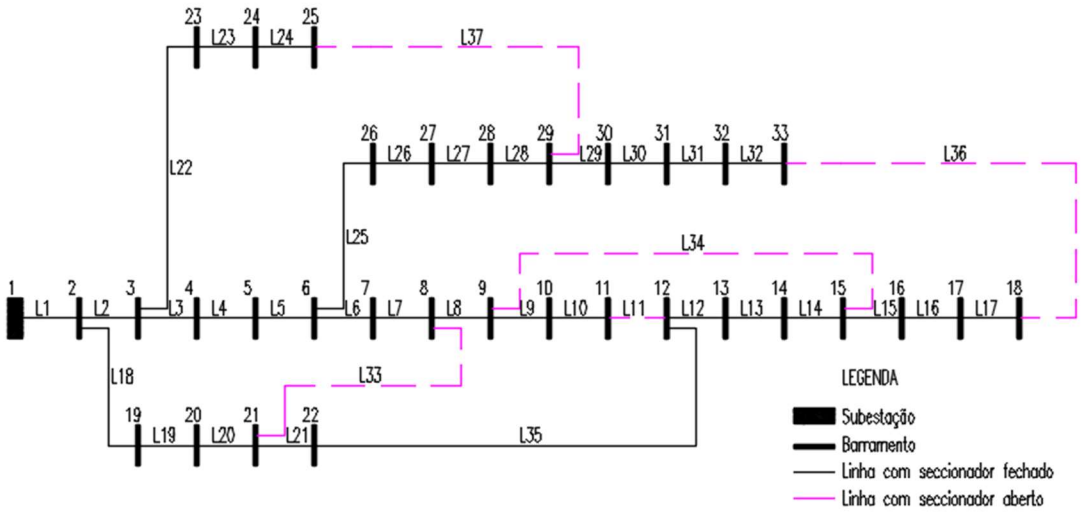


Figura 9: Iteração 1 - Rede IEEE 33 Barramentos

A perda total calculada para a rede da Figura 9 é 1,51 pu. A diferença das perdas é está calculada na equação (3.16)

$$P_{Iter_0} - P_{Iter_1} = 2{,}0029 - 1{,}5130 = 0{,}4899\, pu \quad (3.16)$$

- Iteração 2:

A linha adjacente à linha L11 é a L10. Se o seccionador da linha L10 for desligado, a linha continua radial. A nova configuração da rede está apresentada na Figura 10.

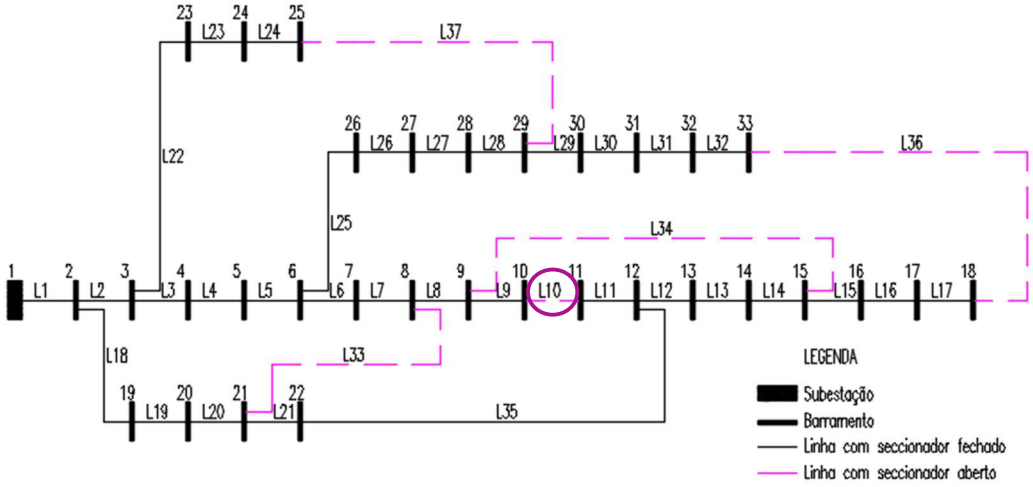


Figura 10: Iteração 2 - Rede IEEE 33 Barramentos

A perda total calculada para a rede da Figura 10 é 1,4899 pu e a diferença de perdas entre as iterações 1 e 2 é 0,0246 pu.

- Iteração 3:

A linha adjacente à linha L10 é a L9. Se o seccionador da linha L9 for desligado, a linha continua radial. A nova configuração da rede está apresentada na Figura 11.

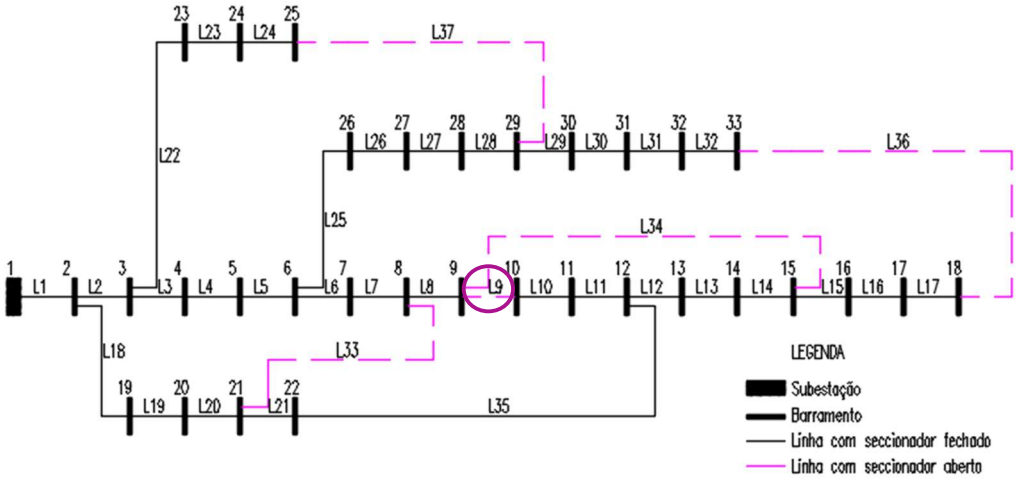


Figura 11: Iteração 3 - Rede IEEE 33 Barramentos

A perda total calculada para a rede da Figura 11 é 1,4664 pu e a diferença de perdas entre as iterações 2 e 3 é 0,022 pu.

- Iteração 4:

A linha adjacente à linha L9 é a L8. Se o seccionador da linha L8 for desligado, a linha continua radial. A nova configuração da rede está apresentada na Figura 12.

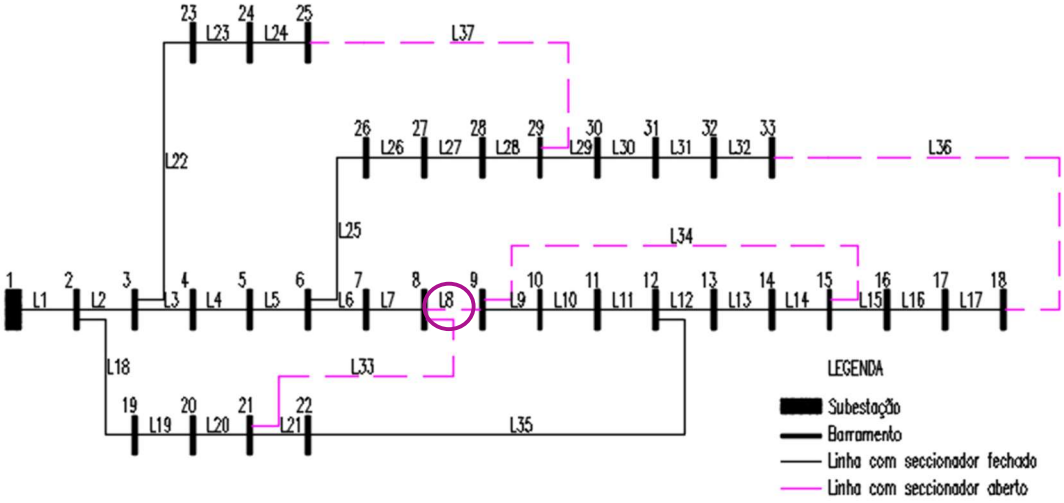


Figura 12: Iteração 4 - Rede IEEE 33 Barramentos

A perda total calculada para a rede da Figura 12 é 1,4495 pu e a diferença de perdas entre as iterações 3 e 4 é 0,0169 pu.

- Iteração 5:

A linha adjacente à linha L8 é a L7. Se o seccionador da linha L7 for desligado, a linha continua radial. A nova configuração da rede está apresentada na Figura 13.

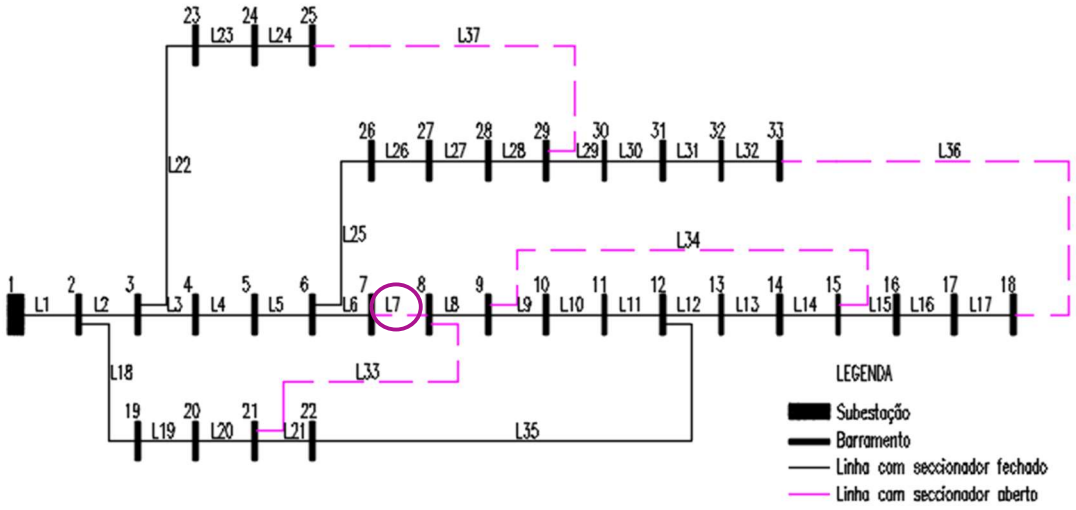


Figura 13: Iteração 5 - Rede IEEE 33 Barramentos

A perda total calculada para a rede da Figura 13 é 1,4268 pu e a diferença de perdas entre as iterações 4 e 5 é 0,0227 pu.

- Iteração 6:

A linha adjacente à linha L7 é a L6. Se o seccionador da linha L6 for desligado, a linha continua radial. A nova configuração da rede está apresentada na Figura 14.

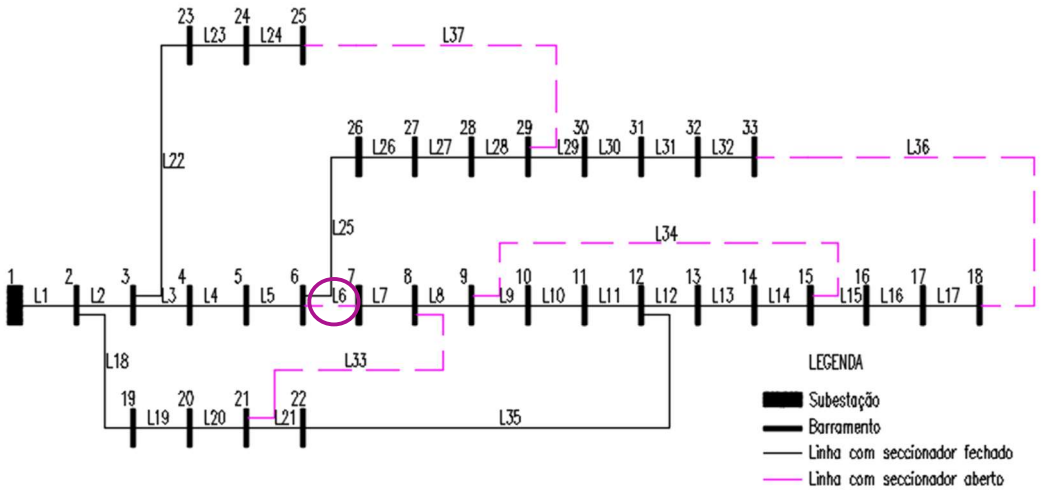


Figura 14 : Iteração 6 - Rede IEEE 33 Barramentos

A perda total calculada para a rede da Figura 14 é 1,4724 pu. A diferença das perdas é está calculada na equação (3.17)

$$P_{Iter_5} - P_{Iter_6} = 1{,}4268 - 1{,}4724 = -0{,}0456\, pu \qquad (3.17)$$

Como a perda calculada na iteração 6 é superior à da iteração 5, a configuração da iteração 6 é desconsiderada, a segunda etapa do primeiro ciclo termina e a rede inicial do segundo ciclo é a que está presente na Figura 13.

e) Resultados obtidos

A Tabela 1 apresenta um resumo de todos os resultados obtidos da aplicação do algoritmo de reconfiguração à rede IEEE 33 barramentos.

RESULTADOS ALGORITMO RECONFIGURAÇÃO - HEURISTICO			
Ciclo	**Iteração**	**Seccionadores abertos**	**Perdas calculadas [pu]**
Inicial		L33, L34, L35, L36, L37	2,0029
1	1	L11, L33, L34, L36, L37	1,5130
	2	L10, L33, L34, L36, L37	1,4884
	3	L9, L33, L34, L36, L37	1,4664
	4	L8, L33, L34, L36, L37	1,4495
	5	L7, L33, L34, L36, L37	1,4268
	6	L6, L33, L34, L36, L37	1,4724
2	7	L7, L8, L34, L36, L37	1,3926
	8	L6, L8, L34, L36, L37	1,3874
	9	L5, L8, L34, L36, L37	2,7314
3	10	L6, L8, L24, L34, L36	1,4005
	11	L6, L8, L24, L34, L36	1,4005
4	12	L6, L8, L34, L36, L37	1,3874
	13	L6, L8, L34, L36, L37	1,3874
5	14	L6, L8, L28, L34, L36	1,4005

Tabela 1: Resultados algoritmo reconfiguração – Método Heurístico

No segundo e quarto ciclo contêm a topologia que representa a menor perda geral obtida para a rede IEEE 33 barramentos, através do método heurístico, que apresenta a perda total de 1,3874 pu e a configuração está representada na Figura 15.

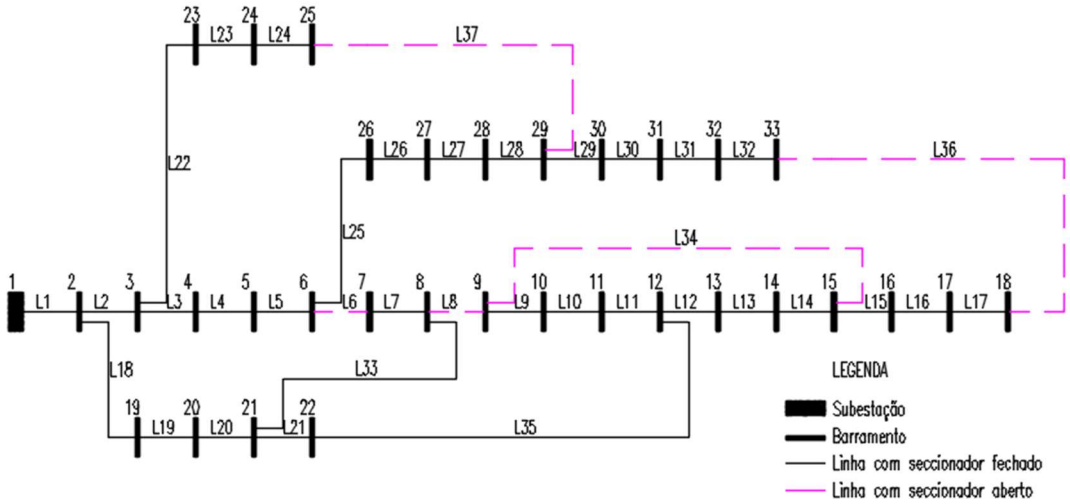


Figura 15: Iterações 8, 12, 13 - Rede IEEE 33 Barramentos

O cálculo da redução total das perdas está presente na equação (3.18)

$$Redução\ total = \frac{(2{,}0029-1{,}3874)}{2{,}0029} \times 100 = 31\% \qquad (3.18)$$

A configuração da Figura 15 apresenta a maior redução total das perdas, que é de 31% em relação à configuração inicial da rede.

f) Comparação dos resultados obtidos com os do artigo

Os resultados obtidos com o algoritmo desenvolvido estão aproximados aos do artigo de referência, conforme está apresentado na Tabela 2.

A diferença entre os resultados do cálculo das perdas na configuração inicial da rede IEEE 33 barramentos apresentados no algoritmo desenvolvido e no artigo, pode ser devido ao fato de que, embora a rede de teste seja *standard*, os parâmetros desta rede não estão disponíveis em uma única fonte, o que pode significar que os valores utilizados no artigo são ligeiramente diferentes dos parâmetros de entrada utilizados no desenvolvimento deste algoritmo.

	Configuração inicial [pu]	**Configuração final [pu]**
Algoritmo	2,0029	1,3874
Artigo	2,0271	1,3578

Tabela 2: Comparação resultados artigo e algoritmo

3.5 Considerações finais

Os resultados obtidos no algoritmo desenvolvido foram satisfatórios quando comparado aos do artigo de referência, no entanto, foram identificadas as seguintes limitações deste método:

- O espaço de busca é restrito;
- O resultado encontrado sempre será o mesmo, independentemente de quantas vezes o código do algoritmo for executado, já que se trata de sequência de passos matemáticos.

Por conta das limitações apontadas acima, outros métodos de reconfiguração topológica foram estudados e, dentre esses, o método da aplicação de algoritmos genéticos ao problema de reconfiguração apresenta maiores possibilidades de identificação de resultados. No capítulo seguinte este tópico será discutido com maior detalhe.

CAPÍTULO 4

4. ALGORITMOS GENÉTICOS APLICADOS À RECONFIGURAÇÃO DE REDES

4.1 Introdução

Compreende-se por algoritmo genético, AG, um conjunto definido de regras e processos com operações finitas que são fundamentadas no processo de evolução biológica em que ocorre a transmissão dos caracteres hereditários nos indivíduos e os mecanismos que asseguram essa transmissão. [12]

Na aplicação dos AG's para solução de problemas reais, ocorre a manipulação de indivíduos que são possíveis soluções para o problema. Os indivíduos mais adaptados passam pelo processo de combinação do material genético que resulta na produção dos filhos e estes filhos podem ou não sofrer alteração do seu material genético. Os melhores indivíduos da população atual, junto com os novos indivíduos gerados, constituem uma nova geração. Desta forma, ocorrem evoluções entre as gerações até que a população que apresenta melhores soluções para o problema é encontrada e o processo termina.

Diversos trabalhos académicos foram produzidos com a aplicação de algoritmos genéticos ao problema de reconfiguração topológica de redes para redução de perdas. Exemplos destes trabalhos são encontrados em [13], [14] e [15].

Dentre as possíveis razões que justificam o desenvolvimento dos métodos com algoritmos genéticos, está a simplicidade de se formular o problema, onde, para a aplicação de reconfiguração, as variáveis de decisão são as manobras de abertura dos seccionadores da linha. Tratando-se de um problema com uma natureza combinatória os AG's são particularmente eficazes em encontrar soluções do problema com elevada qualidade. Permitem igualmente disponibilizar mais do que uma solução com qualidades superiores em problemas não convexos.

4.2 Algoritmo Genético binário e inteiro

Os problemas de reconfiguração da rede podem ser codificados da forma binária, com números reais, inteiros e alfanumérica. Neste trabalho, o AG desenvolvido foi codificado com números inteiros.

Os genes do algoritmo genético representam a menor unidade do cromossoma que são as variáveis de decisão do problema. Na codificação binária, cada gene apenas pode assumir o valor zero ou um. Na Figura 16, é exemplificado um cromossoma com 5 genes binários.

1	0	0	1	1

Figura 16: Exemplo cromossoma binário

Já na codificação com números inteiros, cada gene pode assumir números inteiros, como apresentado no exemplo da Figura 17.

33	15	27	5	23

Figura 17: Exemplo cromossoma codificado com números inteiros

4.3 Características do algoritmo genético

Nos algoritmos genéticos cada indivíduo da população é representado por um cromossoma. Cada cromossoma contém uma possível solução para o problema e eles são implementados na forma de vetores, como demonstrado na Figura 16 e na Figura 17, e cada elemento do vetor é denominado gene.

Uma característica importante dos algoritmos genéticos é que eles formam uma população com várias soluções candidatas, o que faz com que o processo de busca pela solução mais apropriada ao problema seja multidirecional.

Os indivíduos mais adaptados, sobrevivem para as gerações futuras e a forma de determinar o nível de adaptação dos cromossomas é através da função de avaliação ou função objetivo ou ainda *fitness function.*

Figura 18: Estrutura de um algoritmo genético (adaptado de [28])

O fluxograma da Figura 18 apresenta a estrutura de um algoritmo genético que é composta pela população, função de avaliação, seleção, cruzamento, mutação e critério de paragem que serão detalhados a seguir.

4.3.1 População

A população representa um conjunto de indivíduos com possibilidade de solucionar o problema a ser otimizado. Cada indivíduo da população é denominado cromossoma. O tamanho da população está diretamente relacionado à dimensão do

espaço de busca do algoritmo [16] e, por isso, pode ter influência no sucesso dos algoritmos genéticos, já que, dependendo do tamanho da população, se perde a diversidade necessária para convergir para uma boa solução.

Por outro lado, uma população com muitos indivíduos pode prejudicar o processamento computacional do algoritmo.

No presente trabalho consideram-se que todos os indivíduos representam soluções admissíveis do problema, isto é, respeitam as restrições do problema e que representam redes conexas.

4.3.2 Função de avaliação

A função de avaliação, ou *fitness function*, é a componente do algoritmo genético utilizada como métrica de avaliação dos cromossomas apresentados como possíveis soluções para o problema de otimização.

Esta avaliação permite direcionar, a cada geração, o espaço de busca para os melhores resultados.

É fundamental que a *fitness function* seja precisa na representatividade dos resultados obtidos de forma a diferenciar bem as soluções adequadas daquelas inadequadas.

Cada cromossoma da população, passa pela função de avaliação e o resultado obtido é o valor de avaliação.

4.3.3 Seleção

No seguimento do processo de evolução do algoritmo genético, é realizada a seleção dos cromossomas com maiores chances de sobrevivência.

Dentre os métodos de seleção já desenvolvidos pelos pesquisadores, o método da roleta e o método de seleção por torneio são aplicados em diversos estudos.

No método da seleção por roleta, cada indivíduo é representado na roleta de acordo com o valor de avaliação que lhe foi atribuído. Para aqueles que possuem uma melhor função de avaliação é atribuída uma porção maior da roleta e os que possuem piores funções de aptidão recebem menor porção na roleta.

A Tabela 3 contém um exemplo de aplicação do método da roleta. A população do exemplo contém 4 indivíduos, A, B, C e D. Sendo o indivíduo A, aquele que apresenta o maior valor de avaliação e o indivíduo D aquele com o menor valor. A probabilidade de seleção é calculada pela razão entre o valor de avaliação e o somatório dos valores de avaliação.

São realizados sorteiros aleatórios de valores entre 0 e 1. Na coluna "faixa favorável" da Tabela 3 estão indicados quais valores entre 0 e 1 que correspondem a cada um dos indivíduos. Por exemplo, o valor 0, 55 equivale ao indivíduo B, já que a faixa favorável deste indivíduo é entre 0,50 e 0,75.

Indivíduo	**Avaliação**	**Probabilidade de seleção**	**Faixa favorável**
A	5,0	50%	0 – 0,50
B	2,5	25%	0,50 – 0,75
C	1,5	15%	0,75 – 0,90
D	1,0	10%	0,90 – 1,00
Somatório	**10,0**	**100%**	

Tabela 3: Probabilidades de seleção método da roleta

A Figura 19 ilustra as faixas favoráveis de cada indivíduo da população do exemplo apresentado na Tabela 3. A roleta é girada um determinado número de vezes e os indivíduos sorteados pela roleta são selecionados.

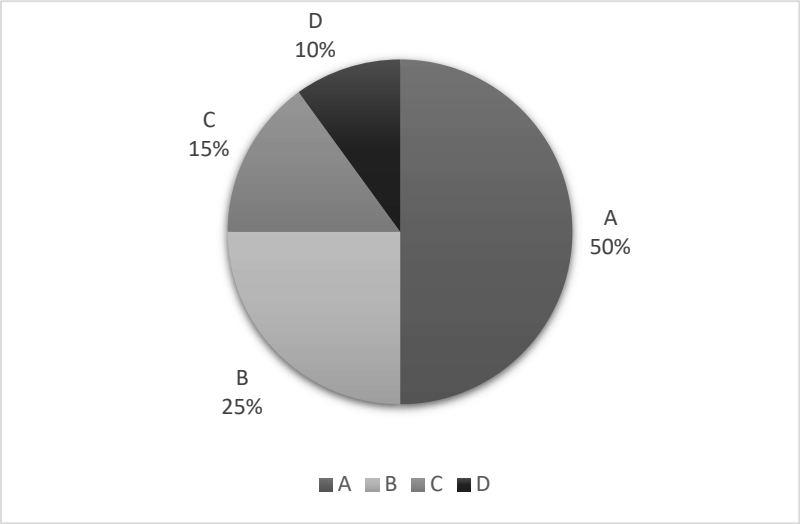


Figura 19: Método de seleção por roleta

No método da seleção por torneio, escolhe-se (normalmente 2) indivíduos de forma aleatória. Os valores de aptidão destes indivíduos são comparados e o vencedor, indivíduo selecionado, é aquele que possui o melhor valor de aptidão. O número de torneios realizados é igual ao número de indivíduos a serem selecionados.

4.3.4 Elitismo

O elitismo é uma estratégia complementar ao método da seleção. Essa técnica é utilizada para aumentar a velocidade de convergência do algoritmo genético, já que mantém no algoritmo os indivíduos com melhores valores de avaliação.

Por outro lado, se a solução do algoritmo genético estiver em um ótimo local, seu desempenho pode ser comprometido, pois neste caso, aumentaria a dificuldade de encontrar a melhor solução global.

4.3.5 Cruzamento

Na etapa do cruzamento (*crossover*) ocorre a combinação entre os genes dos "pais", originando assim os "filhos" que formarão a população da geração seguinte.

Dentre as diversas formas de cruzamento desenvolvidas, existe o cruzamento em um ponto e o cruzamento em dois pontos.

O cruzamento em um ponto está representado na Figura 20. No cruzamento em um ponto é definido um único ponto de cruzamento.

No exemplo, o ponto de cruzamento escolhido fica após o terceiro gene, da esquerda para a direita. Após a definição do ponto de cruzamento, ocorre a troca de material genético entre o indivíduo 1 e o indivíduo 2, gerando assim, os descendentes 1 e 2.

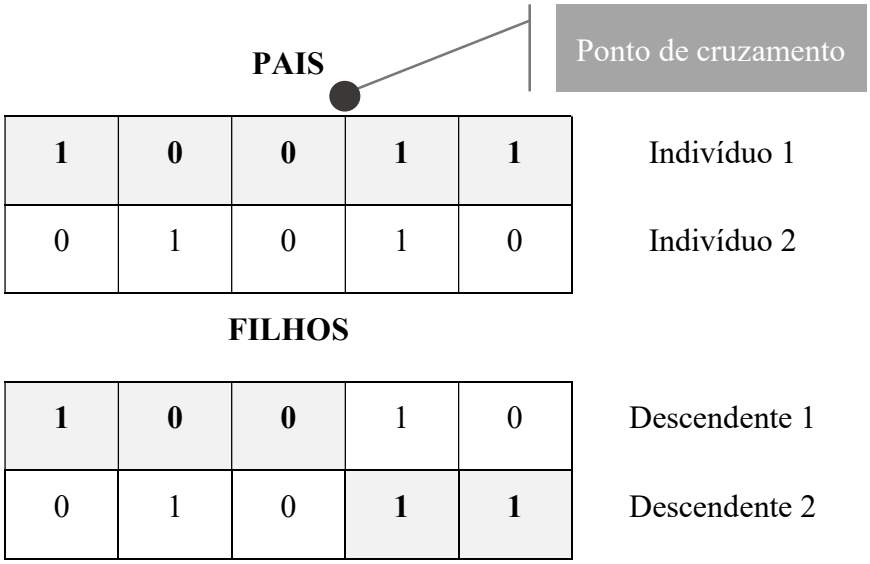


Figura 20: Exemplo de cruzamento em um ponto

O cruzamento em dois pontos está representado na Figura 21.

No cruzamento em dois pontos, são definidos dois pontos de cruzamento. Na Figura 21, estão demonstrados os dois pontos de cruzamento: um após o primeiro gene, da esquerda para a direita e outro após o terceiro gene, da esquerda para a direita.

Após a definição dos pontos de cruzamento, ocorre a troca de material genético dos genes que estão entre os 2 pontos de cruzamento, gerando assim, os descendentes 1 e 2.

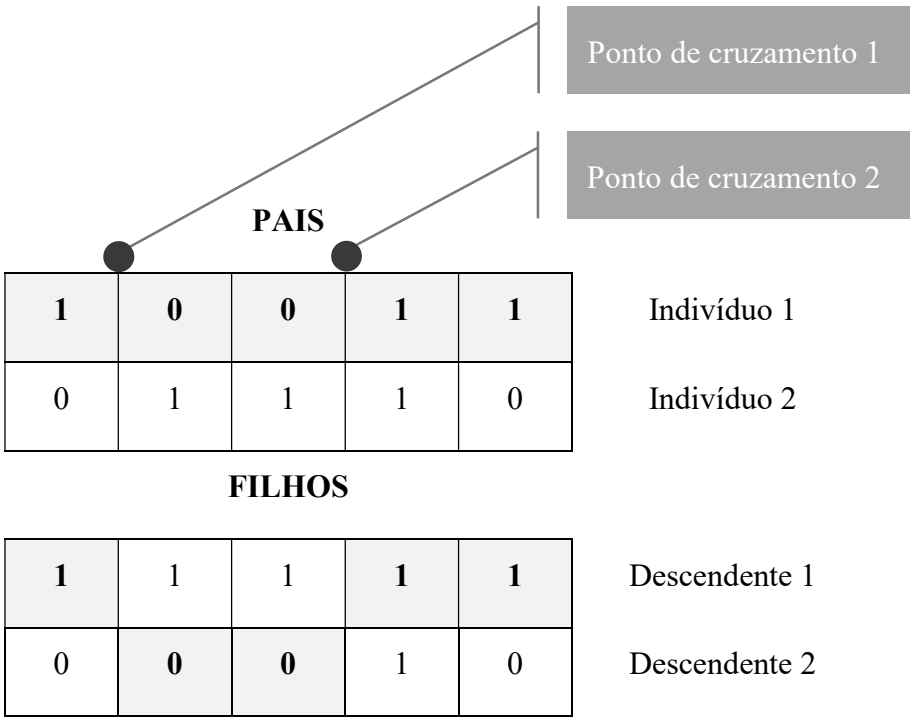


Figura 21: Exemplo cruzamento em dois pontos

O cruzamento pode ser visto como um movimento da solução no hiperespaço. Se o movimento for no sentido correto, a *fitness* aumenta (problema de maximização) ou diminui (problema de minimização).

4.3.6 Mutação

É a operação de mudar aleatoriamente um ou mais genes dentro de um indivíduo. Com a mutação, faz-se uma busca na vizinhança da solução e promove redução do risco de conversão para um valor de mínimo local. É utilizada para garantir diversidade na população.

Apenas uma parte da população passa pela etapa de mutação, a quantidade de indivíduos que passarão pela mutação é definida pela taxa de mutação.

A Figura 22 mostra um exemplo da técnica de mutação de apenas um gene de um indivíduo. O gene de mutação escolhido é o quarto. Como trata-se de um exemplo com genes binários, no gene de mutação, houve a troca o valor 1 para o valor 0.

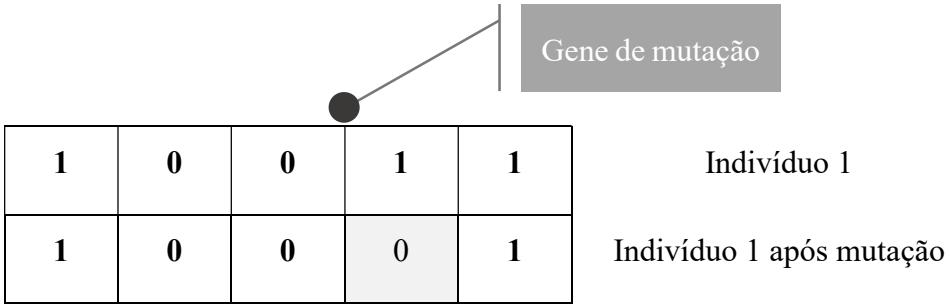


Figura 22: Exemplo mutação

4.3.7 Critério de paragem

Dentre os diferentes critérios de paragem existentes, abaixo estão indicados alguns exemplos:

- Quando atingir a quantidade de número de gerações máximas pré-definida;
- No caso de perda de diversidade da população, ou seja, na geração atual os indivíduos são todos iguais ou os indivíduos são iguais aos da geração anterior. Atinge-se o que se denomina de saturação da população;
- Quando a função objetivo não se altera após um determinado número de gerações pré-definido pelo utilizador, mesmo que haja diversidade de soluções.

4.4 Aplicação dos algoritmos genéticos ao problema de redução de perdas da rede IEEE 33 barramentos

O método de aplicação dos algoritmos genéticos ao problema de redução de perdas proposto nesta secção teve como base o artigo "*Optimal Network Reconfiguration to Reduce Power Loss Using an Initial Searching Point for Continuous Genetic Algoritthm*" [9].

Uma das contribuições do método proposto em [9] é o aumento da chance de se encontrar a solução ótima do problema ao ser inserido o *initial searching point*, ISP, no

espaço de busca inicial. O ISP é determinado por meio de um algoritmo heurístico que será discutido na secção 4.4.1.

A eficácia do estudo realizado é verificada através da comparação entre os resultados obtidos em dois cenários:

- Cenário 1: Aplicação do algoritmo genético à rede IEEE 33 barramentos sem ISP;
- Cenário 2: Aplicação do algoritmo genético à rede IEEE 33 barramentos com ISP.

4.4.1 Determinação do *Initial Searching Point*

O método para determinação o ISP, aplicado à rede IEEE 33 barramentos é demonstrado a seguir:

a) Determinação da configuração radial inicial:

   As linhas com seccionadores abertos na configuração inicial da rede em estudo são identificadas.

   A configuração radial inicial da rede IEEE 33 barramentos, como visto na secção 3.3, tem os seccionadores das linhas L33, L34, L35 e L36 abertos. Desta forma, a configuração inicial do ISP está exibida na Figura 23.

33	34	35	36	37

Figura 23: Configuração inicial do ISP

b) Fechamento do seccionador de uma linha do ISP:
   Na rede em estudo, o seccionador da linha L33 é fechado e a rede assume a configuração da Figura 24.

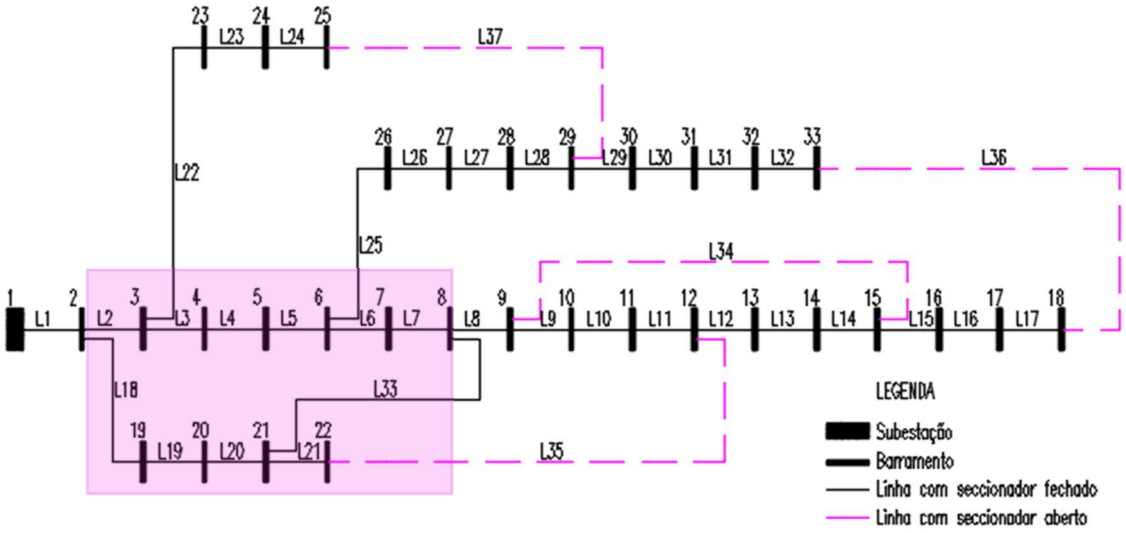


Figura 24: Rede IEEE 33 com uma malha fechada

No caso apresentado na Figura 24, a malha formada pelo fechamento do seccionador da linha L33 é composta, com exceção da linha L33, pelas linhas L2, L3, L4, L5, L6, L7, L18, L19, L20.

c) Cálculo do trânsito de potências:

Após o cálculo do trânsito de potências para a rede da Figura 24, foram identificadas as correntes das linhas pertencentes à malha fechada. Tabela 4.

Linha	**Corrente [A]**
L2	150,2
L3	97,3
L4	90,6
L5	87,5
L6	28,3
L7	20,3
L18	53,3
L19	49,0
L20	44,7

Tabela 4: Correntes das linhas pertencentes à malha fechada

d) Seleção da linha com o menor valor de corrente na malha fechada e abertura do seccionador desta:

A linha L7, de acordo com os valores obtidos em c), é a que possui o menor valor de corrente dentre as linhas da malha fechada. O seccionador da linha L7 é aberto.

e) Verificar se a nova configuração da linha é radial:
Se a linha permanecer radial ou anel aberto após o passo d), a linha L7 assume uma posição no ISP. Caso o contrário, esta linha é desconsiderada, o algoritmo volta para o passo d) e seleciona a próxima linha com menor corrente para ter o seccionador desligado.
No caso da rede em estudo, ela não permanece radial ou anel aberto após a abertura do seccionador da linha L7, com isto, o seccionador da linha L7 volta a condição anterior.
Da Tabela 4, a linha L6 foi identificada como a próxima com menor corrente. Foi verificado que a rede permanece radial ou anel aberto após a abertura do seccionador da linha L6.

f) Substituição da linha com seccionador originalmente aberto pela nova linha definida com seccionador aberto:

A linha L33 foi substituída pela linha L6 e a nova configuração do ISP é exibida na Figura 25.

6	34	35	36	37

Figura 25: Nova configuração do ISP

g) Repetição dos passos b) ao f) para determinar a nova linha a ter o seccionador aberto

h) O algoritmo termina quando todas as linhas com seccionadores abertos na configuração inicial do ISP forem substituídas por novas linhas com seccionadores abertos:

A configuração final do ISP está representada na Figura 26.

6	14	9	32	28

Figura 26: Configuração final do ISP

O fluxograma do algoritmo utilizado para definição do ISP para inserção no espaço de busca inicial do AG aplicado ao problema de minimização de perdas está presente na Figura 27.

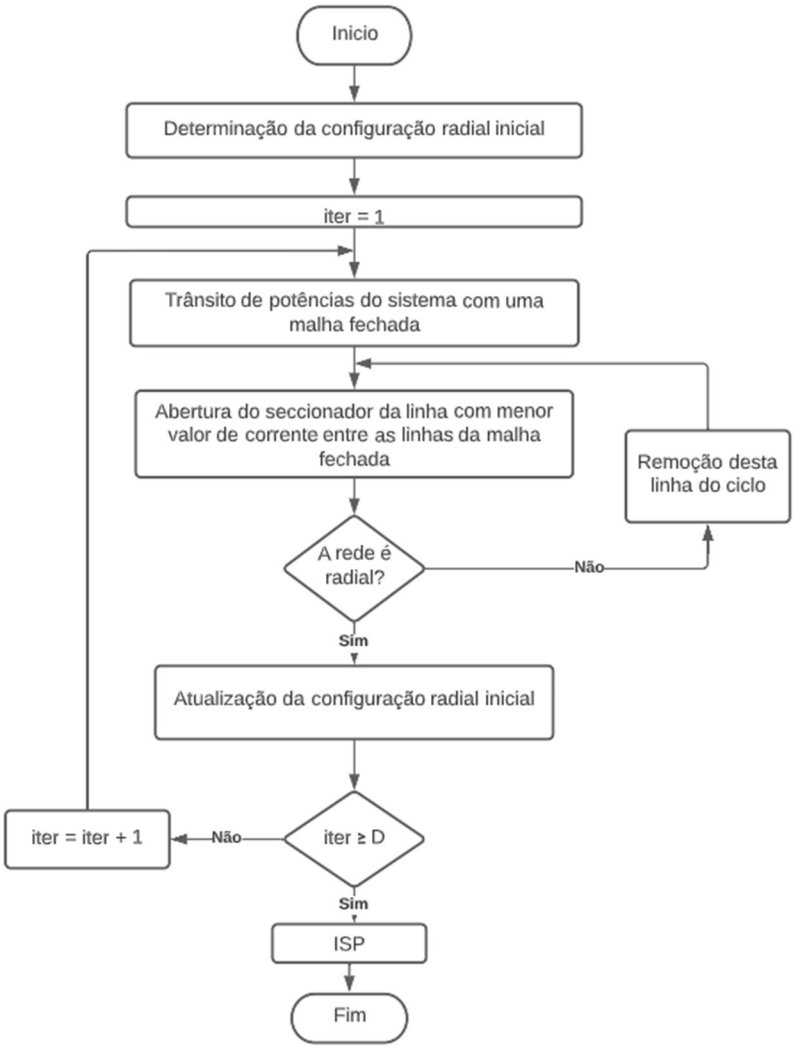


Figura 27: Fluxograma do algoritmo de definição do ISP (adaptado de [9])

O *script* do algoritmo desenvolvido para determinação do ISP encontra-se no anexo A3.

4.4.2 Descrição do Algoritmo Genético

O fluxograma do algoritmo utilizado para desenvolvimento do Algoritmo Genético está presente na Figura 28.

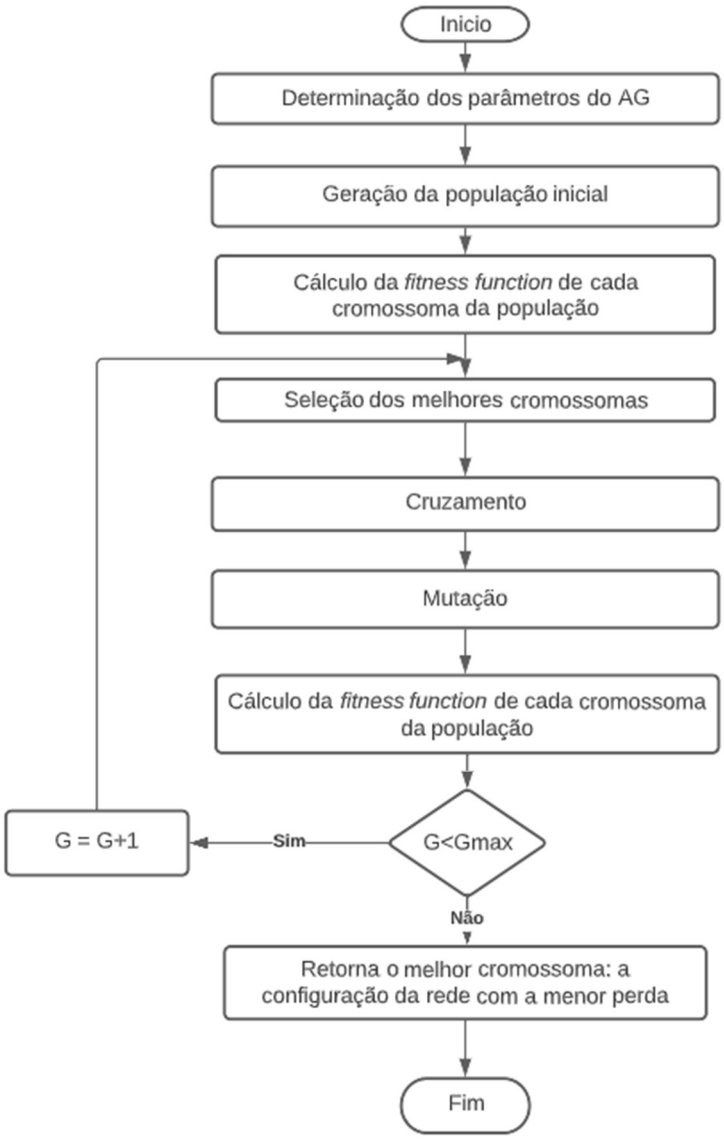


Figura 28: Fluxograma do AG aplicado ao problema de redução de perdas (adaptado de [9])

Os detalhes do Algoritmo Genético desenvolvido para a aplicação no problema de minimização de perdas estão apresentados a seguir:

a) Leitura da rede e definição dos parâmetros:

   Após a leitura e armazenamento dos dados da rede IEEE 33 barramentos, foram definidos os seguintes parâmetros do AG:

   - Tamanho da população: N

- Dimensão do problema: D (quantidade de genes no cromossoma)
- Taxa de seleção: X_{keep}
- Taxa de mutação: X_{mut}
- Quantidade máxima de gerações: G_{max}
- Linhas candidatas: L2 a L37
- Ponto de cruzamento: penúltimo cromossoma

b) Inicialização da população:

Os genes foram codificados com números inteiros, e podem assumir valores desde 2 a 37, que correspondem às linhas candidatas a terem seccionadores abertos na rede em estudo.

Cada gene é aleatoriamente gerado por meio da equação (4.1):

$$X = 1 + S_{maxd} \tag{4.1}$$

Em que Smaxd é um número aleatório entre 1 e 36.

Cada cromossoma é composto por D genes, ou seja, para formar um cromossoma, é necessário gerar 5 genes da rede IEEE 33.

A quantidade de cromossomas gerados depende do cenário em estudo:

- Cenário 1 (Aplicação do algoritmo genético à rede IEEE 33 barramentos sem ISP): São gerados N cromossomas
- Cenário 2 (Aplicação do algoritmo genético à rede IEEE 33 barramentos com ISP): São gerados N-1 cromossomas e o ISP encontrado na Figura 26 é adicionado à população.

c) Verificações da rede

Para cada cromossoma da população, é verificado se este representa uma rede radial, é calculado o trânsito de potências e a *fitness function*.

Para a problemática de redução das perdas ativas de uma rede elétrica, a equação das perdas ativas totais apresentada na equação (4.2) foi definida como *fitness function* do algoritmo genético.

$$f = \sum_{j=1}^{N_l} r_{i,j} \times k_{i,j} \times \left(\frac{P_j^2 + Q_j^2}{|V_j|^2} \right) \quad (4.2)$$

s.a.

$$\boldsymbol{V}_{min} \leq |\boldsymbol{V}_i| \leq \boldsymbol{V}_{max} \quad (4.3)$$

$$0 \leq \left|\boldsymbol{I}_{i,j}\right| \leq \left|\boldsymbol{I}_{i,j\,max}\right| \quad (4.4)$$

Em que:

N_l: Número de linhas da rede em estudo;

$k_{i,j}$: *status* do seccionador localizado no condutor entre o barramento "i" e "j"

$k_{i,j} = 0$ se o seccionador está aberto;

$k_{i,j} = 1$ se o seccionador está fechado;

$r_{i,j}$: Resistência do condutor entre o barramento "i" e o "j" [pu];

P_j: Potência ativa ligada ao barramento "j" [pu];

Q_j: Potência reativa ligada ao barramento "j" [pu];

$\left|\boldsymbol{V}_j\right|$: Módulo da tensão nodal do barramento "j" [pu];

$\boldsymbol{V}_{min}$ e $\boldsymbol{V}_{max}$: Limites admissíveis de tensão no barramento "i" [pu];

$\left|\boldsymbol{I}_{i,j}\right|$: Módulo da corrente elétrica entre o barramento "i" e "j" [pu];

$\left|\boldsymbol{I}_{i,j\,max}\right|$: Valor máximo admissível da corrente elétrica entre o barramento "i" e o "j" [pu];

d) Seleção dos melhores cromossomas

A população foi classificada por ordem crescente de *fitness function* e o melhor cromossoma, o que tem o menor *fitness function,* assume a primeira posição e assim sucessivamente.

São mantidos na população a quantidade de cromossomas correspondente à taxa de seleção, X_{keep}, definida em a).

Os cromossomas são organizados de forma crescente, ou seja, os cromossomas com menores valores de *fitness function* assumem as primeiras posições na população. A quantidade de cromossomas a serem removidos da população é calculada pela equação (4.5)

$$Remover = N \times (1 - X_{keep}) \tag{4.5}$$

Os cromossomas removidos são os de maior valor de *fitness function.*

e) Cruzamento

Os "pais" dos novos cromossomas foram aleatoriamente escolhidos entre os cromossomas mantidos na população e foi realizado cruzamento em um único ponto conforme descrito na secção 4.3.5.

O ponto de cruzamento escolhido foi o penúltimo cromossoma, conforme indicado na Figura 29.

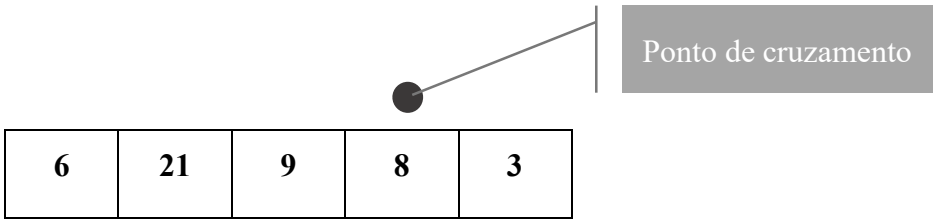


Figura 29: Ponto de cruzamento escolhido

Nesta etapa também foi verificado se os descendentes gerados formam redes radias, caso essa condição seja satisfeita, estes são adicionados à população. Caso contrário, é realizado novo sorteio de "pais" e gerado novos descendentes. Este ciclo é repetido até que uma população esteja novamente com N cromossomas.

f) Mutação

A mutação é realizada apenas na quantidade de cromossomas definido na taxa de mutação e, por causa do elitismo, o primeiro cromossoma da população, ou seja, o de menor valor de *fitness function* não é considerado para a mutação.

Cada cromossoma a passar pela mutação é escolhido aleatoriamente. Para o cromossoma selecionado, o gene a ser substituído também é escolhido aleatoriamente.

O gene selecionado é substituído por cada uma das linhas candidatas e que não fazem parte do cromossoma, é calculada a *fitness function* de cada nova possibilidade que resulte em uma configuração radial ou anel aberto e o valor é armazenado.

Ao final de todas as substituições possíveis, o gene selecionado é substituído pela linha que representa o menor valor de *fitness function* entre todas as opções verificadas.

A Figura 30 ilustra um exemplo de cromossoma selecionado para mutação.

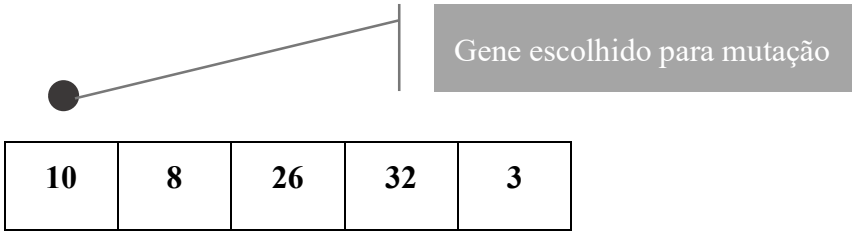


Figura 30: Exemplo de cromossoma mutação

O primeiro gene do cromossoma foi selecionado para a mutação. A *fitness function* da configuração atual é 1,68 pu. As possíveis substituições para ele estão apresentadas na Tabela 5.

Possível substituição	**Cromossoma**					***Fitness function***
9	9	8	26	32	3	1,67 pu
11	11	8	26	32	3	1,70 pu
12	12	8	26	32	3	1,66 pu
13	13	8	26	32	3	1,65 pu
14	14	8	26	32	3	1,63 pu
34	34	8	26	32	3	1,65 pu

Tabela 5: Exemplo substituições mutação

O valor do primeiro gene do exemplo foi substituído por 14, já que apresenta a configuração de menor *fitness function.*

g) Cálculo da fitness *function* de cada cromossoma da população e verificação da condição de paragem

Nesta etapa é realizado o cálculo da *fitness function* e os valores desta geração são armazenados. É verificado se o número de gerações armazenadas ainda não atingiu a quantidade máxima de gerações escolhidas, G_{max}. Em caso afirmativo, uma nova geração é iniciada a partir dos cromossomas da população da geração anterior e os passos d) ao g) são repetidos.

Caso a quantidade máximas de gerações já tenha sido realizada, é verificado qual cromossoma representa a rede com menores perdas, este é indicado como solução para o problema e o algoritmo termina.

4.5 Resultados numéricos

Foram estudados 2 cenários, conforme indicado na secção 4.4, um sem ISP e outro com ISP. Os parâmetros definidos para o Algoritmo Genético aplicado à rede IEEE 33 barramentos, nos dois cenários, são:

- Tamanho da população: N = 8, 10, 20
- Dimensão do problema: D = 5
- Taxa de seleção: $X_{keep} = 0,8$
- Taxa de mutação: $X_{mut} = 0,2$
- Quantidade máxima de gerações: $G_{max} = 50$
- Linhas candidatas: L2 a L37
- Ponto de cruzamento: penúltimo cromossoma

A configuração obtida para o ISP é [6, 14, 9, 32, 28] e apresenta perdas de 1,4041 pu.

Os critérios definidos para análise dos resultados foram:

- Geração convergência: Média da geração em que o valor mínimo da *fitness function* foi verificado após 5 simulações. Equação (4.6).

$$Geração_{convergência} = \frac{\sum Nº\ da\ geração\ com\ a\ melhor\ solução}{5} \quad (4.6)$$

- Maior valor de *fitness function*: Fit_{max}
- Menor valor de *fitness function*: Fit_{min}
- Valor médio de *fitness function*: $Fit_{méd}$

Os resultados obtidos nas simulações com o algoritmo desenvolvido, para os diferentes tamanhos da população definidos e para os 2 cenários e estão resumidos na Tabela 6.

	Cenário 1 (sem ISP)			Cenário 2 (com ISP)		
N	**8**	**10**	**20**	**8**	**10**	**20**
ISP	---	---	---	6, 14, 9, 32, 28	6, 14, 9, 32, 28	6, 14, 9, 32, 28
Configuração final	[8 33 28 31 14]	[33 28 31 8 14]	[14 33 31 8 28]	[33 31 14 28 8]	[33 31 14 28 8]	[33 14 8 31 28]
Geração conversão	21,6	23,2	26,2	23,6	26,4	24,6
Fit_max	5,7186	4,2931	8,0716	4,0167	4,2827	6,5397
Fit_min	1,2999	1,2999	1,2999	1,2999	1,2999	1,2999
Fit_media	1,4421	1,3791	1,3929	1,3237	1,3633	1,3874
% redução perdas	35	35	35	35	35	35

Tabela 6: Comparação entre os dois cenários de simulação

De acordo com a Tabela 6, a configuração de rede que apresenta menores perdas está exibida na Figura 31. A redução percentual de perdas relativamente à característica inicial da rede para esta configuração é de 35%

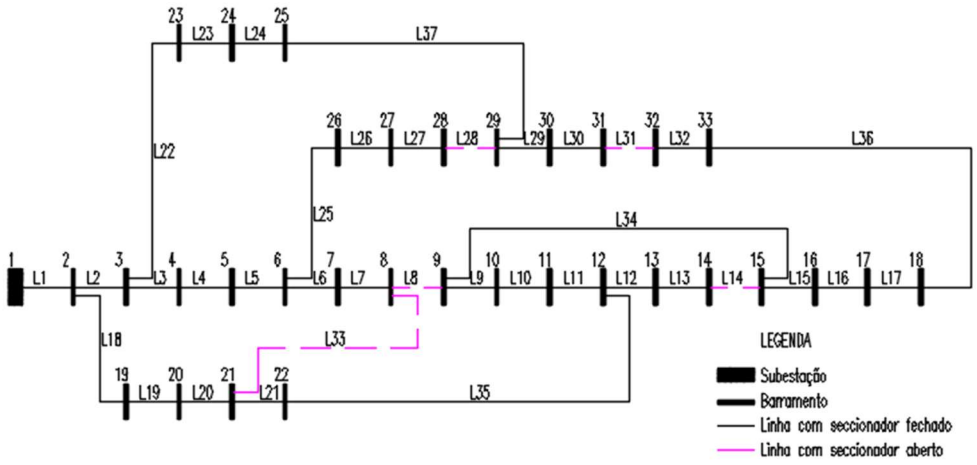


Figura 31: Configuração rede IEEE 33 barramentos com menores perdas

A Figura 32 apresenta a comparação entre os valores máximos da função de avaliação, ou seja, os valores mais desfavoráveis do algoritmo genético. Os valores fit_max das populações do cenário 1 são mais desfavoráveis do que aqueles do cenário 2.

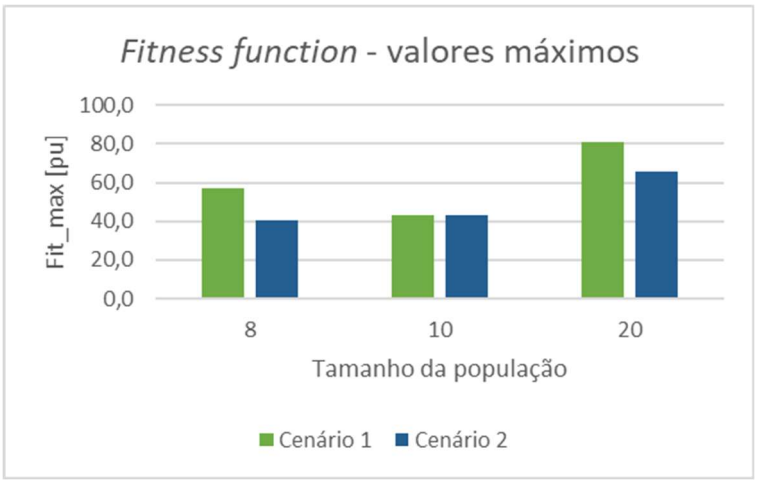


Figura 32: Comparação valores máximos fitness function

Os valores médios da *fitness function* para o cenário 2 são inferiores aos do cenário 1, para todos os tamanhos de população, o que pode ser indicativo de que o ISP promove maior direcionamento do AG para a solução do problema.

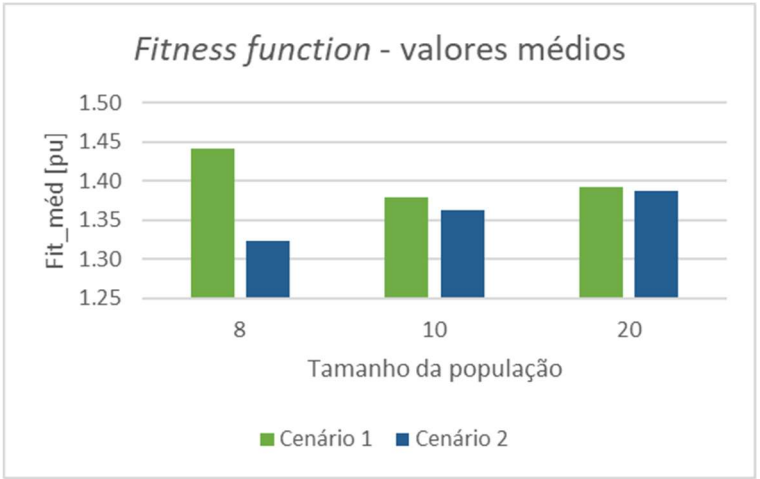


Figura 33: Comparação valores médios fitness function

Por outro lado, a Figura 34 indica que, mesmo com o direcionamento do ISP, foram necessárias mais gerações para que o cenário 2 apresentasse a solução mais satisfatória.

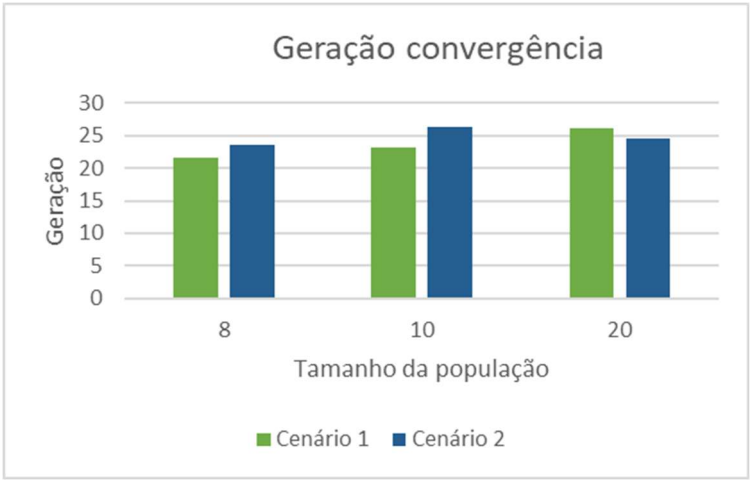


Figura 34 : Comparação geração convergência

A partir da Figura 35 e da Figura 36 é possível identificar que os valores médios da *fitness function* ao longo das gerações tendem a diminuir, o que indica que ocorrem evoluções entre gerações, e que entre a geração 15 e a geração 25 ocorre a saturação em ambos os cenários.

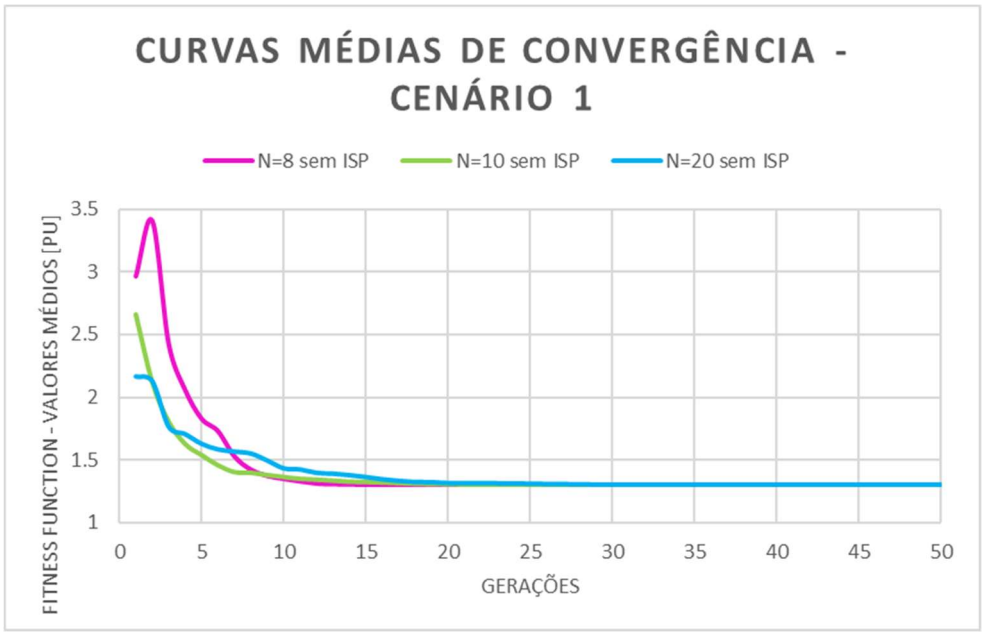


Figura 35: Curvas médias de convergência – cenário 1

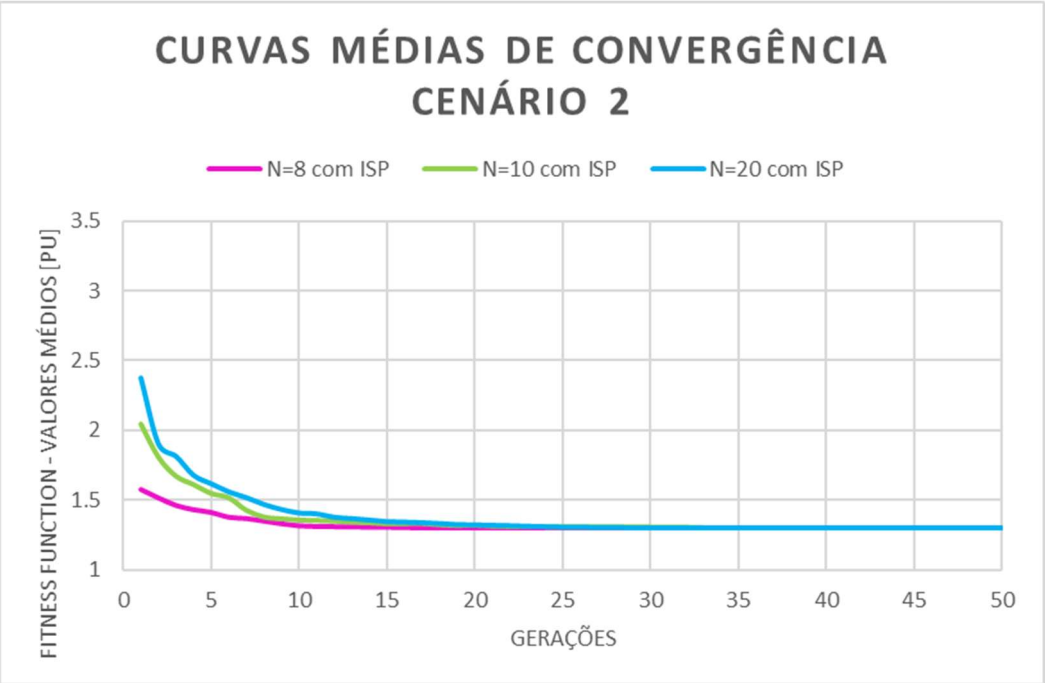


Figura 36: Curvas médias de convergência - cenário 2

Diante dos resultados apresentados, os parâmetros que proporcionam resultados mais otimizados para a aplicação do algoritmo genético à rede IEEE 33 barramentos desenvolvido neste trabalho são:

- Tamanho da população: N = 8, mas a partir da 15ª geração o algoritmo se torna insensível ao tamanho da população;
- Quantidade máxima de gerações: $G_{max} = 20$.

O script do algoritmo genético desenvolvido encontra-se no anexo A4.

4.5.1 Comparação dos resultados obtidos com os do artigo

Os resultados obtidos com o algoritmo desenvolvido estão aproximados aos do artigo de referência, conforme está apresentado na Tabela 7.

Assim como no algoritmo apresentado no capitulo 3, a diferença entre os resultados do cálculo das perdas na configuração inicial da rede IEEE 33 barramentos apresentados no algoritmo desenvolvido e no artigo, pode ser devido ao fato de que,

embora a rede de teste seja *standard*, os parâmetros desta rede não estão disponíveis em uma única fonte, o que pode significar que os valores utilizados no artigo são ligeiramente diferentes dos parâmetros de entrada utilizados no desenvolvimento deste algoritmo.

	Configuração inicial [pu]	**Configuração final [pu]**
Algoritmo	2,0029	1,2999
Artigo	2,0268	1,4873

Tabela 7: Comparação com os resultados do artigo [9]

CAPÍTULO 5

5. FERRRAMENTAS PARA DESENVOLVIMENTO DOS ALGORITMOS

5.1 Introdução

A metodologia de avaliação da eficácia dos métodos estudados neste trabalho consiste na aplicação dos algoritmos desenvolvidos na rede IEEE 33 barramentos, que é a rede utilizada nos artigos de referência, mas com a alteração dos softwares de desenvolvimento.

Nos artigos de referência [8] e [9], os algoritmos foram elaborados com o auxílio do *software* Matlab que é uma plataforma de programação numérica, desenvolvido pela empresa MathWorks, muito utilizada por engenheiros para análise de dados, desenvolvimento de algoritmos e criação de modelos.

Embora muitos pesquisadores optem por desenvolver seus algoritmos em Matlab, esta plataforma possui algumas limitações:

- O utilizador da plataforma não tem liberdade de adaptar as caixas de ferramentas disponíveis para problemas específicos;
- Custo elevado da ferramenta.

5.2 Tecnologias utilizadas

A fim de assegurar maior flexibilidade e controle dos algoritmos elaborados neste trabalho, optou-se por utilizar a linguagem de programação Python 3.9, associada ao ambiente de desenvolvimento Spyder instalado em um computador portátil ASUS ZenBook, Intel Core i5-1135G7 @2.40GHz. Sistema operativo Windows 11 Home.

Nas secções seguintes deste capítulo serão apresentadas as características principais das ferramentas de desenvolvimento adotadas neste trabalho: a interface de desenvolvimento Spyder, a linguagem de programação Python, a biblioteca Pandas e a ferramenta Pandapower.

Figura 37: Tecnologias utilizadas para a elaboração dos algoritmos de redução de perdas

A Figura 37 apresenta as tecnologias utilizadas durante o desenvolvimento dos algoritmos de redução de perdas. Os *scripts* são escritos na plataforma Spyder, na linguagem de programação Python, com funções importadas do módulo Pandapower e estrutura de dados do Pandas.

5.3 Spyder Software

O Spyder, *Scientific PYthon Development Environment*, é uma plataforma de ambiente científico *open source,* ou seja, seu código fonte pode ser editado. Tem uma licença MIT, *Massachussetts Institute of Technology*, que é uma licença de software livre [17]. O anexo A5 contém mais informações sobre o Spyder.

5.4 Python

O Python é uma linguagem de programação *open source* e bastante versátil, pode, por exemplo, ser aplicada em ciência de dados, inteligência artificial e desenvolvimento da Web. Possui uma grande quantidade de bibliotecas disponíveis, dentre elas estão a biblioteca Pandas e a NumPy, ambas utilizadas nos algoritmos apresentados neste trabalho.

Algumas das características da linguagem Python são [21]:

- Possui versões para diversos sistemas operacionais como: Windows, Linux e macOS;
- Contém um interpretador que traduz o código fonte em linguagem de máquina e executa o programa, por isso não gera arquivos executáveis como ocorre na linguagem C.

O anexo A6 contém mais informações sobre o Python.

5.4.1 Pandas e Pandapower

Pandas é uma ferramenta de análise e manipulação de dados, rápida, flexível e de código aberto, construída sobre a linguagem de programação Python. [22]

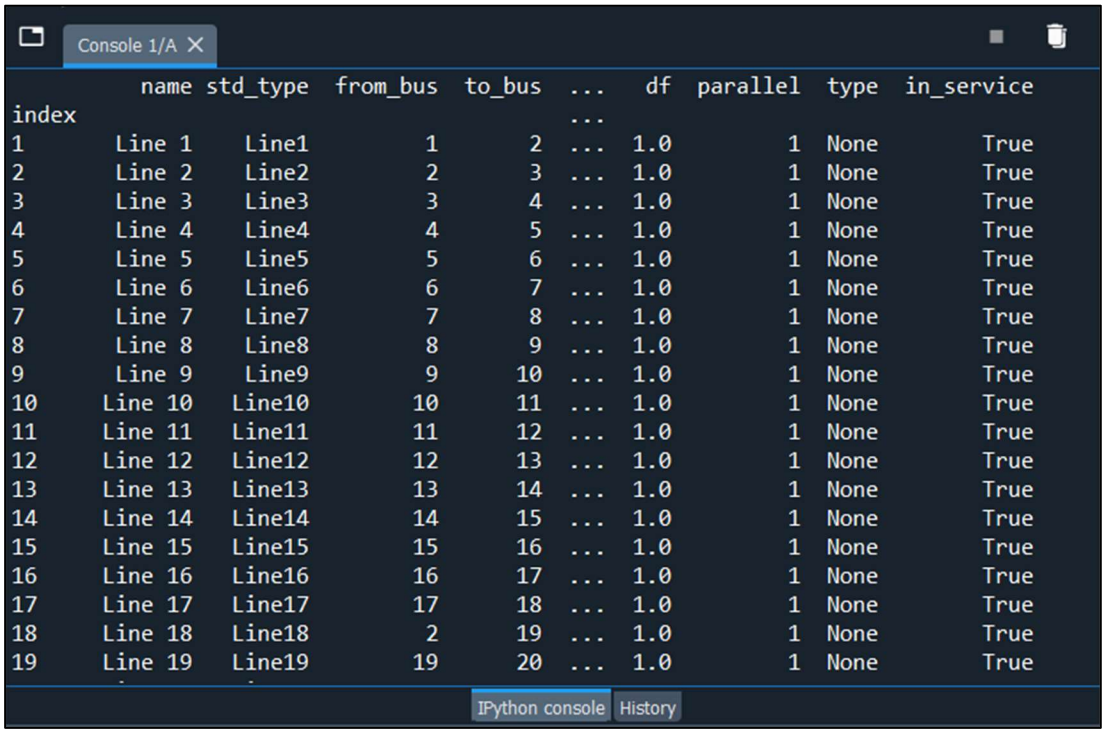
A base do seu desenvolvimento são duas bibliotecas importantes do Python: a matplotlib para a visualização dos dados e NumPy para operações matemáticas. [23]

Duas grandes contribuições do Pandas é o Panda Series, que é um tipo de objeto de armazenamento de dados com a estrutura em forma de lista e o DataFrames, que também é um objeto de armazenamento de dados, mas tem uma estrutura tabular. O anexo A7 apresenta um exemplo de uma variável do tipo lista.

O Pandapower teve como base do seu desenvolvimento a biblioteca de análise de dados do Pandas e as ferramentas de análises de redes de energia PyPower para criar um programa de trânsito de energia dedicado para otimização de redes de energia [24].

Possui uma extensa biblioteca de modelos de sistemas de energia, por exemplo: modelos de cabos, transformadores, condensadores. Além disso, possui ferramentas para o cálculo do trânsito de energia, verificação de sobrecarga nos barramentos, dentre outros.

A estrutura de dados do Pandapower é tabular, ou seja, cada uma das variáveis existentes é representada em forma de tabela que contém todos os parâmetros especificados assim como os resultados das análises efetuadas. Esta forma de representação permite armazenar variáveis de qualquer tipo. As tabelas podem ser facilmente expandidas e personalizadas sem influenciar na funcionalidade do Pandapower. A Figura 38 apresenta um exemplo de dados que foram armazenados em estrutura tabular.



index	name	std_type	from_bus	to_bus	...	df	parallel	type	in_service
					...				
1	Line 1	Line1	1	2	...	1.0	1	None	True
2	Line 2	Line2	2	3	...	1.0	1	None	True
3	Line 3	Line3	3	4	...	1.0	1	None	True
4	Line 4	Line4	4	5	...	1.0	1	None	True
5	Line 5	Line5	5	6	...	1.0	1	None	True
6	Line 6	Line6	6	7	...	1.0	1	None	True
7	Line 7	Line7	7	8	...	1.0	1	None	True
8	Line 8	Line8	8	9	...	1.0	1	None	True
9	Line 9	Line9	9	10	...	1.0	1	None	True
10	Line 10	Line10	10	11	...	1.0	1	None	True
11	Line 11	Line11	11	12	...	1.0	1	None	True
12	Line 12	Line12	12	13	...	1.0	1	None	True
13	Line 13	Line13	13	14	...	1.0	1	None	True
14	Line 14	Line14	14	15	...	1.0	1	None	True
15	Line 15	Line15	15	16	...	1.0	1	None	True
16	Line 16	Line16	16	17	...	1.0	1	None	True
17	Line 17	Line17	17	18	...	1.0	1	None	True
18	Line 18	Line18	2	19	...	1.0	1	None	True
19	Line 19	Line19	19	20	...	1.0	1	None	True

Figura 38: Exemplo de dados de linhas de uma rede modelada em Pandapower

Todos os métodos do Pandas podem ser usados para ler, gravar e analisar com eficiência a rede e os dados dos resultados.

Dentre as análises de redes que podem ser feitas com o Pandapower, destacam-se as seguintes:

- Fluxo de potências: O cálculo do fluxo de potências é baseado no método de Newton-Raphson, mas é possível alterar para outros métodos existentes, como o método Gauss-Seidel;
- Curto Circuito: Realiza o cálculo das correntes de falta em sistemas trifásicos e de curto-circuito monofásicos de acordo com a IEC 60909.

5.4.1.1 Criação de rede com Pandapower

O Pandapower fornece modelos para 13 elementos de uma rede, dentre eles estão os barramentos, linhas, seccionadores, cargas, geradores e transformadores. No site pandapower.com[1] [25] está a informação completa de todos os elementos, assim como a descrição detalhada de cada um deles e exemplos de aplicações.

No desenvolvimento deste trabalho, a rede IEEE 33 barramentos foi modelada em python através do módulo Pandapower. Foram criados e configurados os dados dos barramentos, das cargas, dos cabos, das linhas e dos seccionadores. Os atributos utilizados foram os seguintes:

- Criação de nova rede: pandapower.create_empy_network
- Barramentos: pandapower.create_bus
- Cargas: pandapower.create_load
- Linhas: pandapower.create_line
- Seccionadores: pandapower.create_switch

Foi realizada a exportação da rede modelada para o Excel através do atributo **pandapower.to_excel**. O arquivo exportado contém várias tabelas conforme pode ser visualizado na Figura 39. Cada tabela exportada corresponde a um elemento da rede criada.

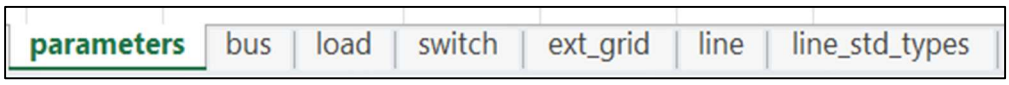


Figura 39: Folhas arquivo Excel exportado

[1] http://www.pandapower.org/about/

O anexo A1 apresenta a estrutura e parâmetros de todos os elementos exportados para o Excel.

A exportação da rede para o excel é vantajosa em vários aspetos, dentre eles:

- Simplifica o processo de modificação de valores da rede inicial, em caso de realização de teste de parâmetros;
- Facilita a visualização dos *inputs* e dos resultados de simulação.

Para a execução dos scripts dos algoritmos desenvolvidos neste trabalho, a rede IEEE 33 barramentos que foi modelada, é importada do Excel para o Python através do atributo **pandapower.from_excel**.

CAPÍTULO 6

6. CONCLUSÕES

Na presente dissertação foram apresentados, desenvolvidos e implementados dois algoritmos para resolver o problema de redução de perdas técnicas em redes de energia elétrica através da reconfiguração topológica. Os dois algoritmos foram aplicados à rede IEEE 33, que possui perdas de 2,0029 pu em sua configuração inicial.

O primeiro algoritmo desenvolvido utiliza o método heurístico que está descrito no capítulo 3. De maneira sucinta, o algoritmo identifica a linha com maior diferença de tensão entre os barramentos dentre as linhas que, inicialmente, estava com seccionador aberto, e a partir dela, encontra o barramento com menor tensão, desliga o seccionador da linha adjacente e liga o seccionador da linha identificada. Depois realiza uma busca entre as linhas adjacentes até que obtenha a configuração de menor perda dentro dos critérios estabelecidos.

Para o primeiro algoritmo, as perdas calculadas após a reconfiguração são 1,38074 pu, o que significa redução de 31% de perdas em relação à configuração inicial da rede.

O método heurístico apresenta resultado satisfatório já que cumpriu o objetivo proposto: redução de perdas. Entretanto, as limitações deste método, por exemplo, espaço de busca restrito, já que as buscas de menores perdas são realizadas apenas entre as linhas adjacentes, foram fatores que motivaram a procura de outro método de estudo.

Dada a natureza combinatória do problema, foram identificados algoritmos evolutivos de redução de perdas por reconfiguração topológica, e com base no artigo [9], foi desenvolvido o segundo algoritmo presente neste trabalho, descrito no capítulo 4.

Trata-se de um algoritmo genético codificado com números inteiros, em que cada gene representa uma linha cujo seccionador está desligado e cada cromossoma representa uma possível nova configuração para a rede em estudo.

Uma característica do algoritmo genético desenvolvido é a possibilidade de inserir um, *Initial Searching Point* – (ISP), na população inicial. A proposta do ISP é obter uma solução de boa qualidade que, associada aos demais cromossomas da população inicial, faz com que seu material genético se espalhe e contribua para que o AG evolua para a solução ótima do problema. O ISP é determinado por meio de um algoritmo heurístico.

Dois cenários de estudo foram criados para avaliação do AG desenvolvido. No cenário 1 foram realizadas simulações sem o ISP e no cenário 2 foi inserido o ISP na população inicial em todas as simulações. Para ambos cenários, foi realizado um estudo paramétrico no sentido de escolher os parâmetros que maximizam o desempenho do algoritmo e garantam uma elevada qualidade das soluções obtidas.

Foi verificado que para aplicação à rede IEEE 33 barramentos, os parâmetros que possibilitam melhores resultados são:

- Tamanho da população: N = 8, embora a partir da 15ª geração o algoritmo aplicado à rede IEEE 33 barramentos se torne insensível ao tamanho da população;
- Quantidade máxima de gerações: $G_{max} = 20$.

Para o algoritmo genético desenvolvido, as perdas calculadas após a melhor reconfiguração identificada são 1,2999 pu, o que significa redução de 35% de perdas.

Comparando-se os dois algoritmos desenvolvidos neste trabalho, método heurístico e método evolutivo, conclui-se que, para o problema de redução de perdas em redes por meio da técnica de reconfiguração iterativa, aplicada à rede IEEE 33 barramentos, o método evolutivo com o algoritmo genético apresenta melhores resultados.

6.1 Trabalhos futuros

Para trabalhos futuros, são propostas as seguintes atividades:

- Desenvolvimento de metodologia para teste de parâmetros:

  Foram realizadas diversas simulações manuais para o teste dos parâmetros, com a automatização deste processo, ocorrerá um aumento da robustez do código.

- Avaliar os algoritmos para outros cenários operacionais da rede;
- Desenvolver o processo de mutação inteligente e adaptado ao problema:

Isto corresponde a incluir um processo de mutação que garanta a melhoria da solução escolhida promovendo movimentos para outras zonas do espaço de soluções.

- Desenvolver e testar outras estratégias que garantam que as soluções obtidas sejam conexas, i.e., que não promovam ilhas;
- Aplicação dos algoritmos desenvolvidos em redes reais com dimensões mais elevadas.

REFERÊNCIAS

REFERÊNCIAS

[1] Regulamento de acesso às redes e às interligações do setor elétrico (RARI) aprovado em anexo ao Regulamento nº 560/2014 publicado na 2ª Série, Nº 246, do Diário da República, de 22 de dezembro de 2014.

[2]E-Redes. "Relatório e contas 2021". url: https://www.e-redes.pt/sites/eredes/files/2022-06/RC_E-REDES%202021.pdf (acedido em 29/08/2022)

[3] B.M.M. Coelho, "Classificação de tipologia de rede da EDP distribuição", Dissertação, Mestrado, FEUP, Porto, julho 2012.

[4] M.V. dos Santos, G.A. Brigatto, L.P. Garcés, "Methodology of solution for the distribution network reconfiguration probem based on improved harmony search algorithm", Research Article, IET Generation, Transmission & Distribution, 2020.

[5] J.P.Sucena Paiva, "Redes de energia eléctrica – uma análise sistémica 4ª Ed.", IST PRESS, 2015.

[6] L.M. Oliveira de Queiroz, "Algoritmos genéticos híbridos para redução de perdas técnicas em redes primárias de distribuição considerando variações de demandas", Dissertação, Mestrado, Universidade Estadual de Campinas, São Paulo – Brasil, 2005.

[7] R. Chidanandappa, Dr.T. Ananthapadmanabha, H.C. Ranjith, "Genetic algorithm based network reconfiguration in distribution systems with multiple DGs for time varying loads", Procedia Technology, 2015.

[8] R.S. Rao, S.V.L. Narasimham "A new heuristic approach for optimal network reconfiguration in distribution systems", International Journal of Applied Science, Engineering and Technology, 2009.

[9] T.T. Nguyen, T.T. Nguyen, N.A Nguyen, "Optimal network reconfiguration to reduce power loss using an initial searching point for continuous genetic algorithm", Willey, 2020.

[10] C.T.Su, C.F. Chang, J.P. Chiou, “Distribution network reconfiguration for loss reduction by ant colony search algorithm”, Eletric Power Systems Research, Volume 75, 2005.

[11] W.M. Dahalan, H. Mokhlis, “Network reconfiguration for loss reduction with distributed generations using PSO”, IEEE International Conference on Power and Energy, Malaysia, 2012.

[12] R.J.Correia da Silva, “Um sistema de gerenciamento e controle operacional de fluxo de potência utilizando técnicas de algoritmos genéticos”, Dissertação, Mestrado, Universidade Federal de Alagoas – UFAL, Maceió – Brasil, 2008.

[13] A.S.A.A. Adail, “Network reconfiguration for loss reduction in electrical distribution system using genetic algorithm”, Thesis, Master, Faculty of Engineering Al-Azhar University, 2012.

[14] M.Ebadi, “Genetic algorithm re-arrangement and distribution network optimization to reduce possible losses”, Faculty of Electronics, University of Tehran, Tehran, Iran.

[15] M.A.S.Jardim, “Reconfiguração de redes de distribuição de energia elétrica usando algoritmo genético multiobjectivo”, Dissertação, Mestrado, Universidade Federal de Minas Gerais, 2011.

[16] L.M.Oliveira de Queiroz, “Algoritmos genéticos híbridos para redução de perdas técnicas em redes primárias de distribuição considerando variações de demandas”, Dissertação, Mestrado, Universidade Estadual de Campinas, 2005.

[17] Techemportugues. “Spyder – Uma alternativa ao Matlab?”. url: https://www.techemportugues.com/2015/11/22/spyder-uma-alternativa-ao-matlab/ (acedido em 29/09/2022).

[18] Spyder. url: https://www.spyder-ide.org/ (acedido em 29/09/2022)

[19] Anaconda distribution. url: https://www.anaconda.com/products/distribution#windows (acedido em 29/09/2022).

[20] Voitto. “O que é Python e para que serve?”. url: https://www.voitto.com.br/blog/artigo/python (acedido em 29/09/2022).

[21] Blog.betrybe.” Python: o que é, como usar, guia pra aprender a linguagem”. url: https://blog.betrybe.com/python/ (acedido em 29/09/2022).

[22] Pandas. url: https://pandas.pydata.org/ (acedido em 29/09/2022).

[23] Voitto. “O que é a biblioteca Pandas. url: https://www.voitto.com.br/blog/artigo/biblioteca-pandas (acedido em 29/09/2022).

[24] Pandapower. “About pandapower”. url: http://www.pandapower.org/about/ (acedido em 29/09/2022).

[25] Pandapower. “Datastructure and elements”. url: http://www.pandapower.org/about/ (acedido em 05/10/2022).

[26] P.M.L. Torres, “Metodologia para redução de perdas e aumento de fiabilidade em redes de distribuição”, Dissertação, Mestrado, ISEL, Lisboa, setembro 2010.

[27] The World Bank. “Electric power distribution losses (% of output)”. url: https://data.worldbank.org/indicator/EG.ELC.LOSS.ZS?view=map (acedido em 21/05/2023).

[28] O algotitmo genético. http://computacaointeligente.com.br/algoritmos/o-algoritmo-genetico/ (acedido em 21/05/2023).

ANEXOS

ANEXO A1 – TABELAS DE PARÂMETROS DE ENTRADA DA REDE IEEE 33 BARRAMENTOS

- Tabela de *inputs* - barramentos

BUS				
index	**name**	**vn_kv**	**type**	**in_service**
1	Bus 1	12,66	b	VERDADEIRO
2	Bus 2	12,66	b	VERDADEIRO
3	Bus 3	12,66	b	VERDADEIRO
4	Bus 4	12,66	b	VERDADEIRO
5	Bus 5	12,66	b	VERDADEIRO
6	Bus 6	12,66	b	VERDADEIRO
7	Bus 7	12,66	b	VERDADEIRO
8	Bus 8	12,66	b	VERDADEIRO
9	Bus 9	12,66	b	VERDADEIRO
10	Bus 10	12,66	b	VERDADEIRO
11	Bus 11	12,66	b	VERDADEIRO
12	Bus 12	12,66	b	VERDADEIRO
13	Bus 13	12,66	b	VERDADEIRO
14	Bus 14	12,66	b	VERDADEIRO
15	Bus 15	12,66	b	VERDADEIRO
16	Bus 16	12,66	b	VERDADEIRO
17	Bus 17	12,66	b	VERDADEIRO
18	Bus 18	12,66	b	VERDADEIRO
19	Bus 19	12,66	b	VERDADEIRO
20	Bus 20	12,66	b	VERDADEIRO
21	Bus 21	12,66	b	VERDADEIRO
22	Bus 22	12,66	b	VERDADEIRO
23	Bus 23	12,66	b	VERDADEIRO
24	Bus 24	12,66	b	VERDADEIRO
25	Bus 25	12,66	b	VERDADEIRO
26	Bus 26	12,66	b	VERDADEIRO
27	Bus 27	12,66	b	VERDADEIRO
28	Bus 28	12,66	b	VERDADEIRO
29	Bus 29	12,66	b	VERDADEIRO
30	Bus 30	12,66	b	VERDADEIRO
31	Bus 31	12,66	b	VERDADEIRO
32	Bus 32	12,66	b	VERDADEIRO
33	Bus 33	12,66	b	VERDADEIRO

- Tabela de *inputs* – cargas

LOAD								
index	**bus**	**p_mw**	**q_mvar**	**const_z_percent**	**const_i_percent**	**scaling**	**in_service**	**type**
1	2	0,100	0,060	0	0	1	VERDADEIRO	wye
2	3	0,090	0,040	0	0	1	VERDADEIRO	wye
3	4	0,120	0,080	0	0	1	VERDADEIRO	wye
4	5	0,060	0,030	0	0	1	VERDADEIRO	wye
5	6	0,060	0,020	0	0	1	VERDADEIRO	wye
6	7	0,200	0,100	0	0	1	VERDADEIRO	wye
7	8	0,200	0,100	0	0	1	VERDADEIRO	wye
8	9	0,060	0,020	0	0	1	VERDADEIRO	wye
9	10	0,060	0,020	0	0	1	VERDADEIRO	wye
10	11	0,045	0,030	0	0	1	VERDADEIRO	wye
11	12	0,060	0,035	0	0	1	VERDADEIRO	wye
12	13	0,060	0,035	0	0	1	VERDADEIRO	wye
13	14	0,120	0,080	0	0	1	VERDADEIRO	wye
14	15	0,060	0,010	0	0	1	VERDADEIRO	wye
15	16	0,060	0,020	0	0	1	VERDADEIRO	wye
16	17	0,060	0,020	0	0	1	VERDADEIRO	wye
17	18	0,090	0,040	0	0	1	VERDADEIRO	wye
18	19	0,090	0,040	0	0	1	VERDADEIRO	wye
19	20	0,090	0,040	0	0	1	VERDADEIRO	wye
20	21	0,090	0,040	0	0	1	VERDADEIRO	wye
21	22	0,090	0,040	0	0	1	VERDADEIRO	wye
22	23	0,090	0,050	0	0	1	VERDADEIRO	wye
23	24	0,420	0,200	0	0	1	VERDADEIRO	wye
24	25	0,420	0,200	0	0	1	VERDADEIRO	wye
25	26	0,060	0,025	0	0	1	VERDADEIRO	wye
26	27	0,060	0,025	0	0	1	VERDADEIRO	wye
27	28	0,060	0,020	0	0	1	VERDADEIRO	wye
28	29	0,120	0,070	0	0	1	VERDADEIRO	wye
29	30	0,200	0,600	0	0	1	VERDADEIRO	wye
30	31	0,150	0,070	0	0	1	VERDADEIRO	wye
31	32	0,210	0,100	0	0	1	VERDADEIRO	wye
32	33	0,060	0,040	0	0	1	VERDADEIRO	wye

- Tabela de *inputs* – linhas

LINE							
index	**name**	**from_bus**	**to_bus**	**length_km**	**r_ohm_per_km**	**x_ohm_per_km**	**in_service**
1	Line 1	1	2	1	0,092	0,047	VERDADEIRO
2	Line 2	2	3	1	0,493	0,251	VERDADEIRO
3	Line 3	3	4	1	0,366	0,186	VERDADEIRO
4	Line 4	4	5	1	0,381	0,194	VERDADEIRO
5	Line 5	5	6	1	0,819	0,071	VERDADEIRO
6	Line 6	6	7	1	0,187	0,619	VERDADEIRO
7	Line 7	7	8	1	0,711	0,235	VERDADEIRO
8	Line 8	8	9	1	1,030	0,740	VERDADEIRO
9	Line 9	9	10	1	1,044	0,740	VERDADEIRO
10	Line 10	10	11	1	0,197	0,065	VERDADEIRO
11	Line 11	11	12	1	0,374	0,124	VERDADEIRO
12	Line 12	12	13	1	1,468	1,155	VERDADEIRO
13	Line 13	13	14	1	0,542	0,713	VERDADEIRO
14	Line 14	14	15	1	0,591	0,526	VERDADEIRO
15	Line 15	15	16	1	0,746	0,545	VERDADEIRO
16	Line 16	16	17	1	1,289	1,721	VERDADEIRO
17	Line 17	17	18	1	0,732	0,574	VERDADEIRO
18	Line 18	2	19	1	0,164	0,157	VERDADEIRO
19	Line 19	19	20	1	1,504	1,355	VERDADEIRO
20	Line 20	20	21	1	0,410	0,478	VERDADEIRO
21	Line 21	21	22	1	0,709	0,937	VERDADEIRO
22	Line 22	3	23	1	0,451	0,308	VERDADEIRO
23	Line 23	23	24	1	0,898	0,709	VERDADEIRO
24	Line 24	24	25	1	0,896	0,701	VERDADEIRO
25	Line 25	6	26	1	0,203	0,103	VERDADEIRO
26	Line 26	26	27	1	0,284	0,145	VERDADEIRO
27	Line 27	27	28	1	1,059	0,934	VERDADEIRO
28	Line 28	28	29	1	0,804	0,701	VERDADEIRO
29	Line 29	29	30	1	0,508	0,259	VERDADEIRO
30	Line 30	30	31	1	0,974	0,963	VERDADEIRO
31	Line 31	31	32	1	0,311	0,362	VERDADEIRO
32	Line 32	32	33	1	0,341	0,530	VERDADEIRO
33	Line 33	8	21	1	2,000	2,000	VERDADEIRO
34	Line 34	9	15	1	2,000	2,000	VERDADEIRO
35	Line 35	12	22	1	2,000	2,000	VERDADEIRO
36	Line 36	18	33	1	0,500	0,500	VERDADEIRO
37	Line 37	25	29	1	0,500	0,500	VERDADEIRO

ANEXO A2 – *SCRIPT* "REDUCAO_PERDAS_METODO_HEURISTICO.PY"

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr  6 16:13:18 2022

@author: eveni
"""

import pandapower as pp
import pandapower.networks
import numpy as np

#LER OS DADOS RELATIVOS A CONFIGURACAO E EQUIPAMENTOS DA REDE E DIAGRAMA DE
CARGAS
net = pp.from_excel("IEEE_33_Bus.xlsx")

#DEFINE ITER = h = k = 1
iter = 0
k = 1 #contador de ciclos
l = 37 #quantidade de linhas
bus = 33 #quantidade de barramentos

#DETERMINA PELO GS A TENSAO NOS BARRAMENTOS E POTENCIA DE PERDAS TOTAL DA REDE
pp.runpp(net, algorithm='gs')


bj = 0
bi = 0
perda_linha = []

print("REDE INICIAL")

resist_linha=net.line["r_ohm_per_km"]*net.line["length_km"]
potencia_ativa=net.res_line["p_to_mw"] #esse valor equivale a pu
potencia_reativa=net.res_line["q_to_mvar"] #esse valor equivale a pu
tensao=net.res_line["vm_to_pu"]*net.bus["vn_kv"]


for x in net.line["to_bus"].index:
    bj=net.line["to_bus"][x]


perda_linha.append(resist_linha[x]*((potencia_ativa[x]**2+potencia_reativa[x]**2
)/tensao[x]**2)*10)

#perda total = somatorio das perdas nas linhas
perda_total=[]
perda_t= 0
for num in perda_linha:
    if num > 0:
        perda_t=perda_t + num
perda_total.append(perda_t)
print("a perda total da rede é", perda_total[iter], "pu ")

iter = iter +1
```

```
barr_estudados = [0]

while k <= 5: # repetir enquanto não forem realizados 5 ciclos completos
    pp.runpp(net)
    print("ITERAÇÃO", iter)
    #VERIFICAR QUAL DAS LINHAS COM  SEC ABERTO TEM MAIOR DIFERENCA DE TENSAO
ENTRE BARRAMENTOS
    #encontrar linhas com seccionador aberto
    Sec_aberto = []
    for i in net.switch["closed"].index:
        if net.switch["closed"].loc[i] == False:
            Sec_aberto.append(i)
    print("Linhas com seccionadores abertos", Sec_aberto)

    dif_barr = 0
    bi = 0
    bj = 0
    b1 = 0
    b2 = 0
    x = 0 #linha com maior diferença de tensao entre barramentos
    maior_v_dif = None

    for i in Sec_aberto:
        bi=net.line["from_bus"][i]
        bj=net.line["to_bus"][i]
        dif_barr = abs(net.res_bus["vm_pu"][bi] - net.res_bus["vm_pu"][bj])
        print("linha", i, "a diferenca de tensao é", dif_barr)
        if(maior_v_dif is None or dif_barr>maior_v_dif):
            maior_v_dif=dif_barr
            b1= bi
            b2 = bj
            x = i
    print("Linha com maior diferenca de tensão entre barramentos", x)

    #DA LINHA SELECIONADA IDENTIFICAR QUAL O BARRAMENTO COM MENOR TENSAO E
DESLIGA A LINHJA ADJACENTE E LIGA A LINHA SELECIONADA

    net.switch["closed"][x] = True
    #print ("O seccionador da linha", x, "foi ligado")

    #verificar qual dos barramentos da linha i tem menor tensão
    barr_menor_tensao=0
    for barr_rep in barr_estudados:
        if b1 == barr_rep:
            barr_menor_tensao = b2
        if b2 == barr_rep:
            barr_menor_tensao = b1
    if barr_menor_tensao ==0:
        if net.res_bus["vm_pu"][b1] < net.res_bus["vm_pu"][b2]:
            barr_menor_tensao=b1
            print ("O barramento a ser estudado é", barr_menor_tensao)
        else:
            barr_menor_tensao=b2
```

```
            print ("O barramento a ser estudado é", barr_menor_tensao)

    opcao_desligar=[]
    indice_linha = 0


    for i in net.line["from_bus"].index:
        indice_linha +=1
        a = net.line["from_bus"][i]
        if a == barr_menor_tensao and indice_linha!= x:
            opcao_desligar.append(i)
    indice_linha = 0
    a = 0
    for i in net.line["to_bus"].index:
        indice_linha +=1
        a= net.line["to_bus"][i]
        if a == barr_menor_tensao and indice_linha!= x:
            opcao_desligar.append(i)
    print("As opções de linhas para desligar são", opcao_desligar)

    #Verifica qual linha é possivel desligar
    op_radial=[]
    verif_bus = 0
    linha_desligada = 0
    for linha_desligada in opcao_desligar:
        if net.switch["closed"][linha_desligada] == False:
            print("a linha", linha_desligada, "já estava desligada")
        else:
            net.switch["closed"][linha_desligada] = False
            pp.runpp(net)

            #TESTE RADIAL
            zeros = 1
            vetor= []
            vetor_bus=[]

            for cont in range (1,bus+1):
                for i in range (1,l+1):
                    if net.line["from_bus"][i]==cont and
net.switch["closed"][i]== True:
                        vetor.append(net.line["to_bus"][i])
                    if net.line["to_bus"][i]==cont and net.switch["closed"][i]==
True:
                        vetor.append(net.line["from_bus"][i])
                vetor_bus.append(vetor)
                vetor=[]
                matriz = []
            for linha in range (0,bus):
                for coluna in range (0,bus):
                    if linha == coluna:
                        matriz.append(len(vetor_bus[linha]))
                    else:
                        matriz.append(0)
```

```
            X=np.array(matriz)
            shape = (bus,bus)
            X=X.reshape(shape)


            for linha in range (0,bus):
                for i in vetor_bus[linha]:
                    X[linha][i-1]=-1

            try:
                XX = np.linalg.inv(X)
            except:
                zeros = 10000
            if zeros !=10000:
                zeros=XX.size - np.count_nonzero(XX)
            if zeros != 0:
                print("Se desligar a linha", linha_desligada, ", a rede NÃO É
radial")
                net.switch["closed"][linha_desligada] = True
                radial = 100

            else:
                print("Se desligar a linha", linha_desligada, ", a rede É
radial")
                op_radial.append(linha_desligada)
                radial = 0

    cont_aberto=0
    for i in net.switch["closed"].index:
        if net.switch["closed"].loc[i] == False:
            cont_aberto+=1

    linha_desconsiderada = x
    if cont_aberto < 5:
        print("Desconsiderar ultima alteração")
        net.switch["closed"][linha_desligada] = True
        net.switch["closed"][x] = False
        linha_desconsiderada = linha_desligada
        linha_desligada=x
        print("Ultima modificação foi desconsiderada")
        print("Linha ligada:",linha_desconsiderada, "Linha desligada:",
linha_desligada)



    Sec_aberto = []
    for i in net.switch["closed"].index:
        if net.switch["closed"].loc[i] == False:
            Sec_aberto.append(i)
    print("Linhas com seccionadores abertos", Sec_aberto)

    while 1: #segunda etapa do ciclo
        #DETERMINA PELO MÉTODO DE GS A TENSAO DOS BARRAMENTOS E POTENCIA DE
PERDAS TOTAL DA REDE
```

```
        pp.runpp(net, algorithm="gs")

        bj = 0
        bi = 0
        perda_linha = []

        resist_linha=net.line["r_ohm_per_km"]*net.line["length_km"]
        potencia_ativa=net.res_line["p_to_mw"] #esse valor equivale a pu
        potencia_reativa=net.res_line["q_to_mvar"] #esse valor equivale a pu
        tensao=net.res_line["vm_to_pu"]*net.bus["vn_kv"]

        for x in net.line["to_bus"].index:
            bj=net.line["to_bus"][x]

perda_linha.append(resist_linha[x]*((potencia_ativa[x]**2+potencia_reativa[x]**2
)/tensao[x]**2)*10)
        #perda total = somatorio das perdas nas linhas
        perda_t= 0
        for num in perda_linha:
            if num > 0:
                perda_t=perda_t + num
        perda_total.append(perda_t)
        print("a perda total da rede é", perda_total[iter], "pu ")
        print ("Diferença de perdas", perda_total[iter-1]-perda_total[iter],
"pu")
        Sec_aberto = []
        for i in net.switch["closed"].index:
            if net.switch["closed"].loc[i] == False:
                Sec_aberto.append(i)
        print("Linhas com seccionadores abertos", Sec_aberto)

        if perda_total[iter]-perda_total[iter-1] < 0:
              print("segunda etapa ciclo")
              iter = iter + 1
              print("ITERAÇÃO", iter)
              linha_p_ligar = linha_desligada
              net.switch["closed"][linha_p_ligar] = True
              print("A linha", linha_p_ligar, "foi ligada")

              # #olha quais barramentos a linha que foi ligada no passo anterior
pertence
              maior_v_dif = None
              bi=net.line["from_bus"][linha_p_ligar]
              bj=net.line["to_bus"][linha_p_ligar]

              #identificar a proxima linha adjacente
              barr_p_estudar =bi

              opcao_desligar=[]
              indice_linha = 0
              for i in net.line["from_bus"]:
                  indice_linha +=1
                  if i == barr_p_estudar and indice_linha!= linha_p_ligar:
                      opcao_desligar.append(indice_linha)
```

```
                indice_linha = 0
                for i in net.line["to_bus"]:
                    indice_linha +=1
                    if i == barr_p_estudar and indice_linha!= linha_p_ligar:
                        opcao_desligar.append(indice_linha)
                print("As opções de linhas para desligar são", opcao_desligar)

                #Verifica qual linha é possivel desligar

                verif_bus = 0
                linha_desligada = 0
                op_radial=[]
                for linha_desligada in opcao_desligar:
                    if net.switch["closed"][linha_desligada] == False:
                        print("a linha", linha_desligada, "já estava desligada")
                    else:
                        net.switch["closed"][linha_desligada] = False
                        pp.runpp(net)

                      #TESTE RADIAL
                        zeros = 1
                        vetor= []
                        vetor_bus=[]

                        for cont in range (1,bus+1):
                            for i in range (1,l+1):
                                if net.line["from_bus"][i]==cont and
net.switch["closed"][i]== True:
                                    vetor.append(net.line["to_bus"][i])
                                if net.line["to_bus"][i]==cont and
net.switch["closed"][i]== True:
                                    vetor.append(net.line["from_bus"][i])
                            vetor_bus.append(vetor)
                            vetor=[]
                        matriz = []
                        for linha in range (0,bus):
                            for coluna in range (0,bus):
                                if linha == coluna:
                                    matriz.append(len(vetor_bus[linha]))
                                else:
                                    matriz.append(0)

                        X=np.array(matriz)
                        shape = (bus,bus)
                        X=X.reshape(shape)

                        for linha in range (0,bus):
                            for i in vetor_bus[linha]:
                                X[linha][i-1]=-1

                        try:
                            XX = np.linalg.inv(X)
                        except:
                            zeros = 10000
```

```
                    if zeros !=10000:
                        zeros=XX.size - np.count_nonzero(XX)
                    if zeros != 0:
                        print("Se desligar a linha", linha_desligada, ", a
rede NÃO É radial")
                        net.switch["closed"][linha_desligada] = True
                        radial = 100

                    else:
                        print("Se desligar a linha", linha_desligada, ", a
rede É radial")
                        radial = 0

            cont_aberto=0
            for i in net.switch["closed"].index:
                 if net.switch["closed"].loc[i] == False:
                     cont_aberto+=1

            linha_desconsiderada = linha_p_ligar
            if cont_aberto < 5:
                 print("Desconsiderar ultima alteração")
                 net.switch["closed"][linha_desligada] = True
                 net.switch["closed"][linha_p_ligar] = False
                 linha_desconsiderada = linha_desligada
                 linha_desligada=linha_p_ligar
                 print("Ultima modificação foi desconsiderada")
                 print("Linha ligada:",linha_desconsiderada, "Linha
desligada:", linha_desligada)

            Sec_aberto = []
            for i in net.switch["closed"].index:
                if net.switch["closed"].loc[i] == False:
                    Sec_aberto.append(i)
            print("Linhas com seccionadores abertos", Sec_aberto)


        else:
            #desconsiderar ultima alteração
            linha_desconsiderada = linha_p_ligar
            net.switch["closed"][linha_desligada] = True
            net.switch["closed"][linha_p_ligar] = False
            linha_desconsiderada = linha_desligada
            linha_desligada=linha_p_ligar
            print("Ultima modificação foi desconsiderada")
            print("Linha ligada:",linha_desconsiderada, "Linha desligada:",
linha_desligada)
            break

    k=k+1
    iter=iter+1
    print("k",k)
    if k == 6:
        print ("As perdas totais são", perda_total)
        menor_perda=min(perda_total)
```

```
        indice = -1
        for indice_menorperda in perda_total:
            indice +=1
            if menor_perda == indice_menorperda:
                print("A menor perda foi", menor_perda, "na iteração",indice )

        print("A redução total das perdas foi
de",((perda_total[0]-menor_perda)/perda_total[0])*100, "%")

print("fim, obrigada!")
```

ANEXO A3 – *SCRIPT* "INITIAL SEARCHING POINT.PY"

```
# -*- coding: utf-8 -*-
"""
Created on Sun Apr 24 12:24:26 2022

@author: eveni
"""

#METODO PARA DETERMINAR O INICIAL SEARCHING POINT
import pandapower as pp
import pandapower.networks
import numpy as np
#import pandas as pd

iter = 0
bus =33
l=37
X = []
XX =[]

#STEP 1: DETERMINAR REDE INICIAL
net = pp.from_excel("IEEE_33_Bus.xlsx")


#STEP 2: DESLIGAR UM DOS SWITCHES
Sec_aberto_inicio = []
for i in net.switch["closed"].index:
    if net.switch["closed"].loc[i] == False:
        Sec_aberto_inicio.append(i)
print("Linhas com seccionadores abertos NO INICIO",
Sec_aberto_inicio)
Sec_aberto_final = Sec_aberto_inicio

while iter <5:
    print("ITERACAO", iter)
    Sec_aberto = Sec_aberto_final
    #desliga o seccionador da iter atual
    net.switch["closed"][Sec_aberto[iter]] = True
    Sec_aberto = []
    for i in net.switch["closed"].index:
        if net.switch["closed"].loc[i] == False:
            Sec_aberto.append(i)
    print("Linhas com seccionadores abertos", Sec_aberto)


    #STEP 3: POWER FLOW
    pp.runpp(net)

    #STEP 4: SELECIONAR LINHA DA MALHA COM MENOR VALOR DE
CORRENTE E ABRIR ESSA LINHA
    #configurar as linhas que ficam em malha para o fechamento
de cada switch
    if net.switch["closed"][33] == True:
```

```
        malha = [2, 3, 4, 5, 6, 7, 18, 19, 20]
    if net.switch["closed"][34] == True:
        malha = [9, 10, 11, 12, 13, 14]
    if net.switch["closed"][35] == True:
        malha = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 18, 19, 20, 21]
    if net.switch["closed"][36] == True:
        malha = [6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 25,
26, 27, 28, 29, 30, 31, 32]
    if net.switch["closed"][37] == True:
        malha = [3, 4, 5, 22, 23, 24, 25, 26, 27, 28]
    a = malha
    while 1:
        corrente_linha_malha= []
        #verificar se a linha já esta no sec_aberto_final
        for rep in Sec_aberto_final:
            for i in a:
                if i == rep:
                    malha.remove(rep)
                    # print("linha removida", i)
       #armazenar as correntes da malha
        cont = 0
        for x in net.res_line["i_ka"][malha]:
            corrente_linha_malha.append(x)
            print("Corrente linha", malha[cont], "é", x)
            cont = cont +1
        menor_corrente = min(corrente_linha_malha)
        #identificar a linha com menor corrente
        for i in malha:
            if menor_corrente == net.res_line["i_ka"][i]:
                linha = i
        print("A linha com menor corrente é", linha)

        #abrir seccionador da linha com menor corrente
        net.switch["closed"][linha] = False
        radial = 0
        #STEP 5: VERIFICAR SE A REDE É RADIAL SE NAO FOR,VOLTAR
PARA O PASSO 4
        pp.runpp(net)
        zeros = 1

        vetor= []
        vetor_bus=[]
        for cont in range (1,bus+1):
            for i in range (1,l+1):
                if net.line["from_bus"][i]==cont and
net.switch["closed"][i]== True:
                    vetor.append(net.line["to_bus"][i])
                if net.line["to_bus"][i]==cont and
net.switch["closed"][i]== True:
                    vetor.append(net.line["from_bus"][i])
            vetor_bus.append(vetor)
            vetor=[]

        matriz = []
        for linha1 in range (0,bus):
            for coluna in range (0,bus):
```

```
                if linha1 == coluna:
                    matriz.append(len(vetor_bus[linha1]))
                else:
                    matriz.append(0)

        X=np.array(matriz)
        shape = (bus,bus)
        X=X.reshape(shape)

        for linha1 in range (0,bus):
            for i in vetor_bus[linha1]:
                X[linha1][i-1]=-1

        try:
            XX = np.linalg.inv(X)
        except:
            zeros = 10000
        if zeros !=10000:
            zeros=XX.size - np.count_nonzero(XX)
        if zeros == 0:
            # print("É radial")
            print("A rede continua radial se desligar a linha",
linha)
            break
        else:
            net.switch["closed"][linha] = True
            print("nao é radial")
            malha.remove(linha)


    #STEP 6: SUBSTITUIR O SWITH ORIGINAL PELO DESLIGADO NO PASSO
ANTERIOR
    Sec_aberto_final[iter]=linha
    print("Sec aberto final", Sec_aberto_final)
    iter = iter + 1

    #STEP 7: REPETIR STEPS 2 - 6 PARA DETERMINAR O PROXIMO
SWITCH ABERTO


#STEP 8: O ALGORITMO VAI ACABAR QUANDO TODOS OS SWITCHES
ORIGINAIS FOREM SUBSTITUIDOS
if iter == 5: #Numero de switches abertos
    ISP = Sec_aberto_final
    print("INICIAL SEARCHING POINT =", ISP)
```

ANEXO A4 – *SCRIPT* "REDUCAO_PERDAS_METODO_AG_ISP.PY"

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 25 13:36:28 2022

@author: eveni
"""
import pandapower as pp
import pandapower.networks
import random
import numpy as np

#INICIAL SEARCH POINT = [7, 14, 10, 32, 28]
isp = [6, 14, 9, 32, 28]

net = pp.from_excel("IEEE_33_Bus.xlsx")
pp.runpp(net)

bj = 0
bi = 0
radial = 0
deletar =[]
perda_linha = []
populacaoapta = []
resist_linha=net.line["r_ohm_per_km"]*net.line["length_km"]
potencia_ativa=net.res_line["p_to_mw"] #esse valor equivale a pu
potencia_reativa=net.res_line["q_to_mvar"] #esse valor equivale
a pu
tensao=net.res_line["vm_to_pu"]*net.bus["vn_kv"]
for x in net.line["to_bus"].index:
    bj=net.line["to_bus"][x]

perda_linha.append(resist_linha[x]*((potencia_ativa[x]**2+potenc
ia_reativa[x]**2)/tensao[x]**2)*10)
fitness_inicial= 0
for num in perda_linha:
    if num > 0:
        fitness_inicial=fitness_inicial + num
print("a perda da rede original", fitness_inicial, "pu ")

#DEFINICOES
#N: tamanho da população
#d: quantidade de switches abertos
#X: cromossomas
#r1: numero aleatorio entre 0 e 1
#Smaxd: linha aleatoria
#G: quantidade de gerações
#cromossomas_finais: são todos os cromossomas gerados em todas
as populações
l = 37 #quantidade de linhas
N = 8
bus = 33
d = 5
X= []
G=50
```

```
geracao = 1
cromossomas_finais= []
fitness_finais=[]
contador = 0
aux=0
Xaux = []
XXX= []
Cand = 0
Cand1=[]
Xauxa=[]
Keep = round(0.8*N) #Qnt_min:quantidade mínima de cromossomas a
ser mantida na seleção
taxa_mutacao = 0.2

#STEP 1: INICIALIZAR POPULAÇÃO
while 1:
    Xaux = []
    Smaxd =random.sample(range(2,l-1),5) #gera 5 numeros
aleatorios entre 2 e 37
    for i in range (0,d):
        Cand=1+(Smaxd[i]) #cada Cand representa um gene
        Xaux.append(Cand) #Xaux representa um cromossoma no
final do loop for

    #TESTE RADIAL: Só cromossomas radiais são aceites nesta
etapa
    #ligar todos os switches
    for x in net.switch["closed"].index:
        net.switch["closed"][x] = True

    #desligar os switches do cromossoma
    for x in Xaux:
        net.switch["closed"][x] = False

    #verificar se é radial
    try:
        pp.runpp(net)
    except:
        nao_convergiu = 0

    zeros = 1
    vetor= []
    vetor_bus=[]
    for cont in range (1,bus+1):
        for i in range (1,l+1):
            if net.line["from_bus"][i]==cont and
net.switch["closed"][i]== True:
                vetor.append(net.line["to_bus"][i])
            if net.line["to_bus"][i]==cont and
net.switch["closed"][i]== True:
                vetor.append(net.line["from_bus"][i])
        vetor_bus.append(vetor)
        vetor=[]
    matriz = []
    for linha in range (0,bus):
        for coluna in range (0,bus):
```

```
            if linha == coluna:
                matriz.append(len(vetor_bus[linha]))
            else:
                matriz.append(0)

    X=np.array(matriz)
    shape = (bus,bus)
    X=X.reshape(shape)

    for linha in range (0,bus):
        for i in vetor_bus[linha]:
            X[linha][i-1]=-1

    try:
        XX = np.linalg.inv(X)
    except:
        zeros = 10000
    if zeros !=10000:
        zeros=XX.size - np.count_nonzero(XX)
    if zeros == 0:
        perda_linha = []

resist_linha=net.line["r_ohm_per_km"]*net.line["length_km"]
        potencia_ativa=net.res_line["p_to_mw"] #esse valor
equivale a pu
        potencia_reativa=net.res_line["q_to_mvar"] #esse valor
equivale a pu
        tensao=net.res_line["vm_to_pu"]*net.bus["vn_kv"]
        for x in net.line["to_bus"].index:
            bj=net.line["to_bus"][x]

perda_linha.append(resist_linha[x]*((potencia_ativa[x]**2+potenc
ia_reativa[x]**2)/tensao[x]**2)*10)
        fitness= 0
        for num in perda_linha:
            if num > 0:
                fitness=fitness + num
        if fitness!=0:
            XXX.append(Xaux)
            contador +=1

    Xaux = []
    if contador == N-1:
        break
for i in isp: #ADICIONA O ISP NA POPULACAO INICIAL
    Xaux.append(i)
XXX.append(Xaux)

Xi=np.array(XXX)
shape = (N,d)
populacao=Xi.reshape(shape)
print("POPULACAO INICIAL - TODA RADIAL/MALHA",populacao)

while 1:
    X= []
    print("GERAÇÃO:", geracao)
```

```
    #STEP 2: POWER FLOW,CALCULO DA FUNCAO FITNESS E VERIFICAR SE
É RADIAL

    radial = 0
    deletar =[]
    perda_linha = []
    populacaoapta = []
    fitnessfunction= []
    for a in range(0,N): #verifica se os cromossomas da
populacao são radiais
        radial = 0
        #ligar todos os switches
        for x in net.switch["closed"].index:
            net.switch["closed"][x] = True

        #desligar os switches do cromossoma
        for x in populacao[a]:
            net.switch["closed"][x] = False

        #verificar se é radial
        try:
            pp.runpp(net)
        except:
            nao_convergiu = 0

        #TESTE RADIAL
        zeros = 1
        vetor= []
        vetor_bus=[]
        for cont in range (1,bus+1):
            for i in range (1,l+1):
                if net.line["from_bus"][i]==cont and
net.switch["closed"][i]== True:
                    vetor.append(net.line["to_bus"][i])
                if net.line["to_bus"][i]==cont and
net.switch["closed"][i]== True:
                    vetor.append(net.line["from_bus"][i])
            vetor_bus.append(vetor)
            vetor=[]
        matriz = []
        for linha in range (0,bus):
            for coluna in range (0,bus):
                if linha == coluna:
                    matriz.append(len(vetor_bus[linha]))
                else:
                    matriz.append(0)

        X=np.array(matriz)
        shape = (bus,bus)
        X=X.reshape(shape)

        for linha in range (0,bus):
            for i in vetor_bus[linha]:
                X[linha][i-1]=-1

        try:
```

```
            XX = np.linalg.inv(X)
        except:
            zeros = 10000
        if zeros !=10000:
            zeros=XX.size - np.count_nonzero(XX)
        if zeros != 0:
            radial = 100

        else:
            radial = 0


        #remove o cromossoma se não for radial
        if radial ==100:
            deletar.append(a)


        #calcular fitness function dos cromossomas aptos
        if radial == 0:
            perda_linha = []

resist_linha=net.line["r_ohm_per_km"]*net.line["length_km"]
            potencia_ativa=net.res_line["p_to_mw"] #esse valor
equivale a pu
            potencia_reativa=net.res_line["q_to_mvar"] #esse
valor equivale a pu
            tensao=net.res_line["vm_to_pu"]*net.bus["vn_kv"]
            for x in net.line["to_bus"].index:
                bj=net.line["to_bus"][x]

perda_linha.append(resist_linha[x]*((potencia_ativa[x]**2+potenc
ia_reativa[x]**2)/tensao[x]**2)*10)
            fitness= 0
            for num in perda_linha:
                if num > 0:
                    fitness=fitness + num

            fitnessfunction.append(fitness)
            #apenas os cromossomas radiais ficam no grupo
populacao apta
            populacaoapta.append(populacao[a])

    # print("DELETAR", deletar)
    pop=np.array(populacaoapta)
    shape = (N-len(deletar),d)
    pop_Apta=pop.reshape(shape)



    #STEP 3: SELEÇÃO DOS CROMOSSOMAS VÁLIDOS
    print("fitness", fitnessfunction)
    fitness_ordenada = sorted(fitnessfunction)

     #Criar uma lista de cromossomas validos apenas com os
validos (de menor fitness)
     #Mantém os Keep melhores cromossomas validos
```

```
    indice_cromoss = []
    for i in fitness_ordenada:
        for ii in range(0,N-len(deletar)):
            if i==fitnessfunction[ii]:
                if ii not in indice_cromoss:
                    indice_cromoss.append(ii)

    #deletar cromossomas excedentes
    apagar = len(indice_cromoss)-Keep
    if apagar <= 0:
        print("Não existem cromossomas suficientes")

    for i in range(0,apagar):
        indice_cromoss.pop()


    populacao_valida = []
    for i in indice_cromoss:
        populacao_valida.append(pop_Apta[i])

    #STEP 4: CROSSOVER PARA NOVA GERACAO (tradicional)
    #Definir quem são os pais (Método da selecao ROLETA)
    ##Selecionar um switch aleatorio
    qnt = Keep
    populacao_crossover = populacao_valida
    while qnt<N:
        radial1 = 0
        radial2 = 0
        pai_mae =random.sample(range(0,Keep-1),2) #gera numeros
aleatorios nao repetidos
        indice_pai =pai_mae[0]
        indice_mae=pai_mae[1]
        pai=populacao_valida[indice_pai]
        mae=populacao_valida[indice_mae]
        #alpha é o ponto onde vai ser feito o crossover (entre 0
e d-1)
        alpha = 3
        #gerar os filhos para completar a populacao
        filho1=np.array([0]*d)
        filho2=np.array([0]*d)
        filho1 = np.array(pai)
        filho2 = np.array(mae)
        for x in range(alpha,d):
            filho1[x] = mae[x]
            filho2[x] = pai[x]
        #TESTE RADIAL FILHO1
        radial = 0
        deletar =[]
        perda_linha = []
        populacaoapta = []
        fitnessfunction= []
        #ligar todos os switches
        for x in net.switch["closed"].index:
            net.switch["closed"][x] = True
```

```
        #desligar os switches do cromossoma
        for x in filho1:
            net.switch["closed"][x] = False

        #verificar se é radial
        try:
            pp.runpp(net)
        except:
            nao_convergiu = 0

        zeros = 1
        vetor= []
        vetor_bus=[]
        for cont in range (1,bus+1):
            for i in range (1,l+1):
                if net.line["from_bus"][i]==cont and
net.switch["closed"][i]== True:
                    vetor.append(net.line["to_bus"][i])
                if net.line["to_bus"][i]==cont and
net.switch["closed"][i]== True:
                    vetor.append(net.line["from_bus"][i])
            vetor_bus.append(vetor)
            vetor=[]
        matriz = []
        for linha in range (0,bus):
            for coluna in range (0,bus):
                if linha == coluna:
                    matriz.append(len(vetor_bus[linha]))
                else:
                    matriz.append(0)

        X=np.array(matriz)
        shape = (bus,bus)
        X=X.reshape(shape)

        for linha in range (0,bus):
            for i in vetor_bus[linha]:
                X[linha][i-1]=-1

        try:
            XX = np.linalg.inv(X)
        except:
            zeros = 10000
        if zeros !=10000:
            zeros=XX.size - np.count_nonzero(XX)
        if zeros != 0:
           # print("Não é radial")
            radial1 = 100

        else:
            repetido = np.unique(filho1)
            if repetido.size == 5:
                 radial1 = 0
                 populacao_crossover.append(filho1)
                 qnt=qnt+1
```

```
        if qnt==N:
            break

        #TESTE RADIAL FILHO2
        radial = 0
        deletar =[]
        perda_linha = []
        populacaoapta = []
        fitnessfunction= []
        #ligar todos os switches
        for x in net.switch["closed"].index:
            net.switch["closed"][x] = True

        #desligar os switches do cromossoma
        for x in filho2:
            net.switch["closed"][x] = False

        #verificar se é radial
        try:
            pp.runpp(net)
        except:
            nao_convergiu = 0

        zeros = 1
        vetor= []
        vetor_bus=[]
        for cont in range (1,bus+1):
            for i in range (1,l+1):
                if net.line["from_bus"][i]==cont and
net.switch["closed"][i]== True:
                    vetor.append(net.line["to_bus"][i])
                if net.line["to_bus"][i]==cont and
net.switch["closed"][i]== True:
                    vetor.append(net.line["from_bus"][i])
            vetor_bus.append(vetor)
            vetor=[]
        matriz = []
        for linha in range (0,bus):
            for coluna in range (0,bus):
                if linha == coluna:
                    matriz.append(len(vetor_bus[linha]))
                else:
                    matriz.append(0)

        X=np.array(matriz)
        shape = (bus,bus)
        X=X.reshape(shape)

        for linha in range (0,bus):
            for i in vetor_bus[linha]:
                X[linha][i-1]=-1

        try:
            XX = np.linalg.inv(X)
        except:
            zeros = 10000
```

```
        if zeros !=10000:
            zeros=XX.size - np.count_nonzero(XX)
        if zeros != 0:
            radial2 = 100
        else:
            repetido = np.unique(filho2)
            if repetido.size == 5:
                 radial2 = 0
                 populacao_crossover.append(filho2)
                 qnt=qnt+1


    populacao_crossover=np.array(populacao_crossover)
    shape = (N,d)
    populacao_crossover=populacao_crossover.reshape(shape)
    # print("POPULACAO CROSSOVER", populacao_crossover)


    #STEP 5: MUTACAO PARA GERAR NOVOS CROMOSSOMAS

    #Mutacao a partir do segundo cromossoma por causa do
elitismo
    pop_mutacao = populacao_crossover
    t=1
    r=0
    S=0
    fit_mut=[]
    mut = []
    n_pos = 0
    n_min = 0
    verif=0
    mutados = round(taxa_mutacao*N+1)
    while t <mutados:
        rcromos = random.randint(2,N-1)
        rgene = random.randint(0,d-1)
        for uu in range (2,l+1):
            # print("INICIAL",uu,rgene,pop_mutacao[rcromos])
            verif=0
            for aaa in range(0,d): #verifica se o gene do
cromossoma é igual a uu
                if uu== pop_mutacao[rcromos][aaa]:
                    verif=10
            if verif==0:
                pop_mutacao[rcromos][rgene]=uu
                #teste radial
                radial = 0
                zeros =0
                deletar =[]
                perda_linha = []
                fitnessfunction= []
                #ligar todos os switches
                for x in net.switch["closed"].index:
                    net.switch["closed"][x] = True

                #desligar os switches do cromossoma
                for x in pop_mutacao[rcromos]:
```

```
                    net.switch["closed"][x] = False

                #verificar se é radial
                try:
                    pp.runpp(net)
                except:
                    nao_convergiu = 0

                #TESTE RADIAL
                zeros = 1
                # print("Teste radial", pop_mutacao[rcromos])
                vetor= []
                vetor_bus=[]
                for cont in range (1,bus+1):
                    for i in range (1,l+1):
                        if net.line["from_bus"][i]==cont and
net.switch["closed"][i]== True:
                            vetor.append(net.line["to_bus"][i])
                        if net.line["to_bus"][i]==cont and
net.switch["closed"][i]== True:

vetor.append(net.line["from_bus"][i])
                    vetor_bus.append(vetor)
                    vetor=[]
                matriz = []
                for linha in range (0,bus):
                    for coluna in range (0,bus):
                        if linha == coluna:
                            matriz.append(len(vetor_bus[linha]))
                        else:
                            matriz.append(0)

                X=np.array(matriz)
                shape = (bus,bus)
                X=X.reshape(shape)

                for linha in range (0,bus):
                    for i in vetor_bus[linha]:
                        X[linha][i-1]=-1

                try:
                    XX = np.linalg.inv(X)
                except:
                    zeros = 10000
                if zeros !=10000:
                    zeros=XX.size - np.count_nonzero(XX)
                if zeros != 0:
                    # print("Não é radial")
                    r+=1
                else:

                    # print("É radial")
                    #fitness
                    perda_linha = []

resist_linha=net.line["r_ohm_per_km"]*net.line["length_km"]
```

```
                    potencia_ativa=net.res_line["p_to_mw"] #esse
valor equivale a pu
                    potencia_reativa=net.res_line["q_to_mvar"]
#esse valor equivale a pu

tensao=net.res_line["vm_to_pu"]*net.bus["vn_kv"]
                    for x in net.line["to_bus"].index:
                        bj=net.line["to_bus"][x]

perda_linha.append(resist_linha[x]*((potencia_ativa[x]**2+potenc
ia_reativa[x]**2)/tensao[x]**2)*10)
                    fitness= 0
                    for num in perda_linha:
                        if num > 0:
                            fitness=fitness + num
                    # print("fitness",fitness)
                   #
print("RADIAL",uu,rcromos,rgene,pop_mutacao[rcromos], fitness)
                    if fitness!=0:
                        fit_mut.append(fitness)
                        mut.append(uu)
                        # print("fit,uu", fit_mut,mut)

        # print("Fiit_mut", fit_mut)
        # n_min = min(fit_mut) #verifica o menor valor de
fitness e faz a mutacao do cromossoma e gene selecionado
        # n_pos = fit_mut.index(n_min)
        #print("min", fit_mut,n_min,n_pos, mut[n_pos])
        #organizar as candidatas
        fit_mut_ordenada =[]
        fit_mut_ordenada = sorted(fit_mut)

        indice_cromoss = []
        for i in fit_mut_ordenada:
            for ii in range(0,len(fit_mut)):
                if i==fit_mut[ii]:
                    if ii not in indice_cromoss:
                        indice_cromoss.append(ii)

        # n_pos = fit_mut_ordenada[0]
        r=0
        t+=1
        pop_mutacao[rcromos][rgene]=mut[indice_cromoss[0]]
        #
#print("FINAL",mut[n_pos],rcromos,rgene,pop_mutacao[rcromos],n_m
in)
        fit_mut = []
        mut = []

    # print("POPULAÇÃO MUTAÇÃO", pop_mutacao)
    a=1
    b=1

    #STEP 6: FITNESS FUNCTION
    #calcular fitness function dos cromossomas da nova geracao
    fitnessfunction = []
```

```
    for a in range(0,N):
        #ligar todos os switches
        for x in net.switch["closed"].index:
            net.switch["closed"][x] = True

        #desligar os switches do cromossoma
        for x in pop_mutacao[a]:
            net.switch["closed"][x] = False
        try:
            pp.runpp(net)
        except:
            nao_convergiu=0
        perda_linha = []

resist_linha=net.line["r_ohm_per_km"]*net.line["length_km"]
        potencia_ativa=net.res_line["p_to_mw"] #esse valor
equivale a pu
        potencia_reativa=net.res_line["q_to_mvar"] #esse valor
equivale a pu
        tensao=net.res_line["vm_to_pu"]*net.bus["vn_kv"]
        for x in net.line["to_bus"].index:
            bj=net.line["to_bus"][x]

perda_linha.append(resist_linha[x]*((potencia_ativa[x]**2+potenc
ia_reativa[x]**2)/tensao[x]**2)*10)
        fitness= 0
        for num in perda_linha:
            if num > 0:
                fitness=fitness + num
        fitnessfunction.append(fitness)



    # print("FITNESS", fitnessfunction)


    #salvar os valores das geracoes e fitness
    cromossomas_finais.append(pop_mutacao)
    fitness_finais.append(fitnessfunction)
    populacao = pop_mutacao


    geracao +=1
    if geracao > G:
        print("--------------ANÁLISES DAS",G, "GERAÇÕES---------
--------")
        break

#Análises dos resultados
nao_radiais = 0
invalidos = 0
validos = 0
cromossomas_nao_radiais = []
cromossomas_invalidos=[] #perda total superior a perda inicial
cromossomas_validos=[]
fitness_validas = []
```

```
aux = 0
G_valido = []
N_valido = []
zeros = 0
for a in range(0,G):
    for aa in range (0,N):
        #VERIFICAR RADIAL
        #ligar todos os switches
        for x in net.switch["closed"].index:
            net.switch["closed"][x] = True
        # print("Cromossoma final",cromossomas_finais[a][aa])
        #desligar os switches do cromossoma
        for x in cromossomas_finais[a][aa]:

            net.switch["closed"][x] = False

            #verificar se é radial
        try:
              pp.runpp(net)
        except:
              nao_convergiu=0
        #TESTE RADIAL
        vetor= []
        vetor_bus=[]
        for cont in range (1,bus+1):
            for i in range (1,l+1):
                if net.line["from_bus"][i]==cont and
net.switch["closed"][i]== True:
                    vetor.append(net.line["to_bus"][i])
                if net.line["to_bus"][i]==cont and
net.switch["closed"][i]== True:
                    vetor.append(net.line["from_bus"][i])
            vetor_bus.append(vetor)
            vetor=[]
        matriz = []
        for linha in range (0,bus):
            for coluna in range (0,bus):
                if linha == coluna:
                    matriz.append(len(vetor_bus[linha]))
                else:
                    matriz.append(0)

        X=np.array(matriz)
        shape = (bus,bus)
        X=X.reshape(shape)

        for linha in range (0,bus):
            for i in vetor_bus[linha]:
                X[linha][i-1]=-1

        try:
            XX = np.linalg.inv(X)
        except:
            zeros = 10000
        if zeros !=10000:
```

```
            zeros=XX.size - np.count_nonzero(XX)
        if zeros != 0:
            # print("Não é radial",zeros)
            nao_radiais+=1

cromossomas_nao_radiais.append(cromossomas_finais[a][aa])
        else:
            # print("É radial")
            if 0<fitness_finais[a][aa]<fitness_inicial:

cromossomas_validos.append(cromossomas_finais[a][aa])
                fitness_validas.append(fitness_finais[a][aa])
                validos+=1
                G_valido.append(a)
                N_valido.append(aa)
            else:
                invalidos+=1

cromossomas_invalidos.append(cromossomas_finais[a][aa])
print("Não radiais", nao_radiais)
print("Inválidos", invalidos)
print("Válidos", validos)
Xi=np.array(cromossomas_nao_radiais)
shape = (nao_radiais,d)
cromossomas_nao_radiais=Xi.reshape(shape)
#print("Cromossomas não radiais", cromossomas_nao_radiais)
Xi=np.array(cromossomas_invalidos)
shape = (invalidos,d)
cromossomas_invalidos=Xi.reshape(shape)
#print("Cromossomas invalidos", cromossomas_invalidos)
Xi=np.array(cromossomas_validos)
shape = (validos,d)
cromossomas_validos=Xi.reshape(shape)
# for i in range(0,len(cromossomas_validos)):
#     print("Cromossomas validos", cromossomas_validos[i],
fitness_validas[i])


#criterio de fecho: media da fitness a cada geracao
soma = 0
media = 0
for a in range(0,G):
    for aa in range (0,N):
        soma = fitness_finais[a][aa] +soma
    media = soma / N
    print("MEDIA FITNESS G", a+1,"é", media)
    soma = 0
#valores evolucao do pior
maior = 0
b = 0
bb = 0
for a in range(0,G):
    for aa in range (0,N):
        if fitness_finais[a][aa]>maior:
            maior = fitness_finais [a][aa]
            b = a
```

```
            bb = aa
    print("PIOR FITNESS G", a+1, cromossomas_finais[b][bb],
"fitness", fitness_finais[b][bb])
    maior = 0
    b=0
    bb=0
#valores evolucao do melhor
menor = 100000
b = 1000
bb = 1000
for a in range(0,G):
    for aa in range (0,N):
        if fitness_finais[a][aa]>0 and
fitness_finais[a][aa]<menor:
            menor = fitness_finais [a][aa]
            b = a
            bb = aa
    if b != 1000:
        reducao_perdas= ((fitness_inicial-
fitness_finais[b][bb])/fitness_inicial)*100
        print("MELHOR FITNESS G",b+1,
fitness_finais[b][bb],cromossomas_finais[b][bb], "redução
perdas",reducao_perdas)
    else:
        print("Não tiveram bons cromossomas na geração",a+1)
    menor = 100000
    b=1000
    bb=1000
```

ANEXO A5 – SPYDER

O Spyder foi escrito em Python e projetado para a linguagem de programação Python.[18]. O ambiente de desenvolvimento é interativo e possui editor para escrita do código, explorador de variáveis, gráficos e arquivos e console para apresentação dos resultados, conforme apresentado na Figura 40.

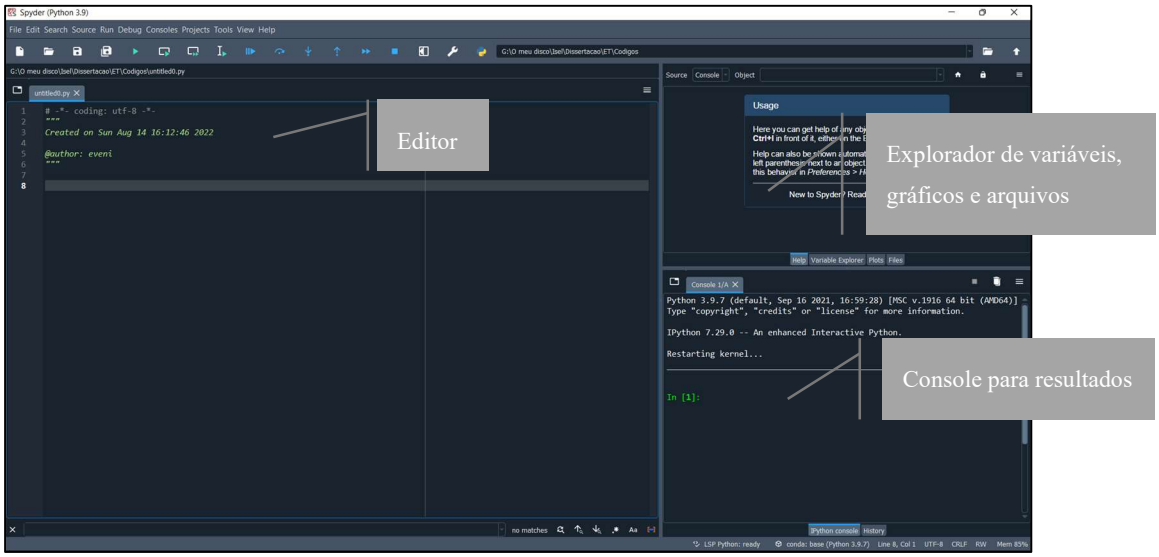


Figura 40: Spyder Software

Dentre diversos meios de efetuar a instalação do Spyder Software, no website spyder-ide.org [18] é recomendada a realização do download e instalação da plataforma Anaconda, que inclui o Spyder e outros pacotes úteis para o Python. O download da plataforma Anaconda pode ser feito no website anaconda.com [19].

ANEXO A6 – PYTHON

A linguagem de programação Python foi desenvolvida pelo programador e pesquisador Holandês Guido Van Rossum [20]. É uma linguagem de alto nível, ou seja, está mais próxima da linguagem humana do que da linguagem da máquina.

Foi projetada para facilitar a escrita do código e para ser utilizada tanto em aplicações mais simples quanto em programas mais complexos. A estrutura dos códigos escritos em Python apresentam alguns aspetos marcantes tais como: não se utiliza ponto e virgula (;) ao final de cada instrução e uma variável pode armazenar mais de um tipo de variável. A Figura 41 mostra um exemplo de código Python.

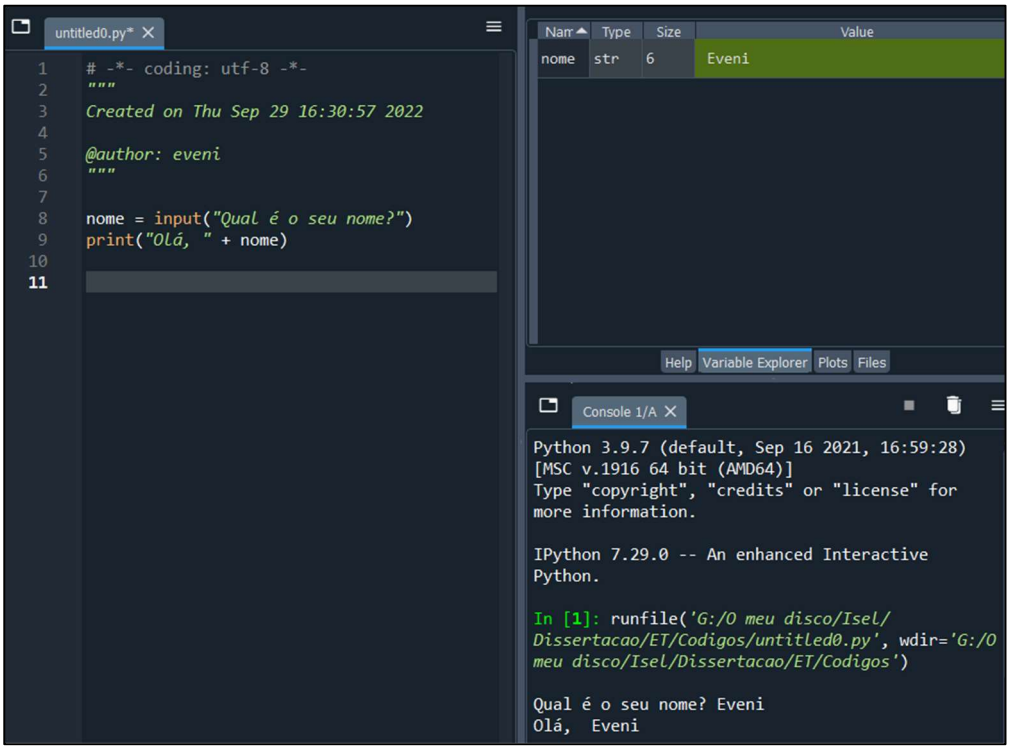


Figura 41: Exemplo código Python

ANEXO A7 – PANDAPOWER

A Figura 42 apresenta um exemplo de uma variável, barr_estudados, do tipo lista que armazena 10 valores.

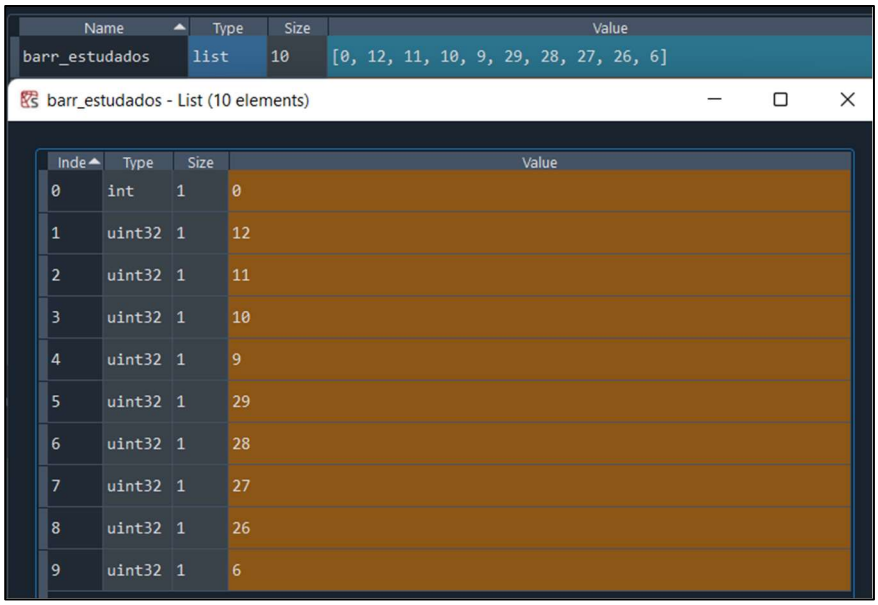


Figura 42: Exemplo de armazenamento de dados em formato de lista