LIA YAMAMOTO

USO DE *SIMULATED ANNEALING* E ALGORITMO GENÉTICO NO PROBLEMA DA RECONFIGURAÇÃO DE UMA REDE DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA

CURITIBA

2004

LIA YAMAMOTO

USO DE *SIMULATED ANNEALING* E ALGORITMO GENÉTICO NO PROBLEMA DA RECONFIGURAÇÃO DE UMA REDE DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Ciências, pelo Programa de Pós-Graduação em Métodos Numéricos em Engenharia – Programação Matemática, dos Setores de Tecnologia e de Ciências Exatas, da Universidade Federal do Paraná.

Orientador: Prof. Volmir Eugênio Wilhelm, Dr. Eng.

CURITIBA
2004

LIA YAMAMOTO

USO DE *SIMULATED ANNEALING* E ALGORITMO GENÉTICO NO PROBLEMA DA RECONFIGURAÇÃO DE UMA REDE DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre em Ciências, na Área de Concentração em Programação Matemática, do Programa de Pós-Graduação em Métodos Numéricos em Engenharia da Universidade Federal do Paraná, pela comissão formada pelos professores:

Orientador: _____

Prof. Volmir Eugênio Wilhelm, Dr. Eng.
Departamento de Matemática, UFPR

_____

Prof. Voldi Costa Zambenedetti, Dr. Eng.
Instituto de Tecnologia para o Desenvolvimento - LACTEC

_____

Profa. Maria Teresinha Arns Steiner, Dra.
Departamento de Matemática, UFPR

_____

Prof. José João Rossetto, Dr.
Departamento de Matemática, UFPR

Curitiba, 01 de outubro de 2004

Ao meu marido e aos meus pais.

AGRADECIMENTOS

Agradeço ao meu marido, Marcelo, pelo apoio e compreensão, durante a execução deste trabalho.

À minha mãe e ao meu pai, pelos ensinamentos, orientações dos caminhos certos a serem seguidos em momentos difíceis, apoiando me sempre em todas as decisões.

Ao professor Dr. VOLMIR EUGÊNIO WILHELM pela orientação fornecida, grande dedicação e apoio, fundamentais para a execução deste trabalho.

À professora Dra. MARIA TERESINHA ARNS STEINER aos conselhos e ensinamentos que muito influenciaram na realização deste trabalho.

À COPEL, em especial, ao engenheiro Fernando Gruppelli, pelo incentivo e conhecimentos fornecidos para o embasamento deste trabalho.

Ao LACTEC, especialmente ao Dr. Voldi Costa Zambenedetti e Eng. Mario Klimkowski, pelos ensinamentos passados essenciais ao início deste trabalho.

Aos professores do mestrado que se dedicaram fielmente ao ensinamento durante todo o curso.

Aos professores Dr. Yuan JinYun, Dr. Luiz Carlos Matioli e Dr. Celso Carnieri pelas orientações durante estudos iniciais ao trabalho.

Ao professor Ademir Alves Ribeiro que desde o início muito me apoiou, com muita paciência se dedica ao ensinamento, e pelo qual apresento grande admiração.

Aos funcionários do CESEC, Maristela e Eliseu, pela dedicação apresentada em todos os momentos.

Aos colegas de curso que, durante a realização dos estudos, muito me apoiaram.

Aos funcionários do departamento de matemática, Kimie e Lauro, que sempre se apresentaram muito pacientes no atendimento.

E às pessoas que, de alguma forma, direta ou indiretamente, auxiliaram neste trabalho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

As redes de distribuição de energia elétrica são projetadas de forma a minimizar as perdas de energia elétrica. Devido a vários fatores, como a expansão da cidade e o conseqüente aumento de consumo de energia, as distribuidoras são levadas a expandir a rede original, por vezes, de forma não otimizada. Tal expansão tende a elevar as perdas elétricas que podem ser reduzidas em função de nova configuração da rede.

Este trabalho descreve a aplicação de algoritmo genético, *simulated annealing* e de dois algoritmos híbridos na abordagem do problema de redução de perdas na rede de distribuição de energia elétrica. Considerando a configuração atual, propõe-se o fechamento de alguns trechos abertos e a abertura de outros trechos fechados visando a otimização da rede.

A procura de uma reconfiguração ótima para a rede de distribuição de energia é um problema de otimização combinatória, em que as técnicas heurísticas possuem a característica de fornecerem uma solução final próxima da solução ótima. Na aplicação dos quatro algoritmos, a função objetivo considerada é a perda de energia, que depende de dois fatores que são a resistência dos cabos que compõem a rede e o carregamento destes.

ABSTRACT

The electric distribution networks are originally projected to minimize energy losses. There are many factors that changes the energy demand, for an example with the development of the cities, the load increases and the expansion of the power utility network, many times without an optimal configuration of the system. Those development can cause an energy loss increase which can be reduced by a new configuration.

This research describes the genetic algorithm, simulated annealing applications and also two hybrid algorithms to deal with the electric power distribution systems reduction losses. Starting with the actual configuration, the purpose is to make a change of the open/closed status of the switching devices. The change selection is made by an optimization process.

The optimal electric distribution feeder reconfiguration is a combinatorial optimization problem which the results of heuristics techniques are close to the optimal solution. On electric distribution system, the objective function is the energy loss, which depends on the branches resistance and also the branches currents.

CAPÍTULO 1

1 INTRODUÇÃO

Ao longo das últimas décadas, segundo a Agência Nacional de Energia Elétrica [1], o consumo de energia elétrica apresentou altos índices de expansão, fruto do crescimento populacional concentrado nas zonas urbanas, do esforço de aumento da oferta de energia e da modernização da economia.

Assim, as redes de distribuição de energia elétrica construídas originalmente de modo a minimizar as perdas elétricas, são expandidas. Essas expansões ocorrem geralmente em etapas, não sendo feitas de forma otimizada, levando a perdas elétricas maiores. Por exemplo, na Figura 1 a rede de distribuição elétrica (trechos de traçado contínuo) foi constituída inicialmente de forma otimizada em relação às perdas elétricas. Após certo período de tempo a rede foi ampliada (trechos com traçados pontilhados) para atender novos pontos de demanda (círculos preenchidos).

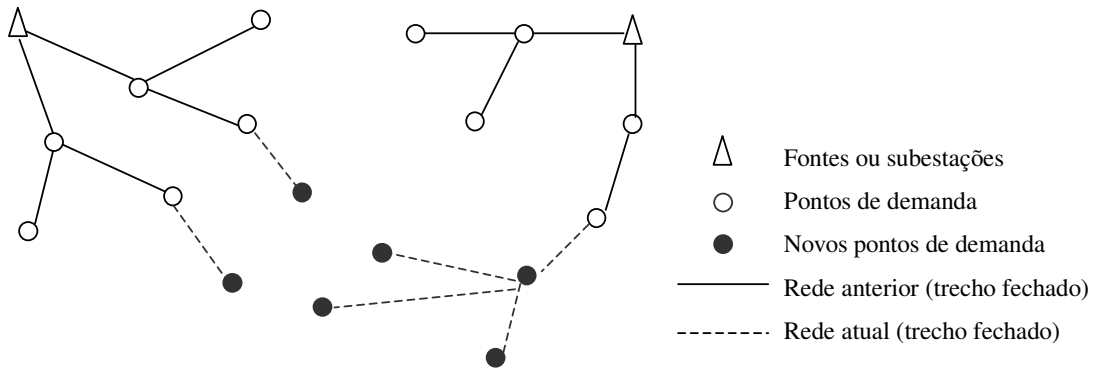


FIGURA 1 – **Configuração de uma rede de distribuição de energia elétrica**

Portanto, devido à expansão da rede é interessante proceder uma configuração da mesma para reduzir as perdas elétricas. A reconfiguração consiste, basicamente, em transferir pontos de demanda entre alimentadores através da abertura de alguns trechos e fechamento de outros.

No estudo da rede de distribuição de energia elétrica, além da reconfiguração, também pode-se considerar a recomposição ou restauração da rede e uma futura expansão. A recomposição considera os casos de possíveis mudanças da configuração da rede devido a ocasionais problemas que ocorram em pontos específicos na distribuição. A expansão pode ser considerada como um planejamento otimizado de uma rede futura, em que são criados novos trechos. Ambos os casos não estão sendo considerados neste trabalho.

Encontra-se na literatura uma grande gama de técnicas de reconfiguração de redes de distribuição de energia elétrica. Estas podem ser classificadas geralmente em dois grandes grupos: as técnicas heurísticas e as técnicas baseadas em modelagem matemática.

Dentre as várias técnicas pode-se citar Civanlar *et al.* [7], Shirmohammadi e Hong [18], Augugliaro *et al.* [2], Lin *et al.* [12], Borozan e Rajakovic [4], Carpaneto e Chicco [6] e Carnieri *et al.* [5] que utilizaram métodos heurísticos no estudo da configuração da rede de forma a obter uma configuração que apresente a menor perda elétrica. Huddleston *et al.* [11] formularam uma função quadrática para descrever o problema de perda elétricas, que permite a análise de múltiplos alimentadores simultaneamente. Sarma e Rao [17] formularam o problema para a resolução através de programação inteira binária. Baseado no estudo de Sarma e Rao, Volpi *et al.* [20] desenvolveram uma metodologia para reconfiguração da rede de distribuição de energia através de programação binária. Chiang e Jean-Jumeau [8] realizaram a formulação através de um problema não diferenciável e multi-objetivo, sendo considerada a redução de perdas e o balanço de carga.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é implementar e analisar quatro métodos heurísticos baseados em *simulated annealing* e em algoritmo genético.

Os objetivos específicos são:

- Realizar um levantamento bibliográfico de técnicas de reconfiguração de redes de distribuição de energia elétrica;
- Implementar quatro métodos heurísticos;
- Aplicar os métodos em redes de distribuição de energia elétrica fictícias;
- Analisar os quatro métodos em termos das soluções encontradas, do tempo de processamento e número de iterações;
- Verificar qual método apresenta melhor desempenho em relação ao objetivo específico anterior.

1.2 IMPORTÂNCIA DO TRABALHO

Segundo o Ministério de Minas e Energia [13], ao todo em 2002, o sistema de energia elétrica brasileiro apresenta aproximadamente 3% de perdas da energia transmitida e distribuída internamente.

De acordo com a Agência Nacional de Energia Elétrica [1], em 2000, o Brasil apresentou um consumo total de 306.207.515 MWh e a Copel, Companhia Paranaense de Energia Elétrica, correspondeu a 5,45% do mercado nacional, ou seja, 16.673.942 MWh.

A redução de perdas no sistema de distribuição de energia elétrica tem importância significativa pois representa uma redução do custo econômico para as empresas e, consequentemente aos consumidores. Além disso, com a redução pode-se obter um melhor aproveitamento da energia elétrica gerada e um aumento da qualidade dos serviços fornecidos aos consumidores.

1.3 METODOLOGIA

Neste estudo utilizar-se-á *simulated annealing* e algoritmo genético na abordagem do problema da reconfiguração da rede de distribuição de energia elétrica visando a redução de perdas. A meta é minimizar a função das perdas de energia de um sistema de distribuição elétrica simplificado. O cálculo da perda é representado pela soma das perdas de energia de todos os caminhos formados pela rede no abastecimento dos pontos de demanda de energia.

Supõe-se que a atual configuração da rede esteja próxima da configuração que apresenta menor perda de energia. O modelo de otimização procura efetuar troca do *status* de alguns trechos. Alguns trechos abertos (sem passagem de corrente) são fechados e outros que se apresentam fechados (há passagem de corrente) são abertos, atendendo algumas restrições elétricas. Assim ocorrerá transferência da demanda de energia entre os alimentadores, ou mudança da rota que abastece consumidores, reduzindo perdas elétricas. A cada iteração, executada pelos métodos de otimização, são escolhidos os trechos que serão abertos e os que serão fechados e se ocorrer uma diminuição das perdas, então tem-se uma nova configuração da rede. Ao iniciar uma nova iteração tem-se o uso da última configuração aceita e, assim, continua-se a executar o processo de otimização.

Inicialmente serão implementados dois métodos, um baseado em *simulated annealing* e outro em algoritmo genético. Em seguida serão implementados dois métodos híbridos. O processo de otimização desses métodos dá-se da seguinte forma:

(i) O processo de otimização do método híbrido AG/SA, inicia com o algoritmo genético e faz o uso da melhor configuração obtida para continuar o processo de otimização através do *simulated annealing*.

(ii) No método híbrido SA/AG, primeiramente, é executado o *simulated annealing* e, a partir do grupo das melhores soluções, faz o uso do algoritmo genético na busca de uma melhor configuração. Este modelo faz o uso de *simulated annealing* para gerar a população inicial necessária ao algoritmo genético na busca de uma configuração da rede que forneça menor perda elétrica.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em quatro capítulos e dois apêndices.

No primeiro capítulo tem-se a introdução do trabalho, a descrição dos objetivos, a importância e a metodologia adotada, além da descrição da estrutura do trabalho e de algumas definições de termos freqüentemente utilizados neste estudo.

O segundo capítulo descreve o problema da reconfiguração da rede de distribuição de energia elétrica e apresenta alguns modelos utilizados em estudos

anteriores. Além disso, são descritas as técnicas de *simulated annealing* e algoritmo genético.

O terceiro capítulo descreve os dois métodos híbridos implementados e analisados. Apresenta-se, também, a aplicação dos quatro métodos de otimização através de três exemplos de redes não reais, discutindo-se os resultados obtidos.

A conclusão do presente estudo e as recomendações para estudos futuros são apresentados no capítulo quatro.

Os dados utilizados pelas redes exemplificadas e os programas executados no *Matlab* são apresentados nos apêndices A e B, respectivamente.

CAPÍTULO 2

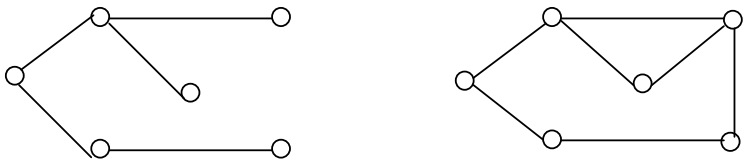
2 RECONFIGURAÇÃO DE SISTEMAS DE DISTRIBUIÇÃO

Para a realização do estudo de minimização das perdas de energia elétrica do sistema de distribuição faz-se necessário, primeiramente a visualização do sistema de distribuição real como uma rede, formado por trechos e nós, assim permitindo uma formulação de um modelo matemático para solucionar o problema de otimização.

No estudo de um sistema de distribuição de energia elétrica, a rede deve apresentar algumas características importantes. A rede deve ser conexa, pois a desconexidade implica na existência de algum nó não ligado ao sistema e isto significa que consumidores não recebem energia, como ilustrado na figura 2(b). A rede, também, deve ser radial (figura 3(a)), não apresentando ciclos fechados como na figura 3(b).

(a) Rede conexa (b) Rede desconexa

FIGURA 2 – **Rede conexa e rede desconexa**



(a) Rede Radial (b) Rede não radial

FIGURA 3 – **Rede Radial e rede não radial**

No presente estudo foi considerada uma rede simplificada, isto é, todo o sistema de distribuição de energia foi simplificado de forma que cada ponto de carga, ou nó, representa um transformador de energia, e a carga considerada representa um somatório das demandas supridas a partir deste transformador (figura 4 e figura 5). Um trecho faz a conecção entre dois nós (ligação entre os nós 3 e 4). Um conjunto de trechos consecutivos compõe os alimentadores da rede, que representam as rotas simplificadas dos cabos de distribuição, que levam a energia elétrica aos postes (toda a extensão de 1 a 5).

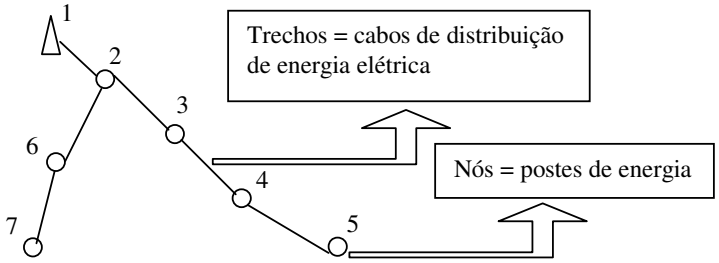


FIGURA 4 – **Simplificação da rede de distribuição de energia elétrica**

A cada trecho é associada uma chave; quando considerado o trecho fechado implica que esta chave apresenta o *status* fechado e há passagem de corrente pelo trecho. Quando o trecho apresentar o *status* aberto, não permite a passagem de corrente, estando a chave associada aberta.

Há a possibilidade de simplificar uma rede de distribuição através da diminuição do número de trechos e nós através da consideração de um região inteira como um bloco de carga. Por exemplo, na figura 5, o bloco formado pelos quatro nós com demanda 20 A, 15 A, 10 A e 5 A foram considerados como um único nó com demanda de 50 A, desde que nenhum dos trechos entre nós seja manobrável. Segundo Mussi [15], um bloco é um componente conexo da rede cujas chaves estão em trechos que ligam o componente com outras partes da rede. Este componente é representada por um único nó, onde não existe a possibilidade de mudança de configuração dentro deste bloco ou nó. Nestes casos, a demanda de energia do bloco é considerada apenas neste ponto de carga e os cálculos elétricos relativos ao sistema interno é excluído.

Carnieri *et al*. [5] e Volpi *et al*. [20] executaram simplificações na rede de distribuição de energia elétrica em seus estudos.

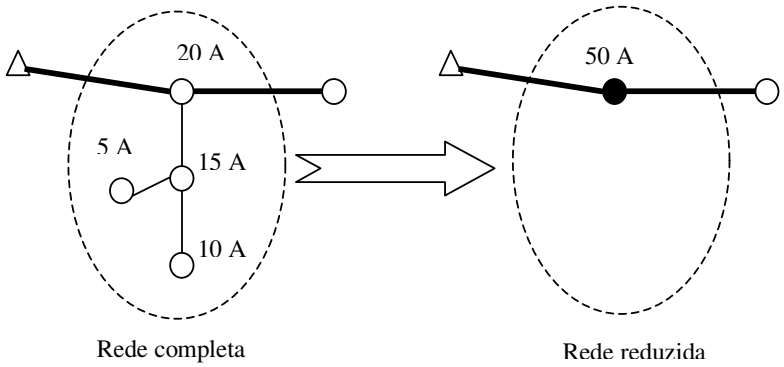


FIGURA 5 – **Exemplo de rede reduzida**

A reconfiguração das linhas de distribuição de energia é realizada através da abertura/fechamento das chaves do sistema, observando sempre que o sistema apresente uma forma radial. A troca de *status* das chaves permite transferir uma determinada quantidade de carga de um alimentador a outro, ou mudar o caminho percorrido até que a energia chegue em um ponto distante da fonte ou subestação, obtendo, assim, uma minimização de perdas. Entretanto, de acordo com Civanlar *et al.* [7], em redes de grande dimensão, o número de possibilidades de configurações é muito grande, tornando inviável o teste de todas as chaves, o que faz necessária a utilização de modelos que facilitem a identificação de uma boa configuração.

Na procura de uma boa configuração, em uma situação real, dois importantes aspectos devem ser considerados, como mencionado por Civanlar *et al.* [7]: (i) capacidade de estimar a mudança dos resultados das perdas com um mínimo de esforço computacional e (ii) um critério que possa ser utilizado para a eliminação das chaves que não se deseja mudar o *status* para diminuição da dimensão do problema, eliminando, assim, as chaves que não implicam na minimização da perda.

Geralmente o sistema de distribuição de energia é feito para atender as demandas dos picos de carga. As cargas, porém, são bastante diversificadas e apresentam uma grande variação no decorrer de um dia, resultando em picos de cargas em tempos diferentes para áreas diferentes. Assim, uma determinada região pode apresentar um alimentador muito carregado em um período, enquanto que ao mesmo tempo outro alimentador permanece com pouca carga. Entretanto, em períodos de pouca demanda de energia o sistema pode ser ineficiente ou não utilizar toda a sua capacidade.

A reconfiguração pode ser utilizada em um planejamento para um futuro aumento ou uma futura diminuição de carga de uma determinada região, ou mesmo, como uma ferramenta de operação periódica, permitindo, assim, uma otimização constante de acordo com as mudanças de cargas atribuídas às subestações.

Vários estudos são feitos com o interesse de minimizar perdas elétricas no sistema de distribuição, utilizando variadas metodologias na tentativa de melhorar as soluções obtidas. Dentre os métodos usados destacam-se as técnicas heurísticas baseadas em algoritmo genético e *simulated annealing*.

Na literatura encontram-se vários estudos que objetivam a reconfiguração do sistema de distribuição de energia elétrica visando a minimização de perdas elétricas. A seguir serão apresentados alguns desses trabalhos.

2.1 ALGUNS MODELOS DE RECONFIGURAÇÃO DE REDE

Civanlar *et al.* [7], em 1988, para reduzir as perdas elétricas em um sistema de distribuição, propuseram um método que, primeiramente, fecha todas as chaves abertas da rede, e então utiliza o método *load flow*, que fornece o fluxo de carregamento do trecho, calculado através de uma função simplificada, que mantivesse a rede radial ao abrir um dos trechos e fechar outro. Esse estudo possibilita a análise de apenas um par de alimentadores, a cada passo, realizando a transferência das cargas entre os dois alimentadores apenas quando houvesse a diminuição de perdas. Civanlar preocupou-se com a redução do número de trechos a serem considerados no estudo. Portanto ao fechar um trecho, que faz a ligação entre as fontes de energia, os trechos considerados possíveis de serem abertos são apenas os trechos restantes que formam esta ligação. Esse processo é repetido até não se trocar mais o *status* dos trechos.

Shirmohammadi e Hong [18], em 1989, determinaram a melhor configuração do sistema de distribuição elétrica através do fechamento inicial de todas as chaves abertas da configuração atual. Em seguida, analisou o sistema através do método *power flow*, abrindo as chaves dos trechos que apresentassem o menor fluxo de corrente, de forma a considerar apenas configurações radiais. A cada passo é realizada a abertura da chave que apresenta a menor corrente, até a abertura de

todos os ciclos previamente fechados. A figura 6 mostra o fluxograma do algoritmo utilizado pelo autor.

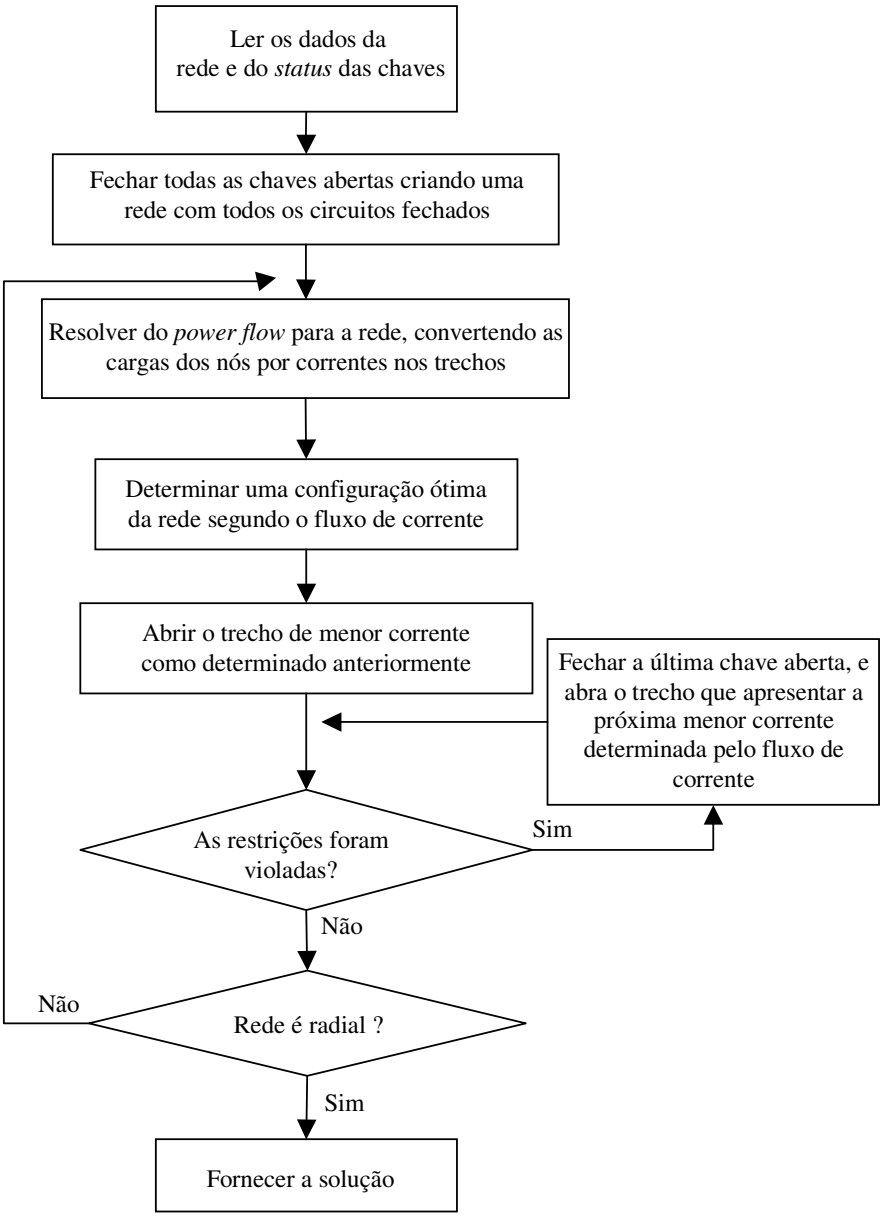


FIGURA 6 – **Fluxograma do algoritmo utilizado por Shirmohammadi e Hong**

Huddleston *et al.* [11], em 1990, formularam o problema para as perdas elétricas do sistema através de uma função quadrática com restrições de igualdade e desigualdade baseadas nas correntes. É um método que considera as tensões

nos trechos e permite realizar uma reconfiguração do sistema considerando vários circuitos simultaneamente para a redução das perdas.

Chiang e Jean-Jumeau [8], em 1990, formularam o problema de forma multi-objetiva e não-diferenciável com restrições de igualdade e desigualdade. A metodologia utilizada apresenta dois estágios. No primeiro estágio, utiliza-se na otimização a técnica *simulated annealing* modificado, em que são implementadas algumas restrições antes da análise de aceitação de uma nova solução. No segundo estágio é aplicado *simulated annealing* para o problema obtido através do método $\in$-*constraint*. Este método transforma um problema multi-objetivo em um problema com uma única função objetivo, através da adoção de um dos objetivos como função objetivo e os demais como restrições.

Augugliaro *et al.* [2], em 1995, utilizaram um procedimento principal e duas sub-rotinas heurísticas para otimização do problema de perdas elétricas. Primeiramente, utilizaram *power flow* para a geração de uma configuração radial inicial. Uma das sub-rotinas também utiliza power flow com algumas diferenças em sua formulação mantendo correntes e potências constantes, não abrindo simultaneamente as chaves que formam *loops*, adotando a abertura progressiva de chaves. Em outra sub-rotina é realizada a transferência de cargas da mesma forma que propôs Civanlar *et al.* [8]. O fluxograma apresentado pelo autor encontra-se na figura 7.

Sarma e Rao [17], em 1995, propuseram um método de programação inteira binária para a minimização de perdas de um sistema. A formulação se baseou-se no somatório das perdas de todos os trechos do sistema, sendo que a função foi desenvolvida para que pudesse utilizar apenas variáveis binárias. Esse método considera várias chaves a cada nova análise e converge para a melhor configuração da rede.

Borozan e Rajakovic [4], em 1997, propuseram uma metodologia em que a análise da adoção de um resultado envolvesse a solução teórica com a possível implementação prática. Essa metodologia é mais complexa, pois inicia-se utilizando um método heurístico baseado em *load flow*. Para a minimização do sistema, o resultado deve ainda ser submetido a uma verificação da relação custo - benefício. A figura 8 mostra um diagrama da aplicação do seu trabalho na análise da adoção de uma reconfiguração do sistema de distribuição de energia.

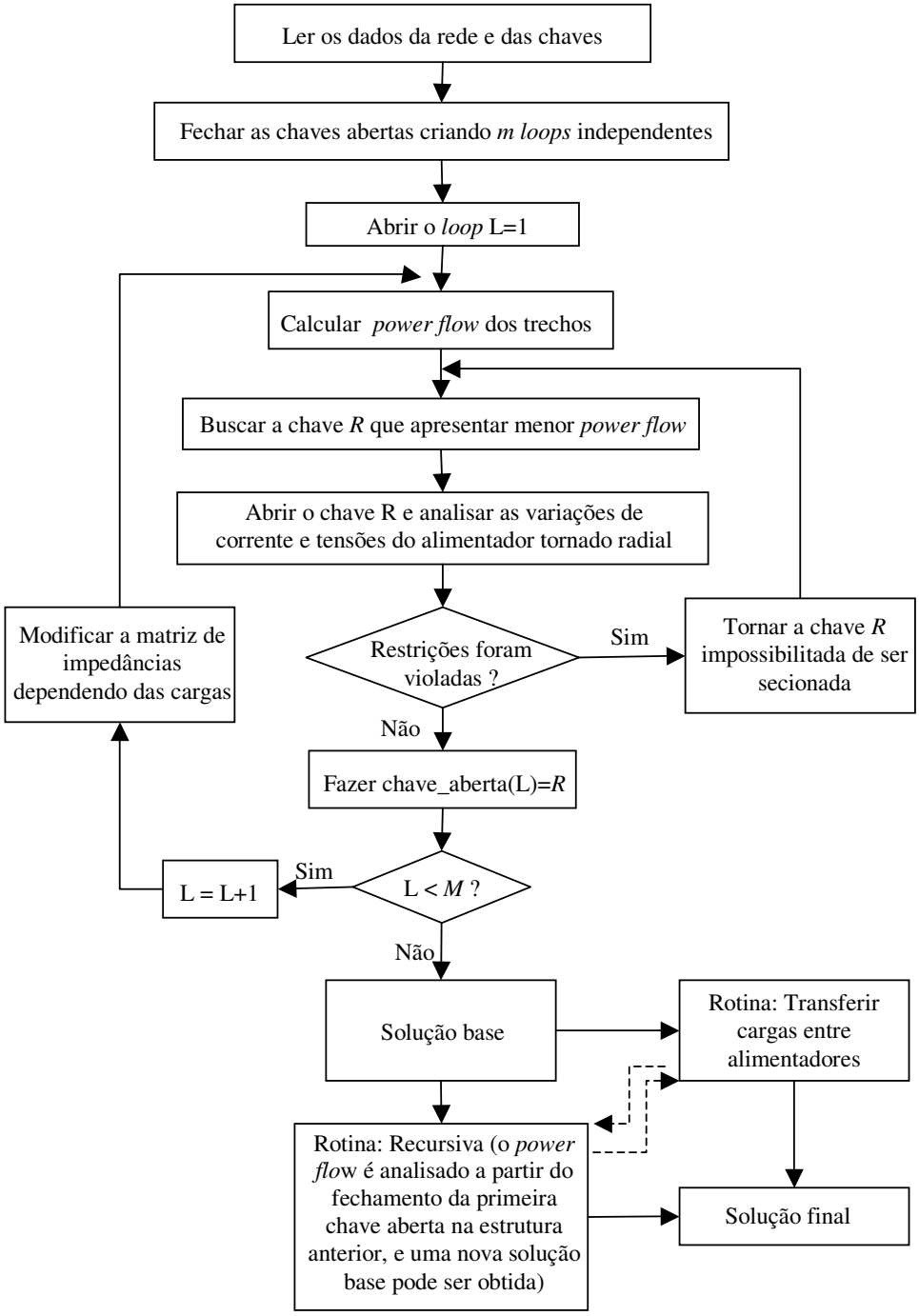


FIGURA 7 – **Fluxograma do algoritmo heurístico de Augugliaro *et al*.**

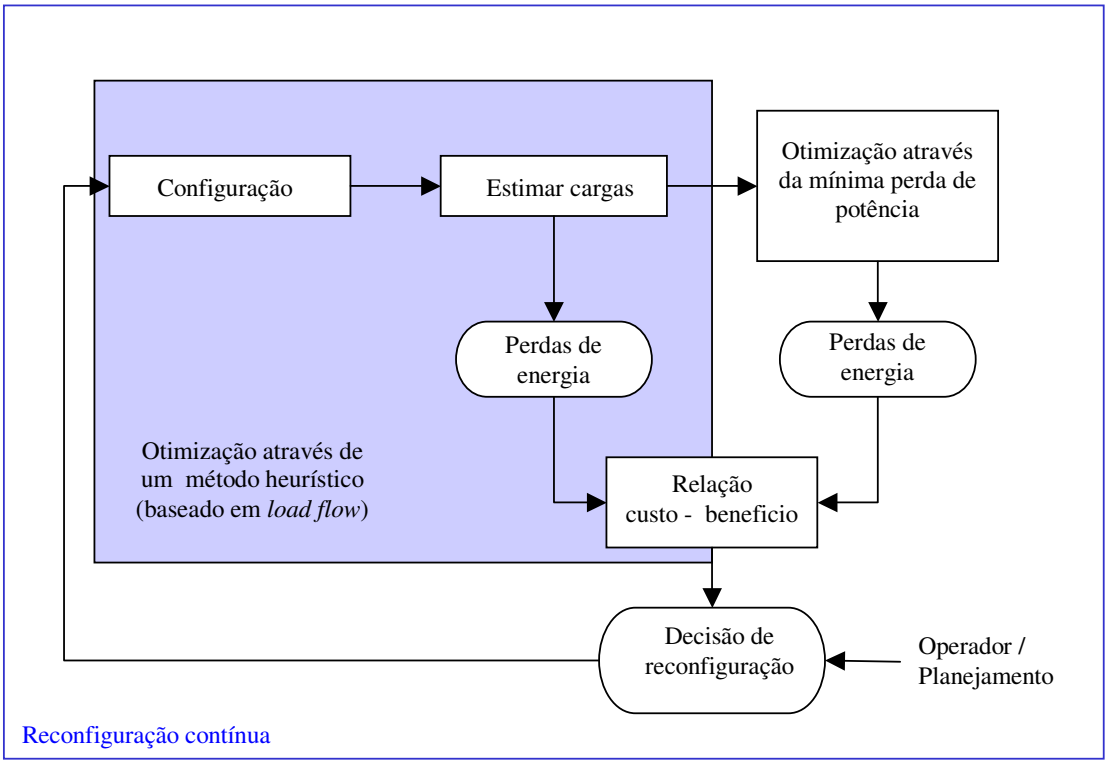


FIGURA 8 – **Diagrama da ferramenta de análise desenvolvida por Borozan e Rajakovic**

Lin *et al.* [12], em 2000, utilizaram um método com as seguintes etapas: primeiramente é calculado o *load flow*, seu resultado é usado como estado inicial para a execução do algoritmo genético até a convergência, ou maximização da função redução das perdas elétricas, e novamente a solução é analisada pelo *load flow*.

Carpaneto e Chicco [6], em 2001, utilizaram *simulated annealing*, em seu estudo para a resolução do problema de otimização com restrições. Para a formulação do modelo matemático, todas as restrições foram incluídas na função com o uso de fatores de penalidade. Assim, é utilizada uma única função objetivo com penalidades, sujeita a apenas configurações radiais. Entretanto foram verificadas dificuldades na escolha dos parâmetros do algoritmo para o alcance do mínimo global, e foi verificada a necessidade de um grande tempo de processamento computacional.

Em 2003, Carnieri *et al.* [5] aplicaram um método heurístico de dois estágios utilizando a regra do mínimo fluxo, *load flow*, a metodologia usada foi baseada no

trabalho de Augugliaro *et al.* [2]. A figura 9 apresenta o fluxograma das etapas para a solução do problema de reconfiguração que objetiva a obtenção de uma configuração otimizada do sistema de distribuição elétrica, a qual foi apresentada à concessionária COPEL, que atua no estado do Paraná.

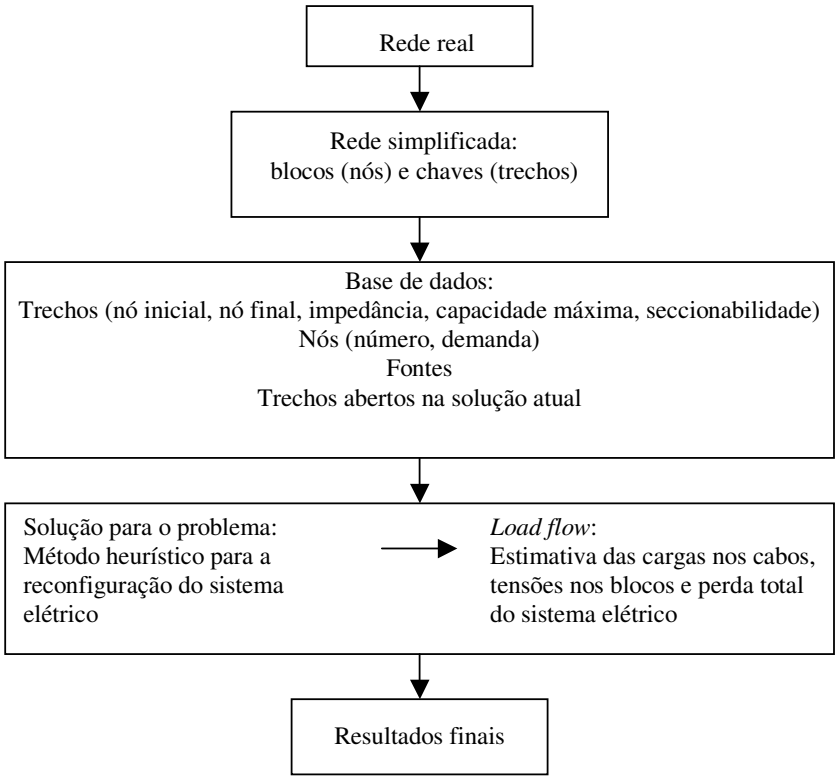


FIGURA 9 – **Fluxograma da solução do problema de reconfiguração de uma rede de distribuição elétrica apresentada por Carnieri *et al.***

Volpi *et al.* [20], em 2003, utilizaram programação binária na minimização de perdas de energia elétrica, baseando-se no trabalho de Sarma e Rao [17]. Em seu desenvolvimento a perda total do sistema foi considerada como o somatório das perdas de todos os trechos e as restrições incluíram radialidade e possíveis problemas de *loops* entre alimentadores e interno a eles.

2.2 *SIMULATED ANNEALING* (SA)

S*imulated annealing* (SA) é uma técnica heurística utilizada em problemas de otimização combinatória. SA é como um algoritmo de descida (procura por um mínimo), que inicia com uma solução inicial, gerando outras soluções vizinhas, e calcula os custos de todas elas, se o custo for o menor, então esta nova solução é aceita, caso contrário a solução é descartada. O processo é repetido até que não hajam melhorias. Às vezes, o ótimo obtido é local (não global) e uma forma de evitar isto é executar a técnica utilizando diversas soluções iniciais diferentes, e a solução final escolhida será a melhor dentre as soluções mínimas locais obtidas.

A técnica s*imulated annealing* permite a aceitação de uma configuração que forneça um "pior" valor para a função objetivo evitando, assim, a convergência para um mínimo local. Essa aceitação é determinada por um número aleatório e é controlada através de uma probabilidade.

2.2.1 Analogia ao processo físico

Quando um metal é aquecido até seu ponto de fusão, sua energia interna é alta e assim suas moléculas se movem rapidamente. Quando a temperatura é reduzida, as moléculas vão gradativamente diminuindo sua velocidade de movimento, à medida em que a energia interna também diminui. Assim, próximo ao ponto de congelamento, o metal se torna sólido, e o estado final das moléculas do metal são determinadas pelos seus comportamentos ou pela velocidade de resfriamento. O metal pode resultar em uma forma amorfa, sem uma forma definida como o vidro ou como um cristal com muitos defeitos em sua estrutura, quando o resfriamento for realizado de forma rápida, o que chamamos de processo *quenching* (esfriamento rápido). Ou, ainda, pode resultar em um cristal, onde todas as suas moléculas estão alinhadas e correspondem a uma configuração de mínima energia do sistema, quando o resfriamento é executado lentamente, chamamos de processo *annealing* (recozimento para uma recristalização).

Analogamente a esse processo natural onde pode-se chegar à menor energia interna de sólidos, pode-se trabalhar com problemas de otimização combinatória. Os

diferentes estados do sólido correspondem as diferentes soluções do problema, e a energia do sistema corresponde a função objetivo a ser minimizada.

De acordo com Eglese [10], em 1953, Metropolis *et al.* apresentou um algoritmo que simula o processo de um grupo de átomos a uma dada temperatura, a procura do equilíbrio térmico. A cada iteração é calculada a diferença, δ, da energia do sistema, onde um átomo é aleatoriamente substituído. Se $\delta<0$, a substituição é aceita, porém, se $\delta>0$, a mudança é aceita com a probabilidade

$$e^{-\delta/k\times T},$$

onde *T* é a temperatura e *k* uma constante física chamada de constante de Boltzmann. Aplicando-se o algoritmo de Metropolis várias vezes, a cada temperatura são executadas várias iterações, e o sistema encontra o equilíbrio térmico para cada temperatura.

Assim, os diferentes estados do metal correspondem as diferentes soluções viáveis de um problema de otimização combinatória, sendo que a energia do sistema corresponde a função objetivo a ser minimizada. Segundo Nara [16], o primeiro a utilizar este algoritmo para otimização foi Kirkpatrik, em 1983, que denotou-o algoritmo *simulated annealing*.

No quadro é apresentado o relacionamento entre o processo físico e o processo de otimização funcional.

Relações entre o processo físico e o processo de otimização funcional

Processo Físico	Processo de Otimização
Estado	Solução
Energia	Custo ou Função Objetivo
Estado de Transição	Soluções Vizinhas
Temperatura	Parâmetro de Controle
Ponto de Congelamento	Solução Heurística

Segundo Barbosa [3], ao iniciar com uma temperatura relativamente alta, e com decréscimo gradual, espera-se que ao final do processo, o sistema estacione em um estado de energia globalmente mínima, por analogia com a física. Através

dessa aplicação, pode-se ignorar a constante de Boltzmann, e reescrever a probabilidade de aceitação de uma configuração de energia mais alta como $e^{-\delta/T}$.

Uma característica importante de s*imulated annealing* é essa aceitação de configurações que apresentam maior energia, que pode parecer pior, porém permite que o método não convirja para um mínimo local, podendo convergir para um melhor resultado (talvez o mínimo global).

Por exemplo, na figura 10, encontra-se o gráfico de uma função f(x). Suponha que deseja-se determinar o mínimo global B de f(x) a partir do ponto x_0. *Simulated annealing* pode convergir de x_o para x_1 e depois de x_1 para B (mínimo global). Entretanto é possível que o processo de otimização convirja para x_2 e, assim, para A.

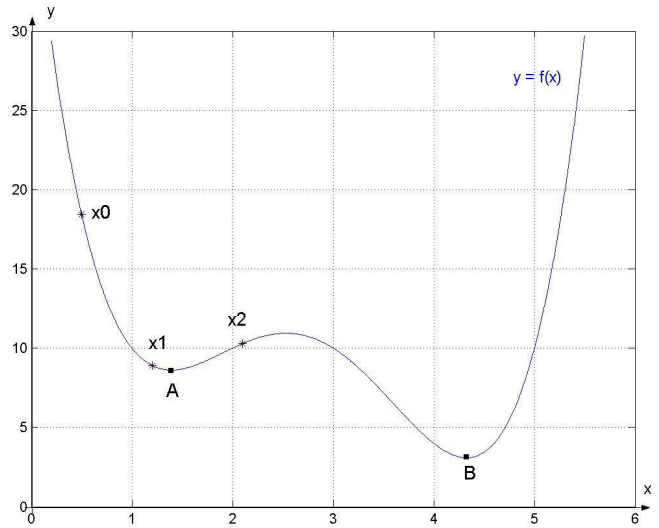


FIGURA 10 **– Característica de busca do s*imulated annealing***

FONTE: NARA [16]

Ao começar com uma temperatura *T* alta, é aceita qualquer tipo de configuração e, a medida que o valor da temperatura decresce, as configurações que possuem uma maior energia tem diminuída a sua probabilidade de aceitação. Assim o processo, inicialmente, trabalha com uma enorme aceitação sem limitar as configurações. Portanto não tendendo caminhar apenas para um mínimo local.

A temperatura *T* diminui segundo a equação $T^{k+1} = \rho\ T^k$, onde ρ assume valores de 0,80 a 0,99, até alcançar o ponto de congelamento, quando o algoritmo

pára. Este ponto de congelamento não necessariamente implica na temperatura zero, podendo ser considerada uma temperatura muito baixa pré-determinada.

2.2.2 Algoritmo propriamente dito

O algoritmo simplificado do s*imulated annealing* é o seguinte:

inicio (*i*,energia(*i*),*T,* α)
  enquanto (critério de parada)
    para *k*=1:*L* (nº de iterações por nível de temperatura)
      gerar vizinho(*j* de *i*)
      calcular energia(*j*)
      se (energia(*j*)<energia(*i*))
        então *i* =*j*
        energia*(i*)= energia*(j)*
        senão
          calcular p = $e^{[energia(i)-energia(j)]/T}$
          se (p>*rand*(0,1)[1])
            então *i* =*j*
            energia*(i*)= energia*(j)*
            senão rejeitar *j*
      fim {se}
    fim {para}
    $T = \alpha.T$
  fim {enquanto}
fim

A figura 11 mostra um fluxograma do algoritmo *simulated annealing*.

[1] um número aleatório uniformemente distribuído entre 0 e 1.

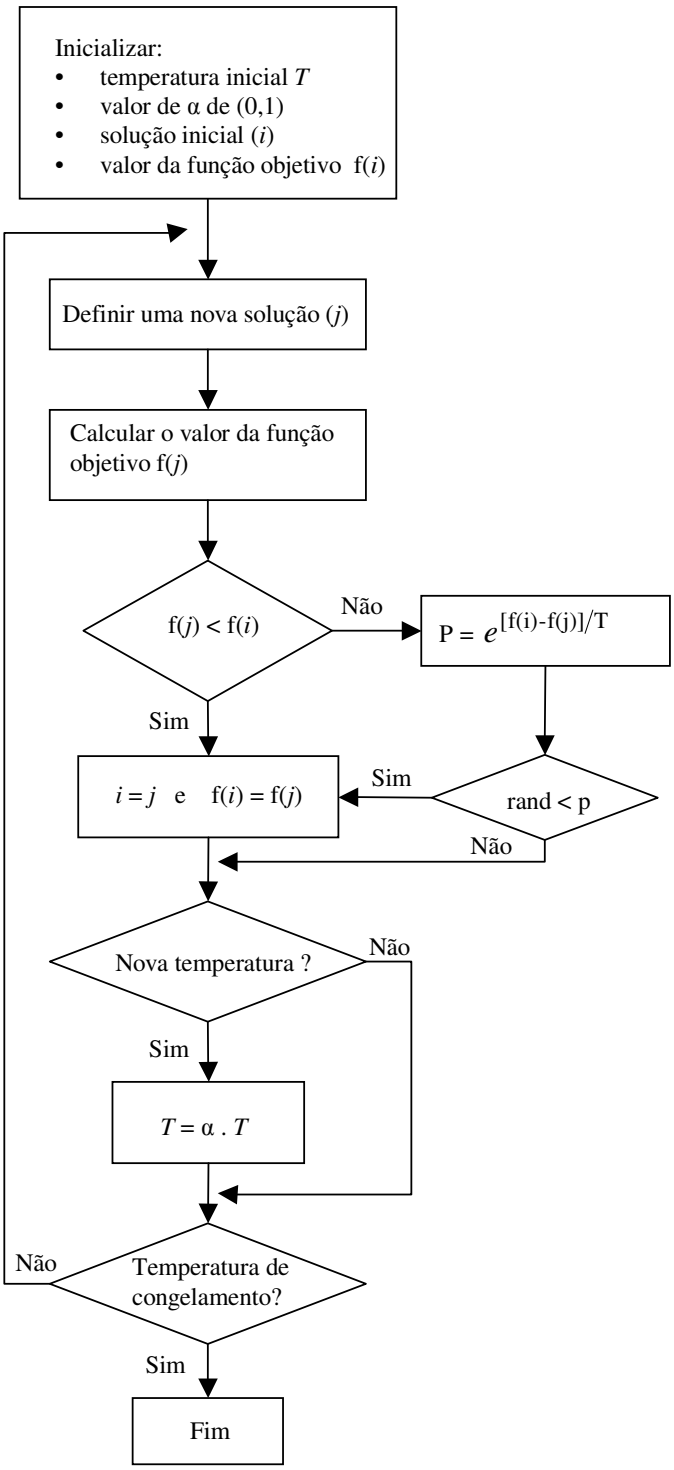


FIGURA 11 – **Fluxograma do algoritmo *simulated annealing***

Para o uso do *simulated annealing* no problema da reconfiguração de rede de distribuição elétrica são necessários alguns dados iniciais:

- trechos existentes
- trechos abertos
- resistência dos trechos
- carga nos nós
- nós fonte (subestações)

Algoritmo:

Passo1: Estima-se uma temperatura inicial *T*, permitindo um espaço de busca, de novas soluções, sem limitações.
Através da configuração original obtém-se a energia(*i*) inicial.

Passo2: Enquanto não for alcançado o critério de parada, vá para o Passo3.
Alcançado o critério de parada, vá para o Passo8. O critério usado no trabalho será detalhado na próxima seção.

Passo3: Para *k* de 1 a *L* (número de iterações por nível de temperatura).
Para que, de forma gradativa, haja o esfriamento do material até seu estado final de 'congelamento', ou seja, haja uma aumento na restrição da variação da pesquisa de uma nova solução, o valor da variável L é calculada por:
L = número máximo de iterações / número de trechos abertos
Após L iterações, vá para o Passo7.

Passo4: Gerar nova configuração de forma aleatória (*j*), sendo uma configuração factível, radial, não permitindo fechar o circuito ou interligar duas subestações.
Calcular energia(*j*).

Passo5: Se (energia(*j*)<energia(*i*)). Então *i=j*, vá para o Passo3.
Caso contrário calcular p = $e^{[\text{energia(i)-energia(j)]}/T}$.

Passo6: Se (p>*rand*(0,1)) , onde *rand*(0,1) é um número aleatório uniformemente distribuído entre 0 e 1.
Então *i=j*, vá para o Passo3.
Caso contrário rejeitar *j*, vá para o Passo3.

Passo7: Diminua o valor de *T*, de forma a diminuir o espaço de busca das novas soluções. Vá para o Passo2.

Passo8: Pare. A solução (*i*) é a melhor configuração da rede.

2.3 ALGORITMO GENÉTICO (AG)

O algoritmo genético (AG), segundo Mitchell [14], foi inventado por John Holland, nos anos sessenta. Seu objetivo foi estudar os fenômenos de desenvolvimento e adaptação à natureza. Holland apresentou, no seu livro *Adaptation in Natural and Artificial Systems*, em 1975, o algoritmo genético e sua estrutura teórica.

O AG é um algoritmo de busca baseada na evolução natural e na genética. O segredo da adaptação e sobrevivência das reproduções é um exemplo das pesquisas do estudo biológico, onde os mais fortes sobrevivem e os mais fracos são eliminados. Fazendo com que a população evolua de forma a melhorar em função de uma determinada característica dentro do meio em que vive.

2.3.1 Representação cromossômica e inicialização

Cada solução possível, dentro de um espaço de busca, é representada como uma seqüência de elementos, onde cada elemento é chamado de gene, e cada uma dessas seqüências formadas pelos genes são os cromossomos, ou também chamados de indivíduos, deste modo cada indivíduo é formado por um único cromossomo. E, cada indivíduo distinto codificado por uma seqüência diferente de genes.

Na utilização do algoritmo genético, primeiramente, define-se um grupo de soluções factíveis, que denomina-se população inicial, formada por um grupo de indivíduos, ou cromossomos, distintos.

Neste trabalho, a codificação do indivíduo foi realizada de forma que cada gene representa um trecho aberto (figura 12), portanto os elementos de cada cromossomo representam as chaves abertas de uma configuração do sistema de distribuição de energia. Desta forma, a codificação dos elementos de cada indivíduo,

desse problema, não se altera com a diferença de ordem. Porém apresentam distinção através da diferença dos genes.

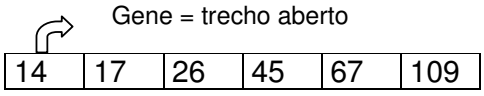


Figura 12 – **Representação de um indivíduo**

A população inicial é representada por um grupo de diferentes configurações viáveis para o sistema em estudo. Estes indivíduos, normalmente, são gerados de forma aleatória ou através de algum processo heurístico.

A população formada pelas configurações candidatas a solução do problema de otimização deve ser ordenada, do melhor ao pior indivíduo, de modo que, em um problema de minimização de uma função, o indivíduo que apresentar menor valor para a função objetivo é considerado o melhor indivíduo, e tem a maior possibilidade de sobrevivência no ambiente. Portanto este ocupa a primeira posição na ordenação.

O valor da função objetivo de cada indivíduo é denominado de *fitness*, que pode ser entendido como adequabilidade ao ambiente. No presente problema de minimização, a finalidade é obter a solução que apresente o menor *fitness*, portanto o indivíduo que apresenta menor *fitness* é o de maior adequação ao sistema.

Desta lista de indivíduos haverá recombinações as quais formarão novos indivíduos transmitindo parte de seu material genético às gerações futuras. Essas novas gerações serão analisadas de modo que a população seja sempre formada pelos indivíduos que apresentem melhor *fitness*, ou sejam mais adequados ao sistema.

2.3.2 Seleção

Para o processo de recombinação do algoritmo genético é realizada uma seleção dentre o grupo de indivíduos pertencentes a população. Assim, através de dois indivíduos selecionados, faz-se uma troca genética para geração de novos indivíduos.

A seleção dos dois indivíduos que efetuarão o processo de recombinação é executada através da fórmula proposta por Mayerle, de acordo com Corrêa *et al.* [9], dada por:

$$\text{Indivíduo selecionado} = m+1-\left\lfloor \frac{-1+\sqrt{1+4\times rand\times(m^2+m)}}{2} \right\rfloor,$$

onde: *m* é o número de indivíduos da população, *rand* é um número aleatório uniformemente distribuído entre 0 e 1, e o símbolo $\lfloor b \rfloor$ é o maior número inteiro menor do que b. Por exemplo, se m = 5 e rand = 0,5, tem-se:

$$\text{Indivíduo selecionado:} \quad 5+1-\left\lfloor \frac{-1+\sqrt{1+4\times 0{,}5\times(5^2+5)}}{2} \right\rfloor$$

$$= 5+1-\lfloor 3{,}4051 \rfloor$$

$$= 5+1-3$$

= 3 (terceiro indivíduo)

2.3.3 *Crossover* ou Recombinação

O *crossover* é o processo de reprodução realizada pelos indivíduos selecionados pelo processo anterior, descrito em 2.3.2. Neste processo há uma troca de genes entre os dois indivíduos selecionados, que são denominados pais, representando um processo sexuado, formando novos indivíduos que são chamados filhos.

Neste estudo, os elementos de um indivíduo representam os trechos que podem ser abertos da rede.

Após a seleção dos pais, a troca dos seus elementos realiza-se de forma aleatória, onde os elementos são escolhidos aleatoriamente para a formação dos filhos, como é ilustrado na figura 13, onde a esquerda temos os pais e a direita os filhos.

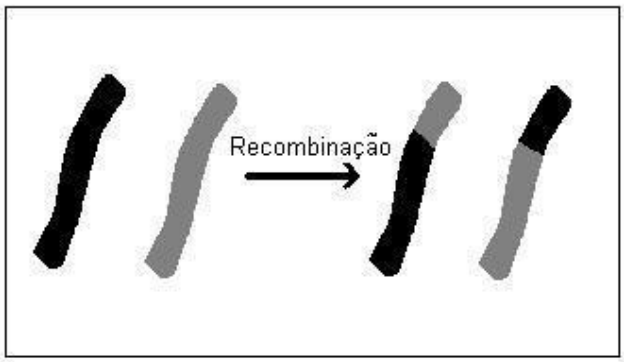


FIGURA 13 – **Processo de recombinação**
FONTE: TANOMARU [19]

2.3.4 Mutação

A mutação é um processo de busca aleatória que neste estudo é realizado apenas quando os filhos obtidos no processo de recombinação não forem viáveis. Escolhe-se aleatoriamente um dos filhos e realiza-se uma troca de elementos por outros possíveis, como demonstra a figura 14. Deste modo, forma-se um novo indivíduo viável que será analisado quanto à adequabilidade ao ambiente, podendo ser aceito ou não no grupo, ou na população, que continuarão a fornecer o material genético às gerações futuras.

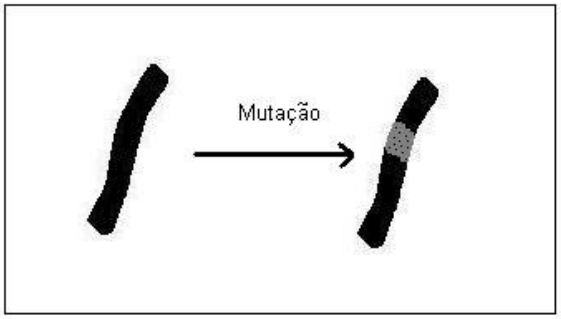


FIGURA 14 – **Processo de mutação**
FONTE: TANOMARU [19]

2.3.5 Algoritmo propriamente dito

O algoritmo simplificado do algoritmo genético é o seguinte:

início

gerar população inicial, composta por *m* indivíduos
calcular *fitness* dos indivíduos da população
ordenar os indivíduos, do menor para o de maior *fitness*
enquanto (critério de parada)
seleção
crossover e/ou mutação, para gerar o(s) filho(s)
calcular *fitness* do(s) filho(s)
se *fitness*(filho(s)) < *fitness*(m-ésimo indivíduo)
então filho(s) são inseridos na população
reordenar os indivíduos da população
rejeitar o(s) último(s) indivíduos da população,
para que permaneça com *m* indivíduos
senão rejeitar filho(s)
fim {se}
fim {enquanto}

fim

A figura 15 mostra a representação do algoritmo genético através de um fluxograma.

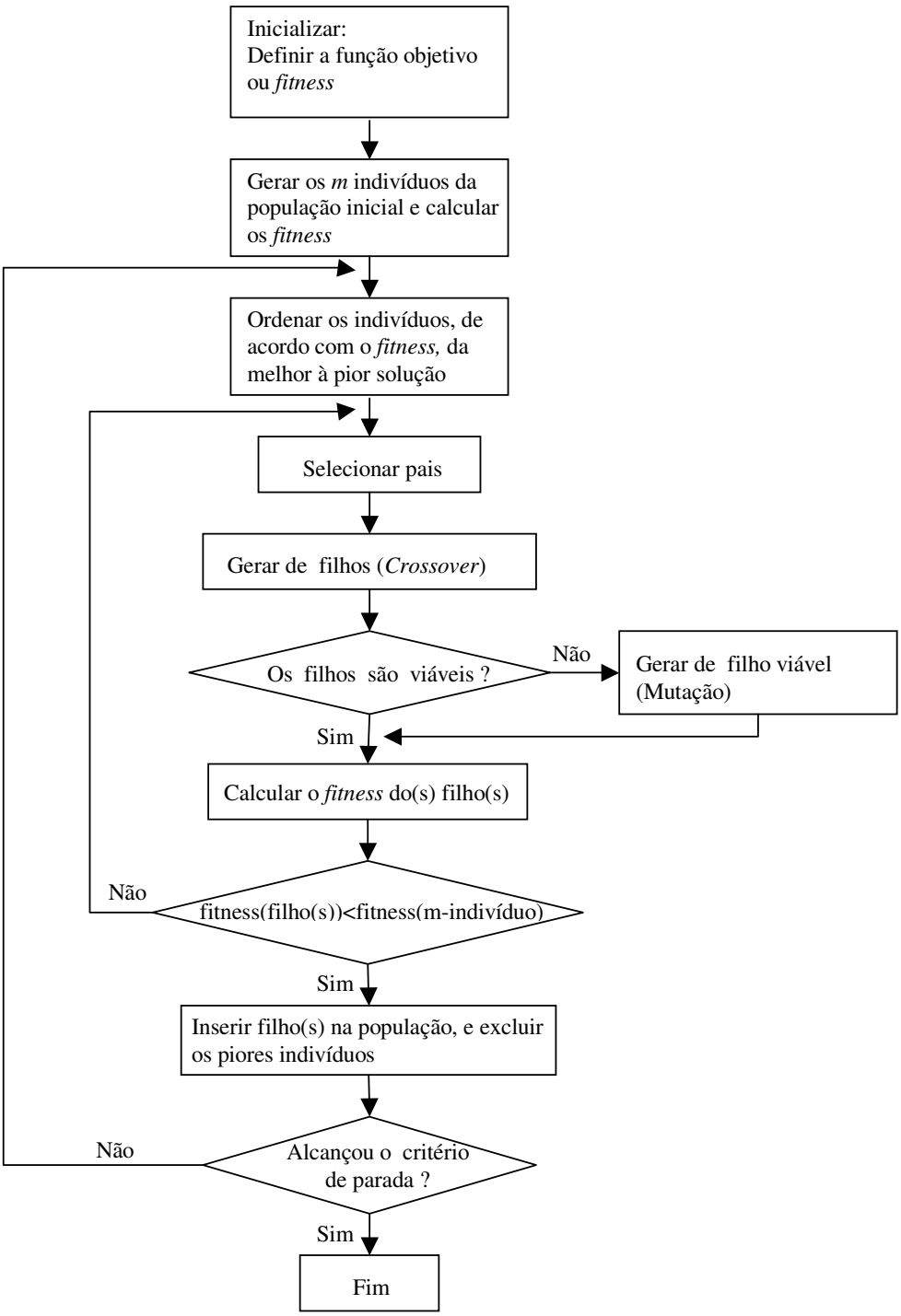


FIGURA 15 – **Fluxograma do algoritmo genético**

No uso do algoritmo genético para reconfiguração da rede de distribuição de energia elétrica são necessários os dados iniciais, abaixo citados. A população é

representada pelas possíveis configurações para o sistema, e os processos de *crossover* e mutação são executados para a criação de novas possíveis configurações da rede.

Dados iniciais:

- trechos existentes
- trechos abertos
- resistência dos trechos
- carga nos nós
- nós fonte (subestações)

Algoritmo:

Passo1: Inicialização

Gerar população inicial, aleatoriamente, formada por *m* indivíduos viáveis.

Calcular o *fitness* de cada indivíduo da população inicial.

Ordenar a lista da população, iniciando pelo indivíduo de menor *fitness* para o maior.

Passo2: Enquanto não for alcançado o critério de parada, vá para o Passo3.

Alcançado o critério de parada, vá para o Passo8. O critério usado no trabalho será detalhado na próxima seção.

Passo3: Seleção

Selecionar dois indivíduos distintos dentre os indivíduos que formam a população inicial, através da seleção:

Selecionado = $m+1-\left\lfloor \frac{-1+\sqrt{1+4\times rand\times(m^2+m)}}{2} \right\rfloor$, onde: *m* é o número de indivíduos da população, *rand* é um número aleatório uniformemente distribuído entre 0 e 1 e o símbolo $\lfloor b \rfloor$ é o maior número inteiro menor do que ou igual a b.

Passo4: *Crossover*

Gerar novos indivíduos, onde os dois previamente selecionados são os pais e, de forma aleatória, faz-se uma troca de elementos entre eles formando assim novos indivíduos, os filhos.

Passo5: Testar a viabilidade dos filhos gerados no Passo4.
Se, no mínimo, um dos filhos for viável, vá ao Passo7.
Caso contrário, vá ao Passo6.

Passo6: Mutação
Escolher aleatoriamente um dos filhos, gerados no Passo4.
Fazer a mutação, formando um novo indivíduo, um novo filho.
Verificar a viabilidade deste filho.
Se o filho for factível, vá ao Passo7.
Senão, vá ao Passo3.

Passo7: Calcular o *fitness* do(s) filho(s) gerado(s).
Se *fitness*(filho) < *fitness*(m-ésimo indivíduo), eliminar o pior indivíduo (m-ésimo) e inserir o filho na população. Em seguida, reordenar os indivíduos da população, a partir daquele que apresenta menor *fitness* até o de maior.
Vá para o Passo2.
Caso contrário, rejeitar o filho e vá para o Passo3.

Passo8: Parar. A população é o grupo das m melhores soluções para configuração da rede. Sendo o primeiro indivíduo, referente a configuração que apresenta o menor valor para a função objetivo.

2.4 EXEMPLO DO USO DE *SIMULATED ANNEALING* E ALGORITMO GENÉTICO

Dado o exemplo de rede de distribuição de energia elétrica composta por três circuitos (figura 16), uma configuração que apresenta menor perda de energia elétrica será calculada utilizando as duas técnicas: *simulated annealing* e algoritmo genético.

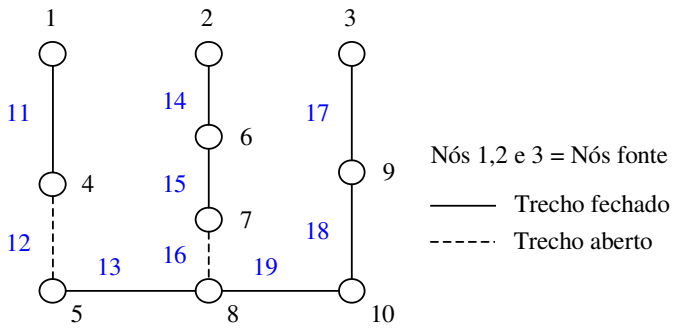


FIGURA 16 – **Configuração inicial da rede exemplo**

Os dados do exemplo encontram-se abaixo. A matriz D de demanda dos nós, tem em sua primeira coluna o número do nó e na segunda a demanda do nó, em ampère. A matriz R de resistência dos trechos, na sua primeira coluna mostra a nomenclatura do trecho e na segunda a respectiva resistência, em ohms. A matriz F das fontes, enumera os nós que são as subestações da rede de distribuição. O vetor x da solução inicial mostra os atuais trechos abertos da rede que apresenta a perda elétrica inicial perda(x), em watts, cujo calculo será detalhado na próxima seção.

- Demanda dos nós:

$$D = \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 3 & 0 \\ 4 & 50 \\ 5 & 10 \\ 6 & 50 \\ 7 & 20 \\ 8 & 20 \\ 9 & 50 \\ 10 & 30 \end{bmatrix}$$

Resistência dos trechos:

$$R = \begin{bmatrix} 11 & 0{,}14 \\ 12 & 0{,}13 \\ 13 & 0{,}13 \\ 14 & 0{,}14 \\ 15 & 0{,}13 \\ 16 & 0{,}13 \\ 17 & 0{,}14 \\ 18 & 0{,}14 \\ 19 & 0{,}13 \end{bmatrix}$$

- Nós fonte: F = [1 2 3]
- Trechos abertos, solução inicial: x = [12 16] , perda(x) = 48,4 W

➢ **Aplicando *simulated annealing*:**

- Temperatura inicial: $T = 1$ e parâmetro: $\alpha = 0{,}5$
  Temperatura de congelamento: $T<0{,}2$ (enquanto $T>0{,}2$ executar).
  A cada temperatura executar duas iterações (L=2)

Para L=1: x_1= [11 13] foi gerado aleatoriamente, solução não viável, pois gera desconexidade da rede em que os nós 4 e 5 não são alimentados. Além da rede apresentar-se não radial.

Para L=2: x_1= [13 19] foi gerado aleatoriamente, solução viável, energia(x_1)= 45,5W.

Como energia(x_1)<energia(x), onde energia(x) = perda(x), então: x = [13 19] e energia(x) = 45,5 W

$T = T.\ \alpha = 1.\ 0,5 = 0,5$

- Temperatura: T= 0,5 > 0,2

Para L=1: x_1 = [18 19] foi gerado aleatoriamente, solução não viável, que torna o nó 10 não alimentado, portanto uma rede desconexa. Além da rede apresentar-se não radial.

Para L=2: x_1= [13 16] foi gerado aleatoriamente, solução viável, energia(x_1)= 45,7W.

Como energia(x_1)>energia(x), calcular p.

Calcula p = exp(energia(x_1) - energia(x) / T) = exp(45,7 – 45,5 / 0,5) = 0,67

Supondo o número aleatório rand = 0,5

Como p>rand, então x = x_1 = [13 16] e energia(x) = 45,7W.

$T=T.\ \alpha = 0,5.\ 0,5 = 0,25$

- Temperatura: T= 0,25 > 0,2

Para L=1: x_1 = [15 16] foi gerado aleatoriamente, solução não viável, pois a rede torna-se não radial e desconexa, não alimentando o nó 7.

Para L=2: x_1= [19 16] foi gerado aleatoriamente, solução viável, energia(x_1)= 45,5W.

Como energia(x_1)<energia(x), então x = [19 16] e energia(x) = 45,5W.

$T = T.\ \alpha = 0,25.\ 0,5 = 0,125$

- Temperatura: T= 0,125 < 0,2 . Parar

Solução final: configuração de menor perda: x = [19 16] e perda = 45,5W, como mostra a figura 17.

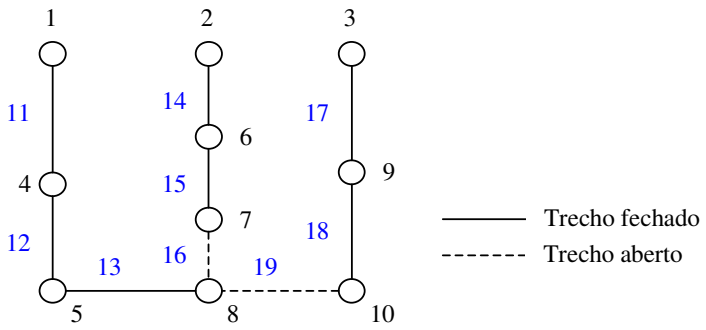


FIGURA 17 – **Configuração final da rede exemplo obtida pelo *simulated annealing***

➢ **Aplicando algoritmo genético:**

- A população inicial forma-se adotando m=2. Assim, esta ficará costituída por 2 indivíduos ou duas configurações. O primeiro indivíduo é a configuração atual da rede; o segundo é uma configuração viável gerada aleatoriamente.

  População: [12 16] indivíduo 1

  [13 19] indivíduo 2

  - Cálculo do *fitness* (função objetivo) dos indivíduos.

  Indivíduo 1: *fitness* = 48,4W

  Indivíduo 2: *fitness* = 45,5W

- Ordenar os indivíduos da população, iniciando pelo que apresenta menor *fitness* para o maior.

  População: [13 19] indivíduo 1 (*fitness* = 45,5W)

  [12 16] indivíduo 2 (*fitness* = 48,4W)

- Seleção de dois indivíduos, denominados pais, para gerar novas configurações, filhos. Neste caso, os dois indivíduos da população.
- *Crossover* ou recombinação

  Pais: [13 | 19] ⇕ [12 | 16] ⟹ [13 16] filho 1, [12 19] filho 2

- Cálculo do *fitness* dos filhos:

  Fitness: Filho 1 = 45,7W

  Filho 2 = 48,1W

- Como *fitness*(filhos) < *fitness*(indivíduo 2), então incluir os filhos na população: [13 19]
  [12 16]
  [13 16]
  [12 19]
- Ordenar população: [13 19] (*fitness* = 45,5W)
  [13 16] (*fitness* = 45,7W)
  [12 19] (*fitness* = 48,1W)
  [12 16] (*fitness* = 48,4W)

  - Permanecer com m=2 indivíduos (melhores) na população: [13 19]
    [13 16]
  - Seleção dos pais: [13 19]
    [13 16]
  - *Crossover*:

Pais: [13 | 19] ⇕ [13 | 16] ⟹ Filhos: [13 16] [13 19]

Os filhos gerados, já existem. Portanto não são considerados como novas soluções. Logo faz-se a mutação.

- Mutação:

  Seleciona-se, aleatoriamente, um dos filhos recém gerados e não considerados: [13 19]

  Em seguida gera-se uma nova configuração, escolhendo aleatoriamente o elemento a ser trocado por um outro qualquer que forme uma solução viável. Trocar o elemento 13 pelo 16, assim temos: [16 19]
- Cálculo do *fitness* da nova configuração: fitness = 45,5W
- Como fitness(novo indivíduo) < fitness(indivíduo 2) , então incluir os filhos na população: [13 19]
  [13 16]
  [16 19]
- Ordenar a população: [13 19] (*fitness* = 45,5W)

[16 19] (*fitness* = 45,5W)

[13 16] (*fitness* = 45,7W)

- Permanecer com os *m* melhores indivíduos na população: [13 19]
  [16 19]
- Suponha alcançado o critério de parada. A população final apresenta o grupo de melhores soluções para o problema da minimização de perdas de energia elétrica.

Solução final: será a configuração [13 19], primeira solução da população, mostrada na figura 18.

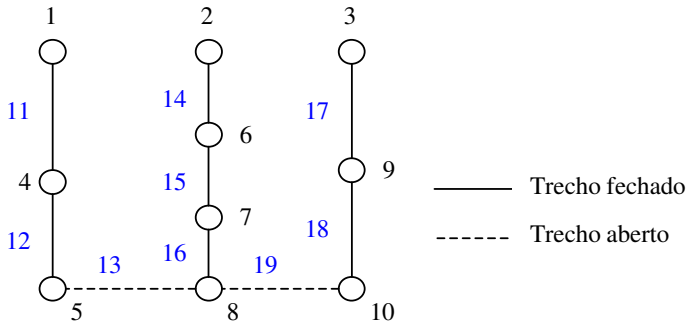


FIGURA 18 – **Configuração final da rede exemplo obtida pelo algoritmo genético**

CAPÍTULO 3

3 MÉTODOS HÍBRIDOS PROPOSTOS E APLICAÇÕES

Os métodos híbridos propostos são processos de otimização que utilizam ambas as técnicas *Simulated Annealing* (SA) e Algoritmo Genético (AG) para fornecer a melhor configuração do problema de minimização de perdas elétricas do sistema de distribuição de energia.

O método híbrido AG/SA faz o uso inicial do algoritmo genético e depois utiliza *simulated annealing*. Nessa técnica, na procura da solução ótima, após a obtenção de uma solução final através do algoritmo genético executa-se o *simulated annealing*.

O método híbrido SA/AG utiliza, inicialmente o *simulated annealing* seguido do algoritmo genético. Nesse processo há a geração da população inicial necessária para executar o algoritmo genético através de uma primeira otimização executada pelo *simulated annealing*.

A utilização de métodos híbridos podem proporcionar uma melhora na busca de uma solução de um problema de otimização. Neste estudo, foi analisada a possibilidade de melhorar a busca de uma nova configuração que apresente menor perda elétrica pelo uso conjunto das duas técnicas. Os métodos híbridos propostos podem atuar de forma a minimizar a perda apresentada pela solução obtida por uma técnica de otimização, através de um segundo processo de minimização por outra técnica.

3.1 MÉTODO HÍBRIDO AG/SA

No método híbrido AG/SA, primeiramente é executado o algoritmo genético, gerando aleatoriamente os indivíduos que vão compor a população inicial. Após sua

execução, é obtida como solução final uma população representada por um conjunto com as melhores configurações obtidas no decorrer do processo para a rede. Dentre as soluções fornecidas é adotada a melhor solução, que apresenta menor perda elétrica para o sistema, como solução inicial para o uso do *simulated annealing*. Então é executado o *simulated annealing*, que ao seu término apresenta a solução final como a melhor configuração da rede.

A utilização dessa metodologia pode melhorar a melhor solução gerada pelo algoritmo genético, que é muito influenciada pelos indivíduos que formam a população e a tendência de gerar um novo indivíduo com elementos diferentes dos indivíduos da população é pequena. Portanto o uso de *simulated annealing* após uma primeira solução pode transformar um mínimo local em mínimo global, pelas características do método de aceitação de soluções com valores da função objetivo maiores na busca do mínimo global.

Para a execução do método AG/SA são necessários os dados abaixo, para que o algoritmo inicie o AG, e após realizar a minimização, o melhor resultado é adotado como configuração inicial para o SA ser executado.

Dados iniciais:

- trechos existentes
- trechos abertos
- resistência dos trechos
- carga nos nós
- nós fonte (subestações)

Algoritmo:

Passo1: Inicialmente, executar o programa que usa o método de algoritmo genético, descrito na seção 3.2.

Ao final de k_1 iterações, armazenar a melhor solução obtida, isto é, o indivíduo que apresentou o menor *fitness*, vá ao passo 2.

Passo2: Executar o programa *simulated annealing*, descrito na seção 3.1, utilizando como solução inicial a solução apresentada no passo 1. Após k_2 iterações, vá ao passo 3.

Passo3: Parar. A solução do passo 2 é a melhor configuração da rede.

A figura 19 apresenta o fluxograma do algoritmo do método híbrido AG/SA.

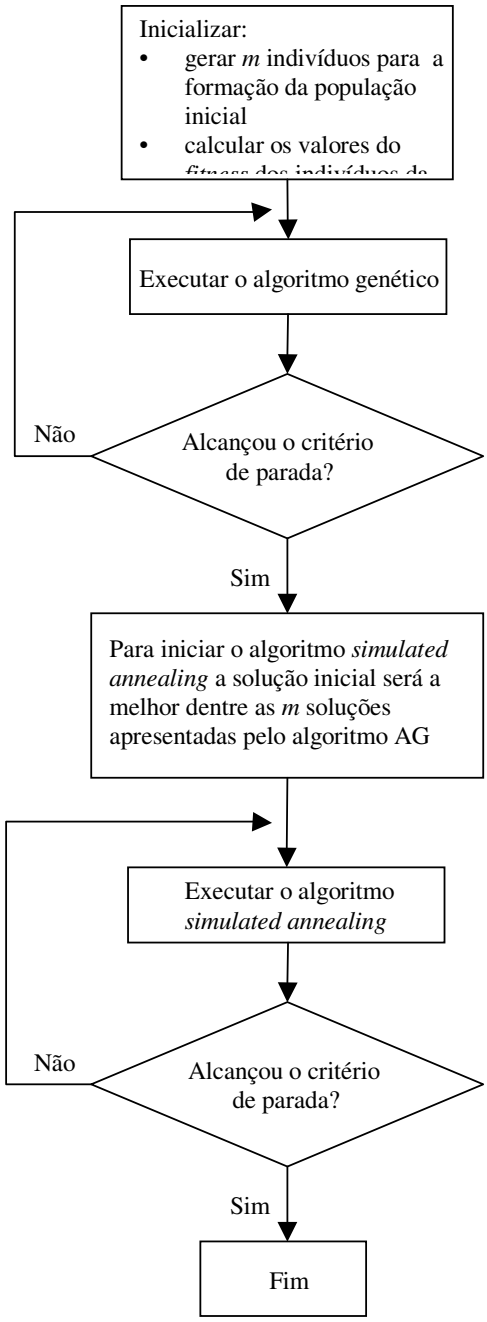


FIGURA 19 – **Fluxograma do método híbrido AG/SA**

3.2 MÉTODO HÍBRIDO SA/AG

Em situações em que não se possui várias soluções iniciais viáveis de um problema, para o uso do algoritmo genético, é possível a utilização de um método de

otimização para a formação da população inicial. Este trabalho propõe o uso de *simulated annealing* na geração dos *m* indivíduos para compor a população inicial para, então, executar a minimização através do algoritmo genético.

Nesta segunda proposta de um método híbrido, inicia-se a otimização através da configuração atual da rede. É executado o *simulated annealing* e entre os melhores resultados apresentados é gerada a população inicial, pelas *m* melhores soluções, para a utilização do algoritmo genético. O algoritmo genético é, então, executado e dentre as soluções apresentadas pela população final, o indivíduo que apresentar menor *fitness*, ou menor perda elétrica, representa a melhor configuração para o sistema.

No uso do método híbrido SA/AG, são necessários os dados iniciais abaixo listados para a execução do SA, lembrando que os trechos abertos fornecidos, correspondem a configuração atual da rede. Após o *simulated annealing*, é executado o AG que, ao seu término, apresenta a solução de menor *fitness* como a melhor configuração da rede.

Dados iniciais:

- trechos existentes
- trechos abertos
- resistência dos trechos
- carga nos nós
- nós fonte (subestações)

Algoritmo:

Passo1: A solução inicial corresponde a configuração atual da rede.
Executar o programa utilizando o algoritmo de *simulated annealing*. Quando o critério de convergência for alcançado vá para o passo 2.

Passo2: Armazenar as *m* melhores soluções obtidas no passo 1, isto é, as *m* melhores configurações que apresentam o menor valor de energia que é a função objetivo.

Passo3: Executar o programa que usa algoritmo genético, adotando como população inicial os *m* indivíduos adquiridos no passo 2. Após alcançado o critério de convergência, vá ao passo 4.

Passo4: Parar. A melhor solução encontrada no passo 3 é a melhor configuração da rede de distribuição de energia elétrica.

A figura 20 mostra através de um fluxograma do método híbrido SA/AG.

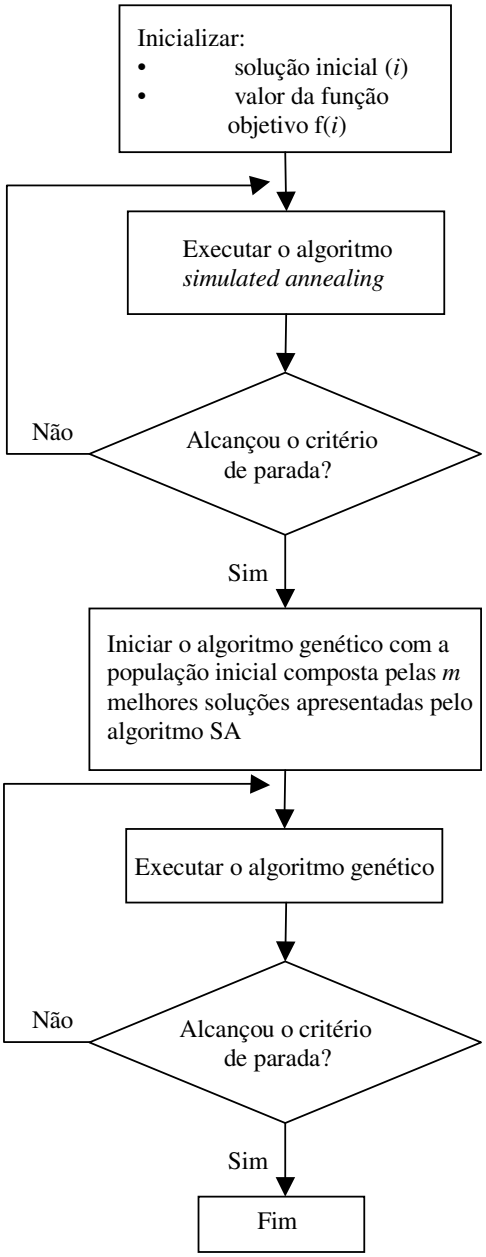


FIGURA 20 – **Fluxograma do método híbrido SA/AG**

3.3 APLICAÇÃO DOS MÉTODOS

A perda de energia elétrica total do sistema de distribuição de energia adotada neste trabalho é o somatório das perdas de potência de todos os trechos que compõem o sistema, como adotado por Sarma e Rao [17], ou seja, a função perda é dada por:

$$\text{Perda} = \sum_{i=1}^{n} r_i . i_i^{\,2}$$

sujeita às restrições de radialidade

onde: r_i é a resistência do trecho i;

i_i é a corrente do trecho i;

n é o número total de trechos que compõem a rede.

Considera-se, para uma simplificação do cálculo da função objetivo de uma rede, a não existência de impedâncias, perdas e oscilações de tensões.

Para o cálculo da função objetivo, a cada iteração, foi necessário o conhecimento detalhado da configuração da rede, ou seja, do caminho percorrido pela corrente desde a subestação (nó fonte) até os nós de demanda. Portanto, nas iterações, durante o processo de otimização, executa-se uma rotina computacional (no *Matlab*), denominada 'caminhos', que fornece a configuração da rede para possibilitar o cálculo do valor da função objetivo, pois a perda a cada trecho depende do valor total da corrente que por ele flui.

Para uma verificação da viabilidade das novas soluções analisadas foi necessário o uso de um outro programa, denominado 'anel'. Este está apresentado no apêndice B e fornece todos os anéis do sistema, ou seja, fornece todos os conjuntos dos trechos que constituem as ligações entre fontes (subestações) isto para que haja radialidade da rede.

No cálculo da função objetivo utilizou-se outra subrotina computacional denominada 'perdatotal' (também apresentada no apêndice B). Esta utiliza informações sobre resistências e correntes que passam nos trechos para calcular as perdas elétricas do sistema.

Neste trabalho aplicou-se os quatro métodos de otimização em três redes de distribuição elétrica fictícias. As resistências dos trechos e as demandas nos nós são dados nas tabelas 13, 14 e 15 no apêndice A.

Foram realizadas simulações, dos três exemplos, utilizando o método *simulated annealing*, o algoritmo genético e os dois métodos híbridos AG/SA e SA/AG, descritos nesta seção. Através das simulações foram verificados o número de iterações, as perdas e tempos de processamento.

O critério de convergência adotado nos métodos foi dado por:

$$erro = \frac{\Delta f}{f} \times 100,$$

onde Δf é a diferença entre dois valores da função objetivo (função perda) e f é um valor da função objetivo, valores esses que serão detalhados na seqüência. Este critério facilita a análise de comparações dos resultados das diferentes metodologias, mesmo que os exemplos apresentem variações significativas nos valores da função objetivo.

Adotou-se como erros máximos para a variável *erro* os valores 0,05 e 0,7. Ou seja, no decorrer do processo de otimização, quando o *erro* for menor que 0,05 ou 0,7, decisões importantes são tomadas nos métodos de otimização. Considera-se um erro de 0,05 rígido em relação a redução desejável nas perdas elétricas devido ao fato das perdas geralmente serem consideráveis. Entretanto um erro igual a 0,7 não exige tanto no processo de otimização, pois 0,7 do total das perdas numa rede extensa é uma quantidade significativa. Adotou-se estes critérios (um rígido e outro não) para verificar o comportamento dos 4 algoritmos (AG, SA, AG/SA e SA/AG) em relação ao tempo de processamento, número de iterações e o valor da função objetivo.

No cálculo do *erro*, utilizou-se duas fórmulas:

$$erro1 = \frac{\Delta f}{f} \times 100 = \frac{|f(meta) - f(x_k)|}{f(meta)} \times 100\,,$$

ou seja, o *erro* na *k*-ésima iteração envolve a diferença entre o valor da função objetivo nesta iteração e a meta traçada para a função perda. A meta para a função objetivo é uma estimativa, bastante realista, feita pelo decisor das perdas elétricas aceitáveis (mínimas possíveis),

$$erro2 = \frac{\Delta f}{f} \times 100 = \frac{\left| f\left(_{k-1}\right) - f\left(x_k\right) \right|}{f\left(x_{k-1}\right)} \times 100 \,,$$

em que o *erro* na *k*-ésima iteração envolve a diferença entre o valor da função objetivo na iteração atual e o valor da função objetivo na iteração anterior.

Estes erros têm importância fundamental no processo de otimização nos métodos híbridos pois determinam o momento em que o processo de otimização deixa de ser feito por uma técnica e passa a ser executado por outra. No caso do método AG/AS, se o *erro*1 (ou *erro*2) for menor que o máximo permitido, o processo de otimização deixa de ser executado pelo algoritmo genético (AG) e passa a ser feito através de *simulated annealing* (SA). No método híbrido SA/AG, o *erro*1 (ou *erro*2) determinam quando o processo de otimização deixar de executado por SA e passa a ser feito pelo AG. Estes erros também determinam quando o processo de otimização (como um todo) é encerrado nos quatro métodos de otimização.

É importante ressaltar que dependendo da meta estipulada pelo decisor, o *erro*1 poderá não ser alcançado e, nestas situações, a técnica de otimização é trocada (de AG para SA, ou de SA para AG) quando um número limite de iterações for alcançado.

Anteriormente, ao introduzir-se os critérios de convergência (*erros*) adotados neste trabalho, foi mencionado que estes definem o momento em que é feita a troca da técnica de otimização nos algoritmos híbridos, bem como determinam o término do processo de otimização (como um todo) nos 4 algoritmos. Em relação ao *erro*2, situação em que não se conhece a perda mínima possível, ou não se deseja atingir uma perda mínima predefinida, adotar-se-á a seguinte estratégia:

i) AG (ou SA): quando o processo de otimização for executado pelo algoritmo genético (*simulated annealing*), o processo de otimização é encerrado quando o *erro* ocorrer em *n* iterações consecutivas;

ii) AG/SA (ou SA/AG): quando o processo de otimização for executado pelo método híbrido AG/SA (SA/AG), executa-se o algoritmo genético (*simulated annealing*) até que ocorra a repetição do *erro2 r* vezes consecutivas. Em seguida é executado o método *simulated annealing* (algoritmo genético) até que o *erro*2 também se repita em *s* iterações consecutivas. Em que $r + s = n$ iterações.

Com relação ao *erro*1 considere o seguinte exemplo: supõe-se que a perda elétrica observada numa determinada rede de distribuição sejam de 100MWh e que o decisor afirma (com bastante confiabilidade) que as perdas poderiam ser de no máximo 90MWh (meta). Neste caso a redução possível é de 10MWh (10% das perdas iniciais) para alcançar a meta. O término da execução dos processos de otimização através dos algoritmos puros (AG e SA) ocorre quando *erro*1 for menor ou igual a 0,05 ou 0,7. No caso do algoritmo híbrido AG/SA (SA/AG) o processo da troca da técnica de otimização de AG para SA (SA para AG) ocorre quando a técnica AG (SA) reduzir em 20% os 10MWh. Neste caso a segunda técnica, que é a SA (AG), será encerrada quando for determinada uma configuração da rede que reduz os demais 80% dos 10MWh. É claro, que se a meta estipulada pelo decisor for irreal, a troca da técnica de otimização nos algoritmos híbridos, e o encerramento dos processos de otimização nos quatro algoritmos se dará quando um número limite de iterações for alcançado.

A escolha da proporção 20% e 80% foi adotada após análise prévia dos tempos de processamento do algoritmo AG/SA considerando as seguintes proporções:

i) 20% de redução da diferença da perda entre o observado e a perda máxima desejável com a primeira técnica e depois 80% com a segunda técnica;
ii) 30% de redução com a primeira técnica e 70% com a segunda técnica;
iii) 40% de redução com a primeira técnica e 60% com a segunda técnica;
iv) 50% de redução com a primeira técnica e 50% com a segunda técnica;
v) 60% de redução com a primeira técnica e 40% com a segunda técnica;
vi) 70% de redução com a primeira técnica e 30% com a segunda técnica;
vii) 80% de redução com a primeira técnica e 20% com a segunda técnica.

As figuras 21, 22 e 23 apresentam graficamente os tempos de processamento considerando as sete diferentes proporções e três redes elétricas (de portes) diferentes. Nestas figuras somente estão ilustrados os tempos de processamento quando o método híbrido AG/SA foi aplicado considerando o *erro*2 menor que 0,7%.

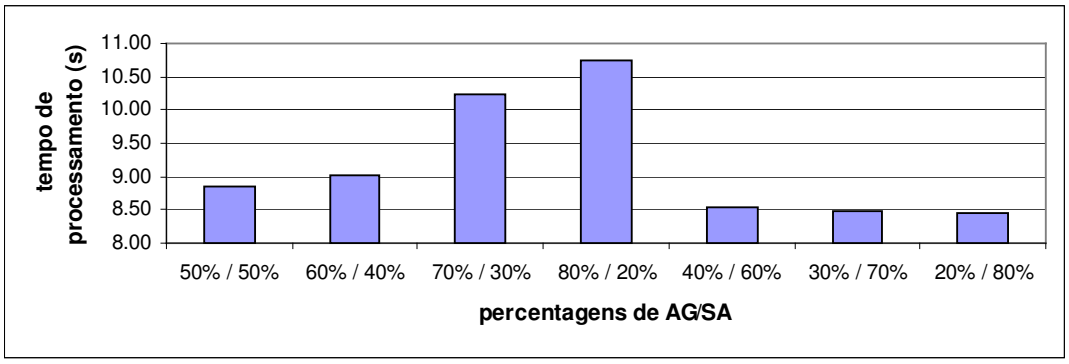


FIGURA 21 – **Tempos de processamento (rede de pequeno porte)**

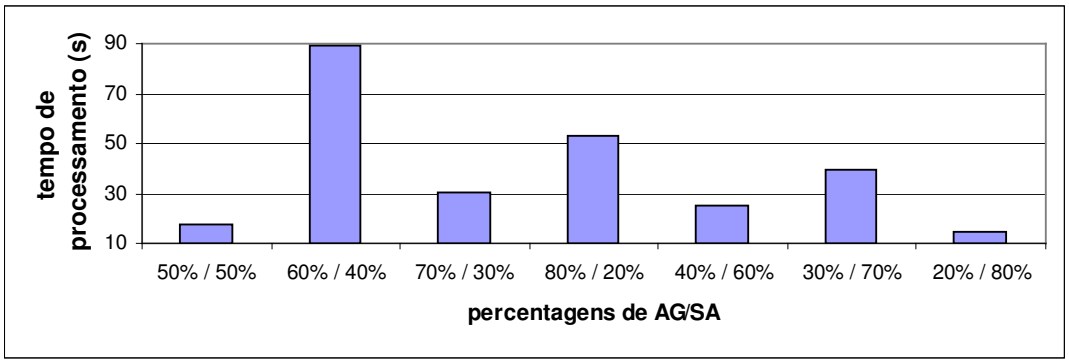


FIGURA 22 – **Tempos de processamento (rede de médio porte)**

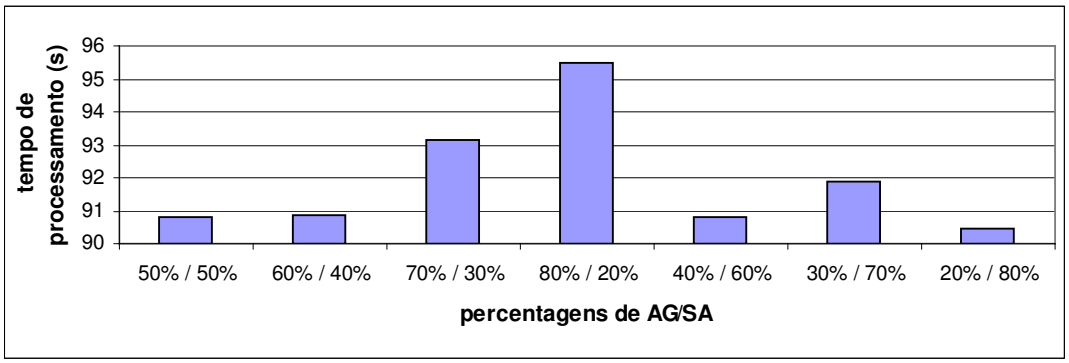


FIGURA 23 – **Tempos de processamento (rede de grande porte)**

Das figuras acima, considerando que os processos de otimização envolviam o cálculo do *erro*1 e *erro*2, e um erro máximo de 0,05, tem-se que a proporção de 20% e 80% levou a tempos de processamento significativamente menores. Esta tendência ocorreu-se para o erro máximo de 0,7.

Procedeu-se uma aplicação dos quatro métodos de otimização em um conjunto de três redes de distribuição elétrica:

i) Pequeno porte: 20 nós com demanda e 20 trechos fechados e 3 trechos abertos;

ii) Médio porte: 50 nós com demanda e 47 trechos fechados e 3 trechos abertos;

iii) Grande porte: 92 nós com demanda e 92 trechos fechados e 8 trechos abertos.

Para cada uma das redes executou-se 16 processos de otimização, dados por:

- quatro algoritmos: AG, SA, AG/SA e SA/AG,
- dois métodos de cálculo dos erros: *erro*1 e *erro*2
- dois diferentes valores máximos para *erro*1 e *erro*2: 0,05 e 0,7;

Ou seja, para uma das redes, executou-se $4 \times 2 \times 2 = 16$ algoritmos de otimização. Vale relembrar que nos algoritmos híbridos trabalhou-se com a proporção 20% e 80%.

Neste trabalho utilizou-se um computador Pentium 4, de 1,8 GHz e 262 Mb de RAM. Os programas foram executados em *Matlab* na versão 5.2.

3.3.1 Rede de pequeno porte

O primeiro exemplo conta com 23 nós, dos quais três são fontes, e 23 trechos, com três abertos. Como mostra a figura 24, na configuração inicial os trechos abertos são os que ligam o nó 6 ao 14, o nó 16 ao 20 e o nó 9 ao 23. Para esta configuração inicial o valor da função objetivo é de 247.752 watts, ou seja, a perda é de 247.752 watts. Os dados da rede encontram-se na tabela 13, do apêndice A.

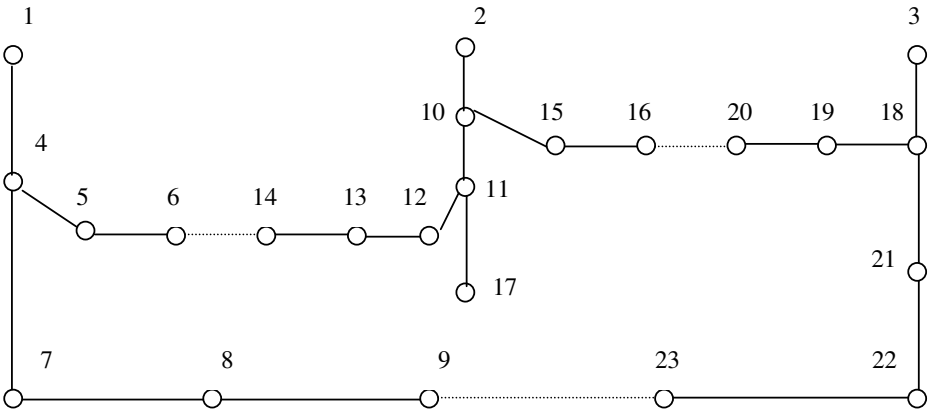


FIGURA 24 – **Configuração inicial da rede de pequeno porte**
FONTE: SARMA E RAO [16]

A nova configuração (figura 25), obtida após a otimização, que apresentou menor valor da função perda, 219.191 watts, representa 88,47% da perda inicial.

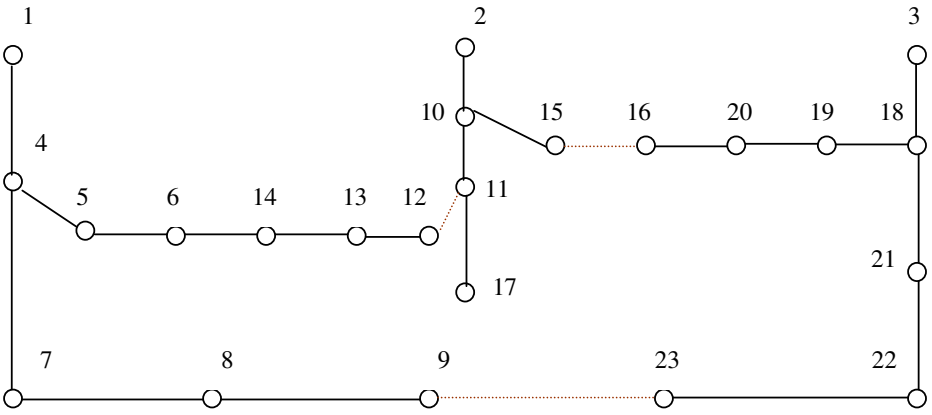


FIGURA 25 – **Configuração ótima da rede de pequeno porte**
FONTE: SARMA E RAO [16]

O processo de otimização utilizado no método híbrido AG/SA, implica que inicialmente usa-se AG e depois SA. Nos três exemplos analisados o critério utilizado para realizar a mudança de AG para SA considera conhecido *a priori* a perda inicial e a perda máxima desejável. Por exemplo, na rede pequena a perda inicial é de 247.752 watts e a perda aceitável é de 219.191 watts (que coincidentemente é a perda mínima). A diferença entre as perdas é de 28.561 watts, e neste caso o AG é executado até que reduza 20% desta diferença, e o restante da otimização é realizada pelo SA. Se casualmente algumas das técnicas não alcançarem a redução antes de um número máximo de iterações (500 para AG e

6.000 para SA) a execução da técnica é interrompida. Se o AG for interrompido continua-se a otimização através de SA, no caso de SA ser interrompido o resultado atual é fornecido como solução final. Esta solução poderá ser maior que a perda máxima permitida originalmente.

Alguns resultados obtidos pelos quatro métodos implementados são mostrados nas tabelas 1, 2, 3 e 4.

Nas tabelas, a seguir, a coluna que fornece o valor da função objetivo fornece, também, a percentagem de redução da perda obtida em relação ao valor da perda inicial.

TABELA 1 – Resultados referentes à rede de pequeno porte usando *erro*1 menor que 0,05 e meta igual a 219.191 W

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	219.191 (11,53%)	17,43	104
AG	219.191 (11,53%)	8,11	48
AG/SA	219.191 (11,53%)	23,28	151
SA/AG	219.191 (11,53%)	5,27	30

Os resultados mostrados na tabela 1 permitem observar que todos os métodos alcançaram a solução final ótima (meta), reduzindo a perda elétrica em 11,53%. O método que convergiu com menor tempo de processamento e menor número de iterações foi o SA/AG.

TABELA 2 – Resultados referentes à rede de pequeno porte usando *erro*1 menor que 0,7 e meta igual a 219.191 W

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	219.191 (11,53%)	15,78	93
AG	219.339 (11,47%)	2,19	10
AG/SA	219.191 (11,53%)	8,83	55
SA/AG	219.191 (11,53%)	5,58	34

Da tabela 2, verifica-se que o algoritmo genético levou o menor tempo computacional e menor número de iterações. Apesar destas vantagens, a solução está 0,0674% acima da meta, entretanto é uma solução consideravelmente próxima da ótima reduzindo em 11,47% da perda inicial. Os outros três métodos alcançaram a meta porém com tempos maiores de processamento.

Na tabela 3, para *erro*2 menor que 0,05, obteve-se o resultado ótimo (meta) na execução de três métodos, SA, AG e SA/AG. Embora o AG/SA foi o método que demandou menor esforço computacional e menor número de iterações.

TABELA 3 – Resultados referentes à rede de pequeno porte usando *erro*2 menor que 0,05

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	219.191 (11,53%)	7,50	44
AG	219.191 (11,53%)	7,80	46
AG/SA	219.339 (11,47%)	5,38	31
SA/AG	219.191 (11,53%)	7,63	47

Ao utilizar os critérios de convergência *erro*2 inferior a 0,7, na tabela 4, a solução final via *simulated annealing* não foi a ótima (meta), reduzindo em 10,60% as perdas iniciais. Os demais métodos atingiram a meta, e o método AG/SA demandou menor tempo computacional e menor número de iterações.

TABELA 4 – Resultados referentes à rede de pequeno porte usando *erro*2 menor que 0,7

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	221.493 (10,60%)	7,95	47
AG	219.191 (11,53%)	6,61	36
AG/SA	219.191 (11,53%)	4,33	25
SA/AG	219.191 (11,53%)	9,42	58

3.3.2 Rede de médio porte

O segundo exemplo apresenta 50 nós, dos quais três são nós fonte. O sistema é composto por 50 trechos, sendo 47 fechados e três abertos. Na configuração inicial (figura 26) a perda é 30.605.175 watts.

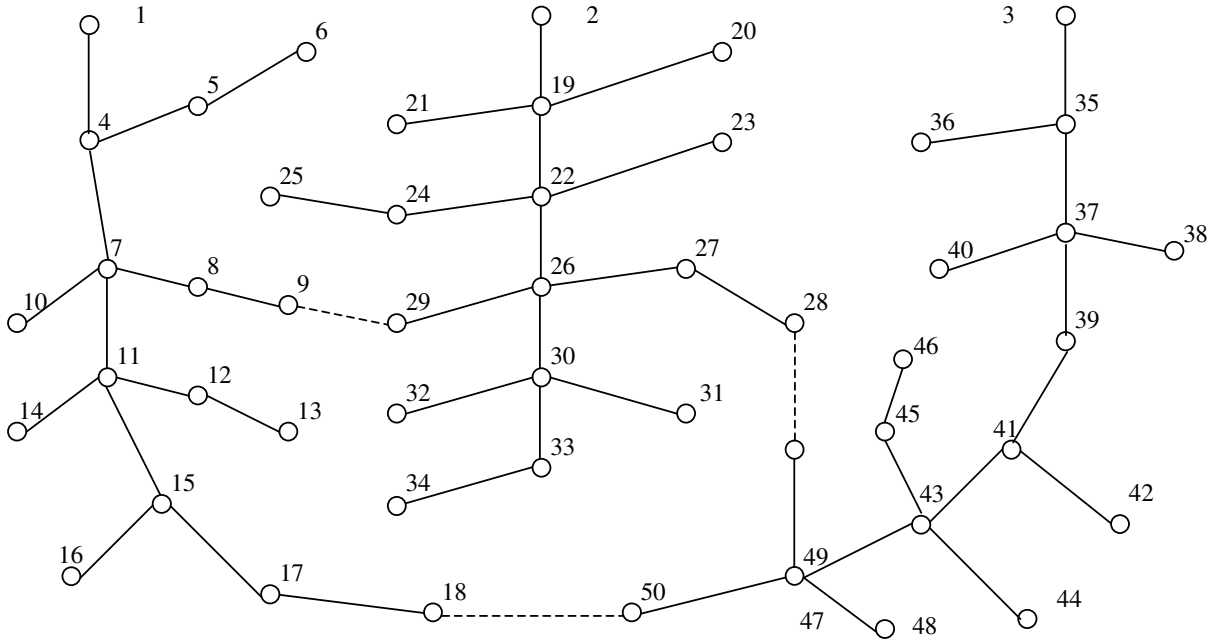


FIGURA 26 – **Configuração inicial da rede de médio porte**

Estipulou-se como meta uma redução em 4,53% das perdas iniciais, ou seja, as perdas deverão ser de no máximo 29.218.075 watts. As tabelas 5, 6, 7 e 8 ilustram resultados obtidos pelos quatro métodos empregados para a minimização das perdas elétricas.

TABELA 5 – Resultados referentes à rede de médio porte usando *erro*1 menor que 0,05 e meta igual a 29.218.075 W

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	29.218.075 (4,53%)	52,14	105
AG	29.218.075 (4,53%)	19,70	39
AG/SA	29.218.075 (4,53%)	14,66	28
SA/AG	29.218.075 (4,53%)	13,95	29

Da tabela 5 observa-se que os quatro métodos alcançaram a solução de perda mínima (configuração dada na figura 27). O método híbrido SA/AG mostrou um menor tempo de processamento. Entretanto, AG/SA executou a tarefa com um número menor de iterações.

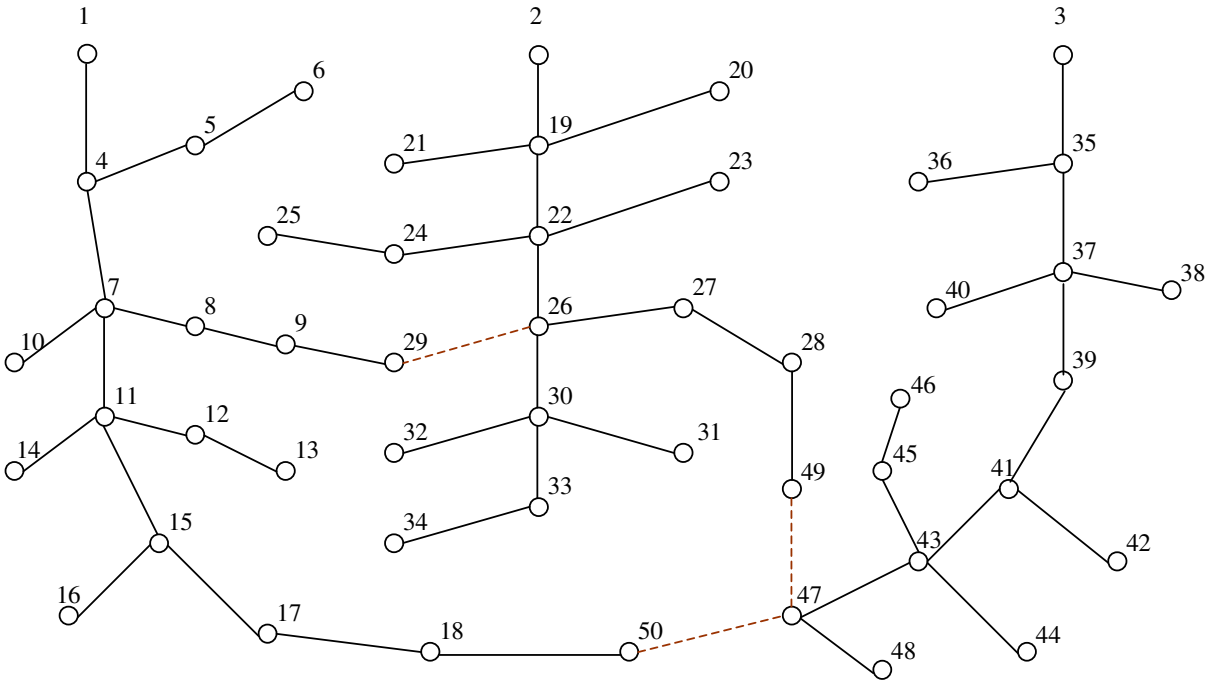


FIGURA 27 – **Configuração com menor perda da rede de médio porte**

Os resultados, fornecidos na tabela 1, referentes à rede de pequeno porte também indicaram o método híbrido SA/AG na obtenção da solução demandando o menor tempo computacional.

Ao impor limite máximo para o *erro*1 igual a 0,7 (tabela 6), o método AG/SA apresentou a melhor solução, de mínima perda, com menor esforço computacional. Diferentemente dos resultados obtidos na rede de pequeno porte (tabela 2) em que o método híbrido SA/AG apresentou melhor desempenho.

TABELA 6 – Resultados referentes à rede de médio porte usando *erro*1 menor que 0,7 e meta igual a 29.218.075 W

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	29.400.175 (3,94%)	19,77	42
AG	29.365.375 (4,05%)	17,22	32
AG/SA	29.218.075 (4,53%)	8,61	14
SA/AG	29.365.375 (4,05%)	10,13	22

Os resultados, considerando *erro*2 menor que 0,05 (tabela 7), mostram que todos os métodos alcançaram a meta (menor perda) e o que apresentou menor tempo de processamento e menor número de iterações foi AG/SA. Da mesma forma, ao adotar o mesmo critério de convergência, o mesmo método apresentou menor tempo computacional para a rede de pequeno porte (tabela 3).

TABELA 7 – Resultados referentes à rede de médio porte usando *erro*2 menor que 0,05

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	29.218.075 (4,53%)	36,17	72
AG	29.218.075 (4,53%)	40,20	85
AG/SA	29.218.075 (4,53%)	20,23	66
SA/AG	29.218.075 (4,53%)	32,22	69

Ao analisar os resultados das simulações, para o critério de convergência adotando *erro*2 inferior a 0,7 (tabela 8), o método híbrido AG/SA apresentou menor número de iterações e tempo computacional, e a solução final apresentada foi a de mínima perda. A rede de pequeno porte, adotando o mesmo critério de convergência, também apresentou melhores resultados através do uso do AG/SA (tabela 4).

TABELA 8 – Resultados referentes à rede de médio porte usando *erro*2 menor que 0,7

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	29.365.375 (4,05%)	33,61	69
AG	29.218.075 (4,53%)	36,36	75
AG/SA	29.218.075 (4,53%)	29,19	60
SA/AG	29.218.075 (4,53%)	29,53	61

3.3.3 Rede de grande porte

A rede, nesta aplicação, é composta por 8 nós fonte, 92 nós com demanda, 100 trechos, dos quais 92 são fechados. A configuração inicial (figura 28) apresenta perdas no montante de 11.277.800 watts. Suponha que a meta é 10.786.750 watts, ou seja, deseja-se reduzir as perdas iniciais em 4,35%.

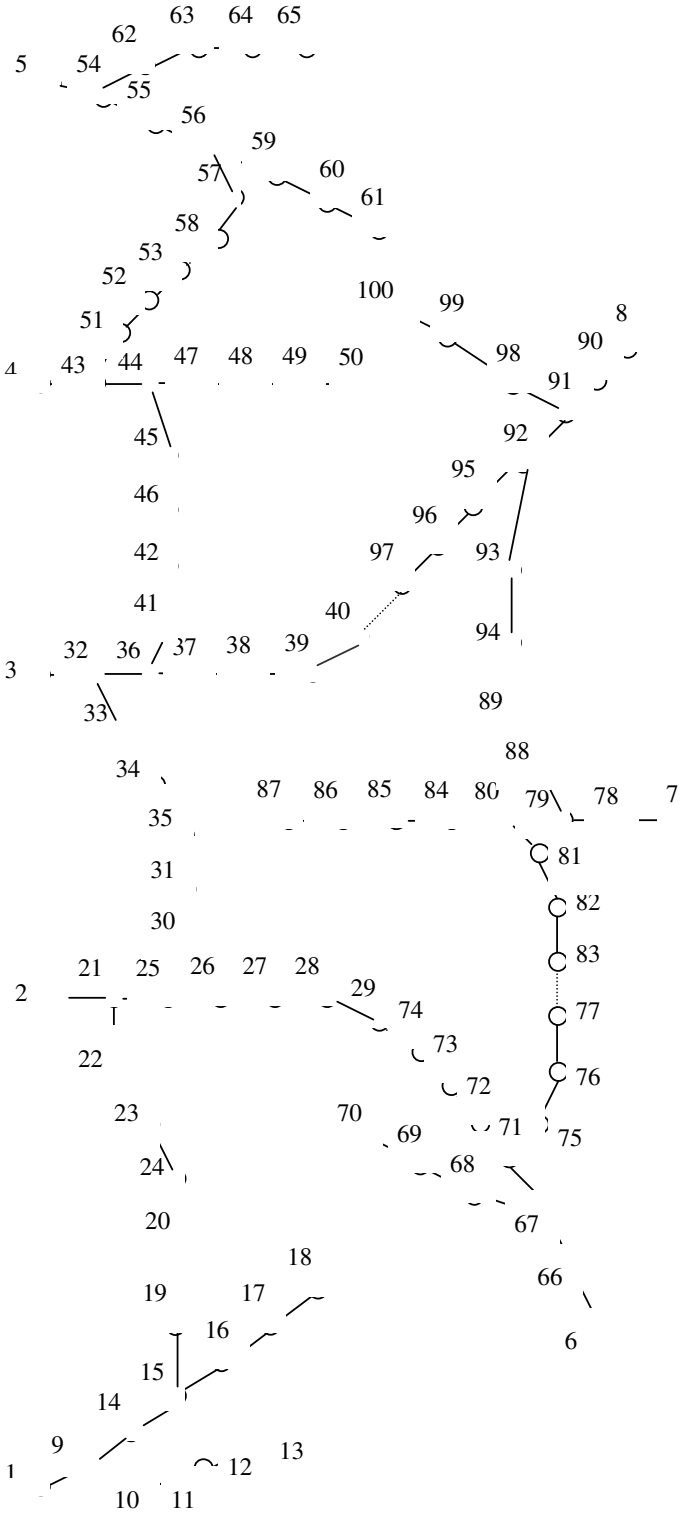


FIGURA 28 - **Configuração inicial da rede de grande porte**

Os resultados dos quatro métodos empregados na minimização das perdas são apresentados nas tabelas 9, 10, 11 e 12.

A tabela 9 contém os resultados obtidos no emprego do critério de parada *erro*1 inferior a 0,05. *Simulated annealing* e o método híbrido AG/SA não convergiram para o *erro*1 menor que 0,05 e foram interrompidos após extrapolar o número máximo de iterações. O algoritmo genético e o híbrido SA/AG praticamente determinaram a solução de menor perda (meta proposta). O menor tempo de processamento e o menor número de iterações foi obtido pelo método híbrido SA/AG. Este método também apresentou melhor desempenho para os exemplos anteriores (tabelas 1 e 5).

TABELA 9 – Resultados referentes à rede de grande porte usando *erro*1 menor que 0,05 e meta igual a 10.786.750 W

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	10.852.850 (3,77%)	74.940,66	6.000
AG	10.790.150 (4,35%)	199,28	238
AG/SA	10.847.000 (3,82%)	58.560,87	6016
SA/AG	10.790.150 (4,35%)	188,27	202

Para *erro*1 inferior a 0,7 (tabela 10), nenhum dos quatro métodos reduziu em 100% a diferença entre a perda inicial e a desejável. O método híbrido SA/AG apresentou melhor solução, reduzindo a perda em 4,07%. Entretanto o algoritmo genético convergiu através de um menor número de iterações e menor tempo computacional. O mesmo desempenho foi observado para a rede de pequeno porte (tabela 2). Na rede de médio porte constatou-se um melhor desempenho através do método híbrido AG/SA (tabela 6).

TABELA 10 – Resultados referentes à rede de grande porte usando *erro*1 menor que 0,7 e meta igual a 10.786.750 W

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	10.839.500 (3,89%)	3.349,16	3.393
AG	10.847.000 (3,82%)	43,42	39
AG/SA	10.904.218 (3,31%)	25.749,14	6008
SA/AG	10.818.650 (4,07%)	49,30	59

Os resultados (tabela 11) mostram que o *simulated annealing* demandou menor número de iterações, porém a solução apresenta a menor redução de perda elétrica. O algoritmo genético alcançou a solução de mínima perda (meta do *erro*1) embora tenha apresentado o maior tempo de convergência (configuração obtida na figura 29). O modelo híbrido SA/AG demandou o menor tempo de processamento.

TABELA 11 – Resultados referentes à rede de grande porte usando *erro*2 menor que 0,05

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	11.016.800 (2,31%)	114,63	124
AG	10.786.750 (4,35%)	290,88	297
AG/SA	10.790.150 (4,32%)	162,80	214
SA/AG	10.824.925 (4,02%)	114,17	163

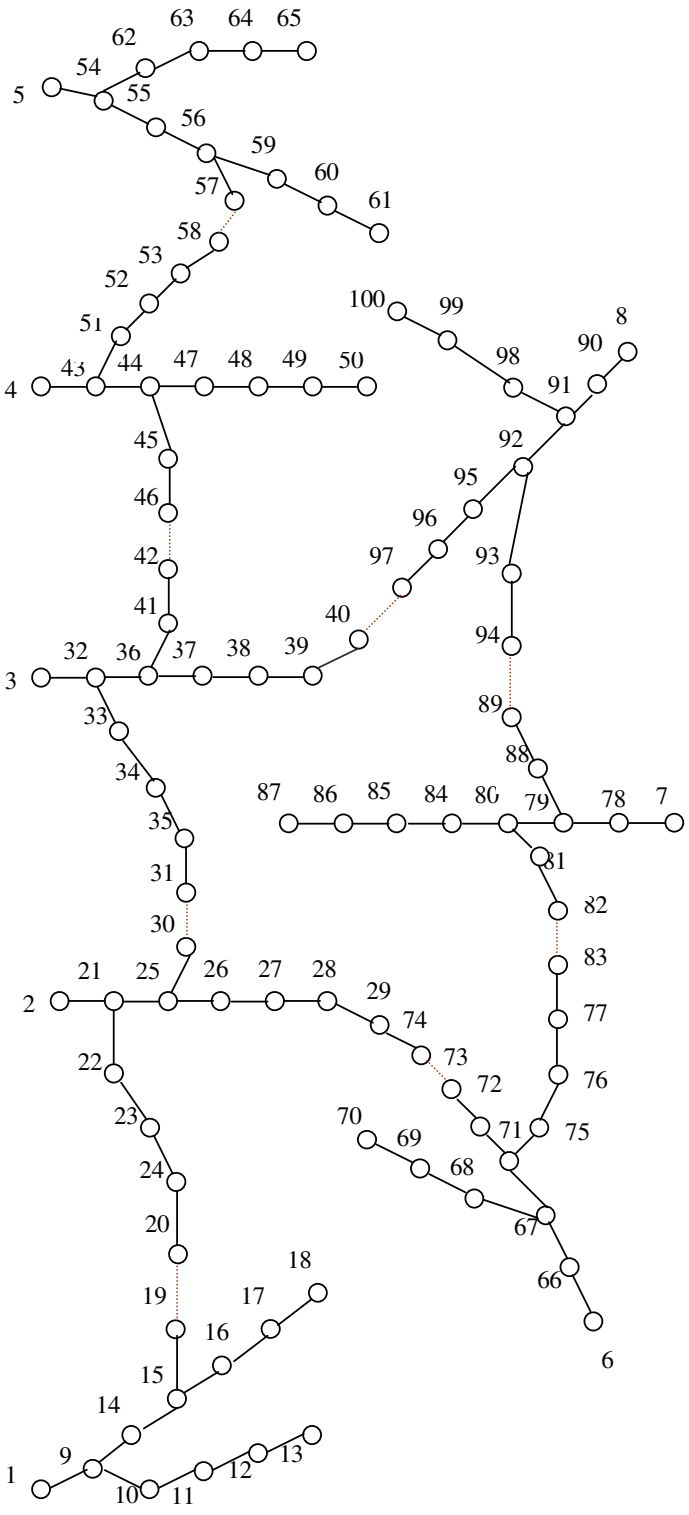


FIGURA 29 – **Configuração com menor perda da rede de grande porte**

A tabela 12 mostra que SA/AG apresentou melhor desempenho dado o menor tempo de processamento e o menor número de iterações. Entretanto a solução alcançada reduz a perda inicial em 3,70% sendo que a solução do algoritmo genético apresenta uma redução de 4,35% de perda elétrica. Por outro lado, AG necessitou de maior tempo computacional.

TABELA 12 – Resultados referentes à rede de grande porte usando *erro*2 menor que 0,7

Método	Valor da função objetivo (W)	Tempo (s)	Número de iterações
SA	11.062.075 (1,91%)	105,28	157
AG	10.786.750 (4,35%)	104,64	121
AG/SA	10.850,.525 (3,79%)	129,69	130
SA/AG	10.860.950 (3,70%)	101,88	108

3.4 CONSIDERAÇÕES FINAIS

No estudo da aplicação dos métodos aplicados na minimização do sistema de distribuição de energia elétrica observou-se que o tempo necessário na otimização do problema aumenta com o aumento do tamanho da rede de distribuição.

Neste trabalho foram considerados dois diferentes erros, *erro*1 e *erro*2, que são possíveis de serem analisados quando se obtém o valor da solução desejável.

Para a análise do estudo dos métodos híbridos, AG/SA e SA/AG, as percentagens adotadas no critério de convergência da otimização foram de 20% e 80%, isto é, inicialmente, executa-se a primeira técnica para minimizar 20% da diferença entre a perda inicial e a desejável (meta), o restante é realizada pela segunda técnica.

As três próximas figuras, ilustram o desempenho dos quatro métodos de acordo com o porte da rede, por tipo de erro e pelo valor do erro. A figura 30 mostra o desempenho em relação ao tempo de processamento; a figura 31 mostra o desempenho em relação ao número de iterações; a figura 32, o desempenho em relação ao valor da função objetivo.

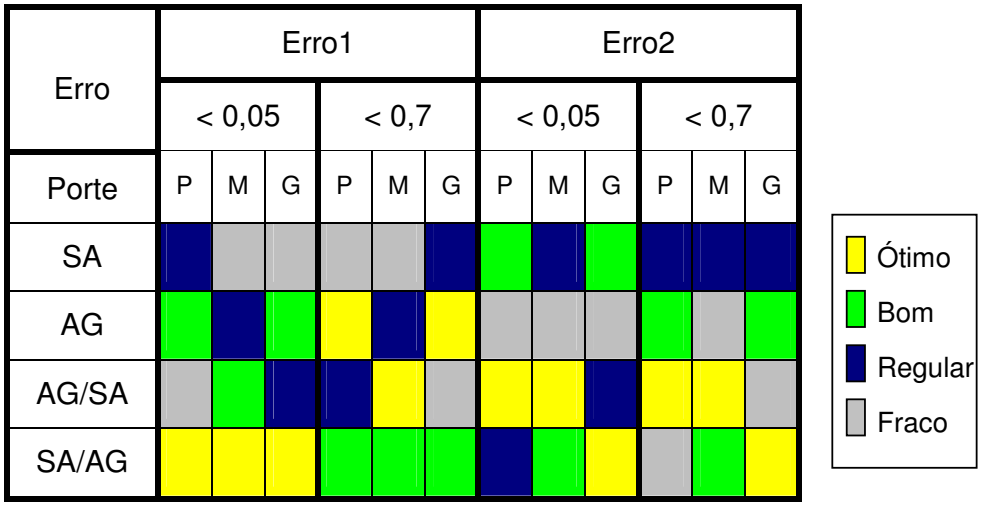


Figura 30 – **Quadro do desempenho dos métodos em relação ao tempo de processamento**

Na figura 30 encontra-se ilustrado o comportamento dos quatro métodos de otimização em relação ao tempo de processamento. Nesta figura, retângulos preenchidos com a cor amarela ( ) indicam o método que apresentou melhor desempenho em relação ao tempo de processamento; retângulos de cor verde ( ) indicam o método que apresentou o segundo melhor desempenho; retângulos de cor azul ( ) indicam o método que apresentou o segundo pior desempenho; retângulos de cor cinza claro ( ) indicam o método que apresentou o desempenho fraco em relação ao tempo de processamento.

Da figura, observa-se que, no geral, o método SA apresentou desempenho abaixo dos demais. Entretanto não é tão óbvio que o método SA/AG tenha sido melhor. Em relação ao erro tem-se: (1) utilizando o *erro*1, *simulated annealing* apresenta o pior desempenho, enquanto que o método híbrido SA/AG o melhor desempenho; (2) utilizando o *erro*2, o algoritmo genético apresentou o desempenho mais fraco, e o método AG/SA apresentou o melhor desempenho.

Na figura 31 encontra-se um quadro resumo do desempenho dos quatro métodos de otimização em relação ao número de iterações. Considerando o *erro*1 como critério de convergência, o método híbrido SA/AG apresentou melhor desempenho e o *simulated annealing* apresentou fraco desempenho; considerando o *erro*2, o comportamento é nítido o fraco desempenho do AG e do bom desempenho do AG/SA.

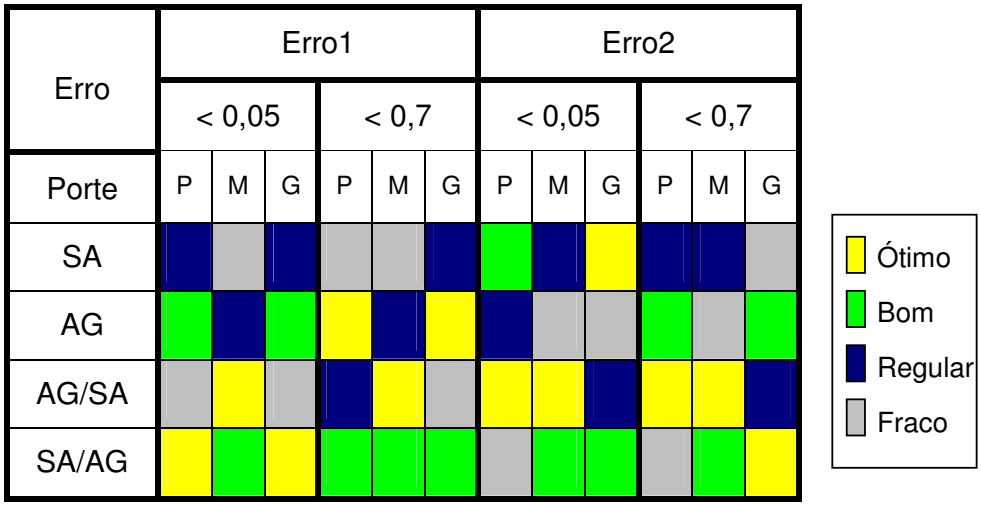


Figura 31 – **Quadro do desempenho dos métodos em relação ao número de iterações**

Em relação a minimização da função objetivo, figura 32, o algoritmo genético apresentou desempenho superior aos demais métodos quando utilizado o *erro*2. Ao usar-se o *erro*1, o melhor desempenho foi do método híbrido SA/AG.

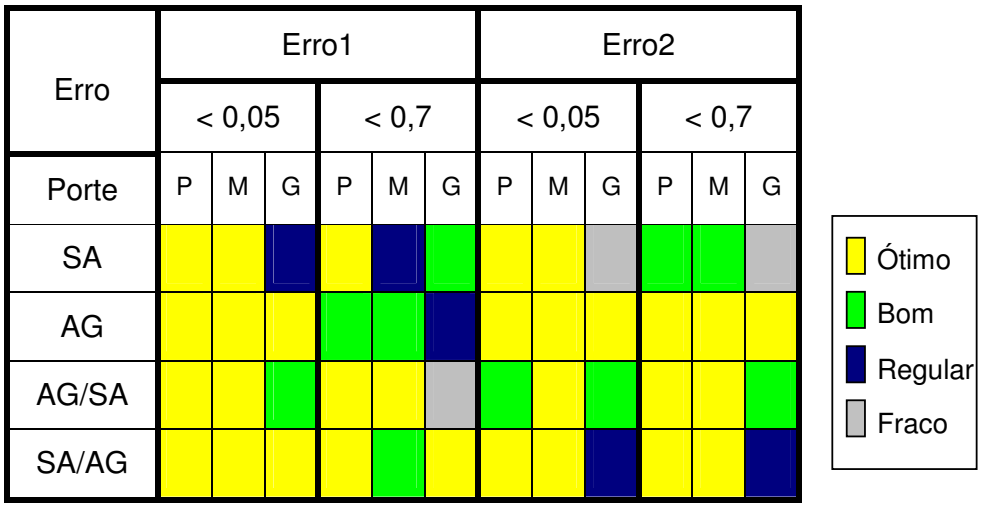


Figura 32 – **Quadro do desempenho dos métodos em relação à função objetivo**

O histograma da figura 33 mostra o desempenho global dos quatro métodos de otimização, considerando as diferentes redes e os diferentes erros.

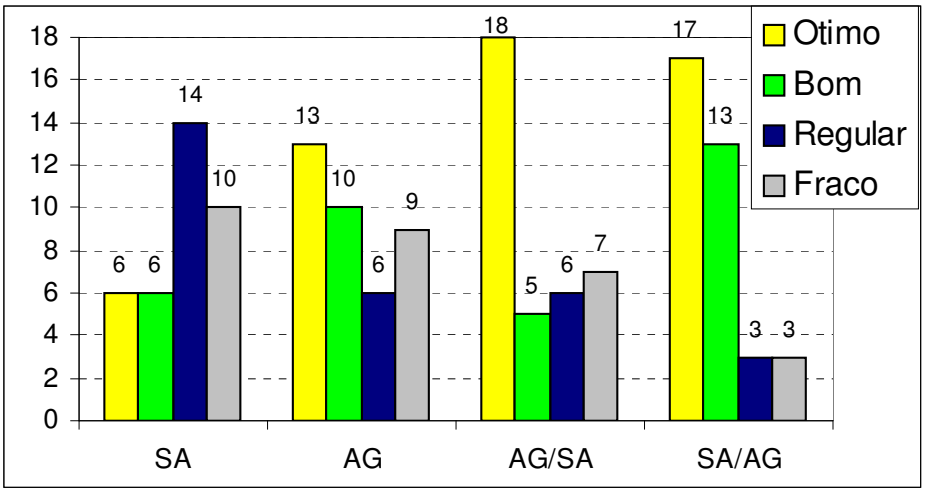


Figura 33 – **Desempenho global dos quatro métodos de otimização**

Desta figura tem-se a freqüência dos resultados dos quatro métodos em relação aos conceitos ótimo, bom, regular e fraco (figuras 30, 31 e 32). Os métodos AG/SA e SA/AG obtiveram um desempenho global superior e o SA desempenho fraco.

CAPÍTULO 4

4 CONCLUSÕES E RECOMENDAÇÕES

Com o constante crescimento dos centros urbanos e desenvolvimento de tecnologias há um conseqüente aumento do consumo de energia elétrica. Portanto há a necessidade de um constante estudo da rede de distribuição de energia elétrica para que ela possa suprir a demanda, de modo a minimizar as perdas elétricas.

Para a redução das perdas é interessante proceder uma melhor configuração da rede de distribuição de energia elétrica. A reconfiguração consiste basicamente em transferir pontos de demanda entre alimentadores através da abertura de alguns trechos e fechamento de outros.

Este trabalho teve por objetivo implementar e analisar quatro algoritmos heurísticos baseados em *simulated annealing* e em algoritmo genético, aplicando-os em redes de distribuição de energia elétrica fictícias, e, assim, verificar os seus desempenhos em relação a qualidade das soluções obtidas, ao tempo de processamento e número de iterações.

4.1 CONCLUSÕES

Este trabalho proporcionou a implementação e aplicação de quatro diferentes métodos na resolução de um problema de minimização de perdas no sistema de distribuição de energia elétrica.

Na implementação surgiram várias dificuldades devido à consideração de apenas redes radiais e conexas como soluções viáveis. Deste modo, foi necessária durante a execução do programa a verificação dos anéis formados pela rede, para que cada um sempre apresentasse uma chave aberta. Além da verificação

constante, a cada iteração, da desconexidade da rede, ou seja, se algum ponto de demanda não estava deixando de ser atendido pela rede.

Os resultados das aplicações dos algoritmos mostraram que não houve apenas um método que apresentasse um melhor desempenho em todas as redes estudadas. Na rede de porte pequeno, para *erro*1 menor do que 0,05, o método híbrido AG/SA apresentou maior rapidez e menor número de iterações, entretanto a solução final não foi a ótima (meta). Para *erro*1 inferior a 0,7 o algoritmo genético convergiu com menor tempo e número de iterações, porém não alcançou a solução ótima. O método AG/SA apresentou melhor desempenho, quando considerado *erro*2 como critério de convergência.

Na rede de porte médio, para o *erro*1 inferior a 0,05, a solução de mínima perda foi gerada com o menor tempo de processamento pelo método híbrido SA/AG. O método AG/SA mostrou melhor desempenho e melhor solução quando considerado o *erro1* menor que 0,7 e o *erro*2 inferior a 0,05 e 0,7.

Na análise da rede de porte grande, considerando o *erro*1 inferior a 0,05, o melhor desempenho foi do método híbrido SA/AG. Para *erro*1 menor que 0,7, o algoritmo genético foi mais rápido embora a melhor solução tenha sido alcançada pelo método híbrido SA/AG. Ao considerar o *erro*2, o método híbrido SA/AG obtiveram menores tempos de processamento e número de iterações, porém o algoritmo genético apresentou a solução de mínima perda.

Ao analisar o desempenho global, considerando o tempo de processamento, número de iterações e valor final da função, dos quatro métodos observou-se que os métodos híbridos AG/SA e SA/AG apresentaram melhores resultados globais. O *simulated annealing* foi o método que mostrou fraco desempenho.

4.2 RECOMENDAÇÕES

Recomenda-se para uma melhor análise de freqüência de execução periódica de reconfigurações do sistema de distribuição elétrica, de determinada região, um estudo relacionando os custos técnicos e econômicos.

Neste estudo, para uma simplificação da função objetivo e suas restrições, não foram consideradas algumas características de um sistema elétrico sendo

importante para a continuidade do estudo a consideração das impedâncias, perdas e oscilações de tensões.

Recomenda-se fazer estudos da implementação dos algoritmos apresentados neste trabalho, em exemplos de rede reais para uma melhor análise de suas eficiências para estes casos. No uso das técnicas híbridas AG/SA e SA/AG, para problemas de dimensões reais, analisar as percentagens mais adequadas de minimização, em cada uma das técnicas, na busca de um tempo mais curto de processamento e uma melhor comparação em relação ao número de iterações necessárias para a busca de uma solução ótima. Desta forma pode-se minimizar o esforço computacional requerido pelas técnicas.

Na procura de uma configuração ótima de uma rede de distribuição de energia elétrica faz-se necessária a realização de várias simulações, para qualquer método heurístico de otimização.

Estudos futuros poderiam analisar a utilização de um método híbrido que aplicasse ao SA/AG novamente o *simulated annealing*, visando obter um melhor resultado final.

Para um estudo mais amplo do desempenho dos métodos usados neste trabalho recomenda-se aplicá-los em outros tipos de problemas de otimização.

REFERÊNCIAS

[1] ANEEL - AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA. Diponível em: <http:\\www.aneel.gov.br> Acesso em: 16 nov. 2004.

[2] AUGUGLIARO, A.; DUSONCHET, L.; MANGIONE, S.. *An efficient greedy approach for minimum loss reconfiguration on distribution networks*, Electric Power Systems Research, p.167-176, 1995.

[3] BARBOSA, V.. *Redes Neurais e Simulated annealing como Ferramentas para Otimização Combinatória*, Investigación Operativa, v.1, p. 125-141,1989.

[4] BOROSAN, V.; RAJAKOVIC, N.. *Application Assessments of Distribution Network Minimum Loss Reconfiguration*, IEEE Transactions on Power Delivery, v.12, n.4, p.1786-1792, 1997.

[5] CARNIERI, C.; STEINER, M.T.A.; MATIOLI, L. C.; MUSSI, N. H.; KALINOWSKI, E. M.; KULIN, C.; BARBOSA, V. B.. *Relatório Copel* – Grupo1 (Parte2). Curitiba, 2003.

[6] CARPANETO, E.; CHICCO, G.. *Performance of the Simulated annealing-based Algorithms for the Optimal Reconfiguration of Distribution Systems*, IEEE Porto Power Tech Conference, 2001.

[7] CIVANLAR, S. ; GRAINGER, J. J.; YIN, H.; LEE, S.S.H.. *Distribution Feeder Reconfiguration for Loss Reduction*, IEEE Transactions on Power Delivery, v.3, n.3, p.1217-1223, 1988.

[8] CHIANG, H.; JEAN-JUMEAU, R.. *Optimal Network Reconfiguration in Distribution Systems: Part 1 : A New Formulation and A Solution Methodology*, IEEE Transactions on Power Delivery, v. 5, n. 4 , p. 1902-1909, 1990.

[9] CORRÊA, E. S.; STEINER, M. T. A.; FREITAS, A. A.; CARNIERI, C.. *A Genetic Algorithm for Solving a Capacitated P-Median Problem*. *Numerical Algorithms*. Kluwer Academic Publishers, v.35, p.373-388, 2004.

[10] EGLESE, R. W.. *Simulated Annealing: A tool for Operational Research*, European Journal of Operational Research, p.271-281,1990.

[11] HUDDLESTON, C. T.; BROADWARTER, R. P.; CHANDRASEKARAN, A.. *Reconfiguration Algorithm for Minimizing Losses in Radial Electric Distribution Systems*, Electric Power Systems Research, p.57-66, 1990.

[12] LIN, W.; SU, Y.; CHANG, S.; TSAY, M.; CHEN, S.. *The Optimal Loss Reduction of Distribution Feeder Based On Special Distribution Transformers Reconnection Using Genetic Algorithm*, IEEE Transactions on Power Delivery, p.1413-1418, 2000.

[13] MINISTÉRIO DE MINAS E ENERGIA. Secretaria de Energia, Coordenação de Balanço de Energia Nacional. Disponível em:< http://www.mme.gov.br> Acesso em: 12 jul. 2004.

[14] MITCHELL, M.. *An introduction to genetic algorithms*. Massachusetts Institute of Technology, 1997.

[15] MUSSI JR., N. H.. *Restauração de Redes de Distribuição com Heurísticas Baseadas no Método Branch and Bound*. Curitiba, 2002. 131f. Dissertação (Mestrado em Métodos Numéricos em Engenharia) – Setor de Ciências Exatas e de Tecnologia, Universidade Federal do Paraná.

[16] NARA, K.. *Simulated Annealing* Applications. In: SONG, Y. H.. *Modern Optimization Techniques in Power Systems*, p.15-38, 1999.

[17] SARMA, N. D.; RAO, K. S. P.. *A new 0-1 integer programming method of feeder reconfiguration for loss minimization in distribution systems,* Electric Power Systems Research, p. 125-131, 1995.

[18] SHIRMOHAMMADI, D.; HONG, H. W.. *Reconfiguration of Electric Distribution Networks for Resistive Line Losses Reduction*, IEEE Transactions on Power Delivery, v.4, n.2, p.1492-1498, 1989.

[19] TANOMARU, J.. *Motivação, Fundamentos e Aplicações de Algoritmos Genéticos*. II Congresso Brasileiro de Redes Neurais, III Escola de Redes Neurais, Curitiba, 1995.

[20] VOLPI, N. M. P.; WILHELM, V. E.; STEINER, M. T. A.; CARNIERI, C.; YUANG, J. Y.; MATIOLI, L. C.; GRUPPELLI JR, F.; KLIMKOWSKI, M.; ZAMBENEDETTI, V. C.; YAMAMOTO, L.; MUSSI JR, N. H.. *Reconfiguração de Sistemas de Distribuição Elétrica Minimizando Perdas elétricas*. IX Iberian Latin-American Congress on Computational Methods in Enginnering, Ouro Preto, 2003.

APÊNDICES

APÊNDICE A – DADOS PARA AS SIMULAÇÕES

As seguintes tabelas apresentam os dados utilizados nas simulações dos três exemplos de redes mencionadas no trabalho. Na primeira coluna está a numeração dos trechos da rede, nas duas colunas seguintes tem-se os nós iniciais e finais, ligados por cada trecho. A quarta coluna indica a resistência de cada trecho e a quinta coluna apresenta a carga demanda, de cada nó final do trecho, e as unidades de medida adotadas foram ohm e ampère, respectivamente.

TABELA 13 – **Dados da rede de pequeno porte**

Trechos	**Nó inicial**	**Nó final**	**Resistência (Ω)**	**Carga no nó final (A)**
1	1	4	0,13	88
2	4	5	0,13	132
3	5	6	0,16	44
4	4	7	0,16	88
5	7	8	0,07	44
6	8	9	0,07	66
7	2	10	0,19	178
8	10	11	0,19	224
9	11	12	0,19	44
10	12	13	0,14	27
11	13	14	0,14	27
12	6	14	0,07	
13	11	17	0,14	204
14	10	15	0,17	44
15	15	16	0,17	44
16	16	20	0,07	
17	3	18	0,19	44
18	18	19	0,17	66
19	19	20	0,14	44
20	18	21	0,19	44
21	21	22	0,14	66
22	22	23	0,07	44
23	9	23	0,07	

Observação: Os nós 1, 2 e 3 são os nós fonte, considerados como subestações, portanto não apresentam demanda.

TABELA 14 – **Dados da rede de médio porte**

Trechos	Nó inicial	Nó final	Resistência (Ω)	Carga no nó final (A)
101	1	4	0,19	250
102	4	5	0,14	300
103	5	6	0,13	250
104	4	7	0,19	200
105	7	8	0,17	350
106	8	9	0,16	400
107	7	10	0,07	100
108	7	11	0,19	400
109	11	12	0,17	300
110	12	13	0,16	500
111	11	14	0,13	200
112	11	15	0,19	300
113	15	16	0,07	150
114	15	17	0,16	250
115	17	18	0,14	150
116	2	19	0,19	250
117	19	20	0,13	200
118	19	21	0,13	200
119	19	22	0,19	200
120	22	23	0,13	300
121	22	24	0,16	350
122	24	25	0,16	400
123	22	26	0,19	300
124	26	27	0,16	300
125	27	28	0,16	300
126	26	29	0,17	400
127	9	29	0,16	
128	26	30	0,19	450
129	30	31	0,16	500
130	30	32	0,17	600
131	30	33	0,16	250
132	33	34	0,14	300
133	3	35	0,19	250
134	35	36	0,13	200
135	35	37	0,19	350
136	37	38	0,13	200
137	37	39	0,19	300
138	39	40	0,13	250
139	39	41	0,19	200
140	41	42	0,13	200
141	41	43	0,19	300
142	43	44	0,13	250
143	43	45	0,17	450
144	45	46	0,14	300
145	43	47	0,19	300
146	47	48	0,13	200
147	47	49	0,17	300
148	28	49	0,16	
149	47	50	0,14	150
150	18	50	0,14	

Observação: Os nós 1, 2 e 3 são os nós fonte, considerados como subestações, portanto não apresentam demanda.

TABELA 15 – **Dados da rede de grande porte**

continua

Trechos	**Nó inicial**	**Nó final**	**Resistência (Ω)**	**Carga no nó final (A)**
101	1	9	0,19	150
102	9	10	0,14	200
103	10	11	0,13	150
104	11	12	0,13	150
105	12	13	0,07	100
106	9	14	0,17	200
107	14	15	0,17	200
108	15	16	0,13	200
109	16	17	0,13	150
110	17	18	0,07	150
111	15	19	0,16	200
112	19	20	0,14	150
113	2	21	0,19	200
114	21	22	0,16	200
115	22	23	0,14	150
116	23	24	0,13	150
117	20	24	0,13	
118	21	25	0,17	200
119	25	26	0,16	200
120	26	27	0,16	200
121	27	28	0,14	150
122	28	29	0,14	150
123	25	30	0,16	150
124	30	31	0,14	150
125	3	32	0,19	200
126	32	33	0,16	200
127	33	34	0,16	150
128	34	35	0,14	150
129	31	35	0,13	
130	32	36	0,17	250
131	36	37	0,16	250
132	37	38	0,16	200
133	38	39	0,14	200
134	39	40	0,14	150
135	36	41	0,14	200
136	41	42	0,13	150
137	4	43	0,19	200
138	43	44	0,17	200
139	44	45	0,14	200
140	45	46	0,13	150
141	42	46	0,13	
142	44	47	0,14	250
143	47	48	0,13	150
144	48	49	0,13	150
145	49	50	0,07	100
146	43	51	0,17	200
147	51	52	0,14	100

				conclusão
148	52	53	0,13	150
149	5	54	0,19	200
150	54	55	0,17	200
151	55	56	0,16	200
152	56	57	0,14	150
153	57	58	0,13	150
154	53	58	0,13	
155	56	59	0,13	150
156	59	60	0,13	150
157	60	61	0,07	100
158	54	62	0,14	200
159	62	63	0,13	150
160	63	64	0,13	150
161	64	65	0,07	100
162	6	66	0,19	200
163	66	67	0,19	250
164	67	68	0,13	200
165	68	69	0,13	200
166	69	70	0,07	150
167	67	71	0,17	200
168	71	72	0,16	100
169	72	73	0,16	150
170	73	74	0,14	150
171	29	74	0,14	
172	71	75	0,16	150
173	75	76	0,16	150
174	76	77	0,14	100
175	7	78	0,19	200
176	78	79	0,19	200
177	79	80	0,17	150
178	80	81	0,16	150
179	81	82	0,16	150
180	82	83	0,14	100
181	77	83	0,14	
182	80	84	0,14	200
183	84	85	0,13	200
184	85	86	0,13	150
185	86	87	0,07	150
186	79	88	0,14	200
187	88	89	0,14	150
188	8	90	0,19	200
189	90	91	0,19	200
190	91	92	0,17	150
191	92	93	0,14	150
192	93	94	0,14	150
193	89	94	0,13	
194	92	95	0,16	200
195	95	96	0,16	200
196	96	97	0,14	150
197	40	97	0,14	
198	91	98	0,13	150
199	98	99	0,13	100
200	99	100	0,07	100

Observação: Os nós 1, 2, 3, 4, 5, 6, 7 e 8 são os nós fonte, considerados como subestações, portanto não apresentam demanda.

APÊNDICE B – PROGRAMAS EM MATLAB

PROGRAMA 1 – ALGORITMO SIMULATED ANNEALING

```
function annealing

tic %contagem de tempo
anterior=0;
global problema
%-------------------------------------------------------------------------------
%Monta a matriz A(nos x nos) guardando o numero dos trechos que os ligam e
B=A+A'
%-------------------------------------------------------------------------------
%-------------exemplo 01----------------------
%P=load('trechos1.m');
%-------------------------------------------------
%-------------exemplo 02----------------------
%P=load('trechos_ex3.m');
%-------------------------------------------------
%-------------exemplo 03----------------------
P=load('trechos_ex5.m');
%-------------------------------------------------
nos=length(P);
A=zeros(nos);
for i=1:nos
   A(P(i,2),P(i,3))=P(i,1);
end
for i=1:nos
   for j=1:i
      if A(j,i)<A(i,j)
         A(j,i)=A(i,j);
         A(i,j)=0;
      end
   end
end
%-------------exemplo 01------------------------
%chaves=load('trechos1.m');
%abertas=load('abertas1.m');
%fmin=219191.78;
%n_iter=500;
%aux_anterior=fmin;
%-------------------------------------------------
%-------------exemplo 02------------------------
%chaves=load('trechos_ex3.m');
%abertas=load('abertas_ex3.m');
% fmin=29218075;
% n_iter=500;
```

```
%aux_anterior=fmin;
%-------------------------------------------------
%------------exemplo 03-----------------------
chaves=load('trechos_ex5.m');
abertas=load('abertas_ex5.m');
fmin=10786750;
n_iter=6000;
aux_anterior=fmin;
%-----------------------------------------------
%-------exemplo 01-------------------------
%chaves_possiveis=[2 3 12 11 10 9 8 14 15 16 19 18 20 21 22 23 6 5 4];
%-------exemplo 02-------------------------
%chaves_possiveis=[105 106 127 126 124 125 148 147 149 150 115 114 112 108];
%-----------------------------------------------
%-------exemplo 03-------------------------
chaves_possiveis=[111 112 117 116 115 114 123 124 129 128 127 126 135 136 141
140 139 138 146 147 148 154 153 152 172 173 174 181 180 179 178 177 186 187
193 192 191 168 169 170 171 122 121 120 119 131 132 133 134 194 195 196 197];
%-----------------------------------------------

n_ab=size(abertas,1);
n_ch=size(chaves,1);

iter=1;
if iter==1      %mostra a funcao perda para a solucao inicial
   atual=abertas;
   caminho=caminhos(A,abertas,P); %usa o programa caminhos
   perda_total=perdatotal(caminho,P); %usa o programa perdatotal
   anterior=perda_total;
   fprintf('\n Iteracao: %d \n',iter);
   fprintf('Valor da Perda: %f \n',perda_total);
   fprintf('Trechos abertos: ');
   fprintf(' %d',abertas(:,1));

   iter=iter+1;
%-------------------------------------
   aneis=anel(caminho,abertas);     %usa o programa anel para achar todos os aneis
                                        da rede
%-------------------------------------
   compr_aneis=size(aneis,2);
   tam_aneis=size(aneis,1);
end
abertas=[];
fim=n_iter-1;
parada=0;
w=tam_aneis;
vetor_w=[];
aux_T=1;
while (fim>0 & parada<n_iter & T>0 & erro>0.05)       %critério de parada
```

```
L=ceil(n_iter/tam_aneis);          %número de iterações a cada temperatura
While (fim>0 & L>0 & erro>0.05)
        if iter>1
           problema=0;
           if aux_T==0
              w=ceil(tam_aneis*T)   %numero de chaves que troca de status
           end
           auxiliar=w;
           resto=tam_aneis-auxiliar;
           rest=resto;
           aleat=0;
           if w>0
              z=w;
              vetor_auxw=[];
              if resto~=0     %se T<1 escolhe as chaves que permanecem abertas
                 while rest>0
                     aleator=ceil(rand*tam_aneis);
                    interac=intersect(vetor_auxw,aleator);
                     tam_interac=size(interac,2);
                    if tam_interac==0
                       vetor_auxw=[vetor_auxw aleator];
                        rest=rest-1;
                    end
                 end
              else          %se T=1 a escolha das chaves abertas continua aleatória
                 vetor_w=[];
                 while z>0
                    aleat_w=ceil(rand*tam_aneis);
                     inter=intersect(aleat_w,vetor_w);
                    tam_inter=size(inter,2);
                     if tam_inter==0
                        vetor_w=[vetor_w aleat_w];
                       z=z-1;
                     end
                 end
              end
              tam_vetor_w=size(vetor_w,2);
              if resto==0     %se T=1 a escolha aleatoria de todas as chaves abertas
                 aaux=1;
                 q=1;
              else          %se T<1 so se escolhe aleatoriamente algumas chaves
                  aaux=resto+1;
                 q=resto+1;
                  vetor=setdiff(vetor_w,vetor_auxw);
                 aux_w=vetor_w;
                  vetor_w=[vetor_auxw vetor];
                 qqq=1;
                  auxab=[];
                 while qqq<resto+1
```

```
            for qqqq=1:tam_aneis
                if vetor_w(qqq)==aux_w(qqqq)
                    n_anel=qqqq;
                end
            end
            auxab=[auxab auxabertas(n_anel)];
            qqq=qqq+1;
        end
        auxabertas=auxab;
    end
        while aaux<tam_aneis+1    %escolha aleatória das chaves que
                                  devem abrir dentro das possíveis
        intersecao=intersect(aneis(vetor_w(aaux),:),chaves_possiveis);
        tam_intersecao=size(intersecao,2);
        aleat=ceil(rand*tam_intersecao);
        auxabertas(q)=intersecao(aleat);
        aaux=aaux+1;
        if q>1
            zaza=q-1;
            for g=1:zaza
                if auxabertas(q)==auxabertas(g)
                    aaux=aaux-1;
                    q=q-1;
                end
            end
        end
        q=q+1;
    end
end
w=auxiliar;
  for r=1:length(auxabertas)
    for t=1:n_ch
        if chaves(t)==auxabertas(r)
            abertas(r,:)=chaves(t,:);
        end
    end
  end
    if problema==0
        caminho=caminhos(A,abertas,P);
        perda_total=perdatotal(caminho,P);

        %-----------------------------------------------------------------
        %analise da adocao ou nao do resultado calculado
        %-----------------------------------------------------------------
        delta=perda_total-anterior;
        if delta>0
          aux=-delta/T;
          prob=exp(aux);
          parada=parada+1;
```

```
                        if prob>rand
                                aux_anterior=anterior;
                                anterior=perda_total;
                               atual=abertas;
                               parada=0;
                        end
                    else
                      aux_anterior=anterior;
                      anterior=perda_total;
                      atual=abertas;
                     parada=0;
                    end
                    fprintf('\n Iteracao: %d \n',iter);
                    iter=iter+1;
                    fim=fim-1;
                end
            end
            %--------------erro1----------------
            aux_erro=abs(fmin-anterior);
            erro=aux_erro / fmin;
            %--------------erro2----------------
            aux_erro=abs(aux_anterior-anterior);
            erro=aux_erro /aux_anterior;
            %-------------------------------------
            L=L–1;
    end
    w=w–1;
end
fprintf('\n\n Valor minimo da funcao: %f \n',anterior);
fprintf('Trechos abertos: ');
fprintf(' %d',atual(:,1));
tempo = toc;
fprintf('\n Tempo total(s): %f',tempo);
fprintf('\n Erro percentual: %f',erro);
```

PROGRAMA 2 – ALGORITMO ALGORITMO GENÉTICO

```
function GA

tic %contagem de tempo
%-------------------------------------------------------------------------------------------------------
%Monta a matriz A(nos x nos) guardando o numero dos trechos que os ligam
%-------------------------------------------------------------------------------------------------------
%-------------exemplo 01--------------------
%P=load('trechos1.m');
%-------------------------------------------------
%-------------exemplo 02--------------------
%P=load('trechos_ex3.m');
%-------------------------------------------------
%-------------exemplo 03--------------------
P=load('trechos_ex5.m');
%-------------------------------------------------
nos=length(P);
A=zeros(nos);
for i=1:nos
    A(P(i,2),P(i,3))=P(i,1);
end
for i=1:nos
    for j=1:i
        if A(j,i)<A(i,j)
            A(j,i)=A(i,j);
            A(i,j)=0;
        end
    end
end

%-------------exemplo 01--------------------
%chaves=load('trechos1.m');
%abertas=load('abertas1.m');
%fmin=219191.78;
%n_iter=500;
%aux_fit= fmin;
%-------------------------------------------------
%-------------exemplo 02--------------------
%chaves=load('trechos_ex3.m');
%abertas=load('abertas_ex3.m');
%fmin=29218075;
%n_iter=500;
%aux_fit= fmin;
%-------------------------------------------------
%-------------exemplo 03--------------------
chaves=load('trechos_ex5.m');
abertas=load('abertas_ex5,m');
fmin=10786750;
```

```
n_iter=6000;
aux_fit= fmin;
%-----------------------------------------------
%-----------------------------------------------
caminho=caminhos(A,abertas,P);
aneis=anel(caminho,abertas);
%-----------------------------------------------
n_ab=size(abertas,1);
n_ch=size(chaves,1);
iter=1;
p=n_ab;
m=5;    %numero de individuos da populacao
ab=abertas(:,1);
populacao=[ab'];

%-----------------------------------------------
%novos individuos para a populacao
%-----------------------------------------------
w=1;
while w<m
%-------exemplo 01------------
%    chaves_possiveis=[2 3 12 11 10 9 8 14 15 16 19 18 20 21 22 23 6 5 4];
%-------exemplo 02-----------
%    chaves_possiveis=[108 112 114 115 150 149 147 148 125 124 105 106 127
126];
%-------exemplo 03-----------
   chaves_possiveis=[111 112 117 116 115 114 123 124 129 128 127 126 135 136
141 140 139 138 146 147 148 154 153 152 172 173 174 181 180 179 178 177 186
187 193 192 191 168 169 170 171 122 121 120 119 131 132 133 134 194 195 196
197];
%---------------------------------
   n_ch_poss=size(chaves_possiveis,2);
   s=1;
   while s<n_ab+1
      random(s)=ceil(rand*n_ch_poss);
      s=s+1;
      q=[1];
      if s>2          %descartar individuo que possua chaves repetidas
         for z=1:s-2
            if random(s-1)==random(z)
               q=[q random(s-1)];
            end
         end
         p=size(q,2);
         if p>1
            s=1;
         end
      end
   end
```

```
    for x=1:n_ab
        pop(x)=chaves_possiveis(random(x));
    end
    for t=1:n_ab        %descartar individuo que abra mais que uma chave dentro de
                            um anel
        k=intersect(pop,aneis(t,:));
        g=size(k,2);
        if g>1
            pop=[];
        end
    end
    populacao=[populacao;pop];
    w=size(populacao,1);
end

%-------------------------------------------------
%calculo do fitness
%-------------------------------------------------
i=1;
for i=1:m
    abertas=[populacao(i,:)]';
    j=1;
    for k=1:n_ab    %montagem da matriz abertas para cada individuo
        for l=1:n_ch
            if chaves(l,1)==abertas(k)
                aux(j,:)=chaves(l,:);
                j=j+1;
            end
        end
    end
    abertas=aux;
    caminho=caminhos(A,abertas,P);
    fitness(i)=perdatotal(caminho,P);
end

%-------------------------------------------------
%ordenar o fitness do menor para o maior
%-------------------------------------------------
fitness_ord=fitness;
populacao_ord=populacao;
for i=1:m
    fitness_ord(i)=min(fitness);
    h=m;
    while h>0
        if fitness(h)==fitness_ord(i)
            fitness(h)=inf;
            populacao_ord(i,:)=populacao(h,:);
            h=1;
        end
```

```
        h=h-1;
    end
end

parada=0;
erro=1;
while (parada<n_iter & erro>0.05)

    r=0;
    while r==0

%-----------------------------------------------
%selecao
%-----------------------------------------------
        g=2;
        select=0;
        while g>0
            auxiliar=sqrt(1+4*rand*(m^2+m));
            j(g)=m+1-(auxiliar-1)/2;
            select(g)=floor(j(g));
            g=g-1;
            if select(1)==select(2)
                g=1;
            end
        end

%-----------------------------------------------
%crossover
%-----------------------------------------------
        auxi=n_ab-1;
        aleat=rand*auxi;
        elem=ceil(aleat);
        for i=1:elem
            troca(i)=ceil(rand*3);
        end
        pai=[populacao_ord(select(1),:);populacao_ord(select(2),:)];
        filho=pai;
        for i=1:elem
            filho(1,troca(i))=pai(2,troca(i));
            filho(2,troca(i))=pai(1,troca(i));
        end
        mut=filho;
        aux_mutacao=[];
        y=1;
        %se existe chaves iguais em um individuo
        x=n_ab;
        a=2;
        while x>1
            for i=1:x-1
```

```
        if a>0 & filho(1,x)==filho(1,i)
            filho(1,:)=[];
            a=size(filho,1);
            aux_mutacao(y)=1;
            y=y+1;
        end
        if a==2 & filho(2,x)==filho(2,i)
            filho(2,:)=[];
            a=size(filho,1);
            aux_mutacao(y)=2;
            y=y+1;
        end
    end
    x=x-1;
end
%se individuo possui mais de uma chave em um unico anel
o=size(filho,1);
while o>0
    t=n_ab;
    while t>0
        k=intersect(filho(o,:),aneis(t,:));
        g=size(k,2);
        t=t-1;
        if g>1
            filho(o,:)=[];
            t=0;
            aux_mutacao(y)=3;
            y=y+1;
            aux_m=o;
        end
    end
    o=o-1;
end
%se existe individuo igual
tam_f=size(filho,1);
auxil=0;
while tam_f>0
    t=m;
    while t>0
        aux1=intersect(filho(tam_f,:),populacao_ord(t,:));
        if size(aux1,2)==n_ab
            filho(tam_f,:)=[];
            t=1;
            aux_mutacao(y)=4;
            y=y+1;
        end
        t=t-1;
    end
    tam_f=tam_f-1;
```

```
        end

        if isempty(filho)

%---------------------------------------------
%mutacao
%---------------------------------------------
            escolha_mut=ceil(rand*2);
            if escolha_mut==1
                if aux_mutacao(1)==2
                    auxilio=aux_mutacao(2);
                elseif aux_mutacao(1)==3 & aux_m==2
                    auxilio=aux_mutacao(2);
                else
                    auxilio=aux_mutacao(1);
                end
            elseif escolha_mut==2
                if aux_mutacao(1)==2
                    auxilio=aux_mutacao(1);
                elseif aux_mutacao(1)==3 & aux_m==2
                     auxilio=aux_mutacao(1);
                else
                    auxilio=aux_mutacao(2);
                end
            end
            mutacao=mut(escolha_mut,:);
            muta=mutacao;
            v=setdiff(chaves_possiveis,mutacao);
            tam_v=size(v,2);
            sorte=ceil(rand*tam_v);
            subst=v(sorte);
            z=n_ab;

            if auxilio==1 | auxilio==2   %se houver repeticao de chaves
                x=n_ab;
                while i>0
                    for i=1:x-1
                        if mutacao(1,x)==mutacao(1,i)
                            t=n_ab;
                            while t>0
                                mutacao(i)=subst;
                                k=intersect(mutacao,aneis(t,:));
                                g=size(k,2);
                                if isempty(k)
                                    poss=intersect(aneis(t,:),chaves_possiveis);
                                    tam_poss=size(poss,2);
                                    q=ceil(rand*tam_poss);
                                    subst=poss(q);
                                    mutacao(i)=subst;
```

```
                end
                t=t-1;
            end
        end
    end
    x=x-1;
  end
end
if auxilio==3   %se houver chaves de um mesmo anel
  t=n_ab;
  while t>0
    k=intersect(mutacao,aneis(t,:));
    g=size(k,2);
    if g>1
      for i=1:n_ab
        if k(1)==mutacao(i)
          indice=i;
        end
      end
    end
    if isempty(k)
      r=intersect(aneis(t,:),subst);
      if isempty(r) | r~=subst
        poss=intersect(aneis(t,:),chaves_possiveis);
        tam_poss=size(poss,2);
        q=ceil(rand*tam_poss);
        subst=poss(q);
      end
    end
    t=t-1;
  end
  mutacao(indice)=subst;
end
if auxilio==4   %se a nova chave estiver no mesmo anel que uma das antigas
                troca-las
  while z>0
    aa=[subst mutacao(z)];
    b=n_ab;
    while b>0
      zz=intersect(aneis(b,:),aa);
      tam_zz=size(zz,2);
      if tam_zz==2
        mutacao(z)=subst;
        z=1;
        b=1;
      end
      b=b-1;
    end
    z=z-1;
```

```
            end
        end
        filho=mutacao;
        %se houver outro individuo igual
        t=m;
        while t>0
            aux1=intersect(filho,populacao_ord(t,:));
            if size(aux1,2)==n_ab
                r=0;
            else
                r=1;
            end
            t=t-1;
        end
    end
end

%-----------------------------------------------
%calculo do fitness dos filhos
%-----------------------------------------------
    n_filho=size(filho,1);
    for i=1:n_filho
        abertas=[filho(i,:)]';
        j=1;
        for k=1:n_ab    %montagem da matriz abertas para cada individuo
            for l=1:n_ch
                if chaves(l,1)==abertas(k)
                    aux(j,:)=chaves(l,:);
                    j=j+1;
                end
            end
        end
        abertas=aux;
        caminho=caminhos(A,abertas,P);
        fitness_filho(i)=perdatotal(caminho,P);

        if fitness_filho~=0 & fitness_filho(i)<fitness_ord(m)
            fitness_ord=[fitness_ord,fitness_filho(i)];
            populacao_ord=[populacao_ord;filho(i,:)];
        end
    end


%-----------------------------------------------------
%reordenar o fitness e excluir os maiores
%-----------------------------------------------------
    tam_fit=size(fitness_ord,2);
    if tam_fit>m
        for i=1:m
```

```
        fit(i)=min(fitness_ord);
        h=tam_fit;
           while h>0
              if fitness_ord(h)==fit(i)
                 fitness_ord(h)=inf;
                 populacao_ord(h,:);
                 popul(i,:)=populacao_ord(h,:);
                 h=1;
              end
                  h=h-1;
              end
          end
        fit=fitness_ord;
        popul=populacao_ord;
else
    fprintf('\n Iteracao: %d \n',iter);
    fprintf('Valor minimo da funcao: %d \n',fit(1));
    fprintf('Chaves abertas: ');
    fprintf('%d ',popul(1,:));
    saida=popul(1,:);
    fitness_ord=fit;
    populacao_ord=popul;
   aux_fit=fit(1);
   %--------------erro1----------------
    aux_erro=abs(fmin-fit(1));
    erro=aux_erro / fmin;
   %--------------erro2----------------
    %aux_erro=abs(aux_fit-fit(1));
    %erro=aux_erro /aux_fit;
   %-------------------------------------
    iter=iter+1;
    parada=parada+1;
end

tempo = toc;
fprintf('\n Tempo total(s): %f',tempo);
fprintf('\n Erro percentual: %f',erro);
```

PROGRAMA 3 – CAMINHOS

```
function [caminhos]=caminhos(A,lig,P);

global problema
%-------------exemplo (leitura funcao: simulacao)---
nos=length(P);
%-------------------------------------------------
n_trlig=size(lig,1);    %numero de chaves de ligacao
for i=1:n_trlig % desliga os trechos abertos = 0
   for w=1:nos
      for t=1:nos
         if A(w,t)==lig(i,1)
            A(w,t)=0;
         end
      end
   end
end
B=A+A';
u=1;

%-----------------------------------------------------------------------------------------------------
%Guarda os caminhos na matriz caminhos(nos, linha impar, e trechos, linha par)
%-----------------------------------------------------------------------------------------------------
%-----------------------------------
%fonte=load('fonte.m');
fonte=load('fontes_ex5.m');
%-----------------------------------
p=length(fonte);
Maux=B;
C=[];
D=[];
E=[];
compr_C=0;
for i=1:p   % monta os caminhos a partir das fontes (apenas 1 caminho)
   q=1;
   D(q)=fonte(i);
   E(q)=0;
   j=nos;
   while j>0
      ref=D(q);
      if Maux(j,ref)~=0
         D(q+1)=j;
         E(q+1)=Maux(j,ref);
         Maux(j,ref)=0;
         Maux(ref,j)=0;
         q=q+1;
         j=nos+1;
      end
```

```
        j=j-1;
    end
    if compr_C~=0   % adapta o tamanho dos vetores caminho para concatenar
        aux2=size(E,2);
        if aux2>compr_C
            w=aux2-compr_C;
            C=[C zeros(tam_C,w)];
        elseif aux2<compr_C
          for g=aux2+1:compr_C
              D(g)=0;
              E(g)=0;
           end
        end
    end
    C=[C;D;E];
    compr_C=size(C,2);
    tam_C=size(C,1);
    D=[];
    E=[];
end
tam_C=size(C,1);
compr_C=size(C,2);
x=2;
k=1;
while  x < compr_C+1     % montar outros caminhos a partir do 2,3,,,, elemento dos
                           caminhos anteriores
    while k < tam_C
        q=x;
        ref=C(k,x);
        aux=0;
        j=nos;
        if ref~=0
            while j>0
                if Maux(j,ref)~=0
                    D(q+1)=j;
                    E(q+1)=Maux(j,ref);
                    Maux(j,ref)=0;
                    Maux(ref,j)=0;
                    q=q+1;
                    aux=1;
                    ref=D(q);
                    j=nos+1;
                end
                j=j-1;
            end
            if aux==1
                for z=1:x
                    D(z)=C(k,z);
                    E(z)=C(k+1,z);
```

```
            end
        end
        if compr_C~=0  % adapta o tamanho dos vetores caminho para concatenar
            aux2=size(E,2);
            if aux2>compr_C
                w=aux2-compr_C;
                C=[C zeros(tam_C,w)];
            elseif aux2<compr_C
                for g=aux2+1:compr_C
                    D(g)=0;
                    E(g)=0;
                end
            end
        end
        C=[C;D;E];
        D=[];
        E=[];
        tam_C=size(C,1);
        compr_C=size(C,2);
    end
    k=k+2;
  end
  x=x+1;
  k=1;
end

%-------------------------------------------------------------------------
%Verifica desconexidade
%-------------------------------------------------------------------------
%no sem alimentacao
for i=1:nos
  if B(i,:)==0
    if ~ismember(i,fonte)
      problema=1;
      return
    end
  end
end
%trecho nao alimentado
for i=1:nos
  for j=1:nos
    if Maux(i,j)~=0
      problema=1;
      return
    end
  end
end

% excluir linhas nulas dos caminhos
```

```
tam_C=size(C,1);
compr_C=size(C,2);
caminhos=[];
i=1;
while i<tam_C
   if C(i,1)~=0
      caminhos=[caminhos;C(i,:);C(i+1,:)];
   end
   i=i+2;
end

%-------------------------------------------------------------------------
%Verifica nao radialidade
%-------------------------------------------------------------------------
Tcam=size(caminhos,1);
j=size(caminhos,2);
aux=j;
i=1;
while i<Tcam+1
   while caminhos(i,aux)==0
      aux=aux-1;
   end
   if ismember(caminhos(i,aux),fonte) %se no ultimo no do caminho tivermos um no
                                                    fonte
      problema=1;
      return
   end
   if i>1
      w=i-2;
      while w>0
         if ismember(caminhos(i,aux),caminhos(w,:)) %Se o ultimo no do caminho
estiver em outro caminho
            problema=1;
            return
         end
         w=w-2;
      end
   end
   i=i+2;
   aux=j;
end
```

PROGRAMA 4 – ANEL

```
function [aneltre]=anel(caminhos,abertas)

%---------------------------------------------------------------------
%Monta os aneis existentes na rede para sempre trabalhar de forma radial
%---------------------------------------------------------------------
Tcam=size(caminhos,1);
j=size(caminhos,2);
Tabe=size(abertas,1);
i=1;
aneis=[];
%---------------------------------------------------------------------
%se o ultimo no do caminho for uma chave aberta concateno o no adjacente da
chave aberta
%---------------------------------------------------------------------

while i<Tcam+1
   aux=size(caminhos,2);
   anel=[];
   aneltre=[];
   while caminhos(i,aux)==0
      aux=aux-1;
   end
   if caminhos(i,aux)~=0
      M=[];
      for z=1:Tabe
         if caminhos(i,aux)==abertas(z,2)
            anel=[];
            aneltre=[];
            for k=1:aux
               anel=[anel,caminhos(i,k)];
               aneltre=[aneltre,caminhos(i+1,k)];
            end
            anel=[anel,abertas(z,3)];
            aneltre=[aneltre,abertas(z,1)];
            M=[M;anel;aneltre];
         end
         if caminhos(i,aux)==abertas(z,3)
            anel=[];
            aneltre=[];
            for k=1:aux
               anel=[anel,caminhos(i,k)];
               aneltre=[aneltre,caminhos(i+1,k)];
            end
            anel=[anel,abertas(z,2)];
            aneltre=[aneltre,abertas(z,1)];
            M=[M;anel;aneltre];
         end
```

```
        end
    end
    i=i+2;
    tam_aneis=size(aneis,1);
    compr_aneis=size(aneis,2);
    compr_M=size(M,2);
    tam_M=size(M,1);
    if compr_M~=0   % adapta o comprimento dos vetores caminho para concatenar
        if compr_M>compr_aneis
            w=compr_M-compr_aneis;
            aneis=[aneis zeros(tam_aneis,w)];
        elseif compr_M<compr_aneis
            g=compr_aneis-compr_M;
            M=[M zeros(tam_M,g)];
        end
    end
    aneis=[aneis;M];
end
%-----------------------------------------------------------------
%concatenar todos os nos que formam um anel
%-----------------------------------------------------------------
tam_aneis=size(aneis,1); %n linha
compr_aneis=size(aneis,2); %n coluna
i=2;
vetortre=[];
if ~isempty(aneis)
    while i<tam_aneis+1
        j=compr_aneis;
        while aneis(i,j)==0
            j=j-1;
        end
        if aneis(i,j)~=0
            k=i+2;
            while k<tam_aneis+1
                q=compr_aneis;
                vetor=[];
                if aneis(k,q)==0
                    while aneis(k,q)==0
                        q=q-1;
                    end
                end
                if aneis(i,j)==aneis(k,q)
                    vaux=0;
                    for p=1:q
                        vaux(p)=aneis(k,q);
                        q=q-1;
                    end
                    vetor=[aneis(i,:),vaux];
                    compr_vetor=size(vetor,2);
```

```
                compr_vetortre=size(vetortre,2);
                tam_vetortre=size(vetortre,1);
                tam_vetor=size(vetor,1);
                if compr_vetortre>compr_vetor    %adapta o comprimento dos vetores
                                                 caminho para concatenar
                    w=compr_vetortre-compr_vetor;
                    vetor=[vetor zeros(tam_vetor,w)];
                 elseif compr_vetortre<compr_vetor
                    g=compr_vetor-compr_vetortre;
                    vetortre=[vetortre zeros(tam_vetortre,g)];
                 end
                vetortre=[vetortre;vetor];
            end
            k=k+2;
        end
    end
    i=i+2;
  end
end
compr_vetortre=size(vetortre,2);
tam_vetortre=size(vetortre,1);
aneltre=[];
for j=1:tam_vetortre
  i=compr_vetortre;
  u=1;
  auxanel=[];
  while i>1
    if vetortre(j,i)==0
      i=i-1;
    elseif vetortre(j,i)==vetortre(j,i-1)
      i=i-1;
    else
      auxanel(u)=vetortre(j,i);
      u=u+1;
      i=i-1;
    end
  end
  if i~=compr_vetortre;     % adapta o comprimento dos vetores para concatenar
    compr_aneltre=size(aneltre,2);
    compr_auxanel=size(auxanel,2);
    tam_auxanel=size(auxanel,1);
    tam_aneltre=size(aneltre,1);
    if compr_aneltre>compr_auxanel
      w=compr_aneltre-compr_auxanel;
      auxanel=[auxanel zeros(tam_auxanel,w)];
    elseif compr_aneltre<compr_auxanel
      g=compr_auxanel-compr_aneltre;
      aneltre=[aneltre zeros(tam_aneltre,g)];
    end
```

```
        aneltre=[aneltre;auxanel];
    end
end
```

PROGRAMA 5 – PERDA TOTAL

```
function [total]=perdatotal(caminhos,P);

%-------------------------------------------------------------------------
%Criar vetor com somatorio de cargas em cada trecho
%-------------------------------------------------------------------------

%-------------exemplo 01----------------------
%I=load('cargas1.m');
%-------------------------------------------------
%-------------exemplo 02----------------------
%I=load('cargas_ex3.m');
%-------------------------------------------------
%-------------exemplo 03----------------------
I=load('cargas_ex5.m');
%-------------------------------------------------
for i=1:length(P)   %monta matriz [trecho carregamento]
   aux=[];
   carreg(i,1)=P(i,1); %trecho
   carreg(i,2)=0;      %carregamento do trecho
   k=2;
   u=1;
   while k<size(caminhos,1)+1
      for j=1:size(caminhos,2)
         if caminhos(k,j)==P(i,1)
            q=j;
            while q<size(caminhos,2)+1
               aux(u)=caminhos(k-1,q);
               q=q+1;
               u=u+1;
            end
         end
      end
      k=k+2;
   end

   for d=1:length(aux) %para nos iguais considerar apenas um
      for j=d+1:length(aux)
         if aux(d)==aux(j)
            aux(j)=0;
         end
      end
   end
   for j=1:length(aux)
      if aux(j)~=0
         carreg(i,2)=carreg(i,2)+I(aux(j),2);
      end
   end
```

```
end

%-----------------------------------------------------------------------
%calculo da perda total
%-----------------------------------------------------------------------

%-------------exemplo 01----------------------
%R=load('resistencias1.m');
%-----------------------------=-------------------
%-------------exemplo 02----------------------
%R=load('resistencias_ex3.m');
%-------------------------------------------------
%-------------exemplo 03----------------------
R=load('resistencias_ex5.m');
%-------------------------------------------------
total=0;
for i=1:length(R)
    funcao(i,1)=R(i,2)*carreg(i,2)^2;
    total=total+funcao(i,1);
end
```