Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Faculdade de Engenharia

Kleber Hochwart Cardoso

Implementação distribuída de auto-cura em redes inteligentes de distribuição de energia elétrica utilizando árvores de extensão mínima

Rio de Janeiro
2014

Kleber Hochwart Cardoso

Implementação distribuída de auto-cura em redes inteligentes de distribuição de energia elétrica utilizando árvores de extensão mínima

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Orientadora: Prof.ª Dr.ª Nadia Nedjah
Coorientadora: Prof.ª Dr.ª Luiza de Macedo Mourelle

Rio de Janeiro
2014

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC / B

C143 Cardoso, Kleber Hochwart
Implementação distribuída de auto-cura em redes inteligentes de distribuição de energia elétrica utilizando árvores de extensão mínima. - 2014.
138f.

Orientadora: Nadia Nedjah.
Coorientadora: Luiza de Macedo Mourelle.
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Faculdade de Engenharia.

1. Engenharia Eletrônica - Dissertações. 2. Sistemas inteligentes - Dissertações. 3. Programação paralela (Computação) - Dissertações. 4. Processamento paralelo (Computadores) - Dissertações. I. Nedjah, Nadia. II. Mourelle, Luiza de Macedo. III. Universidade do Estado do Rio de Janeiro. III. Título.

CDU 004.272.2

Assinatura

Data

Kleber Hochwart Cardoso

Implementação distribuída de auto-cura em redes inteligentes de distribuição de energia elétrica utilizando árvores de extensão mínima

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Eletrônica, da Universidade do Estado do Rio de Janeiro. Área de concentração: Sistemas Inteligentes e Automação.

Aprovado em:

Banca Examinadora:

Prof.ª Dr.ª Nadia Nedjah (Orientadora)
Faculdade de Engenharia - UERJ

Prof.ª Dr.ª Luiza de Macedo Mourelle (Coorientadora)
Faculdade de Engenharia - UERJ

Prof.ª Dr.ª Viviana Cocco Mariani
Departamento de Engenharia Elétrica - UFPR
Centro de Ciências Exatas e de Tecnologia - PUCPR

Prof. Dr. Raphael Carlos Santos Machado
Instituto Nacional de Metrologia, Qualidade e Tecnologia - Inmetro

Rio de Janeiro
2014

AGRADECIMENTOS

Agradeço primeiramente a Deus por me proporcionar a oportunidade de lutar pelos meus sonhos e aos meus pais, que me incentivaram e investiram em minha educação ao longo da vida.

Agradeço à minha esposa Milene e minha filha Gabrielle, pela paciência nos longos períodos de ausência, nas abdicações de momentos de lazer e no apoio nos momentos de fraqueza, na certeza que este período era importante para a minha vida acadêmica assim como meu crescimento pessoal e profissional.

Agradeço aos meus irmãos Brunno e Brenno, aos meus familiares e amigos que compreenderam que por estar me dedicando ao estudo acadêmico não poderia estar presente em certos encontros e confraternizações.

Agradeço aos professores do PEL-UERJ pelo aprendizado ao longo desses dois anos e em especial à Nadia Nedjah e Luiza de Macedo Mourelle, por terem acreditado na minha proposta e com muita paciência me ajudaram a "lapidá-la" com suas orientações e conhecimentos.

Agradeço aos meus colegas de mestrado, pela ajuda nos momentos de dúvida, dividindo comigo os seus conhecimentos e expectativas com os resultados das pesquisas.

Agradeço aos meus amigos e companheiros de trabalho, na Light, por compreenderem meu momento e me auxiliarem com as atividades profissionais, assim como a meu gerente que me deu dispensa de horário para que eu pudesse fazer e concluir este curso

Enfim, agradeço a todos que contribuíram diretamente ou não para que este momento se tornasse realidade.

A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original.

Albert Einstein

RESUMO

CARDOSO, Kleber Hochwart. *Implementação distribuída de auto-cura em redes inteligentes de distribuição de energia elétrica utilizando árvores de extensão mínima.* 2014. 138f. Dissertação (Mestrado em Engenharia Eletrônica) – Faculdade de Engenharia, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2014.

A propriedade de auto-cura, em redes de distribuição inteligente de energia elétrica, consiste em encontrar uma proposta de reconfiguração do sistema de distribuição com o objetivo de recuperar parcial ou totalmente o fornecimento de energia aos clientes da rede, na ocorrência de uma falha na rede que comprometa o fornecimento. A busca por uma solução satisfatória é um problema combinacional cuja complexidade é proporcional ao tamanho da rede. Um método de busca exaustiva se torna um processo muito demorado e muitas vezes computacionalmente inviável. Para superar essa dificuldade, pode-se basear nas técnicas geração de árvores de extensão mínima do grafo, representando a rede de distribuição. Porém, a maioria dos estudos encontrados nesta área são implementações centralizadas, onde proposta de reconfiguração é obtida por um sistema de supervisão central. Nesta dissertação, propõe-se uma implementação distribuída, onde cada chave da rede colabora na elaboração da proposta de reconfiguração. A solução descentralizada busca uma redução no tempo de reconfiguração da rede em caso de falhas simples ou múltiplas, aumentando assim a inteligência da rede. Para isso, o algoritmo distribuído GHS é utilizado como base na a elaboração de uma solução de auto-cura a ser embarcada nos elementos processadores que compõem as chaves de comutação das linhas da rede de distribuição inteligente. A solução proposta é implementada utilizando robôs como unidades de processamento que se comunicam via uma mesma rede, constituindo assim uma ambiente de processamento distribuído. Os diferentes estudos de casos testados mostram que, para redes de distribuição inteligentes compostas por um alimentador, a solução proposta obteve sucesso na reconfiguração da rede, indiferentemente do número de falhas simultâneas. Na implementação proposta, a tempo de reconfiguração da rede não depende do número de barramentos e linhas nela incluídas. A implementação apresentou resultados de custo de comunicação e tempo dentro dos limites teóricos estabelecidos pelo algoritmo GHS.

Palavras-chave: auto-cura. árvores de extensão mínima. computação distribuída. redes de distribuição inteligentes.

ABSTRACT

The characteristic of self-healing, in smart grids, consists of finding a proposal for a reconfiguration of distribution system aiming at restoring the power, partially or completely to supply the network clients, in the event of network failure, which compromises the energy supply. The search for a satisfactory solution is a combinatorial problem whose complexity is proportional to the network size. An exhaustive search-based method is a time-consuming process and often computationally not viable. To overcome this difficulty, techniques for generating minimal spanning trees of the graph, which represents the smart grid, are exploited. However, the majority of studies in this area provide centralized implementations, where the solution for reconfiguration is achieved by a central control system. In this dissertation, we propose a distributed implementation, where each of the network switch collaborates in the development of the solution for reconfiguration. The proposed decentralized solution seeks a reduction in terms of the network reconfiguration time, in case of a single or multiple failures, thus increasing network intelligence. In this purpose, the GHS distributed algorithm is used as a basis for developing a self-healing solution to be embedded in the processing elements that are included within the line commutation switches of smart grid. The proposed solution is implemented using robots as processing units, which communicate via the same network, thereby creating a distributed processing environment. The several tested case studies show that, for smart grids that to have a single distribution feeder, the proposed solution allowed for a successful reconfiguration of the network, regardless of the number of simultaneous failures. In the proposed implementation, the network reconfiguration time does not depend on the number of buses and lines included. The implementation presents results of communication cost and time within the theoretical bounds of the GHS algorithm.

Keywords: Self-healing. minimum spanning tree. distributed computing. smart grids.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

SUMÁRIO

INTRODUÇÃO

REDES Elétricas Inteligentes, do inglês *Smart Grids*, é uma rede elétrica amplamente digitalizada que tem o objetivo de interligar os dispositivos da rede tais como, medidores, sensores, controladores e equipamentos micro-processados, instalados no sistema elétrico. E ainda, elementos de telecomunicações que vem se somando à arquitetura tradicional da rede elétrica com o objetivo de monitorar, gerencia e supervisionar o sistema. Esta nova arquitetura permite a coleta, distribui e atua em informações sobre o comportamento de todos os equipamentos que a compõem, desde a geração até o consumo, para melhorar sua eficiência e segurança do sistema (GELLINGS, 2009).

As redes elétricas inteligentes utilizam da mesma capacidade instalada, para demandar com mais segurança e eficiência, mais energia para os consumidores, pois têm o poder de administrar melhor a sua geração, transmissão e distribuição. Sem excluir a preocupação como a invulnerabilidade do sistema, violação na segurança, falhas humanas e mecânicas.

Um dos principais recursos disponibilizados pelas redes elétricas inteligentes é a funcionalidade de Auto-Cura, também conhecida como *Self-Healing*. Os principais benefícios de um sistema com Auto-Cura são as reduções no tempo de restabelecimento de serviço de fornecimento de energia elétrica, dos custos operacionais para o restabelecimento do fornecimento de energia elétrica e do número de clientes afetados pelas interrupções do fornecimento. Desta forma é possível isolar o defeito da rede elétrica em cerca de 20 segundos (MOREIRA, 2011), e assim melhorar a qualidade do serviço.

Os estudos para a implementação da técnica de Auto-Cura, analisam a problemática de reconfiguração da rede, de forma centralizada, através da modelagem de um problema combinacional de larga dimensão, que busca determinar o estado das chaves existentes na rede de distribuição, para encontrar uma configuração ótima. Esta busca baseia-se em critérios pré-estabelecidos como, minimização de perdas de energia elétrica e a qualidade da distribuição de tensões nos barramentos da rede. Em virtude disto vá-

rias pesquisas são realizadas nesta área a fim de se identificar métodos que otimizem as soluções para melhoria e/ou restabelecimento do sistema, uma vez que os métodos atualmente empregados apresentam grandes dificuldades de resolução de forma otimizada (BERNARDON, 2007).

Estas dificuldades podem ser contornadas, através da utilização da técnica de obtenção da árvore de extensão mínima de um grafo, associado ao modelo de computação distribuída. Este conjunto possibilita o aumento do desempenho, a minimização do tempo de resposta e a melhora na qualidade do resultado.

Problemas de projetos de rede são muito comuns de ocorrerem nas áreas de transportes aéreos ou terrestres, redes de computadores e redes elétricas. Estes problemas podem ser modelados por grafos não-orientados com pesos associados a suas arestas. Quanto maior for a rede maior será a dificuldade de se encontrar uma solução satisfatória, pois devido ao grande grupo de possibilidades a aplicação de um método exato de busca se torna um processo exaustivo e muitas vezes inviável, devido ao tempo de computação empregado para analisar todas as possibilidades (FARIA, 2013). A árvore de cobertura mínima consiste na obtenção de uma árvore geradora de um grafo que apresente um custo mínimo (SEDGEWICK, 2002). O modelo de árvore de extensão mínima ou AGM são comumente aplicado a estes tipos de problemas, pois ele possibilita a identificação do melhor caminho possível, que conecta todos os vértices da rede, dentre todos os caminhos existentes, desde que os pesos de utilização de cada trecho seja conhecido.

A computação distribuída se caracteriza por um conjunto de unidades de processamento que se comunicam em uma mesma rede, com coordenação de atividades oferecendo idealmente um sistema integrado. Possui como características principais a comunicação por mensagens, sistema assíncrono, falhas independentes e heterogeneidade. Como cada equipamento da rede inteligente possui um determinado poder processamento, viu-se a possibilidade de integrar a solução de árvore de extensão mínima à computação distribuída. Desta forma, é possível descentralizar o sistema que proverá a solução de reconfiguração de rede, não sendo mais necessário que todos os dados sejam enviado a uma unidade de processamento centralizada para que sejam tratados e transformado uma solução. Já que estes dados podem agora ser processados em conjunto pelos próprios equipamentos da rede para gerar uma solução, visto que estes passam a se apresentar como um sistema de processamento único.

Esta dissertação está organizada em sete capítulos, cujos conteúdos são resumidamente descritos a seguir.

Inicialmente, o Capitulo 1 traz um levantamento dos trabalhos mais relevantes ao projeto desta dissertação. Neles são apresentados algoritmos para a solução do problema de reconfiguração de rede utilizando o método de árvores de cobertura mínima. Alguns trabalhos com propostas de solução de outros tipos de problemas, também através do uso das técnicas de obtenção de árvore de extensão mínima.

O Capítulo 2 apresenta uma introdução teórica sobre a teoria dos grafos, árvores, árvores geradoras e árvores de extensão mínima. Introduz e descreve os algoritmo mais difundidos e utilizados na busca da árvore de extensão mínima, algoritmo de Prim e algoritmo de Kruskal. É realizada, também, o detalhamento do algoritmo distribuído de GHS, derivado do algoritmo de Kruskal, que fora utilizado como base de desenvolvimento da solução proposta neta dissertação.

O Capítulo 3 mostra a implementação da solução utilizando unidades de processamento distribuídos. Faz uma breve introdução dos robôs Elisa-3 utilizados como unidades de processamento distribuídos. É apresentada arquitetura do algoritmo de Auto-Cura, a criação das variáveis essenciais para a solução, e também o modelo de comunicação utilizado na troca de mensagem entre as unidades de processamento.

O Capítulo 4 apresenta a análise dos resultados obtidos pela implementação descrita no Capítulo 3. Duas diferentes redes de distribuição com um alimentador são utilizadas para testar os casos de otimização da rede, presença de falha simples e falha múltipla. A análise permite a confirmação do desempenho do algoritmo proposto, bem como da sua eficiência reconfiguração da rede de distribuição de energia elétrica. São apresentados os resultados custo de comunicação e a análise de tempo imposto pela implementação proposta, confrontado com seus valores limítrofes apresentados em (GALLAGER; HUMBLET; SPIRA, 1983).

Finalmente, o Capítulo 5 completa esta dissertação, apresentando as principais conclusões obtidas do desenvolvimento do presente trabalho. São apresentadas também possíveis melhorias a partir da solução proposta e as direções para futuros estudos envolvendo o algoritmo GHS.

Capítulo 1

ÁRVORES DE EXTENSÃO MÍNIMA

NESTE capítulo são apresentados os conceitos básicos e definições de grafos, árvores e árvores geradoras. Além disso são discutidos os algoritmos mais conhecidos utilizados para a obtenção de árvores de extensão mínima. Um maior enfoque é dado ao algoritmo de GHS explorado nesta dissertação. Este último é utilizado como base dos estudos realizados para o desenvolvimento de um método descentralizado de auto-cura para as redes de distribuição de energia elétrica.

1.1 Grafos

Os grafos são estruturas discretas, abstratas e intuitivas, comumente empregados para representar a relação entre "objetos". Graficamente, estes objetos são constituídos por *nós* ou *vértices*, interconectados por *arestas*, que configura a relação idealizada.

Um *grafo* é definido pela composição de um outro de vértices e um conjunto de arestas, que ligam pares de vértices do conjunto de vértices. Podendo ser completamente definidos por $G = (V, A)$, onde G denota o grafo formado por V, o conjunto não-vazio de vértices, e A um conjunto de arestas, sendo este formado por pares de elementos (p, q), p e $q \in V$.

A representação das arestas através da notação (p, q), implica que o vértice p está conectado ao vértice q através de uma aresta. Pode ser dito então que p e q são vizinhos.

O exemplo da Figura 1 representa um grafo $G = (V, A)$, onde $V = \{\text{a,b,c,d,e,f}\}$ e $A = \{\text{(a,c), (b,c), (c,d), (d,c), (c,e), (f,f)}\}$.

Conforme a definição de grafos, é possível ter estruturas com arestas representando laços, ocorrendo quando os ponto de origem e destino são os mesmos: (p, p). Outra

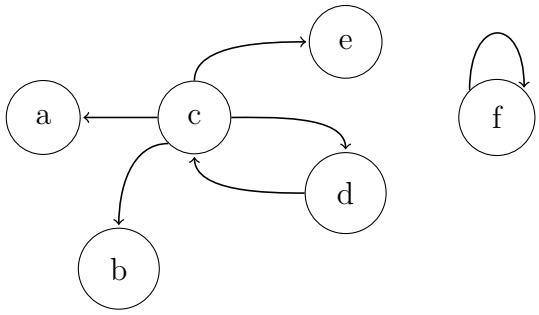


Figura 1: Exemplo de grafo $G = (V, A)$.

estrutura possível de ocorrer e das arestas paralelas, i.e., duas arestas para o mesmo par $(p,\ q)$ representando duas possíveis conexões entre uma mesma origem e destino. Na Figura 1 a aresta (f,f) representa um laço, enquanto que as duas arestas (c,d) e (d,c) representam arestas paralelas.

Um grafo que não possua laços e arestas paralelas em sua configuração, é dito *grafo simples*. A Figura 2 ilustra um exemplo de grafo simples.

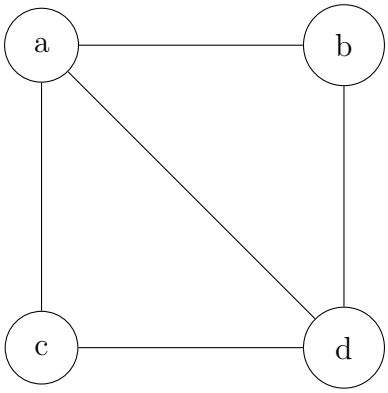


Figura 2: Exemplo de grafo simples com 4 vértices e 5 arestas.

O número de pares de elementos do conjunto A, denotado por NA, de um grafo simples, pode ser obtido através da Equação 1:

$$NA = \frac{Nv(Nv-1)}{2} \tag{1}$$

onde Nv representa o número de vértices do grafo G.

Os grafos podem trazer em sua configuração valores associados à suas arestas, representando uma grandeza qualquer. Dependendo do problema que está sendo descrito, estes valores podem indicar diferentes tipos de grandezas, tais como: comprimento, peso, custo de utilização, penalidade, entre outros. Este tipo de grafo é chamado de *grafo com*

pesos. Um exemplo deste tipo de configuração pode ser verificado no grafo da Figura 3, onde cada uma das arestas tem um valor associado .

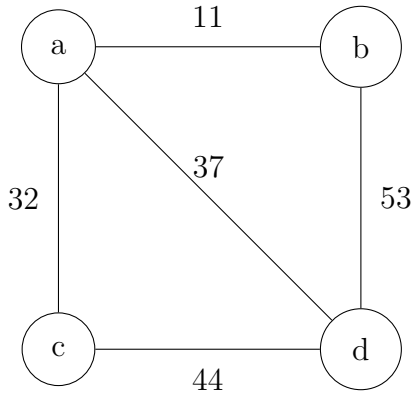


Figura 3: Exemplo de grafo simples com pesos.

Internamente, os grafos podem ser modelados através de uma matriz quadrada, $(Nv+1)(Nv+1)$. Os elementos do conjunto V ocuparão as posições referentes aos índices {(1,1), (2,1), ..., (n,1)} e {($n+1$,2), ($n+1$,3), ..., ($n+1$,n)}. Enquanto que cada aresta existente, que conecta um vértice p e um vértice q, pode ser incluída através da associação de um valor X à célula que fica na interseção dos índices i e j correspondentes aos vértices p e q. Este tipo de representação recebe o nome de *matriz adjacência*.

Tabela 1: Matriz adjacência de grafo simples da Figura 2.

a		X	X	X
b	X			X
c	X			X
d	X	X	X	
	a	b	c	d

Para grafos simples, não existirão indicações de arestas na diagonal principal e a matriz será simétrica em relação à esta diagonal. A Tabela 1 apresenta um exemplo de matriz adjacência de um apresentado na Figura 2

Para a representação de grafos com pesos, os valores associados às arestas são utilizados diretamente para popular as células da matriz adjacência, conforme pode ser visto na Tabela 2, que representa a matriz adjacência do grafo da Figura 3.

Tabela 2: Matriz adjacência de grafo com pesos da Figura 3.

A		11	32	37
B	11			53
C	32			44
D	37	53	44	
	A	B	C	D

1.2 Árvores

Uma árvore é um grafo simples conectado e unidirecional, sem arestas paralelas nem laços. A estrutura é definida por um conjunto finito não vazio de vértices, onde um dos vértices é denominado *raiz* e os demais formam ramificações da árvore ou *sub-árvore*, de forma recursiva. A estrutura da árvore representa uma relação hierárquica entre seus elementos, onde cada vértice pode ter vários sucessores, porém um único antecessor. Elas são conceituadas como um grafo simples $G(V, A)$, conexo e acíclico, formado por Nv vértices e exatamente $Nv - 1$ arestas (ALOISE; CRUZ, 2001).

As árvores são comumente definidas como um grafo sem direção, com a característica de possuir um único caminho entre qualquer par de vértices, ou seja, um grafo pode ser considerado uma árvore se e somente se existir um único caminho entre dois vértices (ROSEN, 1998).

Dado um determinado nível onde a árvore não possui mais descendentes, ou seja, a sub-árvore tem apenas um único vértice, este vértice recebe o nome de *folha*. As folhas da árvore representada na Figura 4 são os nós d, e, G, h e i.

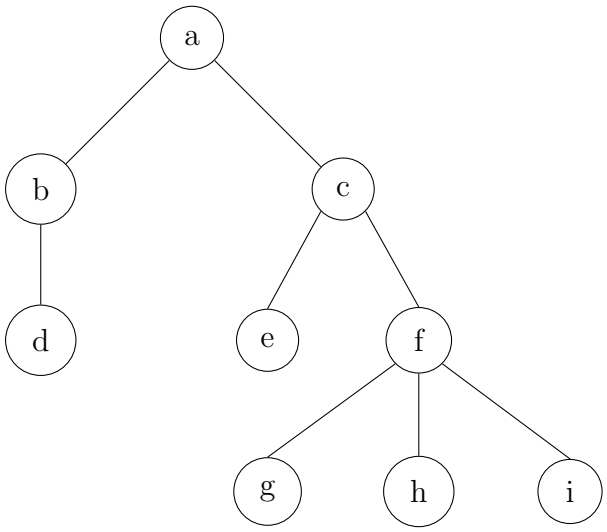
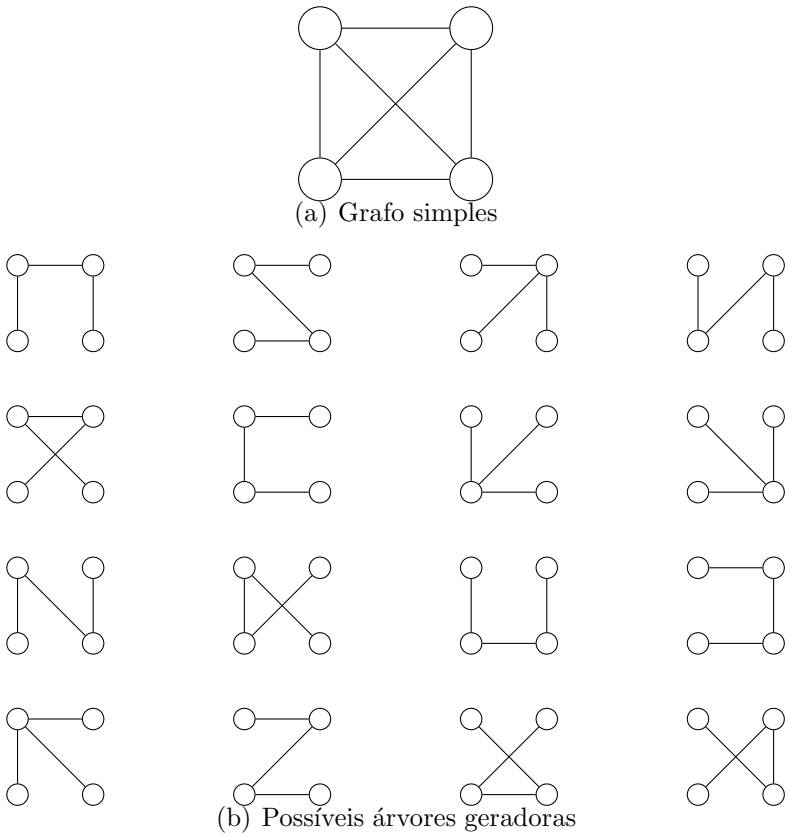


Figura 4: Árvore com raiz a e sub-árvores começando nos nós b, c e f.

Todo o grafo conexo possui ao menos uma árvore associada a sua configuração, que pode ser descrita como sua *árvore de extensão*.

1.2.1 Árvores Geradoras

Uma árvore de extensão de um grafo G consiste em qualquer sub-grafo de G que contenha o menor conjunto possível de arestas conectando todos os vértices do grafo G. Para qualquer grafo simples e conexo, existe pelo menos uma árvore de extensão (ROSEN, 2012). A Figura 5(b) apresenta todas as árvores geradoras possíveis para o grafo apresentado na Figura 5(a).



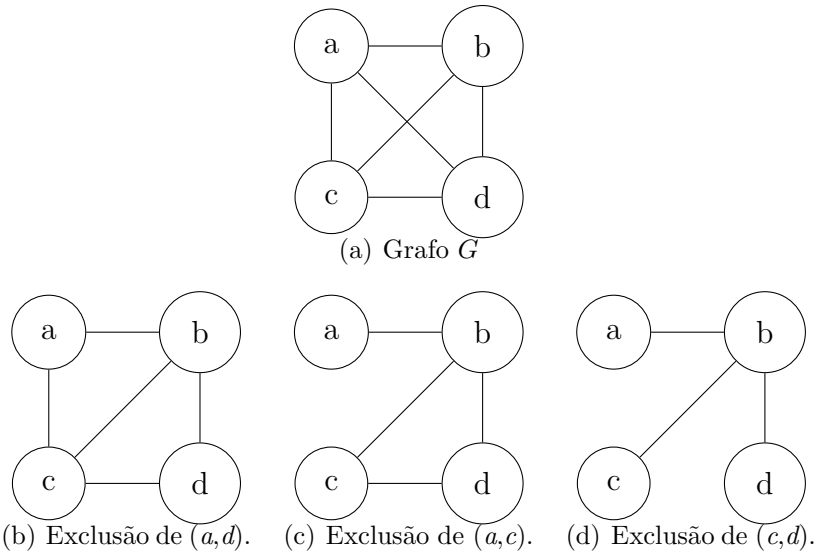
(a) Grafo simples

(b) Possíveis árvores geradoras

Figura 5: Exemplo de grafo conexo simples de 4 nós e 6 arestas, juntamente com todas as suas árvores geradoras.

O processo de obtenção da árvore de extensão é simples. Ele consiste na remoção de arestas do grafo conexo até que este se tornar acíclico, porém continuando conexo. A Figura 6 apresenta um exemplo de passo à passo da obtenção de uma possível árvore de extensão do grafo G ilustrado na Figura 6(a). Inicialmente as arestas (a,d) e (a,c) foram removidas sequencialmente do grafo, conforme apresentado nas Figura 6(b) e 6(c), para quebrar os ciclos formados pelos conjuntos $\{(a,c),(c,d),(d,a)\}$ e $\{(a,b),(b,d),(d,a)\}$. Por

último a arestas (c,d) foi removida, na Figura 6(d), quebrando o ciclo bcd. O grafo obtido é acíclico, não sendo mais possível remover nenhuma aresta sem alterar a característica de conexidade do grafo G. Portanto, está formada uma das possíveis árvores geradoras para o grafo.



(a) Grafo G

(b) Exclusão de (a,d). (c) Exclusão de (a,c). (d) Exclusão de (c,d).

Figura 6: Passos para obtenção de uma árvore de extensão para o grafo G.

1.2.2 Árvore de Extensão Mínima

O conceito de *árvores de extensão mínima* (AEM) é aplicável a todos os grafos conexos simples com pesos. O problema de obtenção da AEM é semelhante ao apresentado na Subseção 1.2.1. Entretanto, neste caso devem ser levados em conta os pesos associados às arestas do grafo, para que dentre as possíveis árvores geradoras, sejam selecionadas apenas aquelas que possuam o menor valor de peso total, i.e, o somatório de pesos das arestas da árvore de extensão encontrada deve ser o menor possível.

Seja um grafo conexo e não direcionado $G = (V, A)$, onde para cada aresta $(p, q) \in A$, tem-se um peso c_{pq} associado. Uma AEM é um subgrafo conexo $G_{V,T}$ do grafo G, tal que $T \subseteq A$ e cuja soma total dos seus pesos, dada pela Equação 2, é minima, considerando todas as possíveis árvores geradoras de G (PRADO; ALMEIDA; SOUSA,).

$$P_T = \sum_{(p,q)\in T} c_{pq} \tag{2}$$

Para um grafo $G = (V, A)$, com pesos associados as suas arestas, o processo simples de obtenção do conjunto de arestas que formam a AEM pode ser obtido utilizando os

passos do Algoritmo 1 que implica na a adição consecutiva de arestas (p, q) *seguras* de um conjunto B, inicialmente vazio. Uma aresta é considerada *segura*, se e somente se, $(B \cup (p, q)) \subseteq T$.

Algoritmo 1 Algoritmo para geração de AEM

```
1: B ← ⊘
2: enquanto B não forma uma AEM faça
3:    para todo (p,q) ∈ A faça
4:       se (B ∪ (p,q)) ⊆ T então
5:          B ← B ∪ (p,q)
6:       fim se
7:    fim para
8: fim enquanto
9: Retorna B
```

1.3 Algoritmos para Árvores de Extensão Mínima

O conceito de AEM é comumente aplicado em problemas de roteamento com a finalidade de se encontrar o melhor caminho dentro de uma rede, que pode ser de transportes aéreos ou terrestres, de computadores e de elétrica. Devido a sua grande aplicabilidade, vários algoritmos para obtenção de AEMs foram desenvolvidos, tais como o algoritmo de Boruvka (LIU, 2001), algoritmo de Dijkstra (BLACK, 2006), algoritmo de Edmond (KEMPE, 2004), entre outros. Porém, existem dois algoritmos de grande destaque na literatura, o algoritmo de Prim (PRIM, 1957; LEISERSON et al., 2001) e o algoritmo de Kruskal (KRUSKAL, 1956; LEISERSON et al., 2001), ambos considerados como algoritmos gulosos para obtenção de AEM. Os passos principais destes 2 algoritmos são detalhados nas Seções 1.3.1 e 1.3.2, respectivamente. Note que estes dois algoritmos formam a base de todos os outros.

1.3.1 Algoritmo de Prim

O algoritmo de Prim (PRIM, 1957; LEISERSON et al., 2001) inicia-se de um vértice escolhido arbitrariamente, e expande-se para formar a árvore de extensão mínima. Este processo funciona através da criação de dois conjuntos básicos de arestas, denominados q e C, onde o primeiro contém inicialmente todas as arestas do grafo e o segundo somente as arestas escolhidas para formar a AEM. A cada passo do algoritmo, uma aresta de menor peso, que conecta um vértice da árvore com um vértice que não pertence à árvore, é escolhida e removida do conjunto de todos as arestas do grafo q e passa a integrar o conjunto C da

árvore de extensão mínima, conforme o pseudocódigo descrito no Algoritmo 2, onde c_{pq} é o peso associado à aresta (p, q).

Algoritmo 2 Algoritmo de Prim para geração de AEM

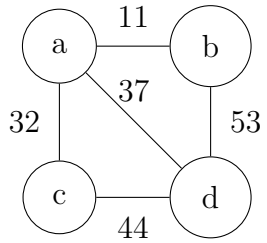
entrada $G = (V, A)$
Ensure: $AEM(V, A_{AEM})$
1: Escolher um vértice p_0
2: $V_{AEM} \leftarrow \{p_o\}$
3: $A_{AEM} \leftarrow \oslash$
4: **enquanto** $V_{AEM} \neq V$ **faça**
5: Seja $(q^*, p^*) = min(p, q), \mid v \in V_{AEM}$ e $q \in V - V_{AEM}$
6: $V_{AEM} \leftarrow V_{AEM} \cup \{q^*\}$
7: $A_{AEM} \leftarrow A_{AEM} \cup \{(p^*, q^*)\}$
8: **fim enquanto**

Este algoritmo é considerado guloso (CORMEN et al., 2002), pois a árvore é expandida a cada etapa absorvendo a aresta do grafo principal, de menor peso possível.

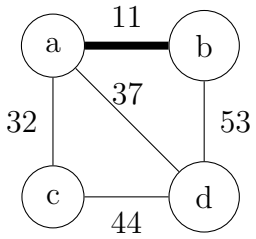
Aplicando o algoritmo de PRIM para o grafo $G = (V, A)$ da Figura 7(a), onde o algoritmo escolheu o vértice b para inicio do processo, inserindo-o no conjunto V_{AEM}. O algoritmo verifica qual o vértice integrante do conjunto $V - V_{AEM}$, que se conecta ao vértice b, cuja a aresta possui o menor peso. Para o exemplo, o vértice a compreende a aresta (a, b) que possui o menor peso, de valor igual à 11, conforme ilustração da Figura 7(b). O conjunto V_{AEM} passa a ser integrado, também, pelo vértice a, e o conjunto solução é atualizado com a aresta (b, a). Em seguida, o é retomada a busca do próximo vértice do conjunto $V - V_{AEM}$ que se conecta à um vértice do conjunto V_{AEM} através da aresta de menor peso existente. No caso do exemplo, o vértice c é selecionado, pois é o vértice que se conecta ao vértice a através da aresta de peso 32, como pode ser visualizado na Figura 7(c). O processo é repetido até que $V_{AEM} = V$, onde neste estágio a AEM do grafo G, apresentada na Figura 7(d.), é obtida.

1.3.2 Algoritmo de Kruskal

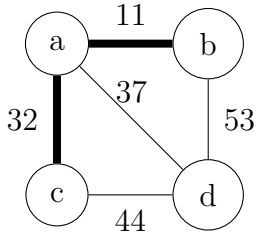
O algoritmo de Kurskal é iniciado com cada vértice do conjunto V sendo seu próprio componente, isto é, um conjunto independente formado por vértices não adjacentes entre si que não estão estritamente contidos em outros conjuntos independentes. O algoritmo passa a escanear o conjunto q, que possui as arestas de A ordenadas pelo valor crescente de peso, com a finalidade de juntar duas árvores diferentes para formar apenas uma, conectando-as através de uma aresta de menor peso. Este algoritmo trabalha com uma



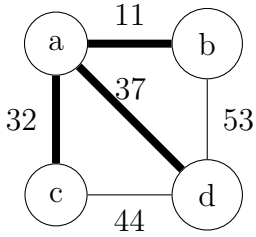
(a) Grafo G.



(b) Integração de a e b à AEM.



(c) Integração de c à AEM.



(d) Integração de d à AEM.

Figura 7: Passos para obtenção de uma árvore de extensão para o grafo G.

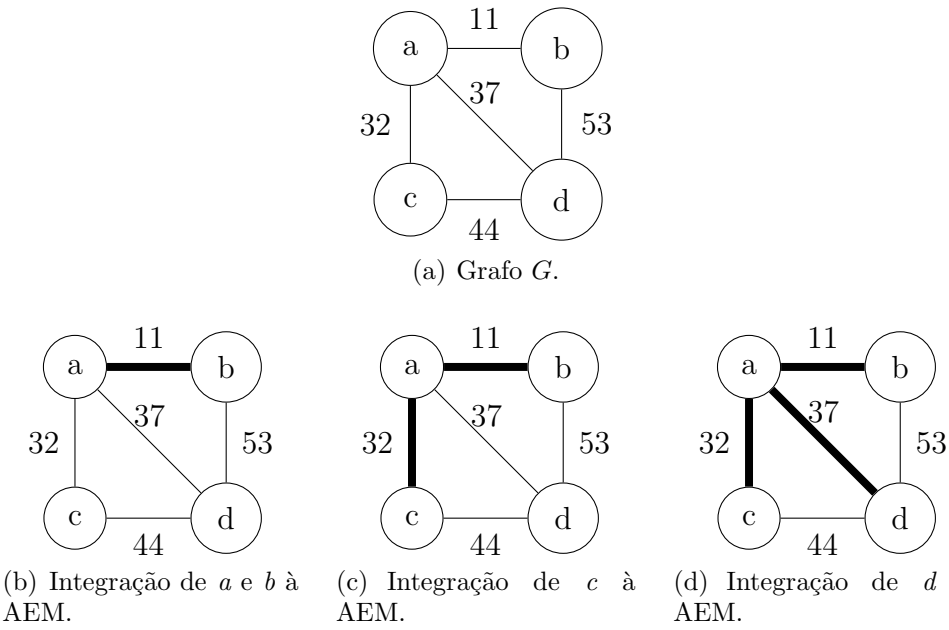
estrutura de dados de conjuntos disjuntos para determinar se uma aresta conecta nós de diferentes componentes, (KRUSKAL, 1956; LEISERSON et al., 2001). Esta estrutura de dados garante que as duas árvores conectadas não contém o mesmo vértice em suas composições. O processo pode ser verificado através do pseudocódigo descrito no Algoritmo 3.

Algoritmo 3 Algoritmo de Kruskal para geração de AEM

entrada $G = (V, A)$
Ensure: $AEM = (V, T)$
1: $T \leftarrow \oslash$
2: **para todo** $p \in V$ **faça**
3: $C_p \leftarrow p$
4: **fim para**
5: $Q = [a_1, a_2, ..., a_{\#A}] \mid a_i$ e $a_i \leq a_{i+1}$
6: **para** $i = 1$ até $\#A$ **faça**
7: Seja $a_i = (q, p)$
8: **se** $C_q \neq C_p$ **então**
9: $T \leftarrow T \in \{(q, p)\}$
10: $C_q \leftarrow C_q \cup C_p$
11: $C_p \leftarrow C_q \cup C_p$
12: **fim se**
13: **fim para**

Para exemplo de operação do algoritmo de Kruskal, é assumindo o grafo $G = (V, A)$ da Figura 8(a). Inicialmente o algoritmo executa a criação dos conjuntos unitários $C_a = \{a\}$, $C_b = \{b\}$, $C_c = \{c\}$ e $C_d = \{d\}$. Em seguida as arestas de A são ordenadas em

um conjunto q, apresentando como resultado $Q = \{(a,b),(a,c),...,(b,d)\}$. O algoritmo seleciona a primeira aresta do conjunto q, relativo ao par (a,b), e verifica se $C_a \neq C_b$. Como os elementos de cada um destes conjuntos são diferentes, a aresta selecionada, conforme verificado na Figura 8(b), é incluída no conjunto T referente às arestas da AEM. Antes da análise da próxima aresta os conjuntos C_a e C_b são atualizados com a união entre os conjuntos. A próxima aresta do conjunto q, dada por (a,c), é selecionada para que a comparação dos conjuntos seja realizada. Sendo satisfeita a condição a aresta passa a integrar o conjunto T, como pode ser verificado na Figura 8(c) e os conjuntos dos vértices são atualizados. Este processo é repetido até que todo o conjunto q tenha sido verificado, gerando como solução $AEM = (V,T)$ representada na Figura 8(d).



(a) Grafo G.

(b) Integração de a e b à AEM.

(c) Integração de c à AEM.

(d) Integração de d à AEM.

Figura 8: Passos para obtenção de uma árvore de extensão para o grafo G.

O algoritmo de Kruskal apresenta um melhor desempenho em relação ao algoritmo de Prim para encontrar a AEM, quando o grafo possui mais vértices que arestas (LEISERSON et al., 2001).

1.3.3 Algoritmo de GHS

Visando a aplicação do conceito de AEM em processamento distribuídos, os algoritmos distribuídos para localização das AEMs têm sido estudados desde 1977. Podemos encontrar dezenas de estudos realizados sobre este assunto onde, em 1983 foi publicado o trabalho que fundamenta o algoritmo distribuído conhecido como GHS (GALLAGER; HUM-

BLET; SPIRA, 1983). Esta publicação teve um impacto significante nas pesquisas sobre computação distribuída e ganhou, em 2004, o premio Edsger W. Dijkstra em Computação Distribuída.

O algoritmo de GHS (GALLAGER; HUMBLET; SPIRA, 1983) recebe este nome devido aos seus criadores, Gallager, Humblet e Spira. Ele foi concebido através da adaptação do algoritmo de Kruskal para o modo de comunicação assíncrono em processamento distribuído.

Em sua definição, para um grafo $G = (V, A)$, onde A é o conjunto de arestas conectadas e bidirecionais, ou seja, permitindo o deslocamento através da aresta no sentido direto e reverso, é assumido que cada vértice possui um peso único. Isto é, o valor de peso não se repete no grafo G, (GALLAGER; HUMBLET; SPIRA, 1983).

Assim, uma unidade de processamento que represente um vértice do grafo executa o algoritmo, que consiste basicamente em três estágios:

1. Envio de mensagens através das arestas adjacentes;
2. Espera pelo recebimento de mensagens respostas;
3. Processamento de mensagens recebidas. Note que para o bom funcionamento do algoritmo GHS, é preciso que as mensagens possam ser transmitidas de forma independente, em ambas as direções sobre uma mesma aresta e chegar no destino com um atraso imprevisível, porém finito, sem erros e em sequência.

O Algoritmo GHS emprega sete tipos de mensagens que são utilizadas na comunicação entre os vértices de um grafo. A seguir é explicado cada um destes tipos:

- `Conectar`: Mensagem enviada através da aresta de menor peso para realizar a união entre dois fragmentos;
- `Iniciar`: Esta mensagem contém um identificador do vértice que está enviando, e é disparada em *broadcast* para todos os vértices vizinhos. Esta mensagem fará com que cada vértice que a receber inicie a identificação de sua aresta de menor peso;
- `Teste`: Mensagem enviada de um vértice à outro a fim de verificar se este encontra-se em um mesmo fragmento;

- `Aceitar`: Mensagem enviada como resposta à mensagem do tipo `Teste` caso o vértice pertença a um fragmento diferente.

- `Rejeitar`: Mensagem enviada como resposta à mensagem `Teste` caso o vértice pertença ao mesmo fragmento;

- `Reportar`: Esta mensagem permite retornar a informação sobre a aresta com menor peso presente em seu fragmento;

- `Trocar`: Esta mensagem é utilizada para realizar a comunicação entre dois fragmentos distintos, através da aresta de menor peso que os conectam.

O algoritmo funciona através da realização de uma sequência de respostas para cada mensagem trocada. Para isso, o algoritmo emprega uma fila do tipo FIFO (*First In, First Out*) para armazenar as mensagens recebidas, viabilizando seus tratamentos uma a uma, em sequência.

Cada uma destas mensagens iniciam diferentes processo, que fazem uso das variáveis existentes nos vértice $q \in V$ descritas abaixo, como base do processo de determinação das arestas que compreendem a AEM.

- Estado do vértice ($EVertice_q$): utilizada para armazenar os estados possíveis do vértice;

- Estados das arestas adjacentes j ($EAresta_j$) do vértice q: utilizada para armazenar os estados possíveis de cada aresta j de um vértice;

- Aresta de fronteira ($AFront_q$): utilizada para armazenar a aresta que tenha o maior potencial de se tornar a ASMP do fragmento;

- Valor do peso associado à aresta de fronteira ($FrontPeso_q$):referente ao valor de peso da possível ASMP;

- Aresta de entrada ($AEnt_q$): variável utilizada para armazenar o RFID do remetente de uma mensagens `Iniciar` que esta sendo processada;

- Aresta em teste ($Ateste_q$): variável responsável por armazenar o vértice adjacente à aresta de peso mínimo dentre as aresta em estado `Básica`, da unidade de processamento que esteja executando o processo `Teste`;

- Contabilizador ($Cont_q$): variável utilizada para realizar a contabilização do número de arestas pelas quais uma mensagem do tipo `Iniciar` foi enviada pela própria unidade de processamento;
- Identificador do fragmento ($IDFrag_q$): utilizada para armazenar a identidade à qual a unidade de processamento pertence;
- Nível do Fragmento $NivFrag_q$: utilizada para identificar em cada um dos vértices qual o nível de fragmento ao qual ele pertence durante a execução do algoritmo GHS.

O algoritmo GHS trabalha através da cooperação de elementos chamados *fragmento* na busca da AEM do grafo. À cada vértice v possui seu nível *NivFrag* que caracteriza seu nível na busca da AEM. Inicialmente cada vértice tem o nível 0 e forma um *fragmento* no estado de `Adormecido`. Existem três diferentes estados, *EVertice*, em que um vértice pode se encontrar:

1. `Adormecido`: Estado inicial dos vértices onde o nível *IDFrag* é igual à zero;
2. `Busca`: Estado no qual o vértice se encontra enquanto está localizando sua aresta de menor peso;
3. `Encontrado`: Estado em que o vértice se encontra quando sai do estado `Adormecido` ou `Busca`.

O despertar de um vértice pode ocorrer de forma espontânea ou ao receber uma mensagem qualquer. Ao despertar, o vértice identifica qual sua aresta adjacente de menor peso e a caracteriza como `Ramo`, além de mudar seu estado *EVertice* para `Encontrado`. Os estados das arestas $EAresta_j$ podem ser caracterizados dentre os três tipos possíveis:

1. `Ramo`: Estado utilizado para classificar uma aresta com possibilidade de pertencer à AEM ao término do algoritmo;
2. `Rejeitada`: Estado que permite identificar que uma aresta não pertencerá a AEM gerada, uma vez que esta possibilitará a formação de um laço;
3. `Básica`: Estado utilizado para caracterizar uma aresta que ainda não foi classificada nem como `Ramo` nem como `Rejeitada`.

No inicio do algoritmo todas as arestas são caracterizadas como `Básica` e ao final do algoritmo não existirá nenhuma aresta neste estado.

Um vértice p, ao executar o processo de despertar, identifica sua aresta adjacente m de menor peso. Em seguida altera $EAresta_m$ para o estado `Ramo`. Através da aresta $m = (p, q)$ no estado `Ramo`, o vértice p solicita uma conexão, enviando uma mensagem do tipo `Conectar`, contendo como argumento o nível do fragmento, onde p se encontra, para o vértice q que se contra em outro fragmento. O Algoritmo 4 apresenta o processo de despertar de um vértice.

Algoritmo 4 Processo de despertar do vértice p

1: Seja $m \leftarrow (p, q)$ uma aresta adjacente de peso mínimo
2: $EAresta_m \leftarrow$ `Ramo`
3: $EVertice_p \leftarrow$ `Encontrado`
4: Envia mensagem `Conectar`$(NivFrag_p)$ via aresta m

E em seguida o vértice p passa a aguardar a resposta do vértice adjacente à aresta m.

Considerando que a aresta pela qual a mensagem foi enviada pelo vértice q é também a aresta de menor valor de peso do vértice vizinho p, a aresta selecionada é indicada como pertencente à AEM, o vértice tem seu nível incrementado e são conectados dando origem a um novo fragmento de nível $NivFrag'$.

Para os vértices com níveis maiores do que 0, supondo que um novo fragmento de nível $f + 1$ foi originado pela combinação de dois fragmentos de nível f através da aresta comum que conecta estes fragmentos. A aresta que foi utilizada para a conexão dos dois fragmentos de nível f passa a ser considerada como *núcleo* do novo fragmento de nível $f + 1$ e o peso associado à esta aresta é utilizado como *identidade* deste fragmento. Em seguida, os vértices adjacentes ao núcleo do fragmento iniciam um novo ciclo de busca, enviando mensagens do tipo `Iniciar` em *broadcast* para os demais vértices que compõem este fragmento. Este processo ocorre através da resposta à mensagem `Conectar` descrita no Algoritmo 5, onde c_j representa o valor do peso da aresta j e $Cont_q$ é uma variável utilizada na contabilização de mensagens do tipo `Iniciar` enviadas através de arestas no estado `Ramo`.

A mensagem `Iniciar` é recebida e retransmitida entre os vetores mais internos até os mais externos ao fragmento, contendo as informações do novo nível e da identidade do fragmento. A mensagem contém também um argumento que força os vértices que

Algoritmo 5 Resposta à mensagem $Conectar(NivFrag_p)$ na aresta j pelo vértice q

```
1: se EVertice_q = Adormecido então
2:    Executar processo de despertar
3: fim se
4: se NivFrag_p < NivFrag_q então
5:    EAresta_j ← Ramo
6:    Envia mensagem Iniciar(NivFrag_q, IDFrag_q, EVertice_q) via aresta j
7:    se EVertice_q = Busca então
8:       Cont = Cont + 1
9:    fim se
10: senão
11:    se EAresta_j = Bsica então
12:       Colocar a mensagem recebida ao final da fila
13:    senão
14:       Envia mensagem Iniciar(NivFrag_q + 1, c_j, Busca) via aresta j
15:    fim se
16: fim se
```

a recebem a iniciarem o processo de busca por uma aresta que possam conectar este fragmento à um outro. Caso ainda existam fragmentos de nível F aguardando resposta à solicitação de conexão de vértices que passaram a integrar o fragmento $L+1$, a mensagem enviada em *broadcast* também será recebida por este vértice fazendo com que este se integre ao fragmento $F+1$. Conforme o processo descrito no Algoritmo 6, onde a variável $AEnt_q$ é utilizada para armazenar o vértice remetente da mensagem `Iniciar` que esta sendo processada, a variável $AFront_q$ utilizada para armazenar a aresta que tenha o maior potencial de se tornar a ASMP do fragmento, a variável *AFrontPeso* referente ao valor de peso da possível ASMP e i representa as arestas diferentes de j.

Quando um vértice q recebe a mensagem `Iniciar`, ele começa sua busca pela Aresta de Saída de Menor Peso (ASMP), que representa a aresta de menor peso que irá conectar este fragmento formado a um outro fragmento através de um vértice p.

No processo de busca pela ASMP, o vértice q seleciona uma aresta de menor peso dentre as arestas que ainda estão no estado `Básica`, e envia através dela uma mensagem do tipo `Teste`. Esta mensagem carrega em seu conteúdo o nível $NivFrag_q$ e a identidade $IDFrag_q$ do fragmento como argumentos. O processo de teste está descrito através do Algoritmo 7, nele é utilizado uma variável $Ateste_q$ responsável por armazenar a aresta de peso mínimo dentre as aresta em estado `Básica`, que está sendo testada como uma possível ASMP e o processo reportar será abordado ainda nesta subseção.

Quando um vértice recebe uma mensagem `Teste`, ele verifica se seu nível de frag-

Algoritmo 6 Resposta à mensagem $Iniciar(NivFrag_p, IDFrag_p, EVertice_p)$ na aresta j pelo vértice q.

1: $NivFrag_q \leftarrow L$
2: $IDFrag_q \leftarrow F$
3: $EVertice_q \leftarrow S$
4: $AEnt_q \leftarrow j$
5: $AFront_q \leftarrow \oslash$
6: $AFrontPeso_q \leftarrow \infty$
7: **para todo** $i \neq j \mid EAresta_i =$ Ramo **faça**
8: Envia mensagem `Iniciar`$(NivFrag_q, IDFrag_q, EVertice_q)$ na aresta i
9: **se** $EVertice_p = Busca$ **então**
10: $Cont_q \leftarrow Cont_q + 1$
11: **fim se**
12: **fim para**
13: **se** $EVertice_q = Busca$ **então**
14: Executar processo de teste
15: **fim se**

Algoritmo 7 Processo de teste executado pelo vértice q

1: **se** Existe arestas adjacentes de peso mínimo no estado $Bsica$ **então**
2: $Ateste_q \leftarrow$ aresta de peso mínimo no estado `Básica`
3: Envia mensagem `Teste`$(NivFrag_q, IDFrag_q)$ via aresta $Ateste_q$
4: **senão**
5: $Ateste_q \leftarrow \oslash$
6: Executar processo de reportar
7: **fim se**

mento é igual ao recebido na mensagem. Caso seja, o vértice envia uma mensagem do tipo `Rejeitar` de volta ao remetente da mensagem e executa um novo processo de teste. Como envio da mensagem `Rejeitar` ao remetente, ambos os vértices, remetente e destinatário, categorizam a aresta que os conecta como `Rejeitada`.

Caso o vértice que esteja recebendo uma mensagem `Teste`, tenha seu identificador do fragmento diferente do recebido na mensagem e, se o nível de fragmento do vértice for maior ou igual do que o recebido na mensagem, o vértice enviará ao remetente desta mensagem uma mensagem do tipo `Aceitar`, garantindo que a aresta testada é uma possível ASMP. Por outro lado, caso o nível do fragmento do vértice que está recebendo a mensagem for menor do que o nível do fragmento presente na mensagem `Teste`, a mensagem será colocada ao final da fila de mensagens. Este processo encontra-se descrito pelo Algoritmo 8.

O processo de resposta às mensagens `Aceitar` e `Rejeitar`, estão descritos nos Algoritmos 9 e 10, respectivamente.

Algoritmo 8 Resposta à mensagem $Teste(NivFrag_p, IDFrag_p)$ na aresta j, pelo vértice q

```
1: se EVertice_q = Adormecido então
2:    Executar processo de despertar
3: fim se
4: se NivFrag_p > NivFrag_q então
5:    Enfileirar mensagem
6: senão
7:    se IDFrag_p ≠ IDFrag_q então
8:       Envia mensagem Aceitar via aresta j
9:    senão
10:      se EAresta_j = Básica então
11:         EAresta_j ← Rejeitada
12:         se Ateste_q ≠ j então
13:            Envia mensagem Rejeitar via aresta j
14:         senão
15:            Executa processo teste
16:         fim se
17:      fim se
18:   fim se
19: fim se
```

Algoritmo 9 Resposta à mensagem *Aceitar* na aresta j, pelo vértice q

```
1: Ateste_q ← ⊘
2: se c_j < AFrontPeso_q então
3:    AFront_q ← j
4:    AFrontPeso_q ← c_j
5: fim se
6: Executa processo reportar
```

Algoritmo 10 Resposta à mensagem *Rejeitar* na aresta j, pelo vértice q

```
1: se EAresta_j = Bsica então
2:    EAresta_j ← Rejeitada
3:    Executa processo teste
4: fim se
```

Após cada vértice ter encontrado as possíveis ASMP, todos os vértices cooperarão pelo processo de reportar, descrito no Algoritmo 11, através do envio da mensagem `Reportar` na identificação da ASMP de todo o fragmento e passará para o estado de `Encontrado`. Caso não existam arestas de saída com peso mínimo o algoritmo está terminado e o fragmento é a própria AEM.

Enquanto que, cada vértice mais interno do fragmento irá aguardar até ter concluído tanto a sua busca pela sua própria ASMP, como ter recebido as mensagens `Reportar` através das arestas mais externas categorizadas como `Ramo`. O vértice então elencará o

menor peso dentre, o peso da ASMP encontrada e os pesos recebidos através das mensagens `Reportar` recebidas. A variável $AFront_q$ do vértice q será carregada com a aresta por onde trafegou a mensagem `Reportar` contendo o menor peso, ou com o vértice adjacente à ASMP, caso o peso desta aresta seja menor do que os pesos recebidos pelo processo de reportar. Em seguida o vértice enviará a mensagem `Reportar` e passará para o estado de `Encontrado` indicando o término de sua busca pela ASMP.

Algoritmo 11 Processo de reportar executado pelo vértice q

1: **se** $Cont_q = 0$ e $Ateste_q = \oslash$ **então**
2: $EVertice \leftarrow$ `Encontrado`
3: Envia mensagem `Reportar`$(AFrontPeso_q)$ via aresta $AFront_q$
4: **fim se**

Eventualmente, os vértices adjacentes à aresta *núcleo* enviam uma mensagem `Reportar` através desta aresta para determinar a ASMP do fragmento e em qual lado do núcleo esta aresta se faz presente.

Após o envio da mensagem `Reportar` pelos dois vértices a aresta *núcleo*, é iniciado o processo de conexão dos dois fragmentos através da ASMP para gerar um novo fragmento. O vértice do *núcleo* mais próximo da ASMP descoberta envia a mensagem `Trocar` em direção à aresta ASMP. O transito desta mensagem partindo do vértice do *núcleo* até a ASMP é possível graças ao mapeamento realizado em cada um dos vértices, durante o processo reportar, através da variável $AFront_q$ que permite traçar este caminho a ser percorrido. Quando o vértice do fragmento que possui a aresta de saída de menor peso que conecta este fragmento à um outro recebe a mensagem `Trocar`, ele envia uma mensagem `Conectar` ao seu vértice vizinho e adjacente a ASMP.

Após o envio da mensagens `Conectar` através da ASMP, duas possibilidades podem ocorrer:

- Caso estes dois fragmento estejam em um mesmo nível $NivFrag$, a troca de mensagens `Conectar` através da ASMP ocorrerá em ambas as direções fazendo com que esta aresta se torne o novo *núcleo* do fragmento gerado que possuirá o nível $NivFrag + 1$, e inicia-se a produção de mensagens `Iniciar` contendo como argumentos os valores de nível e identidade do novo fragmento formado;

- Caso a mensagem `Conectar` seja enviada por vértice que pertence a um fragmento de nível $NivFrag'$ e identidade do fragmento igual à $IDFrag$' a um vértice que

pertença à um fragmento de nível $NivFrag$, onde $NivFrag > NivFrag'$, e identidade do fragmento igual à $IDFrag$, o vértice do fragmento de maior nível procederá com o envio de uma mensagem `Inciar` como resposta à mensagem `Conectar` recebida. Esta mensagem contém como argumentos o nível $NivFrag$ e a identidade $IDFrag$ do fragmento de maior nível e fará com que o fragmento de menor nível seja encorporado ao fragmento de maior nível e passe a cooperar na busca da ASMP.

O processo de resposta à mensagem `Reportar`, o processo de geração da mensagem `Trocar` e o processo de resposta à mensagem `Trocar` estão descritos nos Algoritmos 12, 13 e 14, respectivamente.

Algoritmo 12 Resposta à mensagem *Reportar*($AFrontPeso_p$) na aresta j, pelo vértice q

```
1: se j ≠ AEnt_q então
2:    Cont_q ← Cont_q − 1
3:    se AFrontPeso_p < AFrontPeso_q então
4:       AFrontPeso_q ← AFrontPeso_p
5:       AFront_q ← j
6:       Executa processo reportar
7:    fim se
8: senão
9:    se EVertice_q = Buscar então
10:      Enfileirar mensagem
11:   senão
12:      se AFrontPeso_p > AFrontPeso_q então
13:         Executa processo trocar
14:      senão
15:         se AFrontPeso_p = AFrontPeso_q = ∞ então
16:            Parar
17:         fim se
18:      fim se
19:   fim se
20: fim se
```

Algoritmo 13 Processo de trocar.

```
1: se EAresta_AFront = Ramo então
2:    Envia mensagem Trocar na aresta AFront_q
3: senão
4:    Envia mensagem Conectar(NivFrag_q) na aresta AFront_q
5:    EAresta_AFront ← Ramo
6: fim se
```

A ilustração da Figura 9 mostra um exemplo de grafo G com pesos associados às arestas, para demonstração do processo de formação de novos fragmentos até a formação

Algoritmo 14 Resposta à mensagem *Trocar*

1: Executa processo trocar

da AEM. Todos os vértices encontram-se com o nível de fragmento igual à 0, ou seja, cada vértice representa um fragmento.

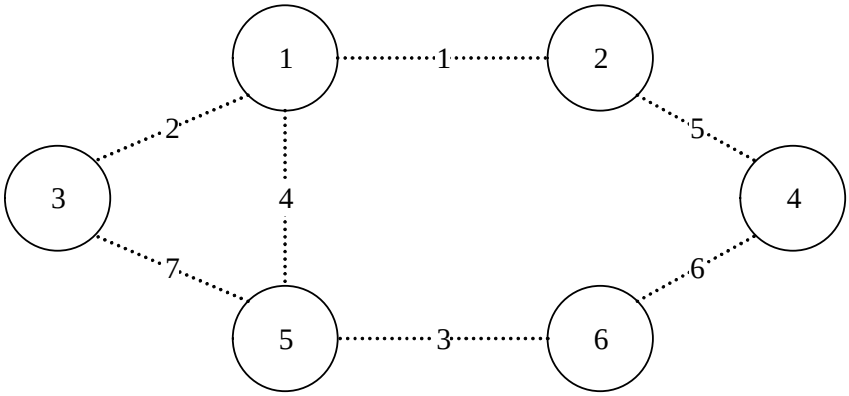


Figura 9: Grafo G.

Supondo que todos os vértices executaram seu processo de despertar, a Figura 10 ilustra os fragmentos de nível 1, formado pelos pares $(1, 2)$ e $(5, 6)$ se conectaram através da troca de mensagens `Conectar` gerando os fragmentos F' e F.

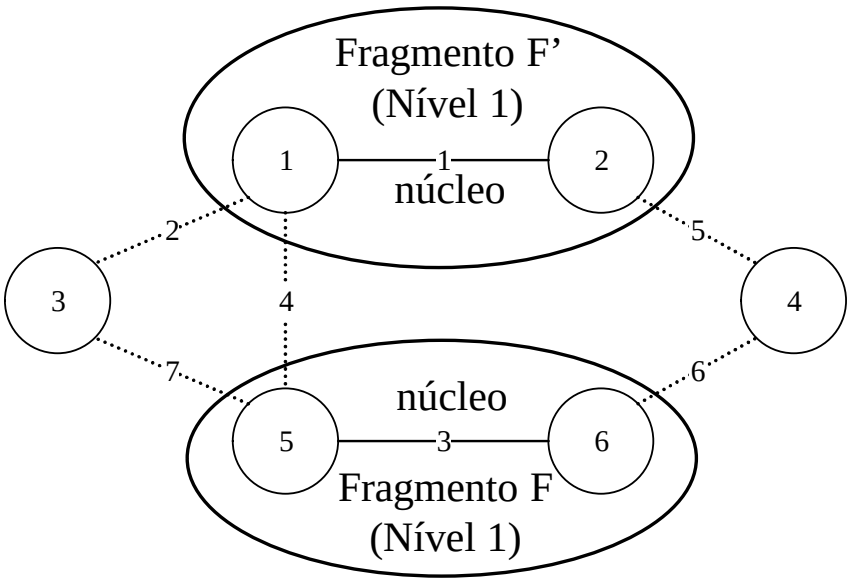


Figura 10: Formação dos fragmentos F' e F.

Na Figura 11 é apresentado o processo de incorporação das arestas de pesos 2 e 5, através do envio das mensagens `Iniciar` pelos vértices 1 e 2 aos vértices 3 e 4 que estavam aguardando a resposta da mensagem `Conectar` enviadas durante o processo de despertar. Por último, pode ser visto na Figura 12 a formação de um novo fragmento F''

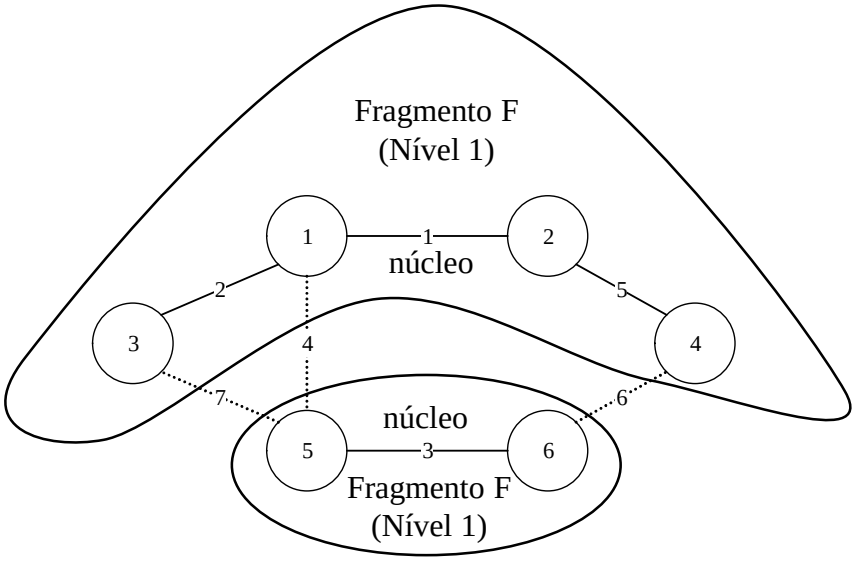


Figura 11: Encorporação dos vértices 3 e 4 ao fragmento F.

através da união dos fragmentos F e F' através da ASMP representada pela aresta de peso 4.

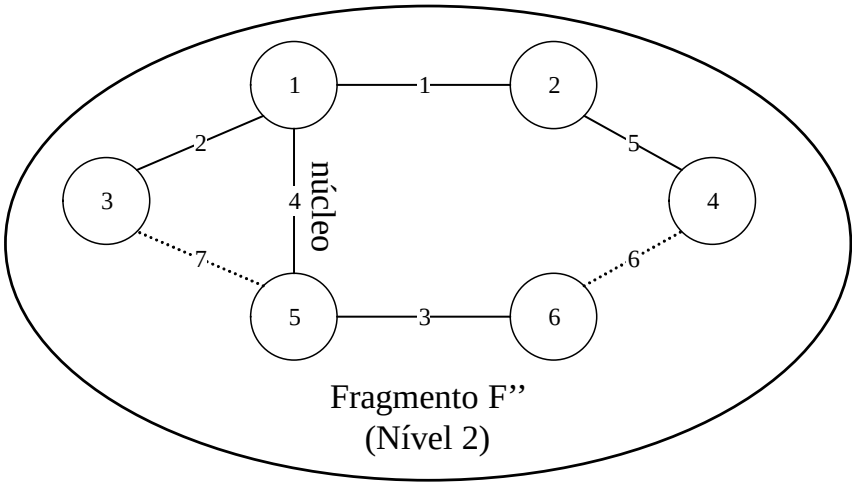


Figura 12: Formação do fragmento F'' representando a AEM do garfo G.

1.4 Considerações Finais do Capítulo

Neste capítulo foram introduzidos os conceitos sobre grafo, árvore e árvores geradoras como base para viabilizar o estudo de modelagem das redes de distribuição de energia elétrica, para execução de algoritmos que possam gerar uma reconfiguração da rede em casos de falhas.

Foram então apresentados os algoritmos de maior aplicabilidade, Prim e Kruskal, na localização da árvore de cobertura mínima de grafos, que corresponde a um grafo

conexo e acíclico.

Além destes dois algoritmo, o capítulo traz uma abordagem aprofundada sobre o funcionamento do algoritmo distribuído GHS, que fora desenvolvido com base no algoritmo de Kruskal.

No próximo capítulo serão apresentados os conceitos básicos e definições definições de grafos, árvores e árvores geradoras. Também serão abordados os algoritmos mais difundidos e utilizados para a geração das árvores de extensão mínima. Será apresentado e detalhado o algoritmo GHS, utilizado como base para o desenvolvimento da solução de auto-cura proposta nesta dissertação.

Capítulo 2

TRABALHOS RELACIONADOS

NESTE capítulo são levantados os trabalhos mais relevantes ao projeto desta dissertação de mestrado, onde foram apresentados algoritmos para a solução do problema de reconfiguração de redes de distribuição utilizando AEM. Além disto outros trabalhos que empregam essa técnica para solução de outros tipos de problemas reais também são comentados.

2.1 Reconfiguração de Redes Usando AEMs

Nesta seção são apresentadas as adaptações que surgiram a partir dos algoritmos básicos de geração de AEM.

2.1.1 Utilização do Algoritmo de Kruskal

Um algoritmo eficiente para localização de árvore de extensão mínima com a finalidade de solucionar o problema de reconfiguração de redes de distribuição de energia elétrica, foi proposto em (MONTOYA; RAMIREZ, 2012). Este tipo de problema possui uma alta complexidade, combinacional, devido ao grande número de chaves que podem ser comutadas. Além disso, a restrição de radialidade típica da rede acrescenta uma complexidade adicional ao problema de reconfiguração das redes de distribuição de grande porte.

Para sua aplicação é necessário que, no mínimo, os dados de topologia da rede, os estados das chaves (aberto/fechado) e os parâmetros dos alimentadores (resistência e reatância indutiva) estejam disponíveis, pois são estes dados que constitui as informações dos pesos de cada caminho a ser analisado para a localização da árvore de extensão mínima.

A solução apresentada baseou-se na minimização das perdas de energia através da tomada de decisão sobre a comutação das chaves na rede de distribuição, com a finalidade de satisfazer a demanda do consumidores. O algoritmo de *Kruskal* (KRUSKAL, 1956), foi utilizado na localização da comutação ótima com o menor valor de perda de energia para a rede reconfigurada.

O algoritmo desenvolvido foi aplicado em a duas configurações de rede distintas para testes e validação. No primeiro teste foi utilizada uma rede de 12,66 kV, hipotética, com 33 nós, 37 ramos, 5 chaves de manobra, potência ativa e reativa das cargas no sistema sendo 3715 kW e 2300 kVAr, respectivamente, e a perda de potência do sistema igual à 197,379 kW. Após a aplicação do algoritmo de reconfiguração a perda da energia ativa passou a ser de 138,825 kW, equanto que a tensão mínima no sistema passou de 0,9178 *p.u* para 0,9593 *p.u*. Em comparação a outros 4 métodos aplicados ao mesmo sistema o algoritmo apresentou a melhor capacidade de minimizar as perdas de potência do sistema e se equiparou aos outros métodos quanto a minimização das perdas de tensão (V_{min}). Para o segundo teste foi utilizado uma rede, também hipotética, de 12,66 kV com 69 nós, 73 ramos, 5 chaves de manobra, potência ativa e reativa das cargas no sistema sendo 3801,5 kW e 2694,6 kVAr, respectivamente e perda de potência do sistema igual à 204,45 kW. Com a aplicação do algoritmo as perdas da energia ativa passou a ser de 80,868kW, e a tensão mínima do sistema passou de 0,9178 *p.u* para 0,9593 *p.u*. Quando comparado aos outros 4 métodos aplicados ao mesmo sistema o algoritmo apresentou a melhor capacidade de minimização das perdas da tensão mínima e gerou a quarta melhor minimização de perda de potência.

Já no trabalho de (SUDHAKAR; SRINIVAS, 2011b), é descrita a implementação de uma metodologia, também baseada no algoritmo de *Kruskal* (KRUSKAL, 1956), para resolver o problema de restauração do sistema de distribuição, considerando um número qualquer de falhas simultâneas, com o intuito de determinar a reconfiguração ótima de operação do sistema e restaurar o valor máximo das cargas afetadas após uma falha.

Para sua operação, o algoritmo, utiliza como dados de entrada os estados das chaves de seccionamento e manobra, além do valor da impedância dos cabos como valor dos pesos associados às arestas. Assim a solução permite a maximização do fluxo de energia nas linhas de transmissão, fazendo com que as perdas de toda a rede sejam reduzidas.

O algoritmo desenvolvido foi aplicado no modelo do IEEE para Sistemas de Dis-

tribuição de Energia Elétrica com 16 barramentos para falhas presentes em apenas uma ou duas linhas, onde não houve a necessidade de calcular as perdas separadamente, pois o algoritmo possui a capacidade de reduzir a impedância total da rede durante o processo de restauração. Após a geração da rede resultante, a implementação proposta verifica a presença de sobrecargas nos alimentadores, decidindo se haverá necessidade de execução de deslocamento ou corte de cargas.

Os autores também apresentaram a implementação em *hardware* do algoritmo para a restauração do sistema de distribuição de uma rede composta de 16 barramentos, utilizando a linguagem *assembly* para programação de um microcontrolador 8051. Usando chaves seletoras, foram inseridos os estados das linhas de transmissão para todas as possibilidades de falha e o resultado das reconfigurações é apresentada através de LEDs associados ao estados das chaves.

2.1.2 Utilização do Algoritmo de Prim

Já em (SUDHAKAR; SRINIVAS, 2011a), foi apresentado o desenvolvimento de um algoritmo para o planejamento e operação do serviço de restauração da rede de distribuição de energia elétrica baseado na busca do melhor caminho do fluxo de potência da rede de distribuição.

A função objetivo apresentada neste trabalho é a maximização da quantidade de energia restabelecida dada uma falha na rede de distribuição de energia elétrica, enquanto as restrições de tensão, radialidade, perdas nas linhas, carga e prioridade dos consumidores são satisfeitas.

O serviço de restauração do sistema foi caracterizado como um problema de otimização combinacional de alta complexidade devido ao grande número de chaves presentes na rede de distribuição. Tal característica pode levar à um longo período de busca para a obtenção de um plano de reconfiguração viável e que satisfaça todas as restrições. A fim de contornar esta adversidade, o algoritmo de Prim foi utilizado para prover o a comutação das chaves da rede de distribuição.

O algoritmo de Prim foi aplicado ao modelo de 16 barramentos do IEEE, (BARRY; LO, 1994), assumindo que o peso de cada aresta do grafo que representa a rede é igual à impedância equivalente de cada ramo da rede de distribuição. Como resultado foi observada uma rede resultante, onde o alimentador3 apresentava 100% de sua capacidade,

o alimentador2 98% e o alimentador1 108% e os limites de corrente nas chaves de cada alimentador (S1, S6 e S12) são respectivamente 100%, 150% (valores máximos aceitáveis pela concessionária local, a Tamil Nadu Electricity Board) e 50% de seus valores nominais. Estes resultados foram comparados aos resultados de duas redes resultantes geradas por outros 2 métodos. A implementação proposta apresentou a segunda maior minimização de perda de potência e a maior minimização da perda de tensão na rede.

Foi demonstrado a necessidade de se garantir a execução de cortes de cargas na rede resultante, caso uma falha ocorra na saída de um dos alimentadores, visto que toda a carga alocada no alimentador que entrou em falha deverá ser deslocada para os outros alimentadores que compõem a rede, o que poderá causar uma sobrecarga.

2.1.3 Utilização do Algoritmo de Dijkstra

Para a localização de uma solução ótima em alta velocidade, foi apresentado por (SUDHAKAR et al., 2004), um algoritmo para comutação das chaves da rede de distribuição com o objetivo de reduzir o tempo de restauração da rede em caso de falhas, isolando somente a sessão onde a falha ocorreu e restaurando a energia da rede. Este algoritmo foi desenvolvido para atuar através da alteração dos estados das chaves dos alimentadores incluídos na rede de distribuição, alterando assim suas configurações. Com isto as tensões das linhas e dos barramentos são redistribuídas e as perdas da rede são minimizadas. O método desenvolvido é baseado no algoritmo de *Dijkstra* (BLACK, 2006), este algoritmo apresenta o menor tempo de resposta para a localização da árvore de cobertura mínima se comparado à outros métodos como a programação dinâmica ou a programação linear.

Neste caso o problema da reconfiguração da rede de distribuição é solucionado operando tanto nas chaves de seccionamento como nas de manobra da rede. Assim, a implementação proposta faz uso de uma árvore de decisão binária para representar os estados das chaves da rede após o processo de reconfiguração

A implementação foi testado em uma rede radial composta por 33 barramentos, 32 linhas e 5 chaves de manobra. Inicialmente foi aplicada uma técnica de redução de rede fazendo com que o sistema passa a ser visto como uma rede composta de 14 barramentos, para redução do tempo de busca da rede de cobertura. Resultados para todas as possibilidades de falhas múltiplas ou isoladas foram apresentados.

2.1.4 Utilização do Algoritmo de Reverse Delete

Outro algoritmo foi desenvolvido e apresentado em (SUDHAKAR; SRINIVAS, 2011c). Seu objetivo consiste na maximização da energia restaurada para as áreas isoladas, após uma falha na rede e capacidade de transferência de potência além da minimização de perdas na rede resultante e do tempo de duração do processo de restauração, satisfazendo as restrições imposta para a rede de distribuição.

Este algoritmo trabalha na localização do caminho ideal do fluxo de potência para uma rede de distribuição, para que caso ocorra uma falha na rede a energia possa ser parcial ou totalmente restabelecida através da comutação das chaves da rede. Tendo sido embasado no algoritmo *Reverse Delete*, por possuir a capacidade de localizar o melhor caminho para o fluxo de potência com base nos valores de impedância dos cabos da rede. Isso confere à rede resultante gerada o menor valor de impedância resultante sendo este diretamente convertido em menores índices de perdas de tensão e potência, proporcionando a maximização de transferência de energia útil dentro do sistema.

A implementação proposta foi testada e os resultados da sua aplicação para dois tipos de configurações de rede de distribuição com falhas em um dos barramentos foram apresentado. A primeira rede é composta por apenas um alimentador e 33 barramentos e a segunda 3 alimentadores e 16 barramentos. A implementação foi especificada em *hardware* utilizando a linguagem VHDL e sintetizada usando a placa de prototipagem XILINX SPARTAN 3.

2.2 Reconfiguração de Rede Baseada em Outros Processos de Busca

2.2.1 Adaptação do Algoritmo de Bellman-Ford

Outra forma de abordagem para solução da localização da reconfiguração ótima da rede de distribuição dada a presença de uma falha é (YUNHAI; YONG, 2002). Neste trabalho é apresentado um algoritmo onde o processo de restauração é dividido, segundo os autores, em ”energização em série”e ”energização em paralelo”através de adaptações de algoritmos de Bellman-Ford e de busca de árvore de cobertura mínima, ambos de caráter polinomial para resolução do problema combinacional, característica da reconfiguração da rede.

O processo de funcionamento do algoritmo é transcorrido da seguinte forma: na primeira fase, os barramentos que possuem cargas prioritárias são energizados um a um, sempre respeitando as restrições consideradas para cada sistema, que ocorrem por conta do reduzido número de barramentos energizados. Para isto, o algoritmo trabalha na localização do caminho ideal de todas as cargas em série por fase. Já na segunda fase, o sistema é dividido em vários subsistemas e restaurado individualmente, devido a restrição de tensão nos barramentos, e algumas cargas não prioritárias passam a ser restauradas uma a uma. Havendo um empate na decisão entre o fechamento de mais de uma chave de manobra, o algoritmo deixa a decisão à cargo do profissional de despacho.

Para acelerar o processo de reconstrução da rede, o custo das linhas são considerados. Tal custo é baseado em fatores como influência do valor de indutância resultante, tempo de operação mão-de-obra, estados dos equipamentos, capacidade de transmissão, etc. Porém neste trabalho o custo foi baseado nas admitâncias das linhas e as reatâncias paralelas foram utilizadas na composição das cargas da rede.

É resaltado pelos autores que o algoritmo tem o propósito de apenas encontrar o fluxo ótimo de potência, que as linhas não são energizadas ao mesmo tempo, e que cada conexão deve atender todas as restrições de segurança para a rede.

2.2.2 Solução Baseada em Algoritmos Genéticos

Além das propostas embasadas na teoria dos grafos, outras propostas para a solução da reconfiguração das redes de distribuição foram investigadas.

Uma das soluções que se destacaram durante os estudos foi a metodologia apresentada em (FERREIA et al., 2013), onde o trabalho apresentado foca na reconfiguração a partir de uma ou mais falhas em alimentadores interconectados utilizando o algoritmo genético para manipular as chaves de seccionamento e manobra da rede. Além disto, o algoritmo visa minimizar o número de manobras, conservando a vida útil dos equipamentos e facilitando o processo de reconfiguração, e maximizar a carga total atendida por um dado chaveamento, levando em consideração o tempo e a frequência de interrupções ocorridas na rede.

A operação do algoritmo utiliza como base a informação dos estados das chaves pré e pós-falta de modo a inferir onde a falha está localizada. Com base nesta informação define-se o espaço de busca de soluções e o número de variáveis de decisão para realização

da reconfiguração das chaves. Os autores tomaram duas premissas como verdadeiras, onde a primeira considera a devida coordenação das proteções que compõem a rede garantindo o estado de pós-falta e a segunda considera a necessidade de isolamento da falha para que as chaves que estão diretamente ligadas ao trecho em falha sejam excluídas do campo de busca do processo de otimização.

Para validação da proposta, os autores, aplicaram o algoritmo a um cenário modelado, com dados reais, composto de 5 alimentadores, 2 chaves telecomandadas para cada alimentador, um disjuntor em cada subestação e 6 chaves de manobra. Todos os alimentadores apresentavam, em sua configuração normal, tensão média de 13,0 kV, além de cargas e extensão: 1) 2,90 MVA e 3,76 km; 2) 5,62 MVA e 6,65 km; 3) 6,42 MVA e 9,71 km; 4) 5,42 MVA e 6,45 km; 5) 6,33 MVA e 7,96 km.

Os testes foram aplicados a 3 cenários distintos de falha, para verificação de sua eficácia tanto para casos de falhas simples ou simultâneas. Para os 3 casos o algoritmo convergiu para o chaveamento ótimo em menos de 30 iterações sem a necessidade de alteração da parametrização inicial.

Foi apontado pelos autores que devido ao comportamento variável das cargas se faz necessário um estudo preliminar da rede, além da possibilidade de implementação de melhorias no algoritmo considerando a minimização de perdas técnicas durante o processo de otimização.

2.3 Planejamento de Redes Através de Árvores de Extensão Mínima

2.3.1 Rede de Distribuição de Gás

Um estudo de caso é apresentado em (MAHDAVI et al., 2010) para a criação de uma programação linear com o intuito de gerar uma configuração ótima da rede de distribuição de gás que ligará as estações de redução de pressão e os consumidores, utilizando a técnica das árvores de extensão mínima. Seu objetivo foi determinar os locais e tipos de estações minimizando os custos de localização e alocação na rede.

Para utilização do método de minimização de custos para alocação dos elementos da rede, os autores desenvolveram modelos matemáticos que resultaram em, 4 funções que refletem os custos de localização/alocação e 32 funções de restrições que formularam os pesos do grafo.

Para a validação do método proposto foi considerado no trabalho um modelo de distribuição da companhia de gás Mazandaran, composto de 119 consumidores com variadas demandas. Através da aplicação da solução propostas foram identificados 9 potenciais localidades para alocação das estações abaixadoras de pressão.

2.3.2 Rede de Distribuição de Energia Elétrica

Um método para o planejamento de uma rede de distribuição de energia é proposto em (WENYU et al., 2004). Este foi baseado e adaptado do algoritmo de Kruskal que interpreta cada dois nós conectados em pontos de carga. Vale ressaltar que o método proposto é estendido para o planejamento de redes em malha, determinando redes mais curtas e adequando a seção transversal dos cabos tronco e ramos. Outro ponto importante do algoritmo é a minimização das perdas e das quedas de tensão verificadas na rede resultante.

Para a operação do algoritmo os pontos de carga são transformados em nós do grafo, todos os caminhos possíveis que ligam os pontos de carga formam as arestas e os custos de construção das linhas (material, mão de obra etc) são tratados como os valores de peso de cada aresta. Enfim o algoritmo de Kurskal é aplicado ao grafo gerado para a obtenção da árvore de cobertura de custo mínimo que refletirá diretamente no esquema a ser construído com o custo de investimento mínimo.

Para a utilização do método de localização da configuração ótima de redes radiais (multiplas fontes), os autores propõem uma adaptação no algoritmo através da utlização do algoritmo de Warshall, que trabalhará com a finalidade de encontrar o caminho mais curto para cada ponto de carga, compondo um conjunto T. Em seguida, o aloritmo de Kruskal é aplicado ao grupo de arestas formadas pelo conjunto T e pelas as arestas formadas pelos nós transversais da rede com a finalidade de conectar todos os pontos de carga.

Muitas vezes, no planejamento de uma rede nova, faz-se uso de uma rede já existente como base. No trabalho foram apresentados dois métodos de operação: I) Caso a rede antiga não sofra reconstrução, o algoritmo considera que os nós de conexão da linha antiga também são pontos de carga e durante o processo de planejamento nenhuma carga é adicionada a rede antiga; II) Caso a rede antiga seja parcialmente reconstruída, as linhas que passarão pela reconstrução são excluídas, então os nós de conexão e as arestas que os conectam são considerados durante o processo de planejamento. Para a parte da rede que

não sofrerá reconstrução, nenhuma carga é adicionada, mantendo sua capacidade máxima respeitada.

É apontado pelos autores a importância de se atentar ao planejamento da alocação dos dispositivos de seccionamento da rede, pois sua presença em rotas muito curtas entre cada alimentador implica em um número expressivo de subredes radias de única fonte. Portanto, deve-se considerar a capacidade máxima da linha como condição limite e verificar a carga total ligada a cada subrede. Outro ponto importante sinalizado, é o fato de que o algoritmo é focado somente na localização da rota com menor custo de investimento e que nem sempre a rede resultante será a melhor em termos, por exemplo, de minimização de perdas de tensão. E para mitigar tais problemas, após a geração da rede resultante a seção dos cabos pode ser devidamente adequada pela equipe de planejamento.

2.4 Considerações Finais do Capítulo

Neste capítulo expostos apresentados os trabalhos mais relevantes ao projeto desta dissertação de mestrado, onde foram apresentados algoritmos para a solução do problema de reconfiguração de redes de distribuição utilizando AEM.

Outros trabalhos focados na solução de reconfiguração de rede de energia elétrica utilizando outros processos de busca também foram apresentados. Outros trabalhos voltados para solução de problemas de replanejamento de redes de elétricas e gás também foram apresentados.

O algoritmo de GHS foi utilizado como base no desenvolvimento da solução de auto-cura que será apresentada no capítulo seguinte.

Capítulo 3

IMPLEMENTAÇÃO DESCENTRALIZADA DA AUTO-CURA DE REDES DE DISTRIBUIÇÃO INTELIGENTES

ESTE capítulo apresenta uma implementação da técnica de auto-cura, além de otimização de perdas, para Redes de Distribuição Inteligentes (RDIs) de Energia Elétrica de forma descentralizada, desenvolvida com base no algoritmo de GHS apresentado no Capítulo 1, e sua implementação em unidades de processamento distribuído. No decorrer deste capítulo, serão apresentados também a estruturação dos dados utilizada pelo algoritmo e arquitetura de comunicação que fora desenvolvida para permitir a troca de mensagens entre as unidades de processamento, necessárias para o funcionamento da solução.

3.1 Modelagem de RDIs por Grafos

O esquemático de uma rede elétrica de distribuição, segundo um conceito simplista, denota o esquema de ligação física das fontes geradoras de energia às cargas consumidoras. A Figura 13 apresenta um exemplo esquemático de rede de um *Alimentador G*. Este *Alimentador* provê energia à um número de *Barramentos de Distribuição NB* igual à 4, representados pelo conjunto $\{B_1, B_2, B_3, B_4\}$. Estes barramentos tem a função de efetuar a segmentação do sistema de distribuição e é onde estão localizados as cargas consumidoras, os elementos de seccionamento do sistema e os *Cabeamentos* de conexão, também denominados *Linhas* de conexão. Estas Linhas conectam cada barramento à seu barramento vizinho, permitindo o fluxo de energia entre o *Alimentador* e toda a rede. Na

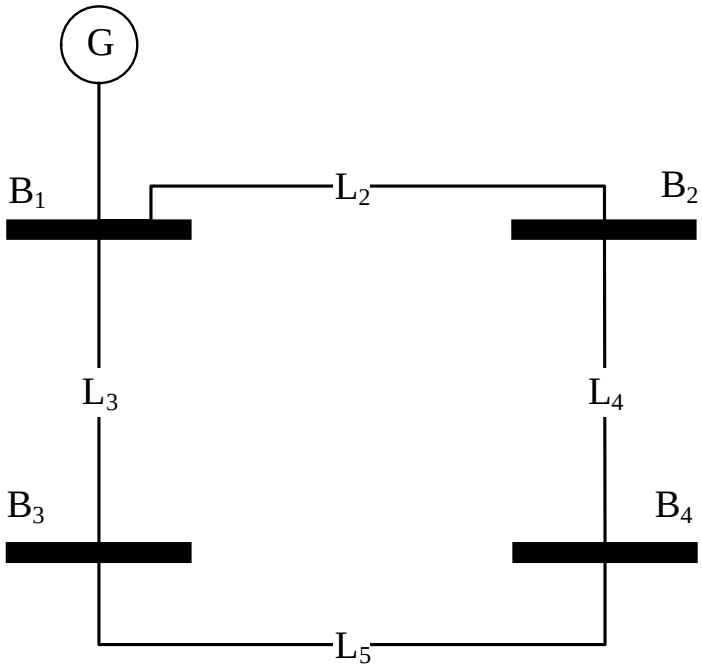


Figura 13: Exemplo de rede de distribuição simplificada.

Figura 13, analisando o barramento B_1, é verificado que seus barramentos vizinhos são B_2 e B_3, já que estes estão conectados à B_1 através das linhas L_2 e L_3, respectivamente. Pode ser dito então que B_1 possui um número de barramentos vizinhos, NV, igual à 2.

Através dos esquemáticos, as redes de distribuição podem ser modeladas utilizando grafos. Este modelo é adotado devido a sua conveniência de abstração intuitiva da rede e pela possibilidade de implementação de algoritmos eficientes, tal como o GHS, utilizando as estruturas de dados e resultados da teoria de grafos, já amplamente estudada.

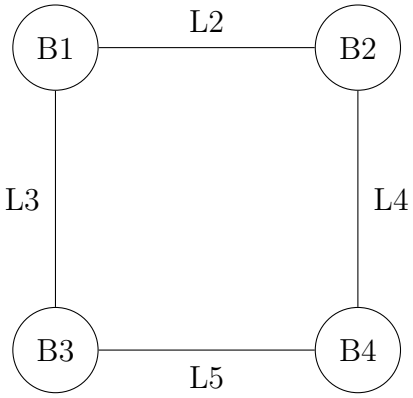


Figura 14: Representação em grafo do exemplo da Figura 13.

Modelando a rede de distribuição utilizando um grafo, cada barramento é representado por um vértice, enquanto que cada uma das linhas, compreendidas entre dois

barramentos, é representada por uma aresta. O exemplo de grafo apresentado na Figura 14 representa a RDI da Figura 13.

Para a utilização das técnicas de obtenção de Árvore de Extensão Mínima, discutidas na Seção 1.2, aplicadas como base da solução, também faz-se necessário a definição dos valores de pesos associados à cada aresta do grafo modelado. Na implementação do algoritmo, a grandeza adotada como referência, para a modelagem dos pesos das arestas, é o valor da impedância (Z) das linhas que compõe cada aresta. Esta associação é justificada, pois dada a minimização da impedância equivalente do sistema de distribuição, as perdas do sistema de distribuição também serão minimizadas, conforme pode ser verificado através das equações de fluxo de potência apresentadas a seguir na Seção 3.2.

3.2 Modelo de Minimização de Perdas do Alimentador

Um dos principais problemas na implementação de auto-cura para redes de distribuição, é a identificação da melhor configuração de uma rede radial (ISONI, 2013) que apresente os menores índices de perdas técnicas de energia (ARANGO; TAHAN, 2005), inerentes às atividades de transporte da energia elétrica na rede, satisfazendo as restrições de tensão de operação do sistema, capacidade de corrente do alimentador e mantendo a característica radial do sistema de distribuição, conforme apresentado em (RAO; NARASIMHAM; RAMALINGARAJU, 2008).

Este problema de minimização da perda total de potência ativa do sistema (P_{perda}), pode ser modelado através da função objetivo f da Equação 3, onde CR é o conjunto de todas as configurações possíveis da rede.

$$f(C*) = min(P_{perda}) \; \forall \; C \in CR \tag{3}$$

Para a compreensão desta formulação, seja o circuito apresentado na Figura 15 como exemplo simplificado de fluxo de carga em um alimentador com barramentos do tipo PQ (MATOS, 2013).

Para a análise de perdas, as grandezas elétricas estudadas em cada um dos barramentos presentes no sistema de distribuição são referentes às potências ativas e reativas de transferência entre barramentos dados por P_{Bi} e jQ_{Bi}, $0 < i \leq n$. São analisadas também as potências ativas (P_{Ci}) e reativas (jQ_{Ci}) das cargas C conectadas aos barramentos,

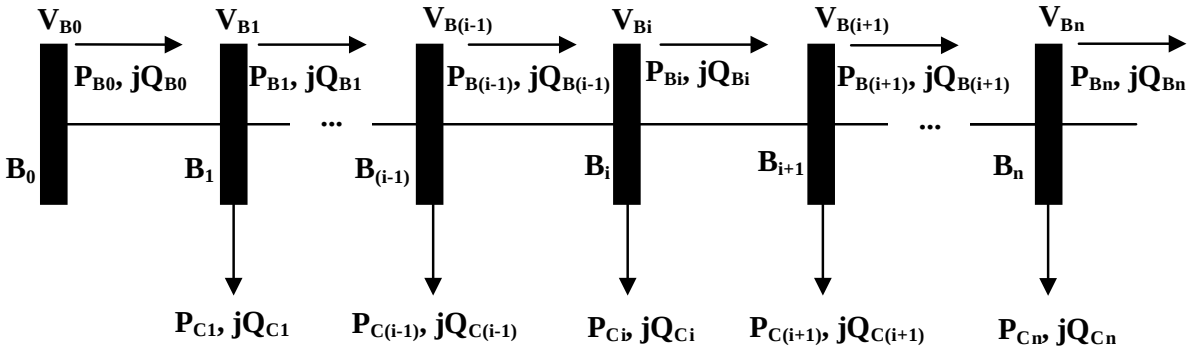


Figura 15: Exemplo de fluxo de potência em barramentos com um alimentador.

além da magnitude da tensão em cada um dos barramentos. Esta magnitude é verificadas através do módulo da tensão ($|V_{Bi}|^2$), pois o fluxo de potência pode apresentar sentidos diferentes, dependendo da potência consumida pelas cargas de cada barramento, fazendo com que a tensão apresente sinais opostos.

Pela complexidade do sistema de distribuição, durante o problema de reconfiguração é assumido que a rede é simétrica, as 3 fases do sistema estão defasadas umas das outras em 120^o, e balanceada, a distribuição por igual das potências das cargas entre as fases. Portanto, as linhas de distribuição são representadas como impedâncias em serie, com demanda constante e com um fluxo de potência aparente (S) dado pela soma das potências ativas (P) e reativas (jQ) das cargas conectadas (C) aos barramentos, conforme a Equação 4.

$$S_{Ci} = P_{Ci} + jQ_{Ci} \tag{4}$$

Tendo como premissa os valores conhecidos de P_0, jQ_0 e V_0 referentes ao primeiro barramento da rede, para os demais barramentos subsequentes denotados por $B(i+1)$, as potências ativa e reativa $P_{B(i+1)}$ e $jQ_{B(i+1)}$ e o módulo de tensão ($|V_{B(i+1)}|^2$) podem então ser calculados através das Equações recursivas 5 – 7, (RAO; NARASIMHAM; RAMALINGARAJU, 2008). Este método é conhecido como *forward update*.

$$P_{Bi,B(i+1)} = P_{Bi} - P_{C(i+1)} - R_{Bi,B(i+1)} \cdot \frac{P^2 + (jQ)^2}{|V_{Bi}|^2} \tag{5}$$

$$jQ_{Bi,B(i+1)} = jQ_{Bi} - jQ_{C(i+1)} - X_{Bi,B(i+1)} \cdot \frac{P^2 + (jQ)^2}{|V_{Bi}|^2} \tag{6}$$

$$|V_{B(i+1)}|^2 = |V_{Bi}|^2 - 2(R_{Bi,B(i+1)} \cdot P_{Bi} + X_{Bi,B(i+1)} \cdot jQ_{Bi}) \cdot \frac{P^2 + (jQ)^2}{|V_{Bi}|^2} \tag{7}$$

Outro método possível de ser implementado é o *backward update*, similar ao *forward update*. Este método é empregado através da aplicação das Equações recursivas 8 – 10.

$$P_{B(i-1)} = P_{Bi} - P_{Ci} - R_{B(i-1),Bi} \cdot \frac{P'^2 + (jQ')^2}{|V_{Bi}|^2}, \quad (8)$$

$$jQ_{B(i-1)} = jQ_{Bi} - Q_{Ci} - X_{B(i-1),Bi} \cdot \frac{P'^2 + (jQ')^2}{|V_{Bi}|^2}, \quad (9)$$

$$|V_{B(i-1)}|^2 = |V_{Bi}|^2 - 2(R_{B(i-1),Bi} \cdot P'_{Bi} + X_{B(i-1),Bi} \cdot jQ'_{Bi}) \cdot \frac{P'^2 + (jQ')^2}{|V_{Bi}|^2}, \quad (10)$$

onde $P'_{Bi} = P_{Bi} + P_{Ci}$ e $jQ'_{Bi} = jQ_{Bi} + jQ_{Ci}$

A potência ativa perdida durante a fluxo de energia elétrica através da linha que conecta um barramento B a outro barramento $B + 1$ é calculado através da Equação 11

$$P_{Perda}(B, B+1) = R_{B,B+1} \cdot \frac{P^2 + (jQ)^2}{|V_i|^2}, \quad (11)$$

e o total de perda de potência ativa do todo o sistema (PS_{perda}) é definido pelo somatório das perdas presentes em cada linha que compõe o alimentador, conforme definida na Equação 12.

$$PS_{perda} = \sum_{B=0}^{NB-1} P_{Perda}(i, i+1) \quad (12)$$

3.3 Arquitetura do Algoritmo de Auto-cura

Durante o desenvolvimento e implementação da solução, as plataformas utilizadas como unidades de processamento distribuído, foram os robôs Elisa-3, desenvolvidos pela GCtronic (GCTRONIC, 2013). Estes robôs são compostos por um processador Atmel ATmega2560 de 8MHz, memória RAM (*Random Access Memory*) de 8KB, uma memória *flash* de 256KB, uma memória EEPROM (*Electrically-Erasable Programmable Read-Only Memory*) de 4KB, um conjunto de sensores, além de um circuito responsável por efetuar comunicação sem fio, por meio de rádio frequência.

A comunicação entre estas unidades de processamento ocorre de forma indireta através de troca de mensagens. As informações trocadas entre as unidades de processamento são enviadas em forma de mensagens, inicialmente, à uma estação rádio-base, por rádio frequência na faixa de 2,4GHz, que repassa esta informação para um computador via comunicação serial. Após o computador estruturar o pacote de dados recebido da estação

rádio-base, no formato de mensagem para transmissão, esta mensagem é encaminhada do computador à estação para que seja entregue à unidade de processamento de destino. A arquitetura de comunicação está representa no diagrama de blocos da Figura 16.

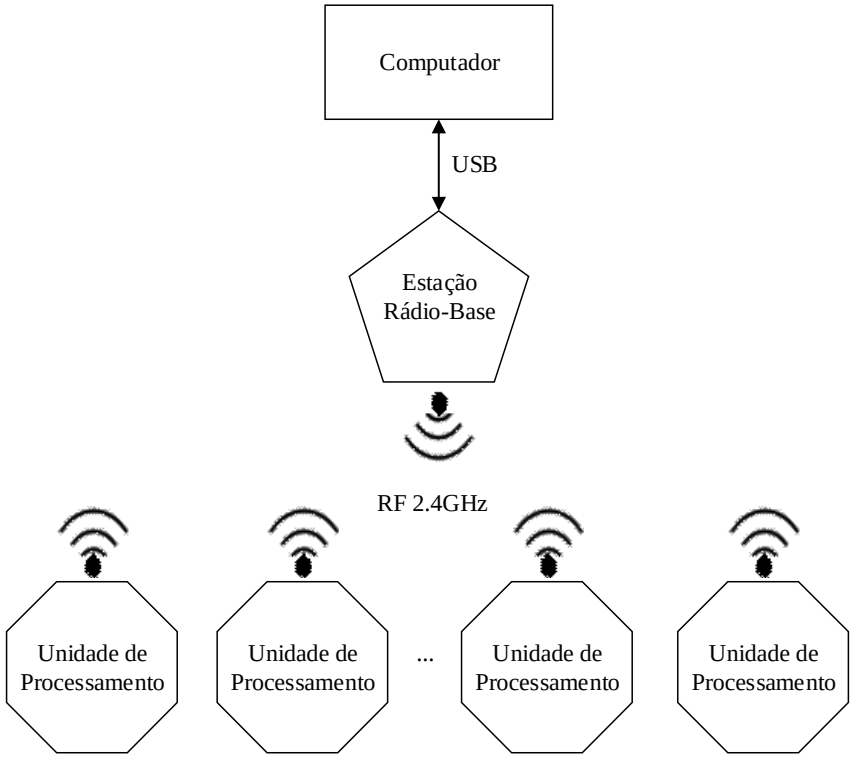


Figura 16: Arquitetura de comunicação.

Durante o processo de comunicação, para a identificação de cada unidade de processamento, faz-se uso de um RFID (*Radio-Frequency Identification*), como elemento de endereçamento, que encontra-se armazenado nos dois últimos *bytes* do microcontrolador interno da EEPROM dos robôs Elisa-3.

A concepção da solução que permitir a auto-cura das RDIs, faz uso de um algoritmo distribuído que opera em cada uma das unidades de processamento. Como este algoritmo deverá estar presente em cada uma das unidades de processamento, optou-se pela geração de um único código para otimizar o carregamento das unidades.

A operação da solução ocorre através das três etapas principais descritas abaixo, e que serão aprofundadas no decorrer deste capítulo.

- **Inicialização**: fase inicial do algoritmo, responsável pela inicialização das principais variáveis de estado do algoritmo GHS. Têm o objetivo de ajustar as informações necessárias para a correta execução do algoritmo;

- **Criação de vizinhança**: nesta fase, o algoritmo verifica se há interrupções nos trechos do cabo do alimentador, monitorado-os pelos controladores das chaves. Cada interrupção impacta na exclusão da aresta do grafo que representa a linha afetada. Ao final desta atualização cada unidade de processamento contém as informações de conexão desta com as suas unidades vizinhas.

- **Reconfiguração de Rede**: contempla a última etapa do algoritmo na qual é utilizado o algoritmo de GHS, buscando a convergência para a AEM do grafo que representa a melhor configuração possível de conexões dos barramentos presentes no alimentador. É responsável por localizar a melhor configuração das linhas da rede de distribuição para que o maior número possível de barramentos tenham seu fornecimento restabelecido.

3.3.1 Inicialização

Esta etapa é a fase inicial da solução. Cada uma das unidades de processamento, que representam os vértices do grafo modelado da rede de distribuição, tem suas variáveis definidas e inicializadas. As principais variáveis de operação do GHS abordadas na Seção 1.3.3, responsáveis por armazenar as informações que utilizadas no funcionamento do algoritmo GHS, são:

- Estado do vértice (*EstVertice*);
- Estados das arestas adjacentes j ($EstAresta_j$);
- Aresta de fronteira (*AFront*);
- Valor do peso associado à aresta de fronteira (*AFrontPeso*)
- Aresta de entrada (*AEnt*);
- Aresta em teste (*Ateste*);
- Contabilizador (*Contb*);
- Identificador do fragmento (*IDFrag*);
- Nível do Fragmento (*NivFrag*)

Para representação de cada um destes parâmetros foi utilizado uma variável do tipo inteiro. Na Tabela 3 é apresentado o mapeamento das variáveis *EVertice* e $EAresta_j$ com suas representações e a quantidade de *bytes* utilizados na formação da mensagem de envio. Para cada uma das demais variáveis, apenas um *byte* foi necessário para a alocação das informações.

Tabela 3: Mapeamento das variáveis de estado dos vértices e das arestas.

Variável	Bytes	Valor	Denominação de Estado
EVertice	1	0	Dormindo
		1	Encontrado
		2	Buscando
$EAresta_j$	1	0	Básica
		1	Ramo
		2	Rejeitada

Além das variáveis essenciais para o algoritmo GHS, também fez-se necessário a criação de dois vetores de inteiros utilizados para armazenar as informações de configuração do grafo como entrada do algoritmo, conforme apresentado abaixo:

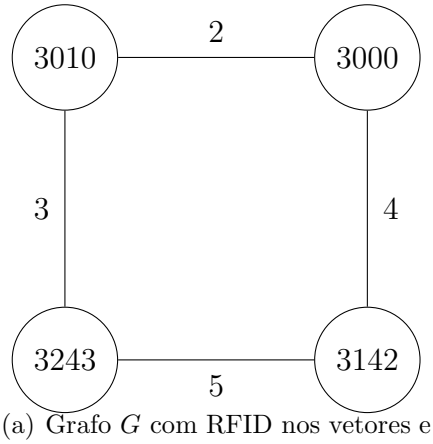
- Vetor de IDs (*Ids*): vetor de inteiros, com número de elementos igual ao total de número de unidades de processamento que compõem o grafo, contendo o RFID de cada uma destas unidades;

- Vetor de pesos (*Pesos*): vetor de inteiros, com número de entradas, *NPesos*, igual ao número de pares unidades de processamento que compõem o grafo. Conforme definido na Seção 1.1, este número é definido por $(NB^2 - NB)/2$.

Como o código gerado na implementação para ser carregado nas unidades processadoras é único, todos os dados referentes à configuração do grafo acabam sendo importados em todas as unidades de processamento, permitindo o processo de mapeamento de vizinhança.

A obtenção dos dados de vizinhança para qualquer grafo poderia ser facilmente obtido carregado-se todos os dados presentes na matriz adjacência que descreve o grafo, em uma matriz NB×NB nas unidades de processamento. Porém, na concepção desta solução existe uma preocupação quanto a limitação do uso de memória das unidades de processamento.

Conforme apresentado na Seção 1.1, a matriz adjacência dos grafos simples possuem duplicidade de informação no mapeamento das arestas, além de não possuírem informações em sua diagonal principal. Por isso optou-se por se utilizar a matriz adjacência triangular, onde a duplicidade e a diagonal principal não ocorrem, possibilitando um menor uso de memória das unidades de processamento.



(a) Grafo G com RFID nos vetores e pesos nas arestas

3010				
3000	2			
3243	3	-1		
3142	-1	4	5	
	3010	3000	3243	D3142

(b) Matriz adjacência triangular do grafo G.

Figura 17: Exemplo de obtenção de árvore geradora para um grafo G.

Utilizando como exemplo o grafo G da Figura 17(a), onde verifica-se a presença de 4 vértices representados com exemplos de RFIDs e apenas 4 arestas com pesos associados das 6 possíveis, caso os vértices estivessem completamente conectados. Para modelar esta situação, para todo e qualquer configuração de rede de distribuição, à cada aresta não existente é associado o valor -1 na matriz adjacência triangular. Como pode ser visto, na Figura 17(b), a presença do valor -1 para as arestas inexistentes mapeados nas posições (3243, 3000) e (3142, 3010). Este valor foi escolhido para representar o valor infinito de impedância, representando o caso real da ausência de linha de distribuição.

Note que no caso de uma descontinuidade em alguma das linhas de distribuição da rede, caracterizando uma falha, o valor -1 substitui o valor do peso da aresta que modela a referida linha. Está ação é adotada, pois no caso uma linha em falha, o equipamento de proteção instalado na chave que alimenta a linha atuará acarretando na impossibilidade de utilização desta linha. Então é assumido que sua impedância tende à infinito, assim como na ausência de alguma linha.

O carregamento da matriz adjacência triangular nas unidades de processamento é realizado através do uso dos vetores *Ids* e *Pesos*. Onde, no vetor *Ids* cada índice k deste

vetor é carregado, sequencialmente, com os RFIDs de todas as linhas da primeira coluna da matriz adjacência triangular.

Após o processo de carregamento do vetor *Ids*, é iniciado o processo de carga do vetor *Pesos*. Excluindo-se a primeira coluna e a última linha da matriz adjacência triangular, o restante dos dados presentes na matriz são referentes aos pesos das arestas do grafo. Estes pesos são carregados no vetor *Pesos* através da linearização, linha por linha, desta matriz.

As Tabelas 4 e 5 ilustram, respectivamente, o carregamento dos vetores *Ids* e *Pesos* com as informações da matriz adjacência triangular do grafo G, apresentado na Figura 17.

Tabela 4: Vetor *Ids* referente ao exemplo da Figura 17.

Posição (i, j) da Matriz	(0, 0)	(1, 0)	(2, 0)	(3, 0)
Índices Vetor *Ids*	0	1	2	3
RFID	3010	3000	3243	3142

Tabela 5: Vetor *Pesos* referente ao exemplo da Figura 17.

Posição (i, j) na Matriz	(1, 1)	(2, 1)	(2, 2)	(3, 1)	(3, 2)	(3, 3)
Índices Vetor *Pesos*	0	1	2	3	4	5
Peso	2	3	-1	-1	4	5

3.3.2 Geração de Vizinhança

Conforme apresentado na Seção 1.3.3, o algoritmo GHS utiliza a troca de mensagens entre os vértices do grafo na busca da AEM. Lembrando que na implementação, cada um dos vértices é a representação de uma unidade de processamento que faz referência à um barramento da RDI. Para que esta troca de mensagens seja possível, cada unidade de processamento deve mapear suas unidades de processamento vizinhas, através dos RFIDs os representam.

Para prover as informações de vizinhança de cada unidades de processamento, foi criada uma matriz de inteiro denominada *Vizinhança* responsável por armazenar os valores representativos dos estados de cada uma das arestas da unidade de processamento, os valores de peso destas arestas e os RFIDs das unidades de processamento incluídos na vizinhança. Esta matriz é formada por três linhas e número de colunas igual à NV. Ela

armazena os valores referentes à $EAresta_j$, peso da aresta j e o RFID do vértice conectado através desta aresta, para cada vértice vizinho.

A população da matriz *Vizinhança* é realizada através do cruzamento de informações dos vetor *Ids* e *Pesos* presetes na unidade processadora. Para tal foi desenvolvido um algoritmo que efetua o cruzamento dos dados destes 2 vetores, possibilitando a modelagem da matriz adjacência triangular. Através desta modelagem é possível localizar as unidades processadoras vizinhas da unidade que está executando o algoritmo, além dos pesos de cada uma das arestas que as conectam.

O algoritmo executa a varredura do vetor *Ids* para localizar e armazenar em uma variável *id* o índice k deste vetor, onde se encontra o RFID correspondente à unidade que está exercendo a busca. Posteriormente, o vetor de *Ids* é novamente percorrido, e ao encontrar um RFID diferente ao da unidade que está executando o algoritmo, o índice k deste RFID é armazenado em uma variável *id'*. Em seguida, os índices armazenados nas variáveis *id* e *id'* são utilizados para encontrar o índice do vetor *Pesos* que esta armazenado o peso da aresta que conecta a unidades de processamento mapeados em *Ids*[*id*] e *Ids*[*id'*].

Sendo o vetor *Pesos* a representação linearizada, linha a linha, da matriz triangular, quaisquer índices *idpeso* do vetor *Pesos* podem ser dados em função da relação entre os índices *id* e *id'*. A Equação 13 apresenta o modelo matemático desta relação.

$$idpeso = \begin{cases} \frac{(id^2 - id)}{2} + id', \text{ se } id < id' \\ \\ \frac{(id'^2 - id')}{2} + id, \text{ se } id > id' \end{cases} \tag{13}$$

Após calcular o *idpeso*, o algoritmo verifica se o valor presente em *Pesos*[*idpesos*] é diferente de -1, indicando que existe uma aresta que interliga as unidades de processamento *Ids*[*id*] e *Ids*[*id'*]. Então as informações referentes à *Ids*[*id'*] e *Pesos*[*idpesos*] são armazenados no vetor *Vizinhança* e o algoritmo retoma a varredura do vetor *Ids* até seu último elemento.

Este processo foi escrito na forma do Algoritmo 15, que tem a função de criar a matriz *Vizinhança* e contabilizar número de seus vizinhos, através da variável *nViz*.

A Tabela 6 a seguir apresenta um exemplo de matriz *Vizinhança* obtida através do Algoritmo 15 para a unidade de processamento de RFID 3010 da Figura 17(a).

Algoritmo 15 Algoritmo para geração da matriz vizinhança para robô *Rid*

```
1: nViz ← 0
2: para todo Ids[k], | 0≤k<NB faça
3:    se Ids[k] = Rid então
4:       id ← Ids[k]
5:    fim se
6: fim para
7: para todo Ids[k], | 0≤k<NB faça
8:    se Ids[k] ≠ Rid então
9:       id' ← Ids[k]
10:      se id < id' então
11:         C ← [id(id − 1)/id'] + id
12:      senão
13:         C ← [id'(id' − 1)/id] + id'
14:      fim se
15:      se Pesos[C] ≠ −1 então
16:         Vizinhos[0][nViz] ← 0
17:         Vizinhos[1][nViz] ← Pesos[C]
18:         Vizinhos[2][nViz] ← Ids[ids']
19:         nViz ← nViz + 1
20:      fim se
21:   fim se
22: fim para
23: Retorna Vizinhos
24: Retorna nViz
```

Tabela 6: Exemplo da matriz vizinhança da unidade de processamento de RFID 3010 da Figura 17(a).

0	Estado da Aresta	0	0
1	Peso	2	3
2	RFID	3000	3243

3.3.3 Reconfiguração de Rede

A etapa de reconfiguração de rede é responsável por encontrar uma proposta de reconfiguração das chaves da rede de distribuição, garantindo a interconexão do máximo de barramentos possíveis permitindo a continuidade do fornecimento de energia elétrica para a maior parte possível do sistema.

Este processo é realizado através da aplicação do algoritmo de GHS descrito na Seção 1.3.3, devido à sua característica de operação distribuída, durante a localização da AEM. Outra propriedade importante do algoritmo de GHS é o fato de que a solução resultante apresenta o menor somatório dos pesos associados as arestas do grafo. Este fato, garante que o sistema apresenta a menor impedância equivalente dentre todas as

possibilidades de reconfiguração.

Devido a troca de inúmeras mensagens durante a execução do algoritmo, foi necessária a implementação de uma fila de mensagens do tipo FIFO (*First In*, *First Out*) em cada uma das unidades de processamento, pois o algoritmo somente consegue tratar uma mensagem por vez. Esta fila consiste em uma matriz de inteiros de dimensões $R \times 7$. A quantidade R de linhas equivale ao comprimento da fila, mensurado como três vezes o número de vizinhos da unidade processadora. Isto é devido a possibilidade de recebimento de, no máximo, uma mensagem `Aceita` ou `Rejeitar`, uma mensagem do tipo `Reportar` e uma mensagem do tipo `Trocar` ou `Conectar` por cada uma de suas arestas do vértice, conforme apresentado em (GALLAGER; HUMBLET; SPIRA, 1983). Já as 7 colunas são devido ao comprimento das mensagens trocadas entre as unidades processadoras. A modelagem destas mensagens serão abordadas na Seção 3.4 deste capítulo.

Durante a execução do algoritmo de GHS pela unidade de processamento é verificada a presença de mensagem no topo fila de mensagens, a mensagem é então processada podendo gerar uma nova mensagem à ser transmitida para um de seus vizinhos. Após a execução do algoritmo de GHS verifica-se a ocorrência da convergência para a AEM. Caso o algoritmo não tenha chegado a solução desejada inicia-se a etapa de envio e recebimento de mensagem para permitir a troca de informação entre as unidades de processamento. Caso receba uma nova mensagem, a unidade de processamento a enfileira na fila de mensagem para ser processada futuramente pelo algoritmo. O Algoritmo 16 apresenta a modelagem deste processo.

Algoritmo 16 Processo de reconfiguração de Rede no robô *Rid*.

1: **enquanto** Houverem mensagens na fila **faça**
2: Executa etapa processamento do Algorítimo GHS, relativo ao tipo de mensagem recebida.
3: Efetua envio e recebimento de mensagens
4: **fim enquanto**

Durante a etapa de envio e recebimento de mensagens, a antena da unidade de processamento sempre é habilitada para o recebimento de mensagem endereçadas ao seu RFID. No entanto, para proceder com a transmissão a unidade de processamento necessita receber uma permissão advinda da estação rádio-base. Em seguida, uma segunda mensagem de coleta permitindo assim a habilitação da antena para transmissão das informações disponibilizadas ao término do processamento da mensagem localizada no início da fila.

Após o envio das informações, a mensagem no primeiro lugar da fila é desenvileirada e o envio de mensagens pela antena é então desabilitado. O Algoritmo 17 apresenta o fluxo da etapa de envio e recebimento de mensagens.

Algoritmo 17 Processo de envio e recebimento de mensagem.

```
1: Recebe mensagem da antena
2: se A mensagem é do tipo permissão de transmissão então
3:    Armazena o conteúdo da mensagem ao final da fila
4: senão
5:    se Existe alguma mensagem pendente de envio então
6:       Envia mensagem
7:       Remove mensagem da fila
8:       Desabilita envio de mensagem
9:    fim se
10: fim se
```

O processo de reconfiguração é finalizado após a convergência do algoritmo de GHS. Cada uma das unidades de processamento apresentará, através $EstAresta_j$ na matriz *Vizinhança*, as possibilidades de estado `Ramo` ou `Rejeitada` para as arestas. Através da interpretação desses estados, o controlador da chave pode decidir entre:

1. Abertura da chave, ocorrendo quando o estado da aresta é igual à `Rejeitada` e chave encontra-se atualmente fechada;

2. Fechamento da chave, quando o estado da aresta é igual à Ramo e chave encontra-se atualmente aberta;

3. Permanência do estado da chave, caso o estado da aresta seja `Rejeitada` e chave se encontre atualmente aberta, ou o estado da aresta seja igual à Ramo e a chave já esteva fechada.

3.4 Modelagem da Comunicação

Conforme apresentado na Seção 3.3, a comunicação entre as unidades de processamento ocorre de forma indireta através do envio de mensagens destas unidades à uma estação rádio-base para que esta possa rotear os pacotes às unidades de processamento de destino.

Desta forma o processo de comunicação transcorre tanto na unidade de processamento, quanto no computador. A primeira tem a responsabilidade de formatar as informações necessárias para a operação do algoritmo de auto-cura e os RFIDs de destino

em forma de mensagens a serem transmitidas para a estação rádio-base. Enquanto a segunda, é responsável por receber as mensagens das unidades de processamento e roteá-las à seus respectivos endereços de destino.

3.4.1 Formato das Mensagens

Durante a fase de reconfiguração de rede e na busca da AEM, o algoritmo GHS procede com a troca de 7 diferentes tipos de mensagens entre os controladores da chave. Para viabilizar a utilização destas mensagens no nível das unidades de processamento, fez-se necessário o uso de um campo do tipo inteiro (*TMsg*), que distingui cada um dos tipos de mensagens a serem enviadas. A Tabela 7 sintetiza a codificação adotada assim como o conteúdo enviado em cada mensagem.

Tabela 7: Codificação e conteúdo dos tipos de mensagens.

Tipo de Mensagem	**Valor do TpM**	**Conteúdo da Mensagem**
`Conectar`	1	NivFrag
`Iniciar`	2	NivFrag, IDFrag, EstVertice
`Teste`	3	NivFrag, IDFrag
`Reportar`	4	VFrontPeso
`Rejeitar`	5	Sem conteúdo
`Aceitar`	6	Sem conteúdo
`Trocar`	7	Sem conteúdo

Cada tipo de mensagem carrega em seu conteúdo diferentes tipos de informação necessárias à execução correta das etapas do algoritmo GHS, tais como, o nível do fragmento (`NivFrag`), a identidade do fragmento (`IDFrag`), o estado do vértice (`EVertice`), o peso da melhor aresta (`VFrontPeso`) e o RFID da unidade de processamento de destino.

As mensagens enviadas pelas unidades de processamento são modeladas conforme o protocolo de comunicação dos robôs Elisa-3. As informações transmitidas são empacotadas em um vetor, chamado *Payload*, do tipo caracter sem sinal com comprimento de 16 *bytes* conforme o formato apresentado na Figura 18.

Payload[0]	Payload[1]	Payload[2]	Payload[3]	Payload[4]	Payload[5]	Payload[6]	Payload[7]	Payload [8-13]	Payload [14]	Payload [15]
PacketId	TMsg	NivFrg	IDFrag	EVertice	AFrontPeso	RFID Destino		Vago	RFID Emissor	

Figura 18: Composição do vetor *Payload*.

As informações utilizadas durante a troca de mensagens, pelo algorítimo de autocura, foram alocadas nas posições compreendidas entre 1 e 7, e nas posições 14 e 15 do vetor *Payload*. A primeira posição do vetor traz em seu conteúdo o *PacketID*, que será utilizado, pelo computador durante o processo de comunicação, como forma de segregação das mensagens válidas e inválidas recebidas através da antena. Para esta implementação a informação é considerada válida caso o conteúdo do *PacketID* seja igual a 2.

Os endereços RFIDs podem ocupar dois *bytes*, e para garantir que qualquer número de unidades de processamento possa ser utilizado na solução, é necessário que esta informação seja enviada usando 2 *bytes*. Este processo é realizado dividindo a informação do RFID em duas diferentes posições do vetor *Payload*, cada uma contendo 8 *bits*, onde a primeira posição armazena os 8 *bits* menos significativos e a segunda posição armazena, os 8 *bits* mais significativos do RFID. Por exemplo, o RFID 3200 tem sua representação em hexadecimal igual à 0x0C80, então a primeiro posição no vetor armazenará o valor de 0x80 e a segunda posição o valor de 0x0C. Isto explica a representação da informação de RFID Destino e RFID Emissor através dos pares *Payload*[6] e *Payload*[7] e *Payload*[14] e *Payload*[15], respectivamente.

3.4.2 Protocolo de Comunicação

A comunicação para roteamento envolve a estação rádio-base e o computador, conforme apresentado na Figura 16, como responsáveis por prover o encaminhamento das mensagens geradas pelas unidades de processamento. O modelo de comunicação entre estes dois elementos é do tipo mestre-escravo, cabendo ao computador o papel de controlador, sendo este então responsável por executar, continuamente a cada milissegundo, a coleta e entrega de mensagens à estação rádio-base.

Para gerenciar o processo de comunicação, o algoritmo de controle e encaminhamento de mensagens utilizado para o sistema de comunicação dos robôs Elisa-3, disponível no site do fabricante (GCTRONIC, 2013), foi remodelado para prover a comunicação do sistema com eficácia. Através deste algoritmo, o computador realiza a gestão das permissões de envio de mensagens de cada uma das unidades de processamento, seguindo um modelo baseado na passagem de permissão em anel (*Token Ring*) (LIMA, 2013). Tal modelo consiste em executar o repasse de modo infinito de uma ficha ao longo de uma fila circular, constituída pelos elementos transmissores. Cada elemento ao receber uma ficha

inicia a transmissão de sua mensagem. Ao término da transmissão ou caso não exista de uma mensagem a ser transmitida, a ficha é descartada fazendo com que o elemento somente volte a transmitir ao receber uma nova ficha.

A fila circular foi viabilizada através de um vetor(*current_address*) de inteiros de comprimento igual ao número de unidades de processamento que compõem a rede, e populado com os RFIDs destas unidades. Durante a execução do algoritmo este vetor é percorrido, e para cada elemento são enviados os três tipos diferentes de mensagens, correspondendo aos três diferentes estágios de comunicação, conforme apresentado na Figura 19.

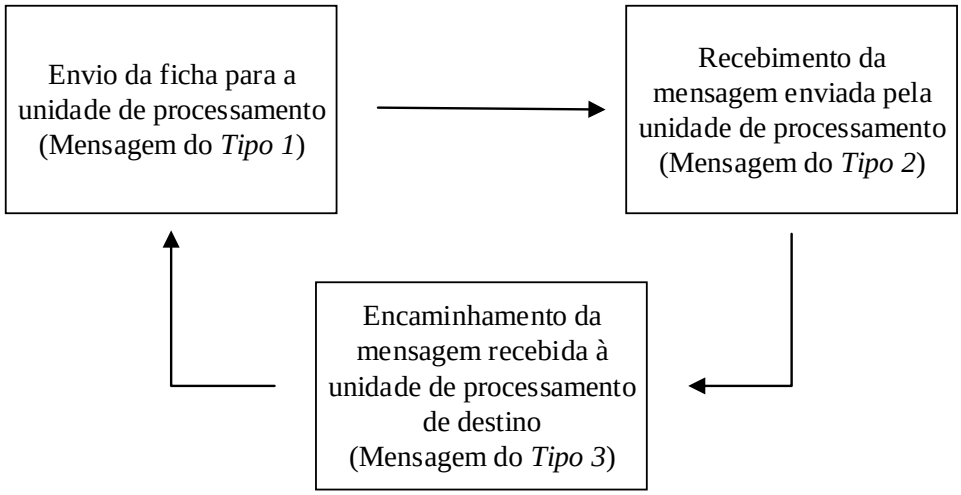


Figura 19: Modelo do ciclo de comunicação.

No primeiro estágio, o computador envia para o primeiro RFID da fila circular a mensagem do *Tipo* 1, que representa a ficha para habilitar o processo de transmissão da unidade de processamento. No segundo estágio o computador envia a mensagem do *Tipo* 2 para realizar a coleta da informação disponibilizada na unidade de processamento, para então encaminhá-la à unidade de processamento de destino através de uma mensagem do *Tipo* 3, encerrando o terceiro estágio da comunicação. O algoritmo passa então para o próximo RFID da fila, executando os três estágios descritos, e ao chegar no último elemento que compõem a fila o processo é recomeçado circularmente. Recomeçando a partir do primeiro RFID da fila.

Durante a passagem do segundo para o terceiro estágio da comunicação, o computador coleta a mensagem recebida e a disponibiliza no *buffer* da estação rádio-base e a armazena em um vetor de dezesseis caracteres, denominado *RX_buffer*. Em seguida, é

verificada a validade da mensagem através da informação do *PacketID* mapeado para a primeira posição do vetor *RX_buffer*.

Com a confirmação da validade da mensagem, o conteúdo do vetor *RX_buffer* é então copiado para um vetor de inteiros sem sinal de sete posições de mensagens recebidas, denominado *MsgRec*. As informações presentes no *RX_buffer*, nas posições compreendidas entre 1 e 5, são convertidas para o interno sem sinal antes de serem armazenadas nas suas correspondentes posições do vetor *MsgRec*, nas posições 0 à 4. Estas posições passam a conter as informações *TMsg*, *NivFrag*, *IDFrag*, *EVertice* e *VFrontPeso* que foram transmitidas pela unidade de processamento. A Figura 20 apresenta o mapeamento das informações contidas na mensagem enviada pela unidade de processamento até sua disponibilização através do vetor *MsgRec* para ser encaminhada ao seu destino.

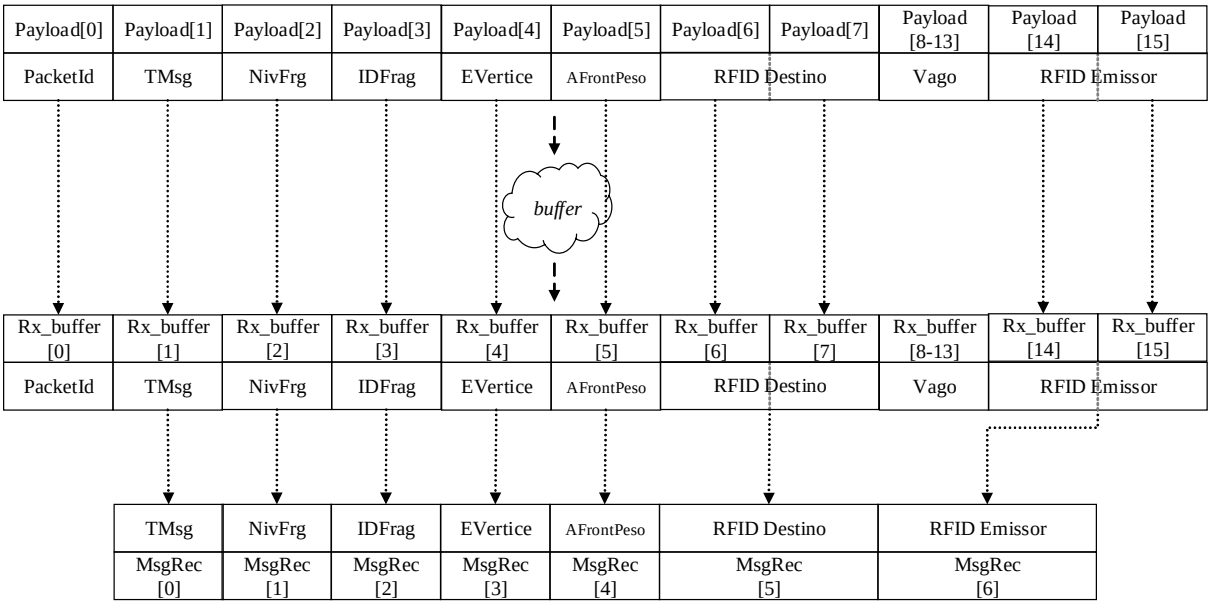


Figura 20: Mapeamento da mensagem na recepção: *Payload*, *RX_buffer* e *MsgRec*.

Dando continuidade ao processo de recuperação dos dados entre os vetores *RX_buffer* e *MsgRec*, os pares de posições 6 e 7 e 14 e 15 do *RX_buffer* são, respectivamente, as partes altas e baixas dos RFIDs Destino e Emissor que foram transmitidos. Antes destas informações serem armazenadas nas posições 5 e 6 do vetor *MsgRec*, os fragmentos de informação passam por um processo de recomposição. Este processo consiste na execução da operação lógica *or* entre a informação da posição mais significativa do RFID deslocada 8 vezes à esquerda e convertida para o tipo inteiro sem sinal e a posição menos significativa convertida para o mesmo tipo caracter sem sinal. As informações mais significativos dos

RFIDs encontram-se alocadas nas posições 7 e 15 do *RX_buffer*, já as menos significativas estão dispostos nas posições 6 e 14 do mesmo.

Ainda durante a transição dos estágios, o algoritmo modela a mensagem recebida da unidade de processamento, que se encontra armazenada no *MsgRec*, segundo o padrão de transmissão da solução, através do *TX_buffer* de 16 caracteres, com o intuito de direcionar as informações contidas nesta mensagem para a unidade de processamento de destino.

A primeira posição do *TX_buffer* é populado com o valor hexadecimal 0x27, que é o comando utilizado para a mudança de estágio do robô. Note que este *byte* não é enviado para a unidade de processamento (GCTRONIC, 2013). A segunda posição do *TX_buffer* é carregado com o valor 4 indicando que esta mensagem é uma mensagem do *Tipo* 3, conforme mostrado na Tabela 8. As posições 2 à 6 do *TX_buffer* são alocadas com as informações pertinentes as posições de 0 à 4 do *MsgRec* (*TMsg*, *NivFrag*, *IDFrag*, *EVertice* e *VFrontPeso*).

Tabela 8: Tipos de mensagens enviados pela estação rádio-base.

Mensagem	**Tx_buffer[1]**	**Conteúdo**
Tipo 1	3	(Sem conteúdo)
Tipo 2	5	(Sem conteúdo)
Tipo 3	4	TMsg, NivFrag, IDFrag, EVertice, VFrontPeso, RFID Origem

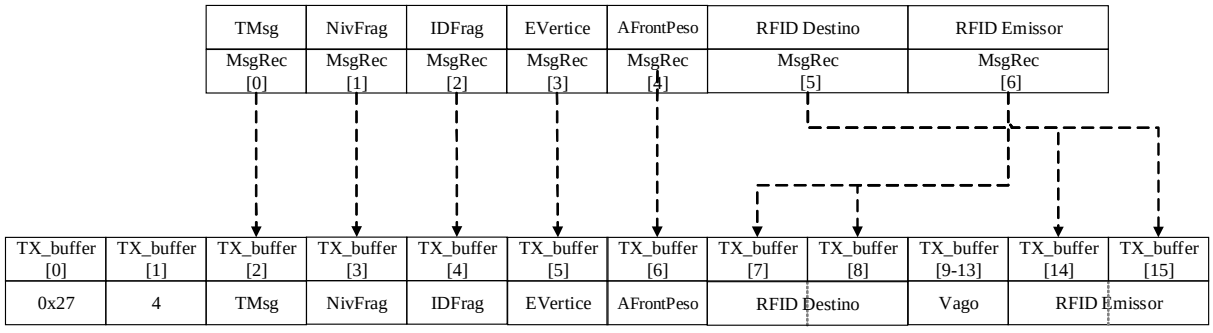


Figura 21: Mapeamento da mensagem para transmissão: de *MsgRec* para *TX_buffer*.

Da mesma forma que durante o processo de envio de mensagens pela unidade de processamento, os RFIDs que irão compor a mensagem a ser enviada pela estação rádio-base, precisão ser fragmentados. O processo de fragmentação segue o mesmo método implementado na comunicação da unidade de processamento. Em virtude disto a

informação presente na posição 6 do *MsgRec*, contendo o RFID da unidade de processamento de onde originou o conteúdo da mensagem, é armazenada nas posições 7 e 8 do *TX_buffer* e a informação presente na posição 5 do *MsgRec* contendo o RFID da unidade de processamento destinatária da mensagem, é armazenada nas posições 15 e 16 do vetor *TX_buffer*. Um modelo deste mapeamento, para envio da mensagem através da estação rádio-base pode ser verificado na Figura 21.

Ao final desta modelagem, inicia-se o terceiro estágio da comunicação responsável por encaminhar as informações geradas pela unidade de processamento de origem à unidade de destino. Ao término do envio, o ciclo de comunicação se reinicia para o próximo elemento da fila circular, garantindo a troca de mensagem necessária para a execução do algoritmo distribuído implementado em cada uma das unidades de processamento.

3.5 Considerações Finais do Capítulo

Este capítulo apresentou a forma de implementação da técnica de auto-cura, com otimização das perdas para redes de distribuição de energia elétrica. Além do método utilizado para modelagem das redes de distribuição na forma de grafos.

A arquitetura da solução desenvolvida foi apresentada através dos seus 3 principais blocos de funcionamento, Inicialização, Criação de Vizinhanças e Reconfiguração de Rede. E por esta ser uma solução distribuída, também foi apresentada a modelagem da comunicação entre os elementos de processamento distribuídos, que se fez necessário para viabilizar a comunicação entre estes através de um protocolo de comunicação próprio para a solução.

O capítulo seguinte apresenta a metodologia adotada para execução dos testes aplicados a solução desenvolvida, bem como a análise realizada sobre os resultados obtidos.

Capítulo 4

ESTUDO DE CASOS

NESTE capítulo são analisados os resultados obtidos através da implementação descrita no Capítulo 3. O algoritmo distribuído de Auto-Curo foi aplicado a duas diferentes configurações de redes de distribuição de energia elétrica para testar a sua capacidade em encontrar uma solução de reconfiguração para a rede, com a finalidade de manter a continuidade de fornecimento de energia. A análise permite a confirmação da proposição elaborada a cerca da minimização dos impactos no fornecimento de energia elétrica, devido às eventuais descontinuidades no sistema de distribuição ocasionados por falhas na rede. Foi também verificada a sua eficiência em termos do tempo de elaboração de uma solução para a reconfiguração da rede.

A Seção 4.1 traz a forma de modelagem de duas redes de distribuição. Já na Seção 4.2 são apresentas a metodologia utilizada durante a realização dos testes para encontrar soluções de reconfiguração da rede de distribuição na presença de descontinuidades de fornecimento, e também são definidos os valores de tempo máximo de execução esperado e limite máximo de mensagens até a convergência, com base nas determinações apresentadas pelos criadores do algoritmo GHS, apresentados em (GALLAGER; HUMBLET; SPIRA, 1983). Os resultados gerados pela solução e as comparações são disponibilizados na Seção 4.3.

4.1 Características das Redes Estudadas

As duas redes utilizadas para a execução dos testes foram baseadas nos modelos de sistema de distribuição de energia elétrica de 14 barramentos (KODSI; CAñIZARES, 2003) e 10 barramentos (FRAG; AL-BAIYAT; CHENG, 1995) do IEEE. Estas redes tem como principal característica a utilização de um único alimentador, que encontra-se segmentado através do uso de barramentos de distribuição para a criação de recursos para manobra da rede,

com o intuito de prover a funcionalidade de recomposição da rede de distribuição em caso de falha. Esta recomposição é efetuada através da operação das chaves de seccionamento e manobra, de cada uma das linhas, que se localizam nos barramentos.

4.1.1 Rede IEEE de 14 Barramentos

O modelo IEEE de 14 barramentos (KODSI; CAñIZARES, 2003) é primeira modelagem rede utilizada, ilustrado na Figura 22. Esta rede é composta por 14 barramentos {B1, B2, ... B14}, que representam os pontos de conexão das cargas consumidoras e 20 linhas {L1, L2, ..., L20}, responsáveis por interconectar os barramentos que compõem a rede.

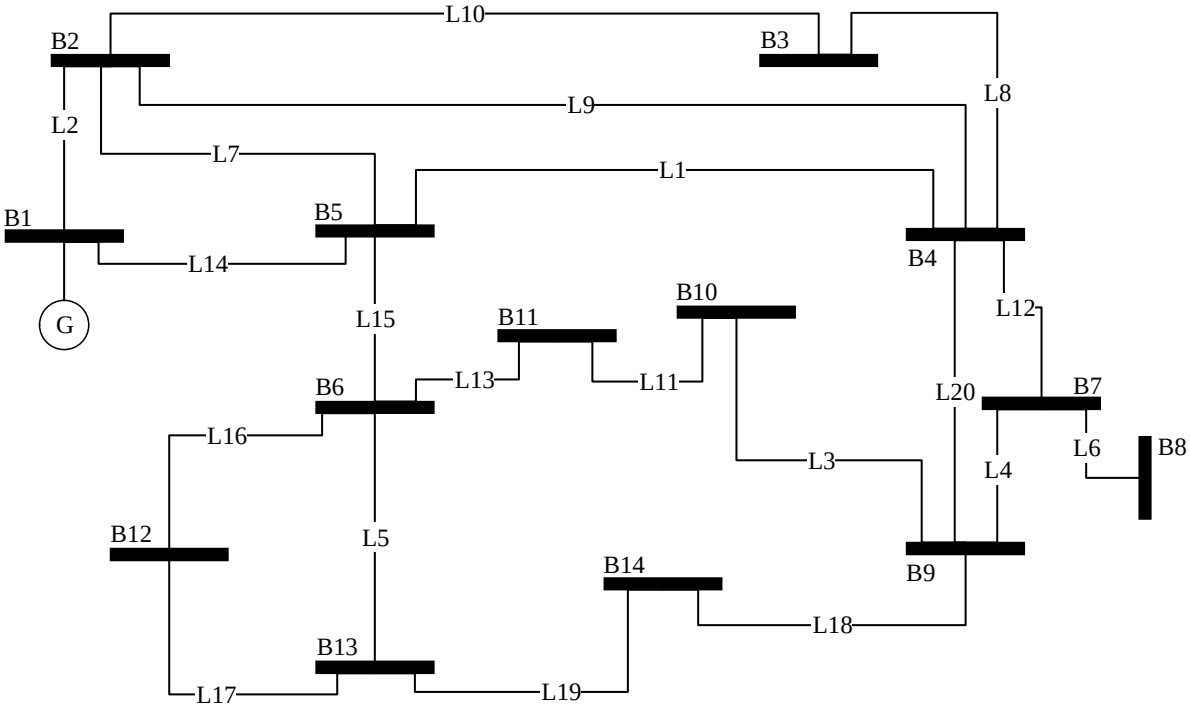


Figura 22: Modelo IEEE de sistema com 14 barramentos (KODSI; CAñIZARES, 2003).

A Figura 22 ilustra o esquema de distribuição de energia elétrica de um alimentador G. Este alimentador prove a energia necessária para atendimento de cada um dos barramentos de distribuição do sistema. Estes barramentos representam a segmentação do sistema de distribuição, além dos pontos de conexão das cargas consumidoras e das chaves de seccionamento e manobra, de cada uma das linhas. Estas linhas tem o papel de levar a energia elétrica de um barramento à outro, por conta disto são consideradas de suma importância para o funcionamento da rede, pois caso ocorra uma falha na linha o barramento ficaria sem alimentação. Pode ser verificada a presença de mais de uma linha conectada aos barramentos, com o intuito de proporcionar recursos para a reconfiguração

da rede de distribuição no caso de falhas, restabelecendo a alimentação do barramento afetado. Porém cada um dos barramentos pode somente estar alimentado através uma única linha, ou seja, caso o barramento possua mais de uma linha como possibilidade de alimentação a chave da linha que possui menor impedância deverá estar fechada enquanto as demais devem permanecer abertas.

A Tabela 9 apresenta o cruzamento de cada uma das linha, com seus valores de impedância disponibilizados em *p.u.* (Sistema por Unidade), que consiste na definição de valores bases para as grandezas elétricas facilitando a possibilidade de suas comparações (MATOS, 2003). E seus respectivos pares de barramentos. Esta tabela encontra-se organizada seguindo a ordem crescente dos valores de impedância de cada uma das linhas.

Tabela 9: Atribuição de pesos com base nas impedâncias.

Linha	**Barramento**	**Impedância**	**Peso**
L_1	B_4–B_5	0,0442	1
L_2	B_1–B_2	0,0623	2
L_3	B_9–B_{10}	0,0903	3
L_4	B_7–B_9	0,1100	4
L_5	B_6–B_{13}	0,1461	5
L_6	B_7–B_8	0,1762	6
L_7	B_2–B_5	0,1830	7
L_8	B_3–B_4	0,1837	8
L_9	B_2–B_4	0,1856	9
L_{10}	B_2–B_3	0,2035	10
L_{11}	B_{10}–B_{11}	0,2089	11
L_{12}	B_4–B_7	0,2091	12
L_{13}	B_6–B_{11}	0,2204	13
L_{14}	B_{11}–B_5	0,2295	14
L_{15}	B_5–B_6	0,2520	15
L_{16}	B_6–B_{12}	0,2838	16
L_{17}	B_{12}–B_{13}	0,2979	17
L_{18}	B_9–B_{14}	0,2988	18
L_{19}	B_{13}–B_{14}	0,3877	19
L_{20}	B_4–B_9	0,5562	20

Conforme apresentado na Seção 3.3.1, o vetor *Pesos* é um vetor de inteiros utilizado para disponibilizar uma parte das informações necessárias para a operação do algoritmo GHS. Por este fato foi preciso efetuar um mapeamento dos valores de impedância de cada linha através de valores inteiros. Isto é obtido através da ordenação crescente dos valores de impedância e em seguida, a atribuição de valores inteiros sequenciais à cada valor na ordenação obtida, conforme pode ser verificado na quarta coluna da Tabela 9.

Os valores populados na quarta coluna da Tabela 9 se fizeram necessário, visto que o algoritmo desenvolvido utiliza um vetor de inteiros representando o vetor *Pesos* apresentado na Seção 3.3.1, para armazenar os valores de peso de cada linha. A utilização de valores inteiros é justificada pela característica de comunicação dos robôs Elisa-3, abordado na Seção 3.4.1.

Com os dados de peso associados à cada uma das linhas, a modelagem da rede de distribuição pode ser realizada na forma de um grafo com pesos, conforme ilustrado na Figura 23. Este grafo permite obter a matriz adjacência, apresentada na Seção 1.1, que contém os dados necessários para o carregamento do vetor *Pesos*, segundo o processo abordado na Seção 3.3. Para a rede do IEEE de 14 barramentos utilizada no teste, este vetor possuirá o número de entradas NE_{Pesos} igual à 91, relativo ao número de combinações sem repetição de pares de barramentos que compõem a rede refletindo as possíveis interconexões entre cada barramentos, dada pela Equação 14.

$$NE_{Pesos} = \frac{NB^2 - NB}{2}, \tag{14}$$

onde *NB* representa o número de *Barramentos de Distribuição* que compõem a RDI.

Outro importante dado de entrada para o algoritmo, viabilizado pela matriz adjacência, é o vetor *Ids*, também apresentado na Seção 3.3, que conterá o RFID de cada uma das unidades representantes dos barramentos na ordem em que aparecem na referida matriz. Para este caso de teste, o número de entradas é igual à 14, pois este é o número de barramentos que compõem a rede.

Após a elaboração da matriz adjacência, através de uma matriz triangular, a Tabela 10 foi obtida. Os dados presentes nesta tabela mapeiam os pesos de cada uma das arestas que compõem a rede IEEE de 14 barramentos em forma de grafo, conforme o processo descrito na Sessão 3.3. A presença do valor -1, pode ser verificado em muitas células desta tabela, com o papel de representar arestas inexistentes dentro do conjunto de possibilidades, melhor detalhado também na Seção 3.3.

Através da tabela obtida, o vetor *Pesos* pode então ser carregado com o conjunto de pesos presentes nas células desta matriz. Já o vetor *Ids* será carregado com os elementos, $\{rfB_1,\ rfB_2,\ ...,\ rfB_{14}\}$, onde rfB_n corresponde ao RFID da unidade de processamento representante do barramento B_i, $1 \leq i \leq 14$.

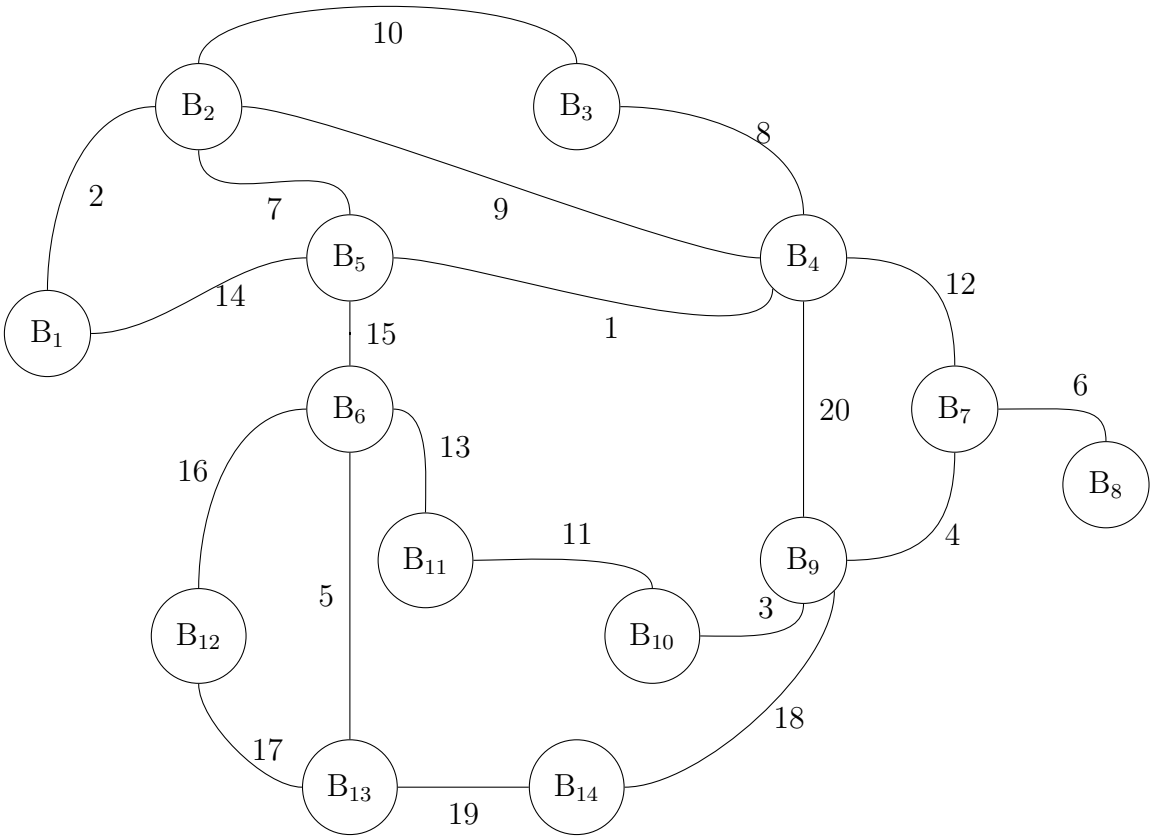


Figura 23: Grafo representando a rede IEEE 14 barramentos.

Tabela 10: Matriz adjacência triangular com pesos baseados nas impedâncias.

B_1													
B_2	2												
B_3	-1	10											
B_4	-1	9	8										
B_5	14	7	-1	1									
B_6	-1	-1	-1	-1	15								
B_7	-1	-1	-1	12	-1	-1							
B_8	-1	-1	-1	-1	-1	-1	6						
B_9	-1	-1	-1	20	-1	-1	4	-1					
B_{10}	-1	-1	-1	-1	-1	-1	-1	-1	3				
B_{11}	-1	-1	-1	-1	-1	13	-1	-1	-1	11			
B_{12}	-1	-1	-1	-1	-1	16	-1	-1	-1	-1	-1		
B_{13}	-1	-1	-1	-1	-1	5	-1	-1	-1	-1	-1	17	
B_{14}	-1	-1	-1	-1	-1	-1	-1	-1	18	-1	-1	-1	19
	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}

4.1.2 Rede IEEE de 10 Barramentos

O modelo IEEE de 10 barramentos (FRAG; AL-BAIYAT; CHENG, 1995) de uma rede de distribuição de energia elétrica , possui em sua configuração 10 barramentos, como pontos

de carga consumidora e 13 linhas, como elementos de conexão entre os barramentos. A Figura 24 apresenta uma ilustração deste modelo de rede.

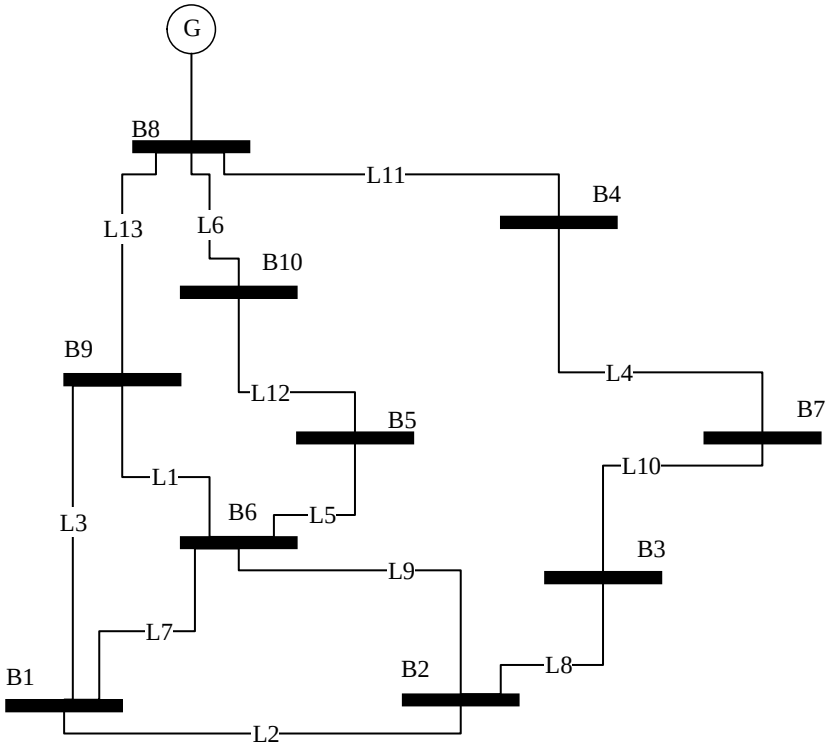


Figura 24: Modelo IEEE de sistema com 10 barramentos (FRAG; AL-BAIYAT; CHENG, 1995).

Com o levantamento dos valores de impedância, convertidos em *p.u.*, de cada uma das linhas do sistema de distribuição disponibilizados na Tabela 11, é possível se notar os valores de peso para cada uma das linhas.

Pode ser verificada a existência dois casos de repetição de valores de impedância, uma para as linhas L3 a L6 com o valor de 0,1649 *p.u* e outro para as linhas L7 a L12 com valor de 0,2474 *p.u.* Vale lembrar que uma das restrições da aplicação do algoritmo GHS é a inexistência de valores de pesos repetidos. Portanto, com a finalidade de desigualar estes valores, durante o mapeamento para valores inteiros à cada um foi atribuído um valor sequencial. Por exemplo, o valor inteiro empregado como peso da linha *L2* foi o valor 2, como esta é a linha antecessora às linhas L3 a L6, os valores empregados à cada uma destas linhas, a fim de eliminar o empate foram os valores inteiros 3, 4, 5 e 6 respectivamente. Tendo em vista que o algoritmo realiza uma busca pelo menor caminho, os valores de peso definidos desta forma sempre terão prioridade em relação aos seus subsequentes. Além disto, caso o algoritmo necessite optar por expurgar algum dos pesos de mesmo valor, a

Tabela 11: Atribuição de pesos com base em impedâncias.

Linha	**Barramentos**	**Impedância**	**Peso**
L_1	B_6–B_9	0,0412	1
L_2	B_1–B_2	0,0825	2
L_3	B_1–B_9	0,1649	3
L_4	B_4–B_7	0,1649	4
L_5	B_5–B_6	0,1649	5
L_6	B_8–B_{10}	0,1649	6
L_7	B_1–B_6	0,2474	7
L_8	B_2–B_3	0,2474	8
L_9	B_2–B_6	0,2474	9
L_{10}	B_3–B_7	0,2474	10
L_{11}	B_4–B_8	0,2474	11
L_{12}	B_5–B_{10}	0,2474	12
L_{13}	B_8–B_9	0,3298	13

ordenação entre estes não terá importância visto que seus valores de impedância são os mesmo, a escolha não impactará na impedância equivalente da rede.

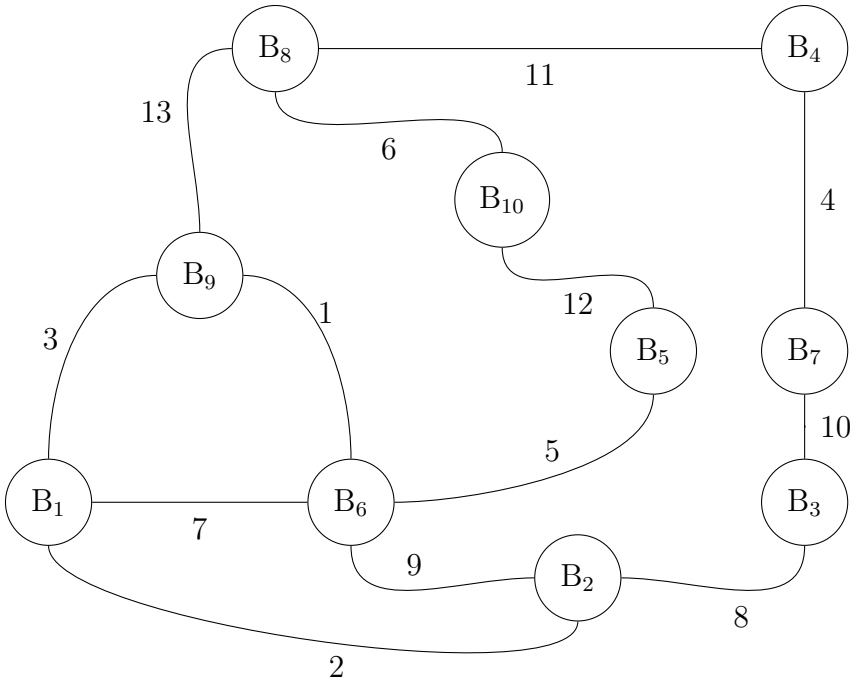


Figura 25: Grafo representando a Rede IEEE 10 barramentos.

De posse dos dados referentes às linhas, seus pesos e os barramentos, modelados a partir do grafo da Figura 25, é possível efetuar a configuração e população dos dados dos vetores de entrada, *Pesos* e *Ids*, do algoritmo de auto-cura.

Neste caso, o vetor *Ids*, tem 10 elementos referentes ao total de barramentos pre-

sentes na rede, identificado por $\{rfB_1, rfB_2, ..., rfB_{10}\}$. Enquanto que o vetor *Pesos* tem um total de elementos igual à 45, conforme definido na Equação 14.

4.2 Metodologia de Avaliação

A implementação desenvolvida foi avaliada por meio das duas redes de distribuição apresentados na Seção 4.1, nas quais foram simulados problemas de descontinuidade simples considerando as linhas. A falha é dita simples já que somente uma das linhas que constituem a rede é considerada em falha. Todos os possíveis casos de falhas simples são avaliados. Uma segunda simulação foi realizado, considerando-se a presença de duas descontinuidades simultâneas.

As avaliações têm como foco o custo de comunicação e a análise de tempo imposto pela implementação proposta, tomando como base a modelagem de um sistema de distribuição de energia elétrica representado por um grafo conexo e acíclico $G = (V, A)$. Para ambos, as características avaliadas, existem valores limítrofes apresentados em (GALLAGER; HUMBLET; SPIRA, 1983), conforme detalhado no restante desta seção.

4.2.1 Custo da Comunicação

Na análise de custo de comunicação, é levado em conta o limite máximo de mensagens (NM_{max}) possíveis de serem trocadas entre os elementos de processamento distribuído até que o algoritmo atinja sua convergência e obtenha a árvore geradora de custo mínimo.

Este limite máximo de mensagens é calculado levando-se em consideração a premissa de que durante todo o processo, uma unidade de processamento pode receber uma única mensagem do tipo `Teste` e enviar uma única mensagem `Rejeitar` por cada uma das arestas que os conectam à seus vizinhos. O custo devido à comunicação pode então ser calculado, para toda a rede como, o dobro do número de arestas ($2A$). Além disto, durante a troca de mensagens do algoritmo GHS, para os casos em que as unidades de processamento estejam em um nível de busca *NivFrag* diferente do inicial, durante a recepção é possível que estas unidades recebam uma única mensagem de dois diferentes tipos, `Iniciar` e `Aceitar`. Enquanto que, durante a transmissão, as unidades de processamento podem enviar, no máximo, uma mensagem `Aceitar` ou `Rejeitar`, uma mensagem do tipo `Reportar` e uma mensagem do tipo `Trocar` ou `Conectar`. Estendendo a possibilidade da troca destes cinco diferentes tipos de mensagens para a faixa de níveis, com limite máximo

de $\log_2 V$, pelos quais uma unidade de processamento pode passar, durante a busca pela AEM, excluindo-se os níveis inicial e o final, pode-se calcular o total de mensagens NM através da Equação 15.

$$NM = 5V(\log_2 V - 1) \tag{15}$$

Ademais, durante a fase de nível zero, as unidades podem receber apenas uma mensagem do tipo `Iniciar` e transmitir uma mensagem do tipo `Conectar`. Por fim, durante o último nível, cada unidade pode transmitir uma única mensagem dos três diferentes tipos, `Teste`, `Rejeitar` e `Reportar`, finalizando o algoritmo. Assim é adicionado um valor menor do que $5V$ mensagens ao total de mensagens trocadas durante os níveis diferentes de zero, conforme apresentado em (GALLAGER; HUMBLET; SPIRA, 1983). Contabilizando as mensagens trocadas durante a execução do algoritmo, chega-se à Equação 16 definida pelo somatório do número de mensagens trocadas para a rejeição de uma arestas $2A$, o número de mensagens trocadas durante os níveis diferentes de zero e último nível $5V(\log_2 V - 1)$, além do número de mensagens trocadas durante o nível zero e durante o último nível $5V$:

$$NM_{max} = 5V \log_2 V + 2A. \tag{16}$$

Note que após a aplicação dos testes, é esperado que o número de mensagens contabilizadas, até a convergência do algoritmo, seja menor ou igual à MN_{max} definido na Equação 16.

A obtenção do número de mensagens trocadas entre as unidades de processamento, durante os testes aos quais as duas redes foram submetidas, foi realizado através do cálculo da média entre o número de mensagens geradas em cada uma das cinco repetições realizadas por testes. A média obtida foi comparada com os valores obtidos através da Equação 16, para cada tipo de rede e cenários, devido à diferença no número de vértices em cada rede e a diferença no número de arestas geradas pela presença das falhas, já que cada aresta em falha deve ser descontada do total de arestas presentes no grafo.

4.2.2 Tempo de Convergência

A análise de tempo tem por objetivo definir o limite máximo de tempo decorrido desde o início de execução do algoritmo até sua convergência. Sendo esta análise realizada através da quantificação do período máximo decorrido em unidades de tempo. Nesta análise, é

assumido que cada unidade de tempo equivale ao tempo dispendido durante a transmissão de uma mensagem.

Como o tempo é um fator de suma importância durante o restabelecimento do fornecimento de energia elétrica, todas as unidades de processamento iniciam no estado de `Busca`, pois no caso das unidades iniciarem em seu estado `Adormecido` seria necessário que esta unidade recebesse uma mensagem qualquer fazendo com que ocorresse a mudança de seus estado e por consequência o inicio de sua comunicação. Como a unidade já parte do estado `Busca`, ela inicia a troca de mensagem independente de ter recebido ou não mensagem de seus vizinhos.

Assumindo que em V unidades de tempo, cada unidade no estado de `Busca` executa a transmissão de uma mensagem `Conectar` à um de seus vizinhos iniciando o processo de busca da AEM. Em $2\,V$ unidades de tempo, cada unidade passa do nível 0 para o nível 1 de busca da AEM, através da propagação da mensagem do tipo `Iniciar`, conforme a característica de funcionamento do algoritmo GHS apresentado na Seção 1.3. Para que todas as unidades de processamento passem do nível 1 para um nível $l + 1$, o algoritmo requer no máximo $5(l + 1)V - 3V$ unidades de tempo.

Sendo assim, as unidades requerem um tempo máximo de $5 \log_2 V - 3V$ para percorrer os níveis de 1 até o nível máximo $\log_2 V$. Partindo desta condição, o tempo máximo, T_{max}, que o algoritmo pode levar até a convergência é expresso pela Equação 17 (GALLAGER; HUMBLET; SPIRA, 1983):

$$T_{max} = 5V \log_2 V, \tag{17}$$

definida pelo somatório do tempo V gasto no início do processo de busca da AEM com a transmissão das mensagens do tipo `Conectar`, o tempo $2V$ de transição do nível 0 para o nível 1 através da propagação das mensagens do tipo `Iniciar`, além de $5 \log_2 V - 3V$ referente ao tempo gasto pelas unidades de processamento para percorrerem do nível 1 até o nível $\log_2 V$.

Durante a fase de teste, o tempo médio de execução, para uma rede com a uma quantidade V de unidades de processamento é independente da quantidade de linhas nela contidas, e deve apresentar-se menor ou igual ao T_{max} encontrado.

A análise do tempo de convergência consiste na comparação entre a quantidade de unidades de tempo necessária até a convergência do algoritmo e o limite máximo de

unidades de tempo obtidas conforme apresentado na Equação 17.

Para esta comparação, inicialmente, foi necessário o cálculo do tempo médio de execução do algoritmo utilizando os valores obtidos em cada uma das cinco repetições executadas para cada um dos testes. Para este tempo de duração, tomou-se como base o número total de *tics* do processador presente no computador tendo sido contabilizado desde o início do processo de comunicação até a convergência do algoritmo. Para cada um dos testes realizados em cada cenário, dividiu-se a média de *tics* obtida, pela média do número de mensagens geradas em cada uma das cinco repetições realizadas por testes. O resultado obtido, referente à média de *tics* por mensagem, consiste no valor da média de tempo transcorrido durante o envio de uma única mensagens durante a execução do algoritmo em cada um dos testes. Em seguida, foi calculado a média de *tics* por mensagem ocorridos em todos os testes, implicando no valor de uma unidade de tempo.

Para a obtenção da quantidade de unidades de tempo necessárias para a convergência da solução em cada cenário de testes, o valor médio de *tics* obtidos em cada testes foi dividido pelo valor de uma unidade de tempo.

4.3 Resultados de Auto-cura

A implementação do algoritmo distribuído foi realizada utilizando o compilador Arduino, versão 1.0.1, com suporte às definições do robô Elisa-3 que é adotado como unidade de processamento para execução e testes do algoritmo. Vale ressaltar que detalhes sobre o Elisa-3 foram abordados no Capítulo 3.

Na implementação da comunicação, foi utilizado o compilador Code::Blocks, versão 8.02, para gerar o programa necessário para operar em conjunto com a estação rádio-base, dotada de um *chip* de rádio *Ultra low power* 2.4GHz *RF Transceiver*, provendo o roteamento das mensagens entre as unidades de processamento. Este programa foi executado epor um computador com processador Intel Core i3 de 1.50GHz, memória RAM de 4GB e sistema operacional Windows 8 de 64 *bits*.

Cada uma das redes, IEEE 14 barramentos e IEEE 10 barramentos, utilizadas para a avaliação da solução proposta foram submetidas à três cenários de testes:

1. O primeiro consiste em uma rede em estado normal, ou seja, sem falha presente, onde será objetivada a configuração de rede que apresente a menor impedância

equivalente conferindo um maior fluxo de potência transferida do alimentador até as cargas conectadas nos barramentos;

2. O segundo cenário, consiste em uma rede que apresenta uma descontinuidade em uma das linhas do sistema caracterizado como falha simples. Este cenário é repetido para que a descontinuidade seja testada em cada uma das linhas do sistema, a fim de se validar a característica de recomposição da rede de distribuição advinda do algoritmo;

3. O terceiro cenário submete as redes a duas descontinuidades presentes simultaneamente, em duas linhas distintas com intuito de validar a capacidade do algoritmo em restruturar redes acometidas por falhas múltiplas.

Como a análise das duplas falhas trata-se de um análise combinatória do total de linhas do sistema consideradas duas a duas, o espaço de busca torna-se muito extenso, sendo 190 e 78 combinações possíveis para as rede de 14 e 10 barramentos, respectivamente. Então, para viabilizar os testes optou-se pela escolha arbitrária de 8 diferentes possibilidades de combinação para a rede IEEE 14 barramentos e 4 diferentes possibilidades para a rede IEEE 10 barramentos, o que representa, respectivamente, aproximadamente 4% e 5% das possibilidades de cada uma destas redes. Para esta escolha tentou-se capturar diferentes possibilidades, tais como, falha uma linha energizada e outra desenergizada, falha em duas linhas energizadas e falhas que pudessem deixar algum barramento sem recurso para recompor seu fornecimento.

Cada um dos testes executados, foram repetidos cinco vezes com a finalidade de minimizar quaisquer possíveis erros de medição tanto do tempo até a convergência quanto a contabilização do número de mensagens trocadas.

Durante a obtenção de cada uma das configurações de rede, os dados referentes aos números de mensagens e o tempo decorrido até a convergência do algoritmo foram levantados para serem analisados e posteriormente comparados com os valores teóricos esperados para o Custo da Comunicação e o Tempo de Convergência.

4.3.1 Redes sem Falhas

O primeiro caso abordado nos testes foram as redes sem falhas. A configuração ótima apresentada pelo algoritmo para a rede IEEE 14 barramentos, é ilustrada na Figura

26. Nela podem ser verificados os estados das linhas desenergizadas, representadas pelos segmentos pontilhados e das linhas conectadas, representadas pelas linhas contínuas. Note que nesta configuração todos os barramentos são conectados ao alimentador G.

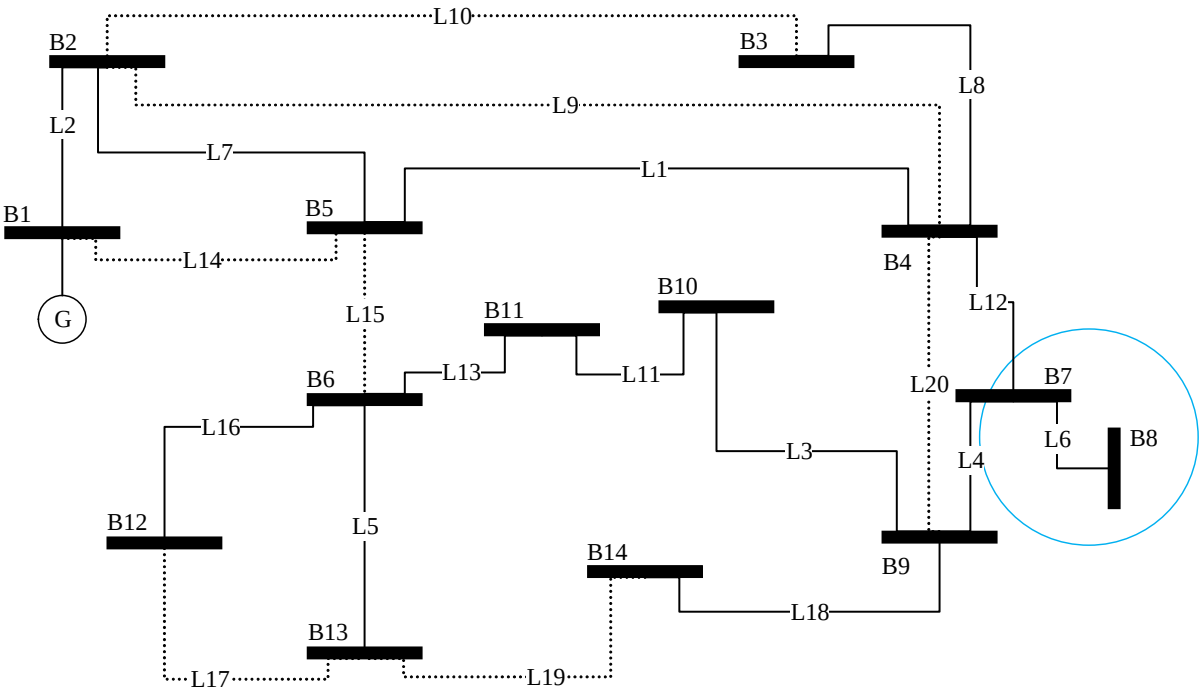


Figura 26: Configuração ótima resultante para rede IEEE 14 barramentos.

Para a rede IEEE 10 barramentos, a implementação gerou a configuração representada na Figura 27, onde as linhas pontilhadas apresentam-se desenergizadas e as demais conectadas.

Para ambos os casos, as redes resultantes apresentam uma proposta de configuração para a maximização do fluxo de potência gerado através da minimização das perdas oriundas da minimização da impedância equivalente do sistema, anteriormente detalhado na Seção 3.2.

A Figura 28(a) apresenta a comparação realizada entre a média do número de mensagens obtidas (*Prático*) e o limite máximo de mensagens esperadas (*Teórico*), gerado de forma algébrica, enquanto que a Figura 28(b) reflete as comparações entre o limite máximo de unidades de tempo e o tempo real transcorrido na execução, ambas para as redes IEEE 10 barramentos (Rede 10) e IEEE 14 barramentos (Rede 14) para o cenário de teste sem falha.

Os resultados de número de mensagens e unidades de tempo obtidos durante a execução da implementação para ambas as redes se apresentam com valores inferiores aos

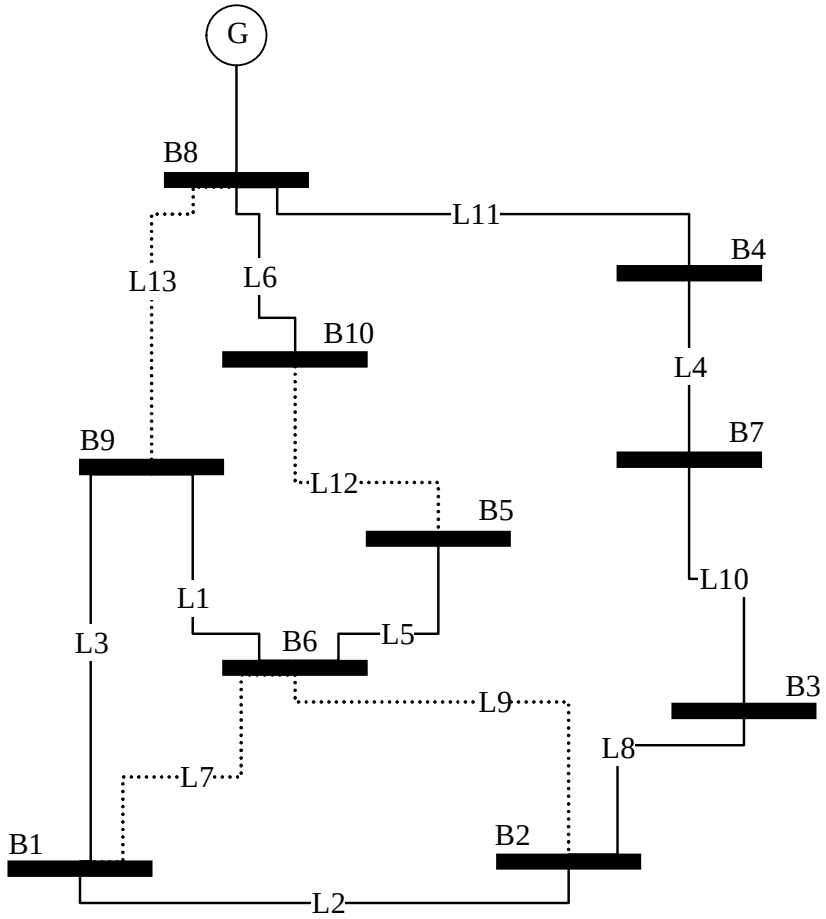
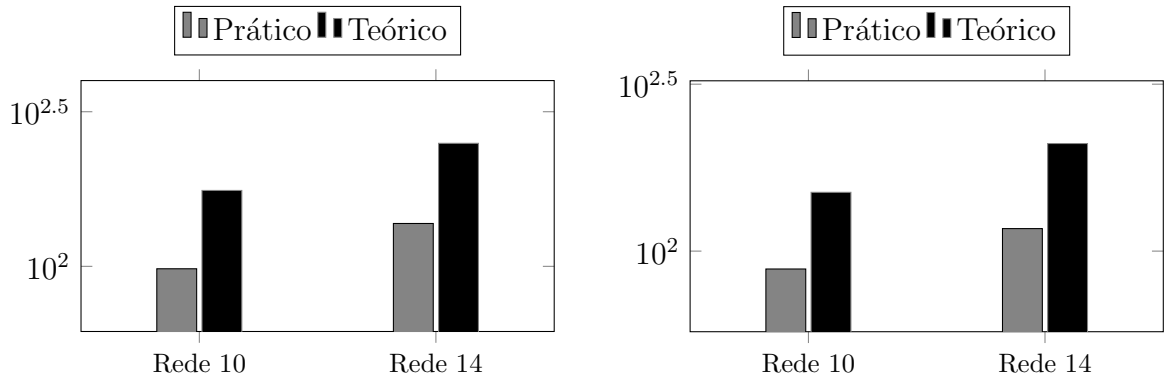


Figura 27: Configuração ótima resultante para rede IEEE 10 barramentos.

limites máximos estabelecidos pelas Equações 16 e 17 conforme estabelecido na Seção 4.2.



(a) Número de mensagens (b) Tempo de reconfiguração

Figura 28: Comparação dos números de mensagens e tempo de de reconfiguração nos casos de falhas simples.

4.3.2 Redes com Falhas Simples

A implementação foi submetida aos testes dos cenários de falhas simples, para o número total de linhas, *nl*, existentes nas redes redes IEEE 14 barramentos e IEEE 10 barramentos. Partindo da premissa de que as cada uma das redes, encontram-se com as configurações ótimas encontradas pela implementação no caso do primeiro cenário (sem falhas).

As propostas de reconfiguração da rede obtidas foram sintetizadas e apresentadas na Tabela 12. As 3 possíveis soluções sintetizadas são:

- Combinação de linhas L_i, $1 \leq i \leq nl$, que deverão ter suas chaves comutadas para que a rede de distribuição seja total ou parcialmente restabelecida;
- Casos de linhas L_i, onde não é possível simular falha na , pois a linha encontrava-se desconectada no sistema utilizado como premissa. Devido a este fato qualquer problema que acometa a linha não sensibiliza a solução. Estes casos são identificados na tabela como *Sem Ação*;
- Casos onde não existem reconfigurações possíveis, identificados como *Não há Recurso*. Como é o exemplo da linha $L6$ da rede IEEE 14 barramentos, que será abordado posteriormente.

Cada uma das configurações propostas para as redes IEEE 14 barramentos e IEEE 10 barramentos, podem ser verificadas no Apêndice A.

A Figura 29 traz um exemplo de reconfiguração proposta pela implementação para a falha presente na linha L_1 da rede IEEE 14 barramentos. Pode ser visto que na Figura 29(a), referente ao estado durante a falha, a linha $L1$ encontrava-se energizada, enquanto a linha $L9$ estava desenergizada. Com a ocorrência da falha na linha $L1$, o barramento $B4$ ficou desenergizado, assim como todo o circuito conectado à ele. A solução implementada, gerou uma nova configuração para a rede, desabilitando a utilização da linha $L1$ e habilitando a linha $L9$, através de suas respectivas chaves. Pois a linha $L9$ é o recurso com menor impedância possível de ser comutada para normalizar o fornecimento de energia da rede, garantindo o menor somatório dos pesos dentre as combinações de linhas possíveis de serem utilizadas, como pode ser visto na Figura 29(b).

Como exemplo de proposta de reconfiguração para rede de IEEE 10 barramentos, pode-se verificar na ilustração da Figura 30. Nela a linha L_2 encontra-se em falha afetando

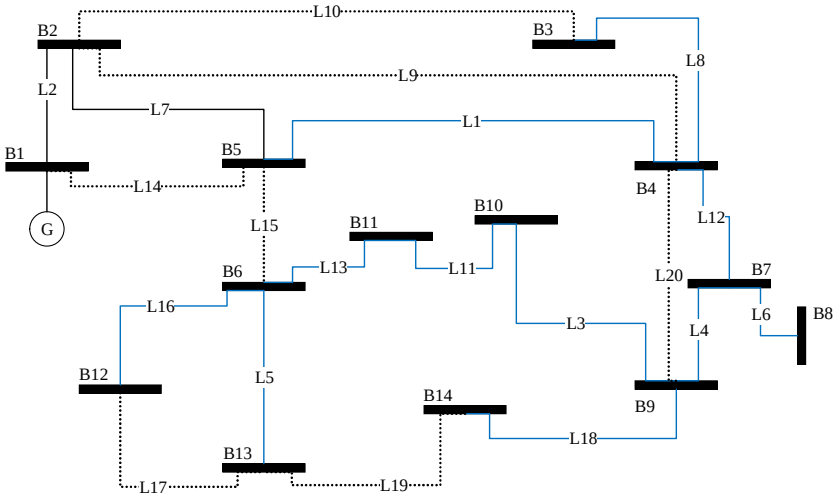
Tabela 12: Reconfigurações para falhas simples das redes IEEE 10 e 14 barramentos.

	IEEE 10 Barramentos	**IEEE 14 Barramentos**
Linha em Falha	Linhas Comutadas	
L_1	L_1, L_7	L_1, L_9
L_2	L_2, L_9	L_2, L_{14}
L_3	L_3, L_7	L_3, L_{15}
L_4	L_4, L_{12}	L_4, L_{15}
L_5	L_5, L_{12}	L_5, L_{17}
L_6	L_6, L_{12}	(Não há Recurso)
L_7	(Sem Ação)	L_7, L_9
L_8	L_8, L_{12}	L_8, L_{10}
L_9	(Sem Ação)	(Sem Ação)
L_{10}	L_{10}, L_{12}	(Sem Ação)
L_{11}	L_{11}, L_{12}	L_{11}, L_{19}
L_{12}	(Sem Ação)	L_{12}, L_{19}
L_{13}	(Sem Ação)	L_{13}, L_{19}
L_{14}	-	(Sem Ação)
L_{15}	-	(Sem Ação)
L_{16}	-	L_{16}, L_{17}
L_{17}	-	(Sem Ação)
L_{18}	-	L_{18}, L_{19}
L_{19}	-	(Sem Ação)
L_{20}	-	(Sem Ação)

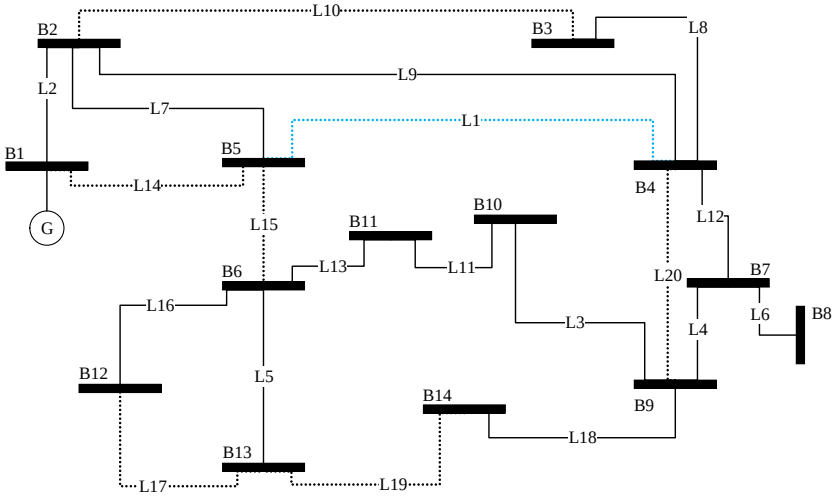
o barramento B_1 e todo o circuito a ele conectado, conforme mostrado Figura 30(a). A proposta de reconfiguração obtida pela solução indica a comutação das linhas $L2$, para abertura de sua chave devido a falha, e $L9$, para fechamento de sua chave restabelecendo o fornecimento de energia aos barramentos afetados. A Figura 30(b) mostra a rede após a reconfiguração das chaves.

Na coluna referente à rede de 14 barramentos, pode ser verificado que exite um caso em que não é possível efetuar a recomposição da rede, pois não há recurso disponível para isso, ou seja, somente há uma única linha que conecta o barramento B_8 à rede. No caso de uma falha na linha $L6$ que conecta o barramento B_8 ao restante da rede de distribuição, conforme mostrado em detalhe na Figura 26, não há outras linhas que possam ser utilizadas como recurso para realimentação do barramento $B8$, ficando este desenergizado.

Os casos de falha nas linhas L_9, L_{10}, L_{14}, L_{15}, L_{17}, L_{19} e $_{20}$ da rede IEEE 14 barramentos e das linhas L_7, L_9, L_{12} e L_{13} da rede IEEE 10 barramentos não apresentam



(a) Rede IEEE 14 Barramentos antes da falha



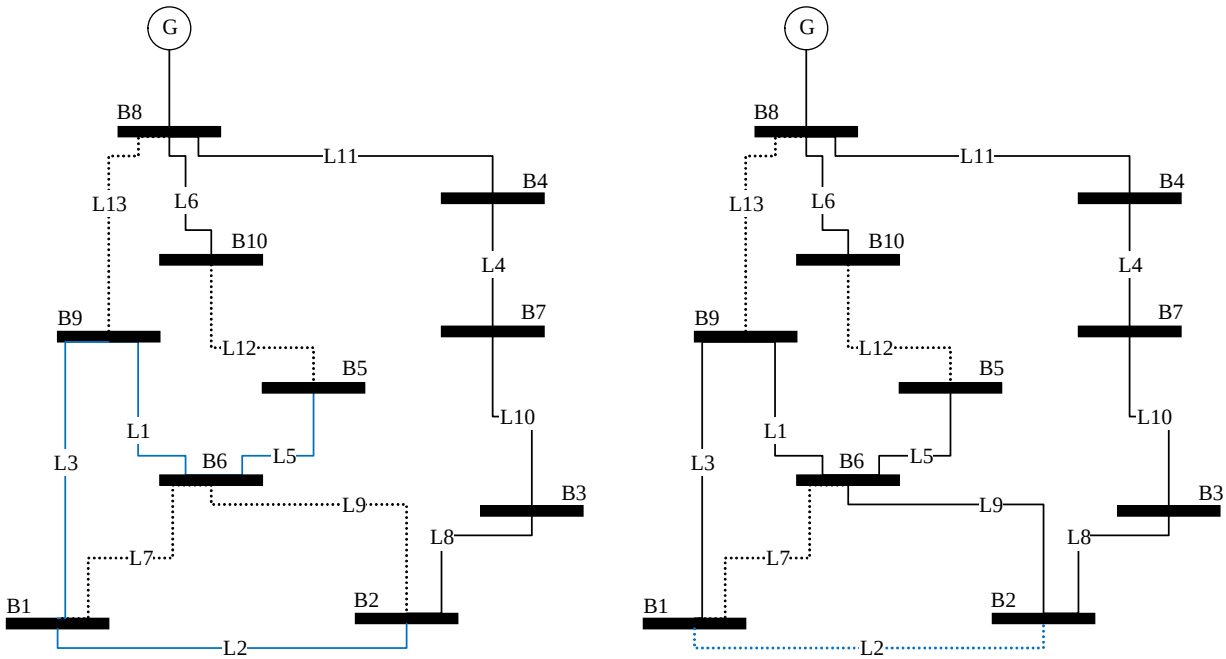
(b) Rede IEEE 14 Barramentos após a falha.

Figura 29: Exemplo de reconfiguração de rede para falha simples para rede IEEE 14 barramentos.

propostas de reconfiguração, visto que estas linhas já se encontram desabilitadas não causando impacto no fornecimento de energia elétrica para os barramentos da rede.

As Figuras 31 e 32 apresentam a comparação realizada entre as médias dos números de mensagens obtidas e o limite máximo de mensagens esperados para cada um dos cenários de falhas simples que apresentaram possibilidades de reconfiguração para rede. Os dados dos gráficos encontram-se identificados pela linha em falha, para as redes IEEE 14 e 10 barramentos.

Uma característica percebida nos resultados dos testes apresentados na Figura 31 é a ocorrência de uma diferença de valores entre os limites máximos de mensagens esperadas para a falha da linha $L6$ da rede IEEE 14 barramentos. Conforme explicado



(a) Rede IEEE 10 Barramentos antes da falha (b) Rede IEEE 10 Barramentos após a falha.

Figura 30: Exemplo de reconfiguração de rede IEEE 10 barramentos para falha simples.

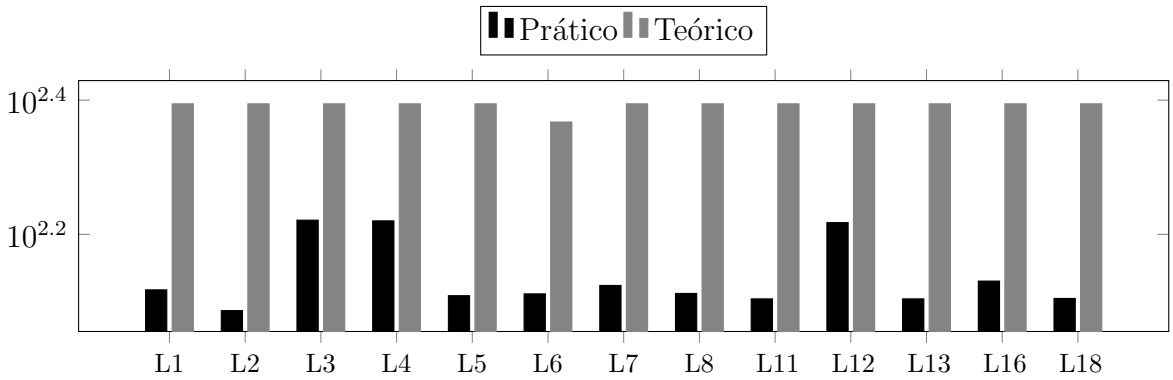


Figura 31: Comparação dos números de mensagens em casos de falhas simples na rede IEEE 14 barramentos.

anteriormente, na presença da falha simples da linha $L6$ para a rede IEEE 14 barramento, o barramento $B8$ ficou completamente isolado da rede, visto que a única linha existente para sua alimentação é a linhas $L6$. Com isto o barramento $B8$ foi expurgado do conjunto de vértices do grafo que representa a rede, para que este seja um grafo conexo, o que fez com que a quantidade total de vértices para esta análise passa de 14 para 13, o que gerou a diferença no limite máximo de mensagens.

Os resultados obtidos e apresentados nas Figuras 33 e 34 dizem respeito às comparações entre o limite máximo em termos de unidades de tempo (*Teórico*) e o tempo

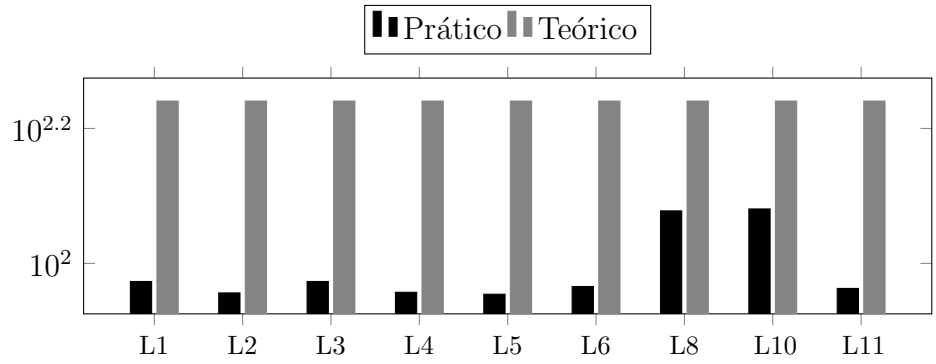


Figura 32: Comparação dos números de mensagens em casos de falhas simples na rede IEEE 10 barramentos.

total transcorrido (*Prático*) na execução de cada um dos testes. Lembrando que estes dados somente puderam ser obtidos para as falhas que apresentaram possibilidades de reconfiguração do cenário de falha simples para cada uma das linhas para as redes IEEE 14 barramentos e IEEE 10 barramentos, respectivamente.

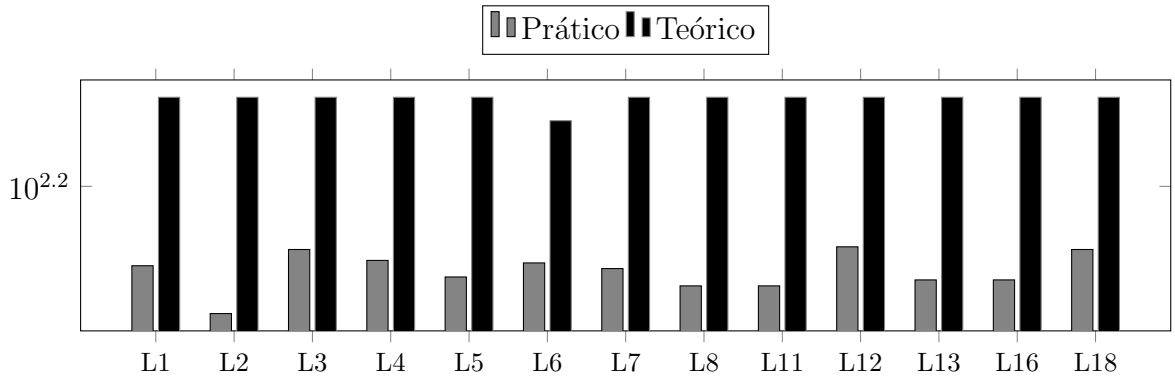


Figura 33: Comparação de tempo de reconfiguração dos casos de falhas simples da rede IEEE 14 barramentos.

Para os dados apresentados nos gráficos das Figuras 33 e 34 pode-se identificar que para todos os casos analisados, os tempos de convergência da implementação se apresentaram abaixo do limite teórico. Além disto, a diferenciação do valor limítrofe calculado para o tempo de convergência da falha na linha $L6$ da rede IEEE 14 barramentos, também acontece tal como para o caso discutidos na análise do número de mensagens. Esta diferenciação também é ocasionada pela diferença do número de vértices que passam a compor o grafo da rede, gerada pela falta de recurso para o restabelecimento da alimentação do barramento $B8$ afetado.

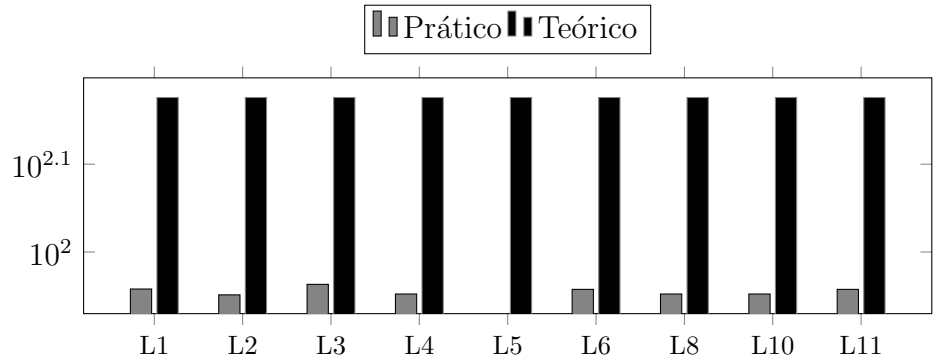


Figura 34: Comparação de tempo de reconfiguração dos casos de falhas simples da rede IEEE 10 barramentos.

4.3.3 Casos de Falhas Múltiplas

Os últimos testes realizados compreendem o cenário de falhas múltiplas, através do qual foi possível identificar a eficiência da implementação em realizar a reconfiguração da rede de distribuição que apresenta mais de uma falha. Os casos de falhas elencados, representados por uma par de linhas L_i–L_j, $1 \leq i \leq nl$ e $1 \leq j \leq nl \mid i \neq j$, além das respectivas linhas que deverão ter suas chaves comutadas são disponibilizados nas Tabelas 13 e 14 para as 2 redes estudadas, respectivamente. No Anexo B pode ser visualizado cada uma das propostas de reconfiguração de rede apresentadas nas tabelas.

Tabela 13: Reconfigurações para as falhas múltiplas das redes IEEE 14 barramentos.

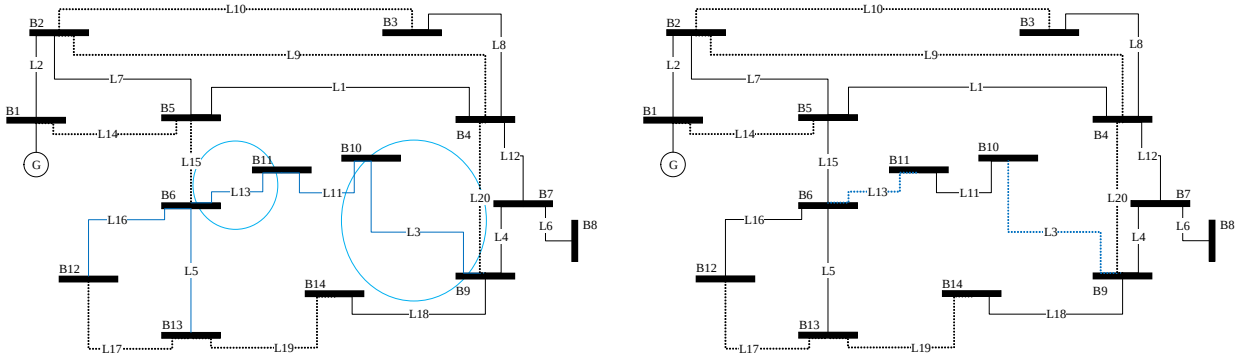
	IEEE 14 Barramentos
Linha em Falha	Linhas Comutadas
L_3–L_{13}	L_3, L_{13}, L_{15}
L_4–L_{20}	L_4, L_{15}
L_7–L_{14}	L_7, L_9
L_9–L_{15}	(Sem Ação)
L_{12}–L_{18}	L_{12}, L_{15}, L_{18}, L_{19}
L_{13}–L_{17}	L_{15}
L_{13}–L_{19}	L_{13}, L_{15}, L_{19}
L_{15}–L_{16}	L_{16}, L_{17}

No teste realizado para a rede de 14 barramentos, onde a falha se localizou nas linhas $L3$ e $L13$, responsáveis por conectar os respectivos barramentos $B10$ e $B11$ à rede de distribuição, a reconfiguração da rede foi efetuada parcialmente, pois não haviam recursos que pudessem contornar a falta de alimentação nestes barramentos. A Figura 35(a) mostra que as únicas linhas que conectam os barramentos $B10$ e $B11$ à rede encontram-se em

Tabela 14: Reconfigurações para falhas múltiplas das redes IEEE 10 barramentos.

	IEEE 10 Barramentos
Linha em Falha	Linhas Comutadas
L_1–L_5	L_1, L_5, L_7, L_{12}
L_3–L_9	L_3, L_7
L_7–L_6	L_6, L_{12}
L_8–L_{10}	L_8, L_{10}, L_{12}

falha. No entanto, o restante da rede pode ser recuperada através da comutação da linha $L15$, conforme a Figura 35(b).



(a) Detalhe das conexões em falha dupla dos barramentos $B10$ e $B11$ à rede.

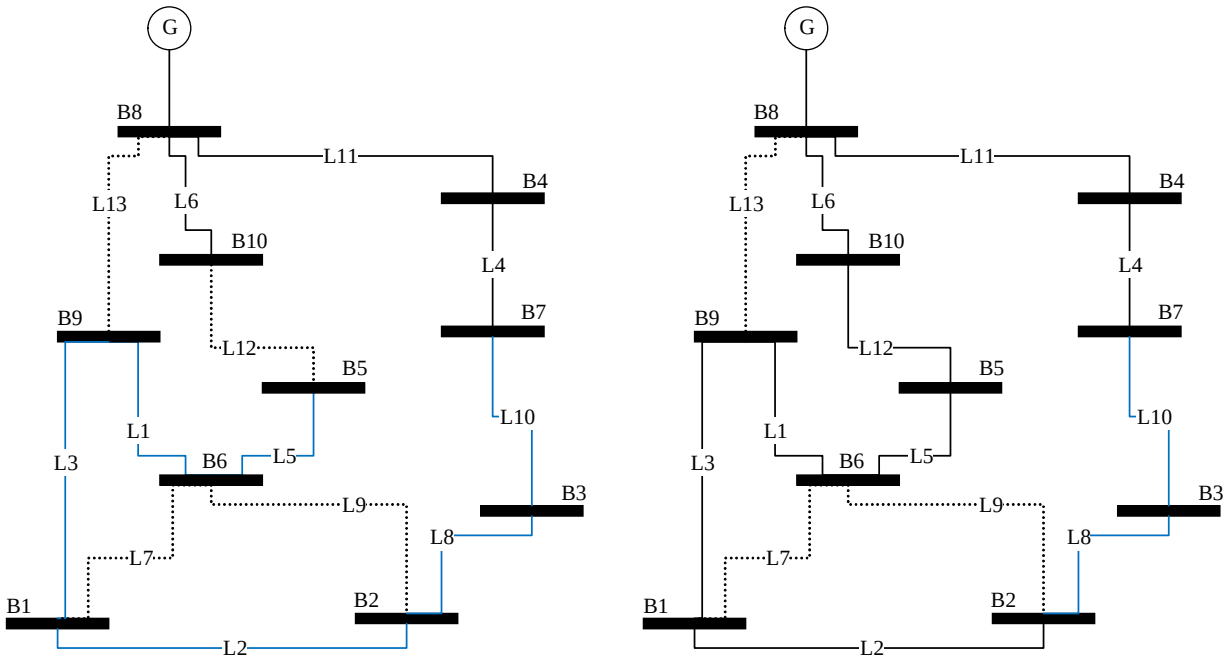
(b) Rede após a falha com reconfiguração parcial.

Figura 35: Exemplo de reconfiguração de rede IEEE 14 barramentos para falhas múltiplas.

Uma mesma ocorrência de reconfiguração parcial pode ser verificado para o caso das falhas nas linhas L_8 e L_{10} da rede IEEE 10 barramentos, conforme verificado na Figura 36(a). Onde, mesmo após a utilização da linha L_{12} como recurso, o barramento B_3 ficou isolado do sistema devido a inexistência de outros recursos para sua alimentação, conforme mostrado na Figura 36(b).

A mesma análise comparativa utilizando da média do número de mensagens obtidas e o limite máximo de mensagens de mensagens, também foi aplicada para o cenário de falhas múltiplas. Os resultados estão disponibilizado nas Figuras 37 e 38, para as rede de IEEE 14 barramentos e IEEE 10 barramentos, respectivamente.

Para os testes de falhas múltiplas também pode ser percebido uma diferença de valores entre os limites máximos de mensagens esperadas para alguns testes, tais como o de falha múltipla para o par de linhas $L_3 - -L_{13}$, da Figura 37 e o par de linhas $L_8 - -L_{10}$, da Figura 38. Nestas situações, os barramentos B_{10} e B_{11}, para a rede de 14 barramentos, e o barramento B_3, para a rede de 10 barramentos, não possuíam outras



(a) Detalhe das conexões em falha dupla dos barramentos $B10$ e $B11$ à rede.

(b) Rede após a falha com reconfiguração parcial.

Figura 36: Exemplo de reconfiguração de rede IEEE 10 barramentos para falhas múltiplas.

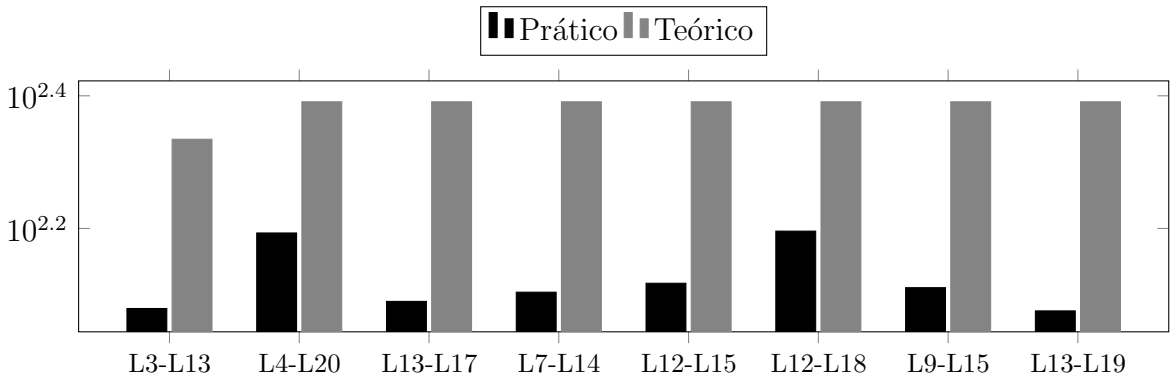


Figura 37: Comparação dos números de mensagens nos casos de falhas múltiplas na rede IEEE 14 barramentos.

linhas de alimentação que pudessem ser utilizadas como recurso para o restabelecimento da conexão de todos os barramento à rede. Em virtude disto, os vértices que representam tais barramentos deixam de fazer parte do conjunto de vértices do grafo conexo utilizado como modelo do sistema de distribuição. A análise para estas falhas utilizam então, um número de vértices menor do que para as demais falhas analisadas, ou seja, para o isolamento dos barramentos B_{10} e B_{11}, para a rede de 14 barramentos a rede passa de 20 para 18 barramentos no total. Já, no caso do isolamento do barramento B_3, para a rede de 10 barramentos a rede passa de 13 para 12 barramentos no total.

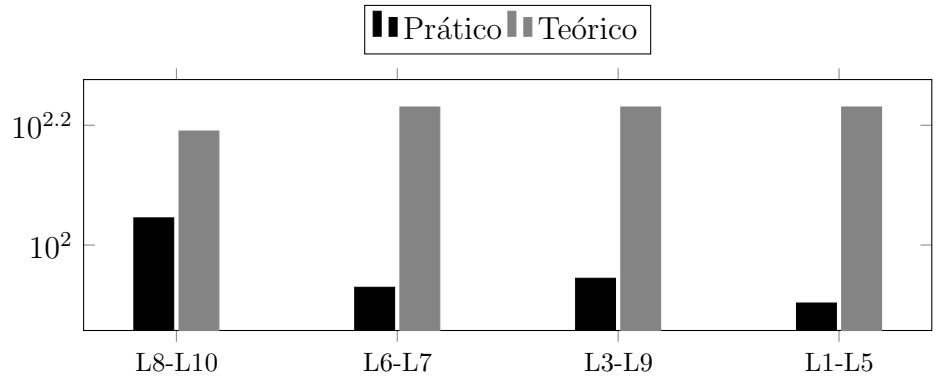


Figura 38: Comparação dos números de mensagens nos casos de falhas múltiplas na rede IEEE 10 barramentos.

Para todos os resultados obtidos, o número total de mensagens utilizadas pelo algoritmo apresentou-se menor do que o limite máximo de mensagens definido pela Equação 16.

Uma segunda característica percebida nos gráficos é a acentuada diferença entre os valores reais e teóricos. Este fato pode ocorrer, pois a formulação apresentada na Subseção 4.2.1, baseia-se na possibilidade de utilização de todos tipos de mensagens possíveis para cada nível no qual o vértice pode percorrer, até a convergência, sem levar em consideração a disposição dos elementos vértices, arestas e pesos que compõem grafo. A configuração destes elementos impacta na quantidade de mensagens trocadas, uma vez que um ou mais vértices podem não fazer uso de toda a gama de mensagens possíveis dependendo da configuração do grafo. Por exemplo, uma unidade de processamento que já tenha definido os estados de suas arestas presentes na matriz *Vizinhança*, definida na Seção 3.3.2, não participará mais do processo de troca de mensagens, dos estados das demais arestas que passarão a compor a AEM.

A formulação do cálculo do limite máximo de tempo para a convergência do algoritmo, também não baseia-se nas características de configuração do grafo modelo da rede de distribuição, tal como a análise do limite máximo de mensagens. A formulação apresentada na Equação 17, utiliza apenas como referência de cálculo o limite máximo de níveis ($\log_2 V$) pelos quais os vértices podem percorrer até a convergência. Isto explica a diferença que ocorre entre os valores de tempo máximo e os valores reais medidos durante os testes, uma vez que os vértices podem não precisar percorrer todos os níveis até a convergência da solução.

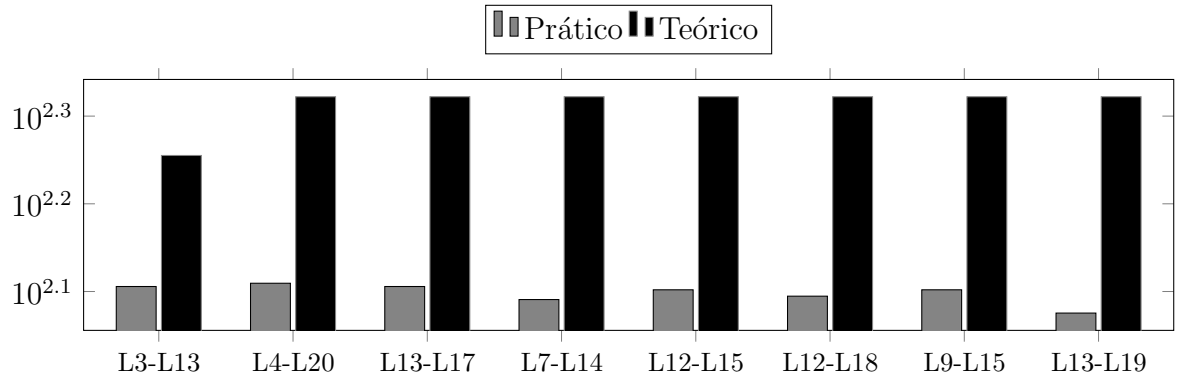


Figura 39: Comparação de tempo de reconfiguração dos casos de falhas múltiplas da rede IEEE 14 barramentos.

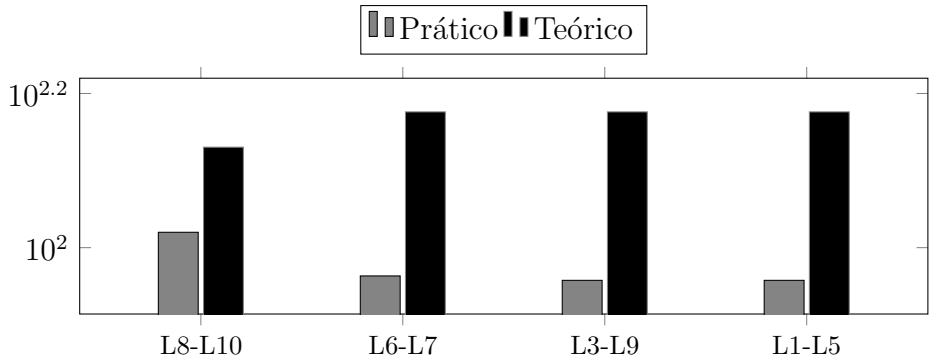


Figura 40: Comparação de tempo de reconfiguração dos casos de falhas múltiplas da rede IEEE 10 barramentos.

4.4 Considerações Finais do Capítulo

Neste capítulo foram apresentados os resultados obtidos pela implementação do algoritmo distribuído GHS para permitir a auto-cura em redes de distribuição de energia elétrica inteligentes, apresentada no Capítulo 3. Durante a modelagem dos dados foi proposta uma forma de remediar à restrição do algoritmo de GHS, quanto a presença de valores iguais dos pesos das arestas. O resultados apresentados comprovam a possibilidade da aplicação desta solução para os problemas de reconfiguração de rede, visto que a implementação conseguiu gerar as comutações das linhas em falha para atingir o restabelecimento do fornecimento de energia elétrica com o maior fluxo de potência possível. Outro ponto de importância foi o intervalo de tempo necessário para a convergência do algoritmo, que esta integrado com o baixo volume de mensagens trocados entre as unidades de processadoras da rede e a velocidade empregada no processo de comunicação.

O capítulo seguinte finaliza este trabalho, abordando suas principais conclusões,

bem como as contribuições mais relevantes da presente dissertação. Também serão tratadas direções para possíveis trabalhos futuros.

Capítulo 5

CONCLUSÕES E TRABALHOS FUTUROS

ESTE capítulo completa esta dissertação, apresentando as principais conclusões obtidas do desenvolvimento do presente trabalho. São apresentadas também possíveis melhorias do algoritmo proposto e as direções para futuros estudos, envolvendo o algoritmo GHS.

5.1 Conclusões

Esta dissertação abordou a estratégia de implementação distribuída da solução de autocura para as redes de distribuição inteligentes. Esta estratégia foi fundamentada na possibilidade de modelagem das redes de distribuição inteligentes em grafos com pesos possibilitando a utilização da técnica de árvores de expansão mínima, onde cada peso representa a impedância do da linha que interliga dois barramentos. Associado à isso foi utilizada a técnica de processamento distribuído para descentralizar a solução, através do algoritmo distribuído GHS (GALLAGER; HUMBLET; SPIRA, 1983). Assim a inteligencia para reconfiguração da rede passa estar presente em cada um dos elementos de chaveamento da rede de distribuição, possibilitando uma menor infraestrutura de comunicação necessária para viabilizar o processamento centralizado.

Este estudo foi motivado pela inovação ao se utilizar um algoritmo descentralizado para executar propostas de reconfiguração de rede, visto que os estudos atuais se concentram na utilização de processamentos centralizados. Seu objetivo foi investigar e desenvolver uma solução descentralizada que consiga efetuar o restabelecimento de energia de uma rede acometida por uma falha, visando a redução do tempo de execução, redução de infraestrutura e melhoria da qualidade do serviço.

A solução de auto-cura foi implementada através dos robôs Elisa-3, utilizados como unidades de processamento distribuído. A arquitetura explora o paralelismo do processamento do algoritmo responsável por encontrar a reconfiguração da rede através da árvore de extensão mínima. Cada elemento processador contribuí com o restante dos elementos processadores, por intermédio da troca de mensagens, na localização do conjunto de linhas que formarão o conjunto solução da reconfiguração.

A arquitetura de comunicação desenvolvida para prover a troca de mensagens entre os robôs Elisa-3 se mostrou limitada, pois ela não ocorre de forma direta. Os robôs necessitam de um elemento que execute o encaminhamento das mensagens, fazendo com que a troca de mensagens entre os robôs ocorra sequencialmente. Caso as unidades efetuassem uma comunicação direta o tempo de execução real do algoritmo seria menor.

A implementação desenvolvida foi validada por meio de duas redes de distribuição compostas por um alimentador, sendo uma rede integrada por 14 barramentos (KODSI; CAñIZARES, 2003) e a outra por 10 barramentos (FRAG; AL-BAIYAT; CHENG, 1995). Para estas redes foram simulados problemas de descontinuidade simples e descontinuidades simultâneas. Todas as linhas de cada uma das rede foi submetida aos testes de falhas simples, já para os testes de falhas simultâneas foram selecionados 4% e 8% das combinações possíveis das redes de 14 e 10 barramentos, respectivamente, devido ao elevado número de combinações de linhas.

Os resultados obtidos mostram que o algoritmo é capaz de gerar uma reconfiguração que permite recuperar parcial ou totalmente o fornecimento e energia da rede. Isto depende diretamente do número de recursos existentes para cada barramento e a localização das falhas. Já que caso uma falha comprometa um barramento que possui apenas uma linha que o conecta a rede, não existe recurso que possa ser utilizado na sua recuperação. O algoritmo seleciona as linhas que possuem o menor valor de impedância, sempre buscando abranger todos os barramentos presentes na rede, criando uma rede na qual o somatório das impedância e das perdas de potência são as menores possíveis.

Devido à limitação no número de unidades de processamento não foi possível realizar a comparação dos resultados com outras soluções com processamento centralizado. Assim as avaliações realizadas tiveram como foco o custo de comunicação e a análise de tempo imposto pela implementação proposta, tomando como base a modelagem de um sistema de distribuição de energia elétrica representado por um grafo conexo e acíclico.

Os dados obtidos para estas duas métricas foram confrontados com os valores máximos teóricos calculados para cada configuração de rede. Todos os dados coletados se mantiveram dentro do limite máximo teórico, validando a correta implementação do algoritmo GHS.

Outro ponto identificado durante os testes foi a ocorrência de perda de mensagens durante a comunicação. A comunicação via RF 2.4GHz, se demonstrou altamente susceptível a interferência do meio, sendo necessária a reinicialização dos destas inúmeras vezes. Em alguns casos a perda das mensagens não influenciava na execução do algoritmo, enquanto que em outros momentos a perda de uma única mensagem impactava diretamente no funcionamento da solução.

5.2 Trabalhos Futuros

Nesta seção, são citadas algumas possíveis modificações nos algoritmos propostos, com o intuito de melhorar seu desempenho. Também são levantadas propostas para futuros trabalhos na área de auto-cura descentralizada.

Uma primeira investigação a ser realizada sobre a solução proposta é a análise do carregamento do alimentador, com o intuito de se evitar sobrecarga. A implementação somente atenta para diminuição das perdas de potência ativa e reativa da rede. Uma medida seria incorporar ao algoritmo um método de seleção para realização da interrupção do fornecimento de energia de barramentos com baixa prioridade de atendimento. O desafio desta implementação está no fato de que as unidades de processamento da rede necessitariam conhecer dinamicamente mais informações da rede do que somente informações de cada linha que os conectam à seus vizinhos.

Uma Segunda investigação a ser feita é a implementação do algoritmo proposto em redes com mais de um alimentador. Os testes realizados nesta dissertação foram todos considerando redes com apenas um único alimentador. O acréscimo de alimentadores nas redes requer um estudo maior sobre a forma de realização de distribuição das cargas em cada um dos alimentadores, para que nenhum alimentador fique sobrecarregado. Além de haver uma preocupação pela garantia de não haver conexões comuns entre os alimentadores, impedindo a ocorrência de um curto-circuito entre alimentadores. A dificuldade para esta implementação está em encontrar um método que possibilite a cada unidade

detectar a linha que deve ser seccionada para que o ocorra o equilíbrio de cargas ocorra e o impedimento do curto-circuito.

Uma terceira investigação consistiria em identificar uma estratégia para que, caso ocorra uma outra falha durante o processo de reconfiguração, os elementos distribuídos da rede tomem conhecimento. A solução desenvolvida não contém um monitoramento constante sobre os estados das arestas enquanto a busca pela reconfiguração está sendo executada. Assim o algoritmo deveria ser reinicializado utilizando os dados mais atuais do estado da rede. O desafio desta implementação está em gerar um processo de comunicação que informe à todos os elementos da rede que o processo precisa ser reiniciado.

Outra investigação que pode ser considerada é o atendimento de cargas prioritárias da rede, a solução desenvolvida não trata prioridade das cargas. Esta prioridade, permitiria por exemplo o isolamento de alguns barramentos que atendem consumidores simples em detrimento de outro barramento que atenda um hospital, no caso em que uma falha da rede possa gerar uma sobrecarga no alimentador. Para esta situação poderiam ser estudados métodos de categorização dos barramentos para que os prioritários tenham um diferencial nos valores de pesos em suas arestas, enquanto que os menos prioritários possam ser excluídos temporariamente da rede.

Uma última investigação que poderá gerar um ganho para a solução desenvolvida, é a identificação das mensagens criticas, ou seja, aquelas que não podem ser perdidas durante a comunicação, para elaboração de uma arquitetura de comunicação mais robusta.

Por fim, a estratégia de reconfiguração de distribuição de energia elétrica de forma distribuída, podem ser utilizadas em aplicações de outras áreas. tal como redes de distribuição de água. Elas podem ser aplicadas no replanejamento de rotas de transito e de redes de distribuição de água, sendo necessário apenas a identificação de quais as grandezas deveram ser

REFERÊNCIAS

ALOISE, D. J.; CRUZ, J. S. *Teoria dos Grafos e Aplicações*. [S.l.: s.n.], 2001.

ARANGO, H.; TAHAN, C. M. V. *Perdas Técnicas de Energia*. [S.l.], 2005.

BARRY, C. S.-L. S. .; LO, C.-Y. A space-efficient short-finding algorithm [vlsi layouts]. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 1994.

BERNARDON, D. P. *Novos Métodos para Reconfiguração das Redes de Distribuição a Partir de Algoritmos de Tomadas de Decisão Multicriteriais*. Tese (Doutorado) — Universidade Federal de Santa Maria, 2007.

BLACK, P. E. Dijkstra's algorithm. *Dictionary of Algorithms and Data Structures*, 2006.

CORMEN, T. H. et al. *Algoritmos Teoria e Prática*. [S.l.]: Ed. Campus, 2002.

FARIA, D. A. M. *Operador de Recombinação EHR Aplicado ao Problema de Árvore Máxima*. Dissertação (Mestrado) — Universidade Federal de Goiás, 2013.

FERREIA, L. R. et al. Solução do problema de self-healing para redes de distribuição radiais através de otimização via algoritmos genéticos. *Periódico Desconhecido*, 2013.

FRAG, A.; AL-BAIYAT, S.; CHENG, T. C. Economic load dispatch multiobjective optimization procedures using linear programming techniques. *IEEE Transactions on Power Systems, Vol. 10, No. 2,*, 1995.

GALLAGER, R. G.; HUMBLET, P. A.; SPIRA, P. M. A distributed algorithm for minimum-weight spanning trees. *ACM Transactions on Programming Languages and Systems, Volume 5, Number 1*, 1983.

GCTRONIC. *Recursos Elisa-3*. [S.l.], Novembro 2013. Disponível em: <http://www.gctronic.com/doc/index.php/Elisa-3>.

GELLINGS, C. *The Smart Grid: Enabling Energy Efficiency and Demand Response*. [S.l.]: Fairmont Press, 2009.

ISONI, M. Arranjos de sistemas de distribuição elétrica para instalações de médio e grande porte. 2013. Disponível em: <www.engeparc.com.br/caribooostfiles/7-SistemasDistribuicao.pdf?>.

KEMPE, D. Lecture notes on the edmonds karp algorithm cs 570. *Periódico Desconhecido*, 2004.

KODSI, S. K. M.; CAñIZARES, C. A. *Modeling and Simulation of IEEE 14 Bus System with Facts Controllers*. [S.l.], 2003.

KRUSKAL, J. B. On the shortest spanning sub-tree and the travelling salesman problem. *Periódico Desconhecido*, 1956.

LEISERSON, C. E. et al. *Introduction to Algorithms*. [S.l.]: The MIT press, 2001.

LIMA, H. S. Arquitetura tcp/ip em redes de dados. 2013.

LIU, Y. Minimum spanning trees. *Periódico Desconhecido*, 2001.

MAHDAVI, I. et al. Optimal gas distribution network using minimum spanning tree. *Industrial Engineering and Engineering Management IEEM*, 2010.

MATOS, A. M. Sistema por unidade. 2003. Disponível em: <http://paginas.fe.up.pt/ mam/sistemapu.pdf>.

MATOS, M. A. Introdução ao trânsito de potências. 2013.

MONTOYA, D.; RAMIREZ, J. A minimal spanning tree algorithm for distribution networks configuration. *Power and Energy Society General Meeting, 2012 IEEE*, 2012.

MOREIRA, R. M. M. *Análise Técnico-Económico de Estratégias de Self-Healing em Smart-Grid*. Dissertação (Mestrado) — Faculdade de Engenharia da Universidade do Porto, 2011.

PRADO, F.; ALMEIDA, T. A.; SOUSA, V. N. Introdução ao estudo sobre Árvore geradora mínima em grafos com parâmetros fuzzy. *Periódico Desconhecido*.

PRIM, R. C. Shortest connection networks and some generalizations. *Bell system technical journal*, 1957.

RAO, R. S.; NARASIMHAM, S.; RAMALINGARAJU, M. Optimization of distribution network configuration for loss reduction using artificial bee colony algorithm. *International Journal of Electrical and Electronics Engineering 2:10 2008*, 2008.

ROSEN, K. H. *Discrete Mathematics and Its Applications*. [S.l.: s.n.], 1998.

ROSEN, K. H. *Discrete Mathematics and Its Applications*. [S.l.: s.n.], 2012.

SEDGEWICK, R. *Algorithms in C*. [S.l.]: Addison-Wesley, 2002.

SUDHAKAR, T.; SRINIVAS, K. Power system reconfiguration based on prim's algorithm. *Electrical Energy Systems (ICEES), 2011 1st International Conference on*, 2011.

SUDHAKAR, T.; SRINIVAS, K. Power system restoration based on kruskal's algorithm. *Electrical Energy Systems (ICEES), 2011 1st International Conference on*, 2011.

SUDHAKAR, T.; SRINIVAS, K. Power system restoration using reverse delete algorithm implemented in fpga. *Electrical Energy Systems (ICEES), 2011 1st International Conference on*, 2011.

SUDHAKAR, T. et al. Supply restoration in distribution networks using dijkstra's algorithm. *Power System Technology, 2004. PowerCon 2004. 2004 International Conference on*, 2004.

WENYU, L. J. . Y. et al. An improved minimum-cost spanning tree for optimal planning of distribution networks. *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, 2004.

YUNHAI, Z.; YONG, M. Optimal algorithm for system reconstruction. *Power System Technology, 2002*, 2002.