UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

EXPLORANDO TÉCNICAS DE BUSCA LOCAL COM MECANISMOS DE *SELF-HEALING* EM REDES DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA

PAULO HENRIQUE AQUINO DA SILVA

PATO BRANCO

2017

PAULO HENRIQUE AQUINO DA SILVA

EXPLORANDO TÉCNICAS DE BUSCA LOCAL COM MECANISMOS DE *SELF-HEALING* EM REDES DE DISTRIBUIÇÃO DE ENERGIA ELÉTRICA

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do Curso de Engenharia de Computação da Coordenação de Engenharia de Computação - COENC - da Universidade Tecnológica Federal do Paraná - UTFPR, Câmpus Pato Branco, como requisito parcial para obtenção do título de Engenheiro.

Orientador Prof. Dr. Marco Antônio de Castro Barbosa

PATO BRANCO

2017

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Engenharia de Computação

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

TERMO DE APROVAÇÃO

Às 13 horas e 50 minutos do dia 21 de novembro de 2017, na sala V008, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, reuniu-se a banca examinadora composta pelos professores Marco Antonio de Castro Barbosa (orientador), Dalcimar Casanova e Marcelo Teixeira para avaliar o trabalho de conclusão de curso com o título **Explorando técnicas de busca local com mecanismos de self-healing em redes de distribuição de energia elétrica**, do aluno **Paulo Henrique Aquino da Silva**, matrícula 01261592, do curso de Engenharia de Computação. Após a apresentação o candidato foi arguido pela banca examinadora. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

_____
Marco Antonio de Castro Barbosa
Orientador (UTFPR)

_____
Dalcimar Casanova
(UTFPR)

_____
Marcelo Teixeira
(UTFPR)

_____
Profa. Beatriz Terezinha Borsoi
Coordenador de TCC

_____
Prof. Pablo Gauterio Cavalcanti
Coordenador do Curso de
Engenharia de Computação

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

SILVA, Paulo H A. Explorando técnicas de busca local com mecanismos de *self-healing* em redes de distribuição de energia elétrica. 2017. 51p. Trabalho de Conclusão de Curso de bacharelado em Engenharia de Computação - Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

Este trabalho propõe um método para minimizar a complexidade de tempo do problema de *self-healing* em redes de distribuição de energia elétrica e apresenta o estudo de sua aplicação em diferentes metaheurísticas. Esse método consiste em fazer um nivelamento as chaves do sistema de distribuição de energia elétrica a partir da área que se encontra sem energia ao ocorrer uma falta. Após o nivelamento, é definido um nível máximo para considerar as chaves que podem ser manobradas, fazendo o bloqueio das demais. Dessa forma, os algoritmos, ao serem aplicados, terão um número reduzido de chaves para manobrarem e, então, poderão fazer uma busca maior em um espaço reduzido de soluções.

Palavras-chave: *self-healing*, *smart grids*, metaheurística, complexidade de tempo.

ABSTRACT

SILVA, Paulo H A. Exploring local search techniques for the problem of self-healing of smart grids. 2017. 51p. Trabalho de Conclusão de Curso de bacharelado em Engenharia de Computação - Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

This work proposes a method to minimizing the time complexity of the problem of self-healing in distribution smart grids and presents the study of its application by different mettaheuristics. This method consists by leveling the switches of the distribution system from the area that is in blackout after occurs a fault. After leveling, defines the maximun level to consider the switches that can be switched, blocking the rest. In this way, the algorithms, when applied, will have a reduced number of switches to work then them can make a bigger search in a smaller search space.

Keywords: *self-healing*, *smart grids*, metaheuristics, time complexity.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

Nas últimas décadas, com o aumento da utilização de tecnologias alimentadas por energia elétrica, o consumo dessa energia tem aumentado consideravelmente (AGENCY, 2008). Os usuários estão demandando maior qualidade e segurança do sistema de abastecimento de energia. Com isso, o uso e a manutenção mais eficientes desses sistemas são o foco nos dias de hoje (MOMOH, 2009). Um novo termo está sendo uma alternativa para gerenciamento dos sistemas elétricos modernos que engloba esses princípios, conhecido como *smart grids*.

Smart grids são redes de energia elétrica automatizadas e objetivam prover uma oferta acessível, confiável e sustentável de energia elétrica (LI *et al.*, 2010). A automação da distribuição provê aumento em eficiência nas operações do dia-a-dia e melhoria no processo de restauração, que inclui responder rapidamente às interrupções de energia, reduzindo erros operacionais e diminuindo a duração dessas interrupções (ZIDAN; EL-SAADANY, 2012).

Uma importante tarefa na estrutura das *smart grids* é a restauração o mais rapidamente possível após a ocorrência de uma falta de energia. O processo de restauração faz com que o sistema retorne a operar normalmente logo após qualquer parte do sistema ter sido perdida devido à interrupção (PEREZ-GUERRERO *et al.*, 2008). Esse processo é feito automaticamente quando há falta de energia em alguma área de cobertura do sistema elétrico, e é conhecido por *self-healing* (JIA *et al.*, 2011). *Self-healing* faz com que o sistema se reconfigure automaticamente empregando, geralmente, técnicas baseadas em inteligência artificial e heurísticas.

1.2 PROBLEMA

Self-healing pode ser visto como um mecanismo de autorrecuperação do sistema de distribuição de energia elétrica. Nesse contexto, esse mecanismo tem por objetivo reconfigurar a rede, após a ocorrência de uma falta, de modo que o sistema isole a área em que a falta ocorreu e reenergize áreas desenergizadas derivadas da área com falta. Esse mecanismo busca conectar o maior número de barras à energia logo após a ocorrência de alguma falta e minimizar o número de operações realizadas com as chaves, de modo que o sistema ainda obedeça as restrições básicas de uma rede de distribuição, como:

- manter a estrutura radial - estrutura básica de uma rede, o qual se assemelha

a uma árvore com a raiz representada por uma subestação;

- manter a tensão para cada consumidor dentro das normas, segundo a ANEEL - Agência Nacional de Energia Elétrica (apresentado na Tabela 1);

- manter o maior número de consumidores conectados - a resolução nº 456 (ANEEL, 2000) estabelece as concessionárias como responsáveis pela prestação de um serviço adequado a todos os consumidores, satisfazendo as condições de regularidade, generalidade, continuidade, eficiência, segurança, atualidade, modicidade das tarifas e cortesia no atendimento; (KAGAN *et al.*, 2009).

O problema de *self-healing* pode ser representado e estudado como um problema de teoria dos grafos, como, por exemplo, o Problema da Árvore Geradora Mínima Capacitada. Existem vários outros problemas que podem ser relacionados a grafos, como Circuito Hamiltoniano, Problema do Caixeiro Viajante, Cobertura de Vértices, Clique, entre outros (GOLDBARG; LUNA, 2000). Esses problemas são da classe NP-Completo (GAREY; JOHNSON, 1979).

Na esfera da computação, os problemas podem ser classificados entre P e NP. Os problemas P são aqueles que podem ser resolvidos de forma ótima por um algoritmo de tempo polinomial. Os problemas NP são aqueles que podem ser verificados em tempo polinomial, isto é, não é conhecido um algoritmo que encontre uma solução ótima em tempo polinomial, mas existe um algoritmo não determinístico que pode verificar em tempo polinomial se uma dada instância de entrada é solução para o problema ou não. Dentro da classe NP, existe uma subclasse conhecida como NP-Completo. Um problema é dito ser NP-Completo se todos os problemas NP podem ser reduzidos, em tempo polinomial, a ele (PAPADIMITRIOU; STEIGLITZ, 1998), ou seja, a partir de uma transformação polinomial qualquer outro problema NP pode ser convertido para esse mesmo problema NP-Completo.

Sendo assim, a restrição de manter a radialidade da rede pode ser visto como um caso particular do Problema da Árvore geradora Mínima Capacitada. Manter o fluxo de potência, respeitando-se os limites de tensão na rede, é um caso particular de problemas de Fluxo em Grafo e variações do clássico Problema do Caixeiro Viajante. Desta forma, pode-se concluir que o problema abordado por este trabalho caracteriza-se como um problema NP-Completo.

1.3 SOLUÇÃO PROPOSTA

A classe NP-Completo têm duas propriedades importantes: (i) Nenhum problema NP-Completo pode ser resolvido por qualquer algoritmo determinístico de tempo polinomial conhecido, e (ii) se há um algoritmo de tempo polinomial para qualquer problema NP-Completo, então existem algoritmos de tempo polinomial para todos os problemas NP-Completos (PAPADIMITRIOU; STEIGLITZ, 1998). As alternativas de soluções para essa classe de problemas são classificadas em técnicas exatas ou aproximadas (PEREZ-GUERRERO *et al.*, 2008).

Algoritmos baseados em técnicas exatas apresentam sempre o melhor resultado possível. Entretanto, não é viável a implementação de tal método para algumas instâncias de problemas com um número elevado de variáveis, devido ao tempo computacional muito elevado. Algoritmos aproximados não necessariamente encontram a melhor solução possível, mas encontram uma solução viável para o conjunto de soluções testadas (KAGAN *et al.*, 2009).

Em função do apresentado, é proposta uma nova metodologia para o problema de *self-healing*. Baseado em técnicas de isolamento de subgrafo e de busca local, partindo de uma solução conhecida, com o objetivo de diminuir a complexidade computacional do problema ao fornecer uma solução viável para o problema sepmre em tempo polinomial.

1.4 OBJETIVOS

1.4.1 OBJETIVO GERAL

Implementar uma estratégia de otimização para o problema de reconfiguração e tratamento de faltas em redes de distribuição de energia elétrica.

1.4.2 OBJETIVOS ESPECÍFICOS

- Implementar uma solução metaheurística para obtenção dos resultados viáveis;
- Implementar uma estratégia de otimização para a melhoria dos resultados obtidos pela metaheurística;
- Realizar experimentação computacional para efeitos de validação e testes.

1.5 JUSTIFICATIVA

A otimização de problemas multiobjetivos é baseada na característica dos problemas reais, no qual se busca satisfazer as restrições do problema de forma que seja a mais conveniente possível. O problema de *self-healing* foi escolhido, pois ele é um problema NP-Completo multiobjetivo, e, uma vez que uma estrutura de solução que cumpre com os objetivos esperados é gerada, essa mesma estrutura pode ser adaptada para outros problemas da mesma ordem.

1.6 ESTRUTURA DO TRABALHO

Este trabalho está organizado da seguinte maneira: no capítulo 2 é introduzido o conceito de *smart grid*, a teoria da complexidade de tempo de algoritmos, além de classificar os tipos de problemas, os métodos de soluções para problemas de otimização e a descrição do estado da arte voltado ao estudo de sistemas *self-healing* para *smart grids*; no capítulo 3 é apresentado a modelagem do problema, o método proposto e a estrutura para o desenvolvimento do procedimento; o capítulo 4 apresenta os parâmetros para a implementação e a estratégia utilizada para o cálculo do fluxo de potência; o capítulo 5 mostra os resultados coletados e descreve o contexto em que o algoritmo foi submetido; e no capítulo 6 é feita a conclusão do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 *SMART GRIDS*

Smart grid, também conhecida como rede inteligente, é uma rede elétrica que pode integrar as ações de todos os usuários conectados à mesma - geradores, consumidores ou ambos - para distribuir eletricidade sustentável, econômica e segura (HARRIS, 2009). Tem por objetivo (i) permitir que a rede seja observável, (ii) fazer com que a rede seja controlável, (iii) melhorar a segurança e desempenho do sistema de energia e (iv) reduzir os custos com operação, manutenção e planejamento do sistema (MOMOH, 2009). As vantagens em se ter uma rede inteligente são (DELGADO-GOMES *et al.*, 2015; HARRIS, 2009):

- aumentar a qualidade e confiabilidade na energia, não somente em operação normal, mas também em interrupções causadas por desastres naturais;
- permitir um papel mais ativo dos consumidores, aumentando suas escolhas como habilitando novos produtos, serviços, e mercados;
- prover aos usuários informação do uso elétrico, permitindo uma implementação de um sistema consciente de energia;
- realizar manutenção preventiva por meio de um sistema de monitoramento de energia contínuo.

A automação da rede de distribuição aumenta a eficiência nas operações do dia-a-dia e melhora o processo de restauração, como responder rapidamente a interrupções de energia, reduzir os erros dos operadores e diminuir a duração de uma interrupção de energia (ZIDAN; EL-SAADANY, 2012). As principais funções da automação na rede de distribuição são detectar e isolar o local da falta e restaurar o serviço (MAMO *et al.*, 2009).

O processo de *self-healing* é descrito como a qualidade do sistema em que possibilita executar medidas corretivas automaticamente e de forma inteligente, quando o sistema for submetido a uma interrupção de energia, para o melhor estado possível, e assim possibilitar a execução de funções básicas sem que as restrições sejam violadas (ZIDAN; EL-SAADANY, 2012). As restrições de uma rede autônoma são similares às de uma rede não autônoma, como (ZIDAN; EL-SAADANY, 2012):

- estrutura de rede radial;

- tensões para os consumidores que devem se manter dentro do limite pré-estabelecido pela ANEEL:

$$V_{min} < V_i < V_{max} \tag{1}$$

onde V_i: tensão no consumidor i, e V_{min}, V_{max}: tensões mínima e máxima aceitáveis pela ANEEL, respectivamente;

- correntes nas linhas devem se manter dentro do limite máximo

$$I_j < I_{max} \tag{2}$$

onde I_j: corrente na linha j, e I_{max}: corrente máxima suportada pela linha.

A estrutura radial é uma rede do tipo acíclica e direcionada, como o exemplo na Figura 1. Os limites de tensão são estabelecidos pela ANEEL, seguindo a Tabela 1. O limite de corrente é determinado pelo tipo de cabo usado pela rede, dependendo de comprimento, bitola e material de cada cabo. A verificação desses limites pode ser realizada por meio de análise de fluxo de potência (STEVENSON, 1982).

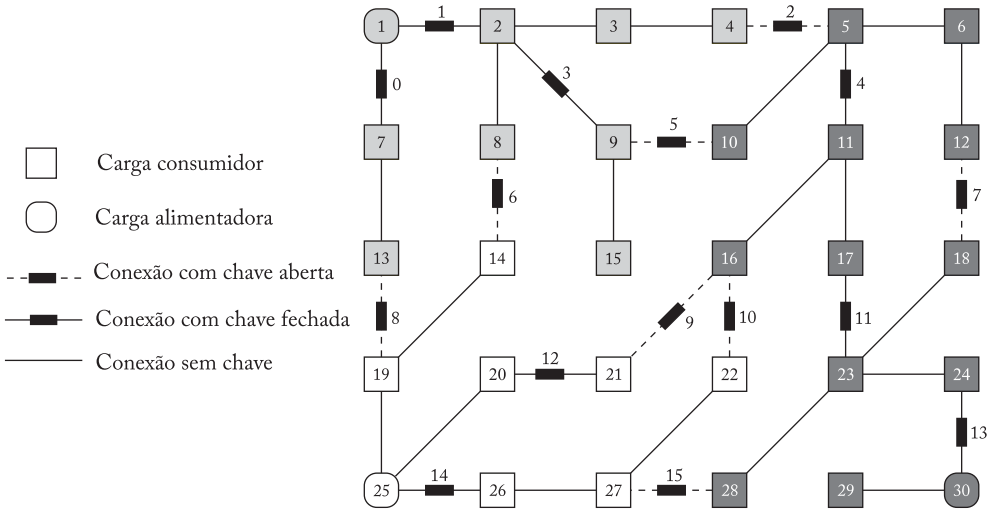

Figura 1: Rede de distribuição radial

Tabela 1: Resolução ANEEL 676/2003: Clientes atendidos com tensão nominal até $1kV$

Classificação da tensão	**Limites de variação da tensão de leitura em relação à tensão nominal em $p.u$ (por unidade)**
	Resolução 676/ANEEL
Adequada	$1,05p.u$ e $0,91p.u$
Precária	$1,10p.u$ e $0,85p.u$
Crítica	Fora das faixas acima

Fonte: Resolução ANEEL 676/2003, 2015

2.1.1 FLUXO DE POTÊNCIA

O estudo do fluxo de potência é de grande importância para planejar e estruturar expansões da rede e ainda melhorar operações do sistema existente. A principal informação obtida a partir desse estudo é a magnitude e o ângulo de fase da tensão de cada barra e o fluxo de potência real e reativo em cada linha (STEVENSON, 1982).

Para cada linha, valores numéricos para a impedância Z_{ij} em série e a admitância total Y_{ij} são necessários para que seja determinado todos os elementos da matriz de admitância das barras. Para cada estudo de fluxo de potência, são dados algumas tensões das barras e o valor da energia aplicada no sistema (STEVENSON, 1982). Essa análise contém restrições com equações e inequações algébricas não lineares, e essas restrições representam as leis de Kirchhoff e operações de limites da rede (ANDERSSON, 2012).

A tensão para uma típica barra do sistema é dada por coordenadas polares, como

$$V_i = |V_i|(cos(\sigma_i) + j \cdot sen(\sigma_i)). \tag{3}$$

A corrente aplicada na rede para a barra i, em termos de Y_{in} é dado pela soma

$$I_i = I_{i1}V_1 + Y_{i2}V_2 + ... + Y_{iN}V_N = \sum_{n=1}^{N} Y_{in}V_n. \tag{4}$$

(STEVENSON, 1982).

Computacionalmente, diferentes métodos de cálculo são apresentados para realizar essas operações, como o método de Gauss-Seidel, o método de Newton-Raphson, entre outros (ANDERSSON, 2012). O método de Gauss-Seidel é um método iterativo no qual se é calculado o conjunto de tensões e correntes a cada iteração, de forma que o valor das tensões e correntes venham a convergir. Essa iteração é repetida até que a variação das tensões calculadas sejam menores que um coeficiente de erro $\epsilon \longrightarrow 0$. O método de Newton-Raphson é realizado a partir de aproximações utilizando expanção em séries de potência para uma função de duas ou mais variáveis (STEVENSON, 1982).

2.2 COMPLEXIDADE DE ALGORITMOS

A complexidade de um algoritmo expressa o esforço computacional necessário para executá-lo, esforço que mede a quantidade de trabalho em termo de tempo ou espaço requerido (TOSCANI; VELOSO, 2009). A complexidade em espaço é medida de acordo com a quantidade de memória utilizada pelo algoritmo, e a complexidade de tempo é medida

de acordo com a quantidade de tempo gasto para a execução. A análise do algoritmo feita pela função de complexidade de tempo pode ser representada por uma função $g(n)$ que indica o maior tempo necessário para o algoritmo resolver uma instância de problema de tamanho n (GAREY; JOHNSON, 1979). É usada a notação O-grande $O(g(n))$ para descrever que $g(n)$ é um limitante superior assintótico do algoritmo (SIPSER, 2007).

Seja a função $f(n)$ a função de complexidade de tempo do algoritmo M_1 para uma entrada de tamanho n. A função $f(n)$ é $O(g(n))$ sempre que existir uma constante c que satisfaça $|f(n)| \leq c * |g(n)|$ para todos os valores de $n \geq 0$ (GAREY; JOHNSON, 1979; SIPSER, 2007). O número de passos que um algoritmo usa sobre uma entrada específica pode depender de vários parâmetros. Por exemplo, se a entrada for um grafo, o número de passos pode depender do número de nós, do número de arestas ou de alguma combinação desses ou de outros fatores. Por simplicidade, o tempo de execução de um algoritmo é computado de acordo com o tamanho da cadeia que representa a entrada sem considerar quaisquer outros parâmetros (SIPSER, 2007; HOPCROFT *et al.*, 2002).

Um problema não é solucionável por apenas um único algoritmo. Existem várias maneiras de resolvê-lo e pode-se buscar resultados diferentes para cada característica, como minimização, maximização ou simplesmente a existência ou não de uma resposta. Os problemas podem ser categorizados em três tipos, de acordo com o aspecto de solução exigida, como, problemas de decisão, problemas de localização e problemas de otimização (GOLDBARG; LUNA, 2000). Problemas de decisão tem por objetivo indicar a existência ou não de uma solução, como resposta *Sim* ou *Não*. Por exemplo, Problema da Existência de Cobertura de Vértices: dado um grafo $G = (V, E)$ e um natural $k \in \mathbb{N}$; existe uma cobertura de vértices $V' \subseteq V$ de G tal que $|V'| \leq k$? Problemas de localização são os problemas que envolvem a procura de um elemento que satisfaça a condição de solução, por exemplo, o Problema de Cobertura de Vértices: dado um grafo $G = (V, E)$ e um natural $k \in \mathbb{N}$; deseja-se a cobertura de vértices $V' \subseteq V$ de G tal que $|V'| \leq k$. Problemas de otimização são aqueles que além de exigirem uma solução, requerem uma qualidade de solução ótima, por exemplo, o Problema da Cobertura Mínima dos Vértices: dado um grafo $G = (V, E)$; deseja-se a cobertura de vértices $V' \subseteq V$ de G tal que não exista uma cobertura de vértices V'' de G de tamanho menor, isto é, com $|V''| < |V'|$ (TOSCANI; VELOSO, 2009). Os problemas ainda podem ser dividido em classes. São elas as classes P, NP, NP-Completo e NP-Difícil (GAREY; JOHNSON, 1979).

A classe P consiste em problemas que podem ser resolvidos por algoritmos determinísticos em tempo polinomial, ou seja, dado o problema, existe um algoritmo determinístico que resolve o problema com uma função de complexidade $O(p(n))$, onde p é

uma função polinomial (TOSCANI; VELOSO, 2009; GAREY; JOHNSON, 1979). Os problemas da classe P são conhecidos como os problemas tratáveis. Exemplos de problemas P são Árvore Geradora Mínima, Problema do Menor Caminho e 2-SAT.

A classe NP é constituída por problemas que podem ser verificados por algoritmos não determinísticos em tempo polinomial, ou seja, dado um problema, existe um algoritmo não determinístico que o verifica com uma função de complexidade $O(g(n))$, onde g é uma função exponencial (TOSCANI; VELOSO, 2009; GAREY; JOHNSON, 1979). Exemplos de problemas NP são Problema de Isomorfismo de Grafos, Problema de Campo Minado e Problema do Ciclo Longo.

A classe NP-Completo consiste em tipos de problemas que estão contidos na classe NP, e se existir um algoritmo que solucione um desses problemas, todos os problemas NP seriam solucionáveis em tempo polinomial (GAREY; JOHNSON, 1979). Ou ainda, pode-se dizer que um problema NP-Completo P satisfaz duas condições: (i) P está em NP; e (ii) todo A em NP pode ser redutível em tempo polinomial a P (SIPSER, 2007). Problemas dessa classe são conhecidos como problemas intratáveis. Exemplos de problemas NP-Completo são Problema do Caixeiro Viajante, Circuito Hamiltoniano e Coloração de Grafos.

A classe NP-Difícil é a classe dos problemas que são polinomialmente redutíveis de problemas NP-Completo, entretanto não é conhecido algum algoritmo não determinístico de tempo polinomial que os resolva, portanto, não é determinado se são da classe NP ou não (TOSCANI; VELOSO, 2009). Exemplos de problemas NP-Difícil são Criptografia, Mineração de Dados e Problema do Quadro de Horários.

O universo de problemas é retratado pela Figura 2. A tabela 2 demonstra a diferença entre as taxas de crescimento para típicas funções de tempo das classes de problemas polinomiais (n, n^2, n^3 e n^5) e exponenciais (2^n e 3^n), em relação ao tamanho (n) da entrada do problema.

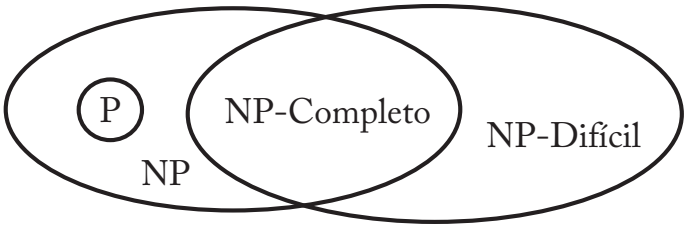

Figura 2: Classes de Problemas
Adaptado de Toscani e Veloso (2009), p.245.

Tabela 2: Comparação de funções de tempo polinomiais e exponenciais.

	n					
$\boldsymbol{f(n)}$	10	20	30	40	50	60
$\boldsymbol{n}$	$0,00001s$	$0,00002s$	$0,00003s$	$0,00004s$	$0,00005s$	$0,00006s$
$\boldsymbol{n^2}$	$0,0001s$	$0,0004s$	$0,0009s$	$0,0016s$	$0,0025s$	$0,0036s$
$\boldsymbol{n^3}$	$0,001s$	$0,008s$	$0,027s$	$0,064s$	$0,125s$	$0,216s$
$\boldsymbol{n^5}$	$0,1s$	$3,2s$	$24,3s$	$1,7min$	$5,2min$	$13min$
$\boldsymbol{2^n}$	$0,001s$	$1s$	$17,9min$	$12,7dias$	$35,7anos$	$366sec$
$\boldsymbol{3^n}$	$0,059s$	$58min$	$6,5anos$	$3855sec$	$2 \cdot 10^8 sec$	$1,3 \cdot 10^{13} sec$

Fonte: Garey e Johnson (1979), p. 7

2.3 MÉTODOS DE SOLUÇÕES PARA PROBLEMAS NP-COMPLETO

Problemas de otimização são naturalmente divididos em duas categorias: aqueles com variáveis contínuas, e aqueles com variáveis discretas, também conhecidos como combinatórios. Nos problemas contínuos, geralmente é procurado por um conjunto de número reais ou até mesmo uma função; nos problemas combinatórios é procurado por um objeto de um conjunto finito ou incontável, como um inteiro, conjunto, permutação ou grafo (GAREY; JOHNSON, 1979; GOLDBARG; LUNA, 2000).

Encontrar uma solução 'ótima' global para uma instância de alguns problemas pode ser demasiadamente difícil, mas sempre é possível encontrar uma solução que seja 'ótima' naquela vizinhança, conhecida como uma solução 'boa' (GAREY; JOHNSON, 1979). Existem vários algoritmos clássicos que são projetados para fazerem buscas por soluções ótimas. Entretanto, nenhum desses algoritmos é robusto pois sempre que o problema é mudado é necessário mudar o algoritmo (MICHALEWICZ; FOGEL, 2013).

Os algoritmos de busca são classificados de acordo com a sua estratégia para encontrar uma solução desejada como, algoritmos de busca sistemática e de busca local.

Por exemplo, o problema da mochila baseia-se em um problema em que se tem um conjunto de itens com diferentes valores e pesos e uma mochila com um limite de peso, e que se busca colocar na mochila objetos desse conjunto de forma que a soma dos valores desses objetos seja a maior possível (CALIFORNIA; KARP, 1972). Seja um problema da mochila com 3 itens e as soluções representadas por vetores de 3 posições onde cada posição representa a ausência (0) ou não (1) do item n na solução. Um algoritmo de busca sistemática irá explorar todas as soluções possíveis a fim de encontrar a solução que tenha o maior custo benefício. Como pode ser visto na Figura 3, o algoritmo gerou todas as possibilidades de combinações para encontrar o resultado melhor (NORVIG; RUSSELL, 2015).

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

Figura 3: Soluções geradas

Um algoritmo de busca local irá explorar somente seus vizinhos, ou seja, a cada etapa será analisado somente os vizinhos de uma solução, como pode ser visto na Figura 4. Seus vizinhos podem ser definidos como a troca de estado de uma variável, e essas combinações são definidas como vizinhança. Esses algoritmos são apropriados para problemas nos quais o custo do caminho para se chegar a solução é irrelevante, e o mais importante é a solução final (NORVIG; RUSSELL, 2015).

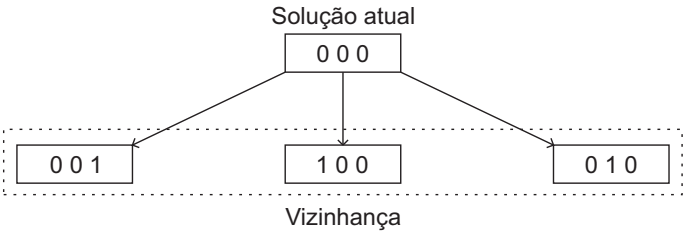

Figura 4: Vizinhança

De acordo com a complexidade do problema, o problema pode ser resolvido por um método exato ou um método aproximado. Para problemas da classe NP-Completo, algoritmos exatos são inviáveis em alguns casos devido ao tempo computacional tomado, tendo-se então a necessidade de utilizar os métodos aproximados. Métodos aproximados geram soluções em um tempo razoável para uso prático, mas não há garantia de encontrar uma solução ótima global (TALBI, 2009; CORMEN *et al.*, 2002).

Nos métodos exatos pode ser encontrado algoritmos como busca exaustiva, *branch-and-bound* e programação dinâmica. Esses métodos de busca sistemática garantem a 'otimalidade' da solução. Entretanto, só podem ser utilizados em algoritmos de tamanho limitado devido a complexidade dos problemas. Esses métodos enumerativos podem ser vistos como algoritmos de busca em árvore (GAREY; JOHNSON, 1979; TALBI, 2009).

Métodos aproximados são utilizados quando não se necessita de uma solução ótima para o problema ou quando essa solução é inviável de ser encontrada em tempo hábil (SIPSER, 2007). Esses métodos podem ser divididos em dois grupos: algoritmos de aproximação e heurísticas (TALBI, 2009; SIPSER, 2007). Os algoritmos de aproximação garantem a 'otimalidade' da solução obtida em relação à solução ótima.

Formalmente, um algoritmo tem uma fator de aproximação de ϵ se sua complexidade de tempo é polinomial e para qualquer instância de entrada gera uma solução α

que

$$\begin{aligned} \alpha \leq \epsilon \cdot s, \text{ se } \epsilon > 1 \\ \epsilon \cdot s \leq \alpha, \text{ se } \epsilon < 1 \end{aligned} \tag{5}$$

onde s é a solução ótima, e o fator ϵ define a garantia relativa de performance. O fator ϵ pode ser uma constante ou uma função do tamanho da instância de entrada (TALBI, 2009).

Problemas NP-Difíceis diferem nessa aproximação. Esses problemas são impossíveis de aproximar independente do fator ϵ. A família de problemas de aproximação são da classe P, onde o problema pode ser aproximado por qualquer fator maior que 1 (TALBI, 2009).

O objetivo principal de algoritmos de aproximação para o problema é encontrar soluções ligeiramente inferiores às soluções ótimas. Esses algoritmos são específicos para cada problema de otimização, o que limita sua aplicabilidade (GAREY; JOHNSON, 1979; TALBI, 2009).

Métodos heurísticos são técnicas de busca local inspiradas em processos intuitivos que procuram uma boa solução a um custo computacional aceitável. Além disso, permitem trabalhar com instâncias de problemas de tamanhos maiores, mas não existe garantia de encontrar soluções ótimas ou quão próximo está de uma solução ótima (TALBI, 2009). Normalmente, uma heurística é empregada em duas situações (LUGER, 2013):

- um problema pode não ter solução exata por causa de ambiguidades inerentes na formulação do problema ou nos dados disponíveis. Por exemplo, o diagnóstico médico. Determinado conjunto de sintomas pode ter várias causas possíveis; os médicos usam heurísticas para escolher o diagnóstico mais provável e para formular um plano de tratamento;

- um problema pode ter uma solução exata, mas o custo computacional de encontrá-la pode ser proibitivo. Por exemplo, o jogo de xadrez. O crescimento do espaço de estados é combinatoriamente explosivo, com o número de estados possíveis aumentando exponencialmente, ou fatorialmente, com a profundidade da busca.

Uma busca heurística bem conhecida é a Subida de Encosta. A Subida de Encosta é o exemplo de heurística mais simples. As estratégias de subida de encosta expandem o estado atual da busca e avaliam os seus filhos. O melhor filho é selecionado para uma

expansão futura. Essa busca pode ser usada por um alpinista impetuoso, mas cego: subir pelo caminho mais íngreme possível até não poder mais avançar. O algoritmo não armazena o histórico da subida, logo não é possível recuperar de falhas da estratégia (LUGER, 2013). Esses algoritmos são desenvolvidos para serem flexíveis, para que possam ser aplicados em diversos problemas. Esta característica viabilizou a construção das chamadas melhores estratégias, comumente conhecidas como metaheurísticas (SOUZA, 2009).

As metaheuríticas são procedimentos destinados a encontrar uma solução boa, eventualmente ótima, consistindo na aplicação de uma heurística subordinada, a qual tem que ser modelada para cada problema (SOUZA, 2009). As metaheurísticas diferem das heuríticas em que elas usam estratégias para tentar escapar de ótimos locais para buscar ótimos globais (BLUM; ROLI, 2003). Ainda, existem dois tipos de metaheurísticas, as construtivas e as de refinamento. Algumas metaheurísticas clássicas são GRASP, *Simulated Annealing* e Busca Tabu.

2.3.1 GRASP

O GRASP (*Greedy Randomized Adaptive Search Procedure* - Processo Adaptativo e Aleatório de Busca Gulosa) é uma metaheurística de construção e consiste em um processo iterativo, no qual cada iteração realiza duas fases, a fase de construção e a fase de busca local. A melhor solução encontrada é mantida como solução. Um pseudocódigo genérico é mostrado no Algoritmo 1 (RESENDE; FEO, 1995; GLOVER; KOCHENBERGER, 2006).

Algoritmo 1: GRASP

```
   Entrada: Instância de entrada
   Saída: Melhor solução encontrada
1  início
2      repita
3          Solucao = ConstruirSolucaoGRASP(InstanciaEntrada);
4          BuscaLocal(Solucao);
5          AtualizaSolucao(Solucao, MelhorSolucao);
6      até Critério de parada não satisfeito;
7      retorna MelhorSolucao
8  fim
```

Na fase de construção uma possível solução é iterativamente construída elemento a elemento. Em cada iteração de construção, a escolha do próximo elemento para ser adicionado é determinado ordenando todos elementos na lista de candidatos de acordo com uma função gulosa. Essa função mede o benefício de selecionar cada elemento. Essa heurística é adaptativa pelo fato de que os benefícios associados a cada elemento é atualizado

a cada iteração da fase de construção. Um componente probabilístico é caracterizado pela escolha aleatória do melhor candidato da lista RCL (*Restricted Candidate List* - Lista de Candidatos Restritiva), mas não necessariamente o candidato do topo. Essa técnica de escolha permite diferentes soluções serem obtidas para cada iteração, mas não necessariamente compromete o poder do elemento adaptativo do método. O Algoritmo 2 mostra o pseudocódigo para a fase de construção (RESENDE; FEO, 1995).

Algoritmo 2: Construtor de Solução GRASP

```
Entrada: Instância de entrada
Saída: Solução criada
1  início
2      Solucao = ∅;
3      para Construção não finalizada faça
4          AtualizaRCL(RCL);
5          s = SelecionaElementoAleatorio(RCL);
6          Solucao = Solucao ∪ {s};
7          AdaptaFuncaoGulosa(s);
8      fim
9      retorna Solucao
10 fim
```

Como o caso de vários métodos heurísticos, as soluções geradas por um algoritmo GRASP não há a garantia de serem ótimos locais considerando as simples vizinhanças. Devido a isso, quase sempre é melhor aplicar uma busca local no intuito de melhorar cada solução construída. Um algoritmo de busca local trabalha de forma iterativa substituindo sucessivamente a solução atual por uma solução melhor em sua vizinhança. Esse processo termina quando nenhuma solução melhor é encontrada na vizinhança. A estrutura de vizinhança N para um problema P relaciona uma solução s do problema com um subconjunto de soluções $N(s)$. A solução s é reconhecida como ótima local se não existir nenhuma solução melhor em $N(s)$. O pseudocódigo para a busca local do GRASP é representada no algoritmo 3 (TALBI, 2009; RESENDE; FEO, 1995).

Algoritmo 3: Busca Local GRASP

```
Entrada: Estrutura de Vizinhança, Solução
Saída: Solução ótima local
1  início
2      repita
3          Encontrar a melhor solução t ∈ N(s);
4          s = t;
5      até s for ótimo local;
6      retorna s
7  fim
```

O ponto chave para o sucesso do algoritmo de busca local consiste na escolha adequada da estrutura de vizinhança, técnicas de busca eficientes, e a solução inicial (RESENDE; FEO, 1995).

2.3.2 *SIMULATED ANNEALING*

Simulated annealing (SA) é uma técnica baseada nos princípios de mecanismos estatísticos que o processo de recozimento requer de aquecer e lentamente resfriar uma substância até que não ocorra mais mudanças. A cada temperatura, o sistema deve prosseguir o tempo necessário para o sistema atingir um estado rígido. E a rigidez da estrutura depende da taxa de resfriamento dos metais (KIRKPATRICK C. D. GELATT, 1983; TALBI, 2009).

Esta técnica inicia a partir de uma solução inicial qualquer, e o procedimento principal consiste em um *loop* que gera aleatoriamente, em cada iteração, um único vizinho s' da solução corrente s (SOUZA, 2009). Considerando um problema de maximização, seja $\Delta = f(s) - f(s')$ com $f(s)$ a função objetivo do problema dado uma solução s. O método aceita o movimento e a solução vizinha para ser a nova solução corrente se $\Delta < 0$. Caso $\Delta \geq 0$ a solução vizinha candidata também poderá ser aceita com uma probabilidade $e^{-\Delta/T}$, onde T é a *temperatura* e Δ é o fator que regula a probabilidade de se aceitar soluções de pior custo (GLOVER; KOCHENBERGER, 2006; SOUZA, 2009). O algoritmo é ilustrado em pseudocódigo pelo algoritmo 4.

A temperatura T assume um valor elevado inicialmente, e é decrementada por uma razão de resfriamento α, tal que $T_k \leftarrow \alpha \cdot T_{k-1}$, com $0 < \alpha < 1$. Com isso, no início tem-se uma chance maior para escapar de máximos locais e, à medida que T se aproxima de zero, o algoritmo se comporta como o método de descida, visto que a probabilidade de se aceitar movimentos de piora diminui ($T \to 0 \Rightarrow e^{-\Delta/T} \to 0$) (SOUZA, 2009). Na primeira parte da busca, a tendência de melhoria é baixa e permite a exploração do espaço de busca. O componente de erro é decrementado vagarosamente fazendo que a busca encontre um máximo local (ou até mesmo global) (BLUM; ROLI, 2003). Na tabela 3 é possível observar algumas probabilidades, dado um $\Delta = 13$, de acordo com a temperatura T.

2.3.3 BUSCA TABU

A Busca Tabu é um procedimento adaptativo auxiliar que guia um algoritmo de busca local na exploração contínua dentro de um espaço de busca originada dos trabalhos de Glover e Kochenberger (2006) (SOUZA, 2009). A palavra *tabu* vem do *Tongan* que

Algoritmo 4: Simulated Annealing

Entrada: $f(.)$, $N(.)$, α, $SAmax$, T_0, s
Saída: Melhor solução encontrada

```
1  início
2      s* = s;
3      Iter = 0;
4      T = T0;
5      para T > 0 faça
6          para Iter < SAmax faça
7              Iter = Iter + 1;
8              Gere um vizinho qualquer s' ∈ N(s);
9              Δ = f(s) − f(s');
10             se Δ < 0 então
11                 s = s';
12                 se f(s') < f(s*) então
13                     s* ← s';
14                 fim
15             senão
16                 Tome x ∈ [0, 1];
17                 se x < e^(−Δ/T) então
18                     s ← s';
19                 fim
20             fim
21         fim
22         T = α · T;
23         Iter = 0;
24     fim
25     s = s*;
26     retorna s;
27 fim
```

Tabela 3: Probabilidade p de aceitação em função da temperatura T

T	$e^{-13/T}$	p
1	0.000002	1.00
5	0.0743	0.93
10	0.2725	0.78
20	0.52	0.66
50	0.77	0.56
10^{10}	0.999...	0.5...

Fonte: Michalewicz e Fogel (2013), p.119

significa algo que não pode ser tocado porque é sagrado (TAO *et al.*, 1991). O nome deste método vem das *Listas Tabu*, que consistem em listas com soluções (ou movimentos) não permitidas. Essas listas podem conter movimentos proibidos devido a alguma restrição do sistema ou até mesmo soluções já visitadas para se evitar ciclos e escapar de ótimos locais.

Esta técnica inicia a partir de uma solução inicial qualquer, e o procedimento principal consiste em um *loop* que gera aleatoriamente, em cada iteração, um único vizinho s' da solução corrente s, tal que, s' não seja uma solução contida na Lista Tabu (SOUZA, 2009).

Considerando um problema de maximização, seja $f(s)$ a função objetivo do problema dado uma solução s. O método sempre aceita o movimento para o melhor vizinho (s') de s, tal que s' não esteja na Lista Tabu (GLOVER; KOCHENBERGER, 2006). O algoritmo é ilustrado em pseudocódigo pelo algoritmo 5.

Algoritmo 5: Busca Tabu

Entrada: $f(.)$, $N(.)$, $|V|$, $|LT|$, BT_{max}, s
Saída: Melhor solução encontrada

```
1  início
2      s* = s;
3      Iter = 0;
4      MelhorIter = 0;
5      LT = ⊘;
6      para Iter − MelhorIter < BT_max faça
7          Iter = Iter + 1;
8          Seja s' = s ⊕ m o melhor elemento de V ⊆ N(s) tal que o
            movimento m não seja tabu (m ∉ LT);
9          Atualizar a lista tabu LT;
10         s = s';
11         se f(s) > f(s*) então
12             s* = s;
13             MelhorIter = Iter;
14         fim
15     fim
16     s = s*;
17     retorna s;
18 fim
```

A Lista Tabu (LT) é iniciada vazia e, ao decorrer do algoritmo, vai armazenando os movimentos realizados já visitadas. Com isso, o algoritmo vai criando sua Lista Tabu e bloqueando movimentos realizados para que o algoritmo busque soluções que ainda não foram exploradas (SOUZA, 2009). A Lista Tabu também pode ser implementada juntamente a outras metaheurísticas para que seja evitado possíveis ciclos e/ou até mesmo para expandir o espaço de busca do problema (GLOVER; KOCHENBERGER, 2006).

2.4 RESTAURAÇÃO DO SISTEMA DE DISTRIBUIÇÃO

A maioria dos sistemas de distribuição não são completamente automatizados. Logo as ações corretivas que são requeridas para a localização e isolação da falta e restauração do serviço são executadas manualmente pelos operadores humanos. Esse processo requer mais tempo que o processo de *self-healing*, no qual o sistema detecta a falta e reconfigura a rede automaticamente. Essa estratégia visa a minimização da carga de trabalho de operadores de campo, provê uma restauração imediata para os consumidores e aumenta a confiabilidade e robustez da rede de distribuição. Abordagens diferentes do problema foram realizadas, utilizando programação linear, programação dinâmica, heurísticas, sistemas inteligentes e redes neurais (ZIDAN; EL-SAADANY, 2012).

Utilizando programação dinâmica, Perez-Guerrero *et al.* (2008) fez uma alternativa de redução de estados. O algoritmo é como um operador permissivo após uma falta no sistema. Faz o uso de um algoritmo dinâmico com o objetivo de reduzir o número de estados agrupando-os com os estados aos seus redores e selecionando o melhor para representá-los. O método é melhor aplicável para sistemas configurados radialmente.

Ferreira *et al.* (2013) define o espaço de busca e o número de variáveis de decisão a partir do local onde a falta ocorreu utilizando o algoritmo genético. Para isso, foi tomado como premissa que o sistema de proteção presente na rede está devidamente coordenado, ou seja, quando uma falta em trecho de consumidor ocorre, tem-se como um estado de pós falta o acionamento da proteção nos equipamentos entre essa falta e a subestação. Outra premissa é a exclusão das chaves ligada diretamente ao trecho com falta do campo de busca no processo de otimização. Kumar *et al.* (2008) apresenta uma variação do algoritmo genético como NSGA-II (*nondominated sorting genetic algorithm-II*), que, diferente do algoritmo genético convencional que converte o problema multiobjetivo para um problema com apenas um objetivo atribuindo pesos para cada uma das funções multiobjetivos, o NSGA-II trata todas as funções juntas em conjunto mantendo a natureza do problema como multiobjetivo.

Foram realizados também trabalhos utilizando lógica *fuzzy*, como o de Das (2006). Das (2006) realizou uma modelagem multiobjetivo da rede de modo a considerar o balanceamento de carga entre os fornecedores de energia, minimizar a variação da tensão nas linhas, minimizar a perda de potência e restringir os valores de corrente nas linhas, além de manter a rede radial e alimentar o maior número de nós possível. Utilizando lógica *fuzzy*, os estados foram classificados entre estados "saudáveis" e estados "não saudáveis" para prover um valor antecipado de cada objetivo.

A partir de heurísticas, McDermott *et al.* (1999) desenvolveu um método heurístico não-linear construtivo para esse problema, derivado de técnicas gulosas. Esse algoritmo toma uma quantidade de iteração maior que outros métodos, mas as permite um modelo mais acurado das restrições e ações de controle. Carpaneto e Chicco (2008) aplica os conceitos de otimização da Colônia de Formigas para minimizar as perdas do sistema de distribuição, fazendo com que o algoritmo inicie a partir da configuração atual e prossegue introduzindo progressivamente variações na configuração baseadas de acordo com suas regras heurísticas. Souza *et al.* (2016) aplicou o sistema inteligente *Opt-aiNet* (*Optimal Artificial Immune Network*) utilizando níveis de demanda variáveis, com o objetivo de minimizar o custo de perdas de energia durante o período de planejamento.

Uma estrutura de controle cooperativo multiagente foi proposto por Zidan e El-Saadany (2012). Os agentes de controle possuem a habilidade inteligente de comunicação e negociação para determinar estados atuais e previstos do sistema e então operar os atuadores e chaves na maneira em que encontrará a melhor forma de atingir seus objetivos satisfazendo suas restrições.

Esses são apenas alguns trabalhos que aplicam técnicas heurísticas e metaheurísticas em problemas de reconfiguração de redes de distribuição. Acredita-se que o estudo de diferentes metaheurísticas aliada com um mecanismo que faça uma redução na vizinhança de uma solução para o problema pode gerar um sistema de *self-healing* capaz de encontrar soluções de melhor qualidade em um menor tempo.

3 MÉTODO PROPOSTO

O problema de *self-healing* pode ser representado em forma de um grafo $G = (V, A)$, onde V é o conjunto de vértices (barras consumidoras/alimentadoras) e A é o conjunto de arestas (chaves/linhas de distribuição), como mostra a Figura 5.

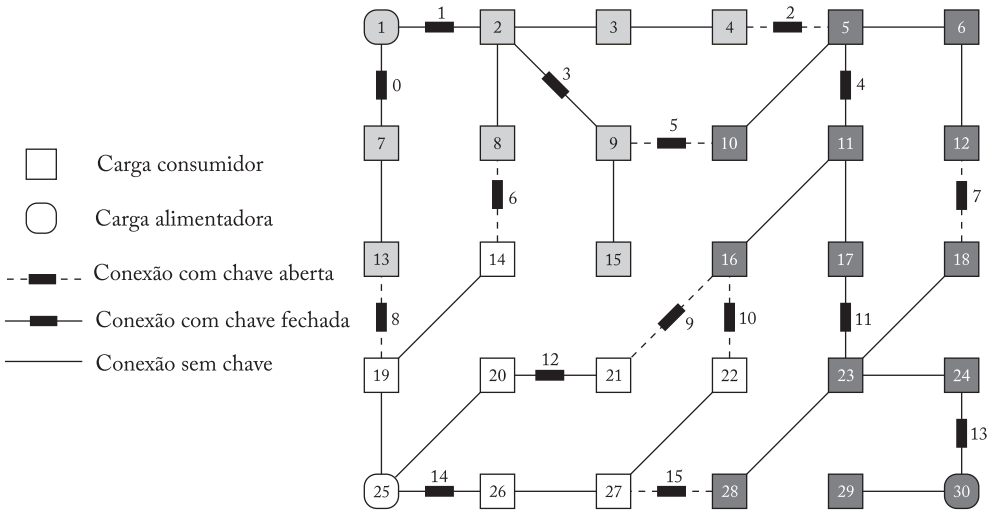

Figura 5: Rede de distribuição radial

A partir de cada alimentador, a estrutura de cada subgrafo é como uma árvore, devido a sua estrutura radial (uma árvore é definida por ser um grafo acíclico e não conexo (CORMEN *et al.*, 2002)). Com isso, um sistema completo pode ser definido como floresta (DEO, 2016), ou seja, um conjunto de árvores. Para cada árvore, sua raiz será uma barra alimentadora interligada às barras consumidoras. Na Figura 5 cada cor identifica as diferentes árvores que compõem o sistema.

Os vértices contêm informações como prioridade, a potência exigida, indicador de subestação e tensão atual. A prioridade é utilizada para que as barras tenham preferência sobre as outras para serem energizadas. A tensão atual é calculada com o fluxo de potência e é utilizado como referência para a verificação da restrição de tensão e corrente. As arestas contêm informações como resistência, reatância, limite de corrente, indicador de chave e corrente atual. A resistência e reatância são usadas no cálculo do fluxo de potência e são necessários para se calcular perdas referentes à rede de distribuição. A corrente atual é calculada pelo fluxo de potência e é usada como referência para a restrição do limite de corrente. Contudo, os dados de uma árvore podem ser representados em forma de tabela como mostra a Tabela 4.

As variáveis a serem tratadas pelo algoritmo de restauração serão as chaves do sistema. Elas serão representadas como um vetor binário de tamanho n para um sistema que contém n chaves. Esse vetor conterá 1 quando a chave estiver fechada e 0 quando

Tabela 4: Dados da rede da Figura 1

Nó i	**Nó j**	**ID da chave**	**Estado da chave**	**Resistência (Ω)**	**Reatância (Ω)**	**Potencia em j (kW)**
1	2	1	FECHADA	1,3	0,5	23,5
1	7	0	FECHADA	0,9	0,35	21,5
2	3	-	-	1,15	0,4	15,2
2	8	-	-	0,4	0,1	23,5
2	9	3	FECHADA	1,8	0,54	15,2
3	4	-	-	1,2	0,61	19,7
4	5	2	ABERTA	1	0,54	32,5
7	13	-	-	0,8	0,35	15,7
9	15	-	-	0,72	0,3	23,3

aberta. Na Tabela 5 mostra como será disposto o vetor para uma rede configurada como mostra a Figura 5.

Tabela 5: Vetor de chaves

ID Chave	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Estado	1	1	0	1	1	0	0	0	0	0	0	1	1	1	1	0

3.1 FORMULAÇÃO DO PROBLEMA

O problema de restauração de serviço pode ser visto como um problema de otimização multiobjetivo e multirrestritivo. Esse problema é considerado ser de grande escala devido à grande combinação de possíveis operações com as chaves que aumentam de acordo com a dimensão da rede. As funções objetivo foram propostas de acordo com o seguinte (ZIDAN; EL-SAADANY, 2012; PEREZ-GUERRERO *et al.*, 2008):

- maximização do número de cargas restauradas considerando suas prioridades

$$Tot_{energ} = Max \sum_{i=1}^{N_{barras}} w_i * P_i * y_i \tag{1}$$

onde $Total_{energizada}$ é a quantidade de barras energizadas; N_{barras} é o número de barras energizadas após a restauração do serviço; P_i é a prioridade da barra i; y_i é o estado da carga na barra i (por exemplo, 1 quando a barra estiver energizada e 0 caso contrário); w_i é a prioridade ou nível de importância da barra i.

- minimização do número de operações realizadas nas chaves, a fim de minimizar o tempo e o custo operacional da restauração

$$Tot_{chave} = Min \sum_{i=1}^{N_s} |x_i - x_{io}| \tag{2}$$

onde $Total_{chave}$ é o quantidade de chaves que trocaram de estado; N_s é o número total de chaves; x_i é o estado da chave i após a rede ser restaurada (1 para chave fechada e o 0 para chave aberta); x_{io} é o estado da chave i após o isolamento da falta.

- maximização das tensões das barras a fim de minimizar a quantidade de barras com tensões baixas ou até abaixo da faixa operacional através do coeficiente de tensões

$$Coef_V = Max \sum_{i=1}^{N_{barras}} (V_i - Lim_{inf}) \tag{3}$$

onde Lim_{tensao} é a variação da tensão mínima do sistema (Min_{tensao}) em relação ao limite inferior de tensão (Lim_{inf}) estipulado pela ANEEL; com isso, valores de Lim_{tensao} negativos representam valores abaixo da faixa operacional e, assim, os valores positivos representam valores dentro da faixa de operação.

- minimização de perda de potências real do sistema

$$Potência_{real} = Min \sum_{i=1}^{N_{linhas}} I_i^2 * R_i \tag{4}$$

onde $Potência_{real}$ é a perda total de potência real do sistema expressa em kW; N_{linhas} é a quantidade de linhas e chaves do sistema; I_i é a corrente na linha i; e R_i é a resistência da linha i.

A prioridade fará com que a barra tenha uma probabilidade maior de ser energizada, de acordo com sua importância. As barras com menor importância terão prioridade igual a 1 e barras com prioridades maiores terão prioridades de valores inteiros maiores. Exemplos típicos de consumidores com prioridades diferenciadas são hospitais, aeroportos e departamentos policiais.

A partir desses objetivos, uma função objetivo foi elaborada a fim de relacionar todos os índices, aplicando-nos pesos para cada um de forma que para cada solução diferente seja pontuada de acordo com sua configuração.

$$FunObj = (1000 * Tot_{energ}/N_{barras}) + (Coef_V * 100) - (Potência_{real}(MW)) - Tot_{chave} * 10 \tag{5}$$

A função objetivo representada pela Função 5 representa a relação entre os índices listados para que seja gerado um *fitness* para cada solução encontrada. O coeficiente de maior peso é a relação de quantidade de barras energizadas pela quantidade de barras totais da rede. O $Coef_V$ demonstra quão penalizado os será a solução, quanto mais nós abaixo da tensão mínima, maior a penalização. A penalização pela potência real dissipada será na escala de MW. E, por fim, um décimo da quantidade de manobras de chaves realizadas pelo algoritmo. Esses valores foram calculados e extraídos a partir de suas unidades de medidas e importância do peso no resultado, tendo como ordem de importância, do maior para o menor, número de barras energizadas, quantidade de chaves manobradas, limite de tensão e perdas.

3.2 *SELF-HEALING*

Self-healing é o mecanismo que realiza a localização e isolação da falta e a restauração do serviço.

A localização e isolação do sistema é feita a partir de um algoritmo de inundação para que toda a área atingida pela falta seja isolada. O algoritmo de inundação inicia pelo local onde ocorreu a falta e percorre por todas as linhas conectadas à falta até encontrar uma chave. Ao encontrar uma chave faz com que essa chave se abra e marca-a com uma *flag* para que a mesma não seja fechada durante o processo de restauração e, assim, repete esse procedimento até que a área em falta seja isolada completamente. A Figura 6 retrata a situação de falta ocorrida no sistema, na qual a falta detectada é isolada, mostrando quais as chaves que serão abertas e bloqueadas nessa etapa.

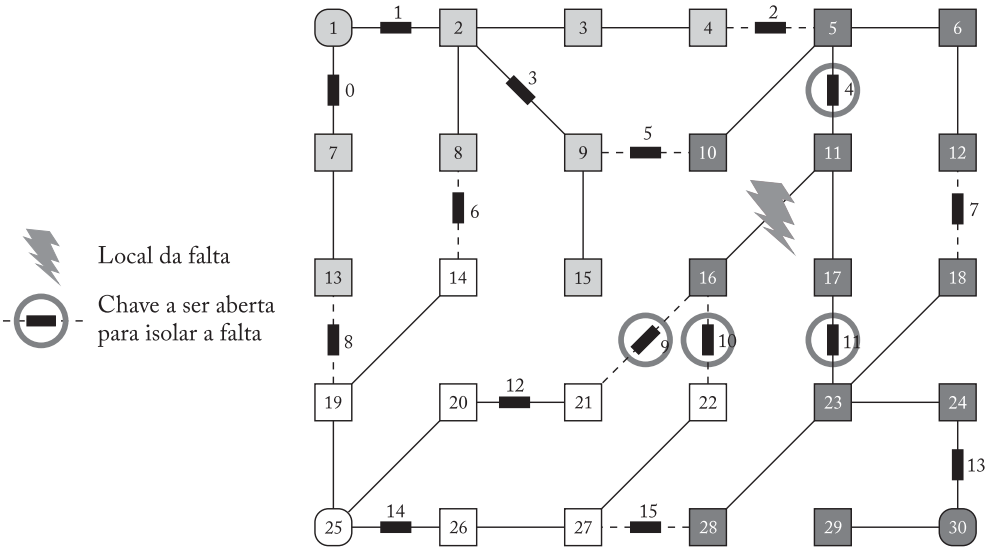

Figura 6: Isolação de uma falta

Além de isolar a falta, poderá ocorrer casos em que haverá áreas fora de serviço.

Há a necessidade de reenergizar essas áreas para que o impacto causado pela falta seja diminuído. Após finalizar essa etapa, o mecanismo inicia o processo de restauração.

O processo de restauração compreende em duas etapas: preparação e restauração do serviço. Os objetivos são o que diferem esses estágios, assim como as circunstâncias em que cada um é submetido (FINK *et al.*, 1995).

3.2.1 PREPARAÇÃO

Nessa etapa é avaliado o estado do sistema após o distúrbio e uma estratégia de restauração é traçada para que os objetivos sejam alcançados respeitando as restrições.

As chaves serão niveladas a partir da área fora de serviço, com exceção da área onde ocorreu a falta. As chaves em que se encontram na área fora de serviço e os *tie-switch* que isolam essa área serão de nível 1, as próximas chaves serão de nível 2 e assim sucessivamente. As chaves com um nível $n \leq N$ é sinalizada como desbloqueada, e as demais são sinalizadas como bloqueadas, onde $n \in \mathbb{N}$ representa o nível de uma chave e $N \in \mathbb{N}$ representa o nível máximo a ser desbloqueado. Quando o algoritmo heurístico de restauração for executado, ele poderá apenas fazer operações com as chaves desbloqueadas.

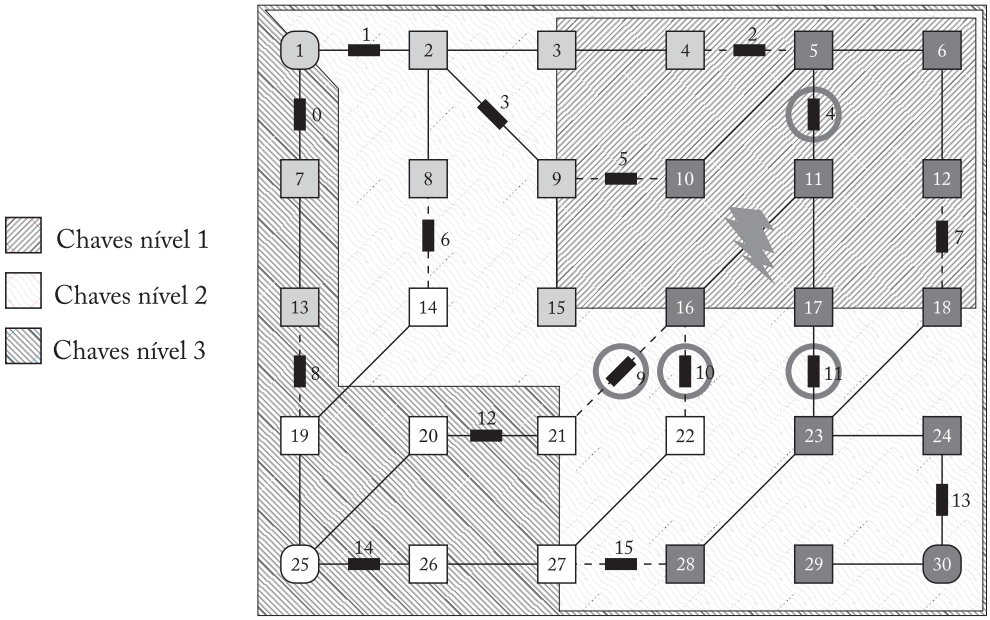

Figura 7: Determinação dos níveis das chaves

3.2.2 RESTAURAÇÃO DO SERVIÇO

Logo após a seção em onde a falta ocorreu for isolada e a preparação do sistema estiver finalizada, é necessário que as áreas sem serviço sejam restauradas. O algoritmo de restauração é aplicado para decidir quais chaves serão operadas de modo com que a

rede continue respeitando suas restrições e que as funções objetivas sejam satisfeitas da melhor maneira possível.

O algoritmo trabalhará com as chaves desbloqueadas em forma de um vetor binário, o qual irá transportar o estado do vetor para as chaves do sistema e realizar o estudo de fluxo de potência para validar essa solução.

4 IMPLEMENTAÇÃO

Para a implementação e desenvolvimento do modelo proposto foi utilizada a linguagem C++, o que proporciona um bom desempenho e a facilidade de integração com mecanismos que proporcionam formas de apresentação dos resultados de maior contraste. Os algoritmos implementados foram as metaheurísticas GRASP, *Simulated Annealing* e a utilização da estratégia da Busca Tabu nesses dois métodos.

4.1 ENTRADA DE DADOS

Nessa etapa é feita a alimentação do sistema com os dados básicos do programa, como

- dados da barras:
  - identificação da barra;
  - indicação se é uma subestação;
  - prioridade;
  - potência ativa e reativa requerida pela barra;
- dados das linhas:
  - barra inicial e final da linha;
  - existência de chave na linha e seu estado (aberta ou fechada);
  - limite de corrente na linha;
  - impedância da linha.

Com isso é possível a construção da matriz de adjacência, um vetor com as barras e um vetor de chaves. Os índices da matriz representam as respectivas barras do sistema.

Considerando os algoritmos implementados, o *Simulated Annealing* requer alguns dados iniciais como taxa de resfriamento (α), temperatura inicial (T_0), número máximo de iterações ($SAmax$) e uma solução inicial válida. Esses parâmetros estabelecidos, apresentados adiante, foram adquiridos após repetidas execuções de forma que o algoritmo obtivesse uma boa razão entre tempo e qualidade da resposta.

O algoritmo GRASP requer uma estratégia para a atualização da lista restrita de candidatos (RCL) em sua fase de construção. Essa estratégia baseia-se em ter, como candidatos dessa lista, as chaves que estejam abertas e com apenas uma das barras energizadas. Assim, a fase de construção irá partir das barras energizadas fazendo com que

as próximas etapas energizem as demais barras sem energia. A cada chave adicionada a solução como fechada, o fluxo de potência é executado para que a RCL seja atualizada.

4.1.1 FLUXO DE POTÊNCIA

A função do fluxo de potência foi implementada a partir do método iterativo de Gauss-Seidel e com o apoio das leis de Kirchhoff, também conhecido como soma das correntes. Esse método é uma estratégia de encontrar as tensões e correntes de forma iterativa de acordo, ou seja, o processo é repetido até que o erro entre os valores passados e atuais sejam o mais próximo ou igual a 0. Para realizar esse processo é necessário encontrar a sequencia correta das barras para realizar os cálculos das correntes e tensões de forma mais eficiente. Uma forma de encontrar essas sequencias é fazer caminhamentos pela árvore gerada do sistema.

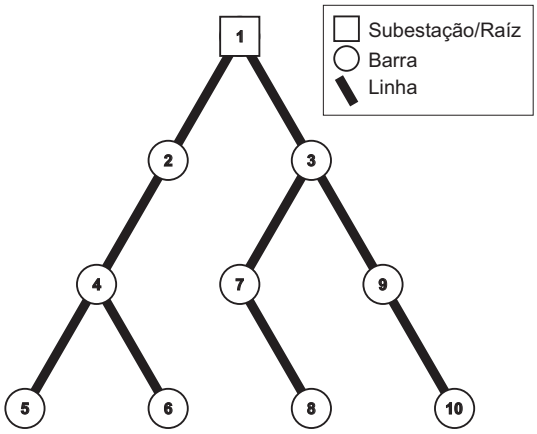

Figura 8: Determinação da sequencia de barras para cálculo das tensões

A sequencia das barras para calcular as tensões pode ser obtida por meio de uma busca em largura, como é mostrado na Figura 8. O nó de início será sempre uma subestação, onde a tensão já se é conhecida. Com os valores da corrente e a impedância da linha entre as barras, é possível usar as leis de Kirchhoff para calcular a tensão da barra seguinte.

A sequencia das barras para calcular as correntes pode ser obtida por meio de uma busca em profundidade, como é mostrado na Figura 9. As barras iniciais são as barras mais distantes da subestação (as folhas da árvore), onde a tensão e a potência requerida já são conhecidas. Com isso é possível calcular a corrente dessas barras e de sua linha adjacente. Com isso, as próximas correntes podem ser calculadas a partir das correntes anteriores.

O algoritmo do fluxo de potência implementado é mostrado no Algoritmo 6. Nessa implementação, além de fazer o cálculo das tensões e correntes do sistema, também

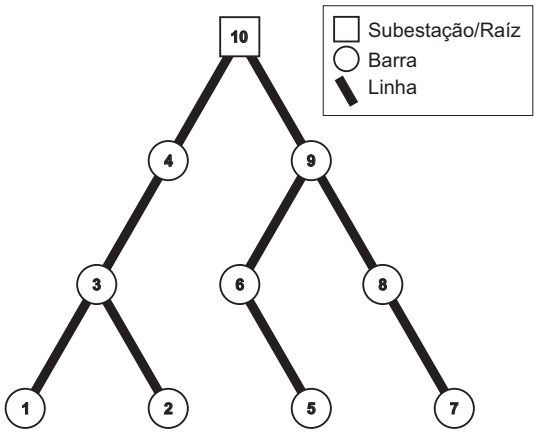

Figura 9: Determinação da sequencia de barras para cálculo das correntes

é feita a verificação na configuração atual da existência de ciclos ao longo da rede. A função retorna *verdadeiro* quando o fluxo estiver correto e $falso$ quando existir algum ciclo.

A partir da iteração das etapas de atualização das tensões e correntes, os valores convergirão para seus valores reais. Essa iteração é feita até que o erro entre as tensões seja 0 ou tão pequeno que seja não significativo.

Algoritmo 6: Fluxo de Potência

```
    Entrada: G
    Saída: boolean
 1  início
 2      pilha = ⊘
 3      vFila = ⊘
 4      iFila = ⊘
 5      para ∀s ∈ G|s é subestação faça
 6          raiz = s
 7          para pilha ≠ ⊘ faça
 8              para ∀s' ∈ G|s' é vizinho de raiz faça
 9                  se s'.cor == branco então
10                      Empilha(pilha, s')
11                      Enfileira(vFila, s')
12                      s'.cor = cinza
13                      s'.pai = raiz
14                      raiz = s'
15                  fim
16                  se s'.cor == preto ∧ s'.pai ≠ raiz então
17                      retorna false
18                  fim
19              fim
20              raiz.cor = preto;
21              Enfileira(iFila, raiz);
22              raiz = Desempilha(pilha);
23          fim
24      fim
25      repita
26          atualizaCorrente(iFila)
27          erro = atualizaTensao(vFila)
28      até erro > erroMínimo;
29      retorna true
30  fim
```

5 RESULTADOS

Para a validação da aplicação do método proposto, foram utilizados modelos de redes de distribuição utilizadas por Das (2006) e por Zidan e El-Saadany (2012), representadas pela Figuras 10 e 11, respectivamente, executados em um computador com processador CORE i7 com 8GB de memória RAM.

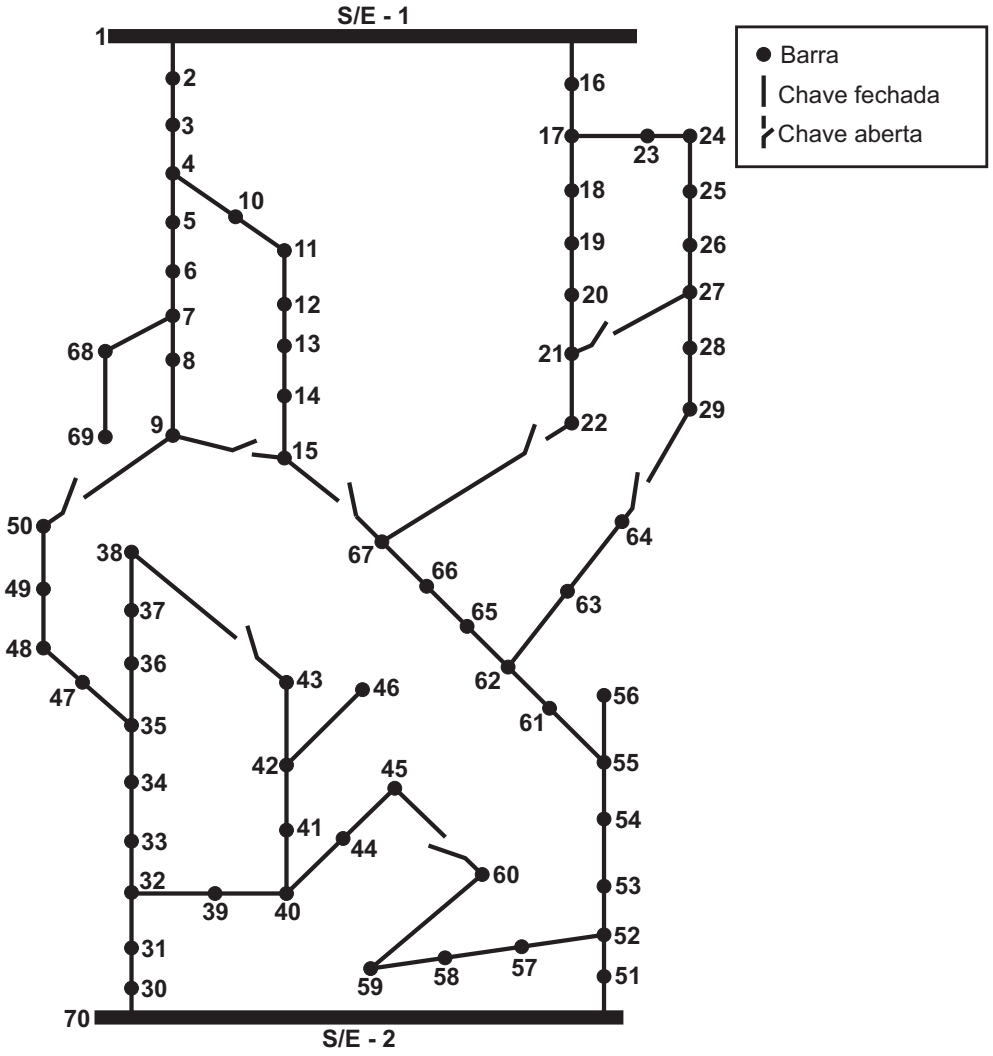

Figura 10: Rede de distribuição de Das (2006)

Zidan e El-Saadany (2012) utilizou uma configuração semelhante a de Das (2006), exceto pela quantidade de chaves contidas no sistema. Para Das (2006) todas as linhas são chaves do sistema totalizando 76 chaves, e para Zidan e El-Saadany (2012) apenas as apresentadas na Figura 11 totalizando 72 conexões, sendo 43 chaves.

Os resultados foram avaliados de acordo com seu desempenho quando submetidos a situação em que ocorre uma falta e com diferentes níveis para a estratégia de nivelamento das chaves para observar o comportamento do sistema em relação das variáveis tempo, valor da função objetivo, quantidade de chaves manobradas, potência dissipada pelas

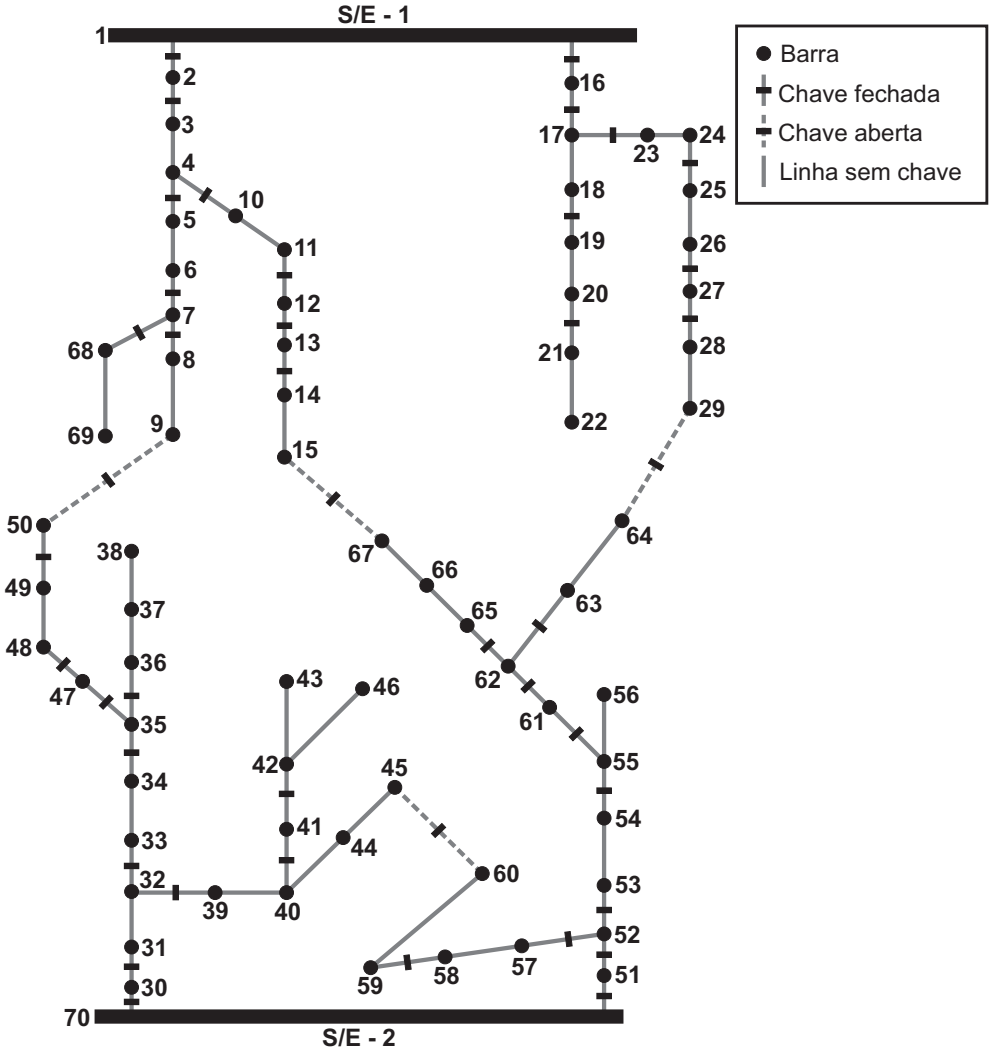

Figura 11: Rede de distribuição de Zidan e El-Saadany (2012)

Tabela 6: Tensão e perda de potência da rede de Das (2006)

Método	Menor tensão (pu)	Perda de Potência Real (kW)
Das (2006)	0.88389	337.45
Gauss-Seidel (implementado)	0.88405	341.28

linhas e pela quantidade de barras energizadas.

Para verificar a precisão e desempenho do fluxo de potência implementado, foi executado a configuração de Das (2006) e os resultados são mostrados na Tabela 6 para demonstrar a exatidão do algoritmo implementado. Também foram calculados os valores da corrente em cada subestação, como é mostrado na Tabela 7.

Os algoritmos foram submetidos a diferentes situações como configuração total da rede, reconfiguração total da rede e reconfiguração após a ocorrência de uma falta. Essas situações foram propostas para a avaliação com o intuito de verificar a eficiência de cada tipo de algoritmo, visto que os algoritmos implementados são de naturezas distintas:

Tabela 7: Corrente nas subestações da rede de Das (2006)

Subestação	Das (2006) (A)	Gauss-Seidel (A)
F1	121.0	121.386
F2	132.3	132.3
F3	197.2	199.841
F4	181.5	181.457

GRASP é um algoritmo que possui uma metaheurística com etapas de construção e de busca local, enquanto o *Simulated Annealing* é uma metaheurística de busca local.

5.1 ALGORITMOS

5.1.1 GRASP

O algoritmo GRASP necessitou que fosse definidas estratégias e parâmetros para sua execução, como estratégia gulosa, a lista RCL e quantidade de iterações.

A fim de se ter uma comparação pela quantidade de iterações realizadas pelos algoritmos em questão, os parâmetros que controlam essa quantidade foram modelados para que a quantidade de iterações fossem bem aproximadas. Onde que, para o critério de parada do algoritmo foi definido uma quantidade de iterações de acordo com os parâmetros do *Simulated Annealing* (α, T_0 e T_f). Na busca local, foi definida uma quantidade fixa de iterações para definir o ótimo local, onde esse valor é igual ao parâmetro $SAmax$ *Simulated Annealing*.

A estratégia gulosa utilizada para adicionar os candidatos à RCL foi adicionar as chaves em que tivessem uma barra energizada, ou seja, uma chave para entrar na RCL deveria ter uma barra energizada e outra não. Com isso, a cada iteração do construtor uma nova barra é energizada. A lista RCL não teve tamanho fixo. Sendo assim, todos os candidatos que satisfizessem a condição gulosa para estar na lista foram adicionados.

Na fase de busca local, foi optado por fazer a troca das chaves aos pares, pois uma vez que todas as barras estiverem energizadas, qualquer movimento singular resultaria na desconexão de um nó, o que resultaria em um *fitness* menor do que se tem. E ao trocar as chaves aos pares, se tem a opção de reconectar uma área de forma mais eficiente.

5.1.2 *SIMULATED ANNEALING*

Os parâmetros do SA foram definidos após os critérios de satisfazibilidade fossem atingidos, como qualidade da resposta e tempo.

O valor de $SAmax$ foi definido como sendo igual a quantidade de chaves contidas

no sistema, onde, que, quanto maior a rede em questão, mais iterações serão necessárias para que a solução seja a melhor em sua vizinhança. O valor de α foi estabelecido de um modo que houvesse um equilíbrio entre tempo e qualidade da resposta.

5.1.3 LISTA TABU

A Lista Tabu foi implementada juntamente com uma nova implementação do GRASP e do SA. Essa estratégia foi escolhida para que os algoritmos não se prendessem a possíveis *loops* e que fosse melhor explorado o espaço de busca.

A Lista Tabu teve o tamanho de 1/3 da quantidade de chaves do sistema. As chaves permanecem bloqueadas pelo determinado número de iterações e então liberadas quando esse número de iterações é expirado.

5.2 CONFIGURAÇÃO TOTAL DA REDE

A configuração total da rede é iniciada com todas as chaves abertas e desbloqueadas, onde os algoritmos terão que configurar toda a rede, como se fosse iniciar uma rede nova. Os algoritmos recebem como estado inicial as chaves do sistema todas desbloqueadas e abertas. O algoritmo deve fazer a configuração total da rede de forma a maximizar a Função 5. Nas Figuras 12 e 13 são mostrados os resultados de cada iteração executada de cada algoritmo derivados da função objetivo.

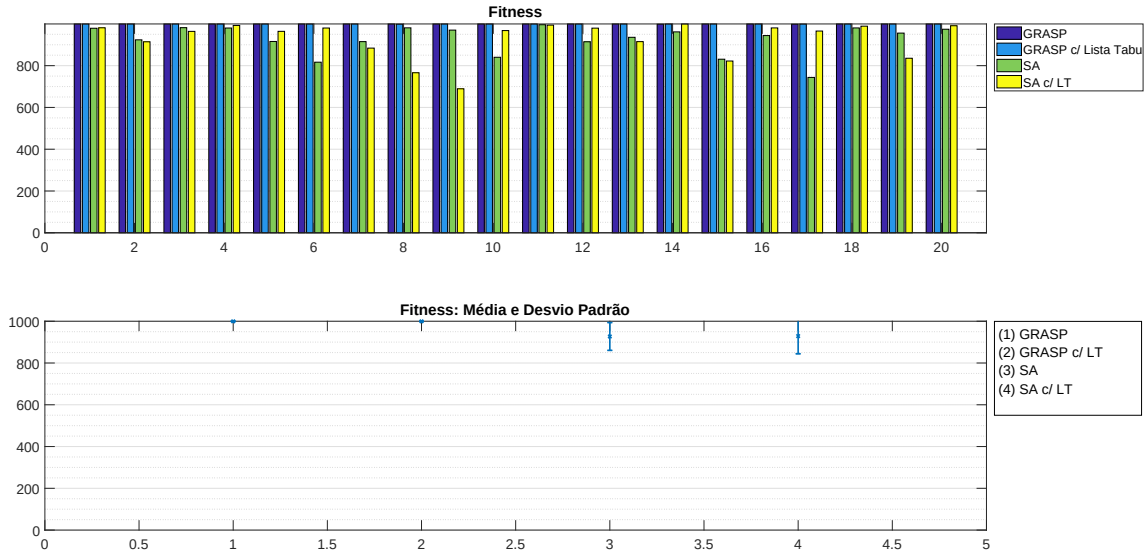

Figura 12: *Fitness* por iteração - rede de Zidan e El-Saadany (2012)

Em todas os casos testados, os os algoritmos conseguiram a energização de todas as 70 barras. Na função objetivo o item que corresponde a quantidade de chaves trocadas foi tida como 0, pois nesse momento foi analisado o desempenho dos algoritmos em realizar

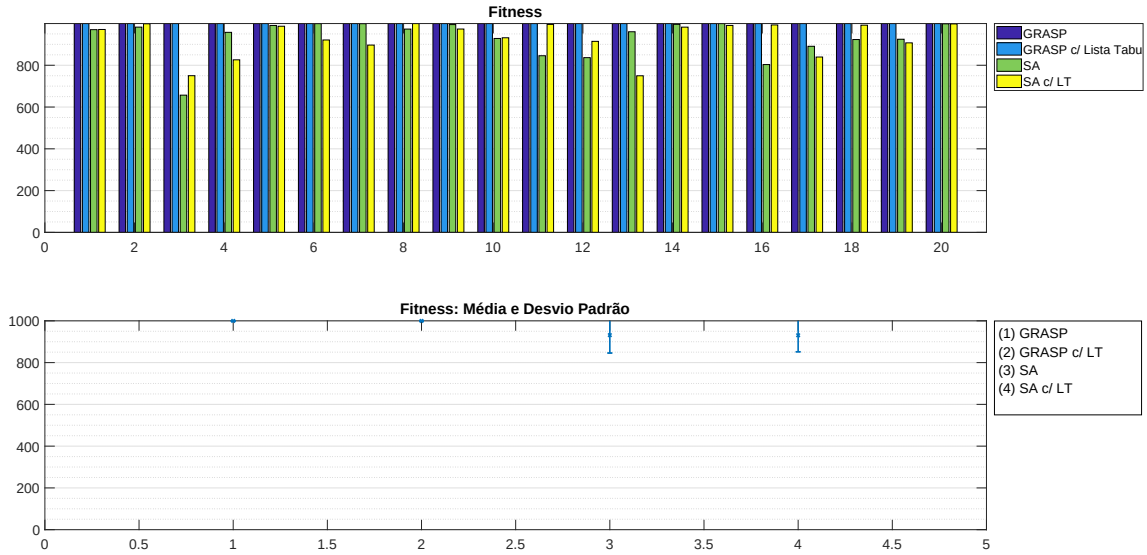

Figura 13: *Fitness* por iteração - rede de Das (2006)

a configuração desde o início.

Para cada solução encontrada, a tensão do nó de menor tensão também foi registrada como mostra nas Figuras 14 e 15.

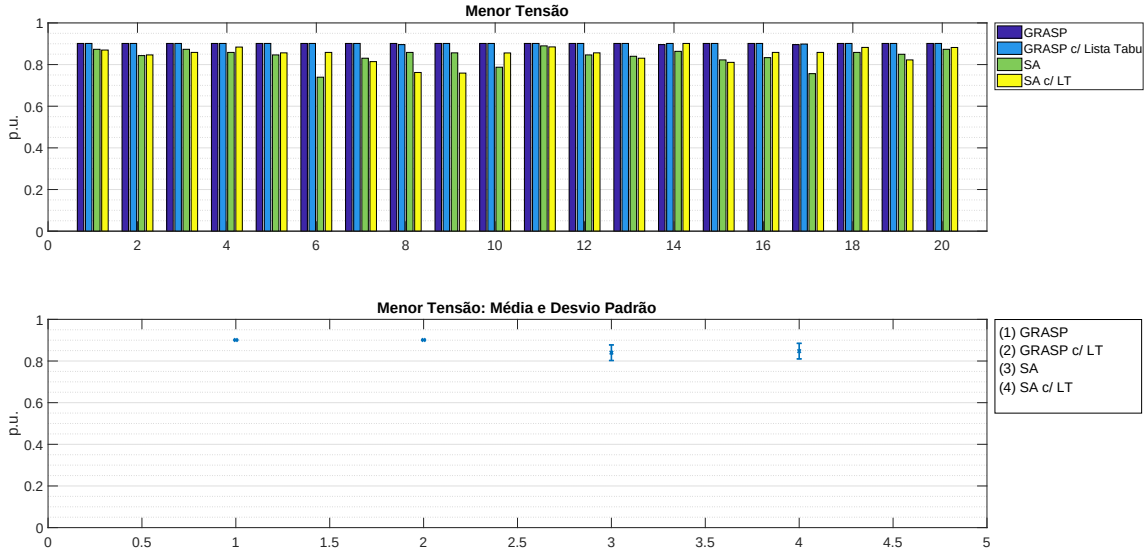

Figura 14: Menor tensão por iteração - rede de Zidan e El-Saadany (2012)

5.3 OCORRÊNCIA DE UMA FALTA

Partindo da configuração original de cada instância, foi aplicada uma falha entre as barras 53 e 54 nas redes de Zidan e El-Saadany (2012) e Das (2006) para o estudo do desempenho dos algoritmos.

Após a ocorrência da falta, inicia-se o procedimento de Preparação, onde os algoritmos localizam o local da falha e a isolam-a. Essa isolação é feita a partir da abertura das chaves mais próximas do local falha adicionando *flags* para essa chaves para que não

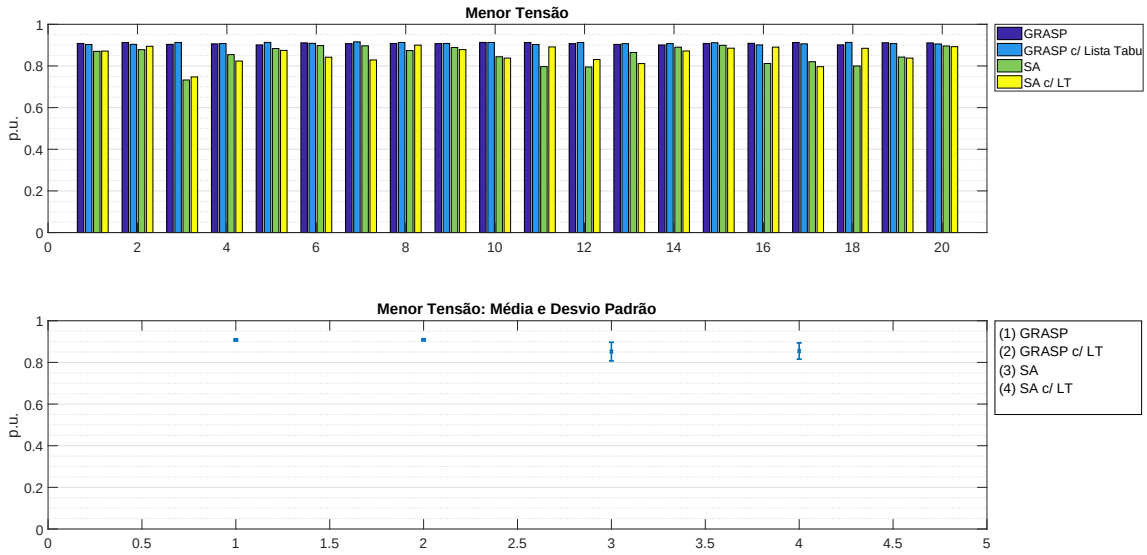

Figura 15: Menor tensão por iteração - rede de Das (2006)

sejam operadas no processo de Restauração.

Após a Preparação, houve uma área em que as barras ficaram desconectadas, ou seja, sem energia. Para que essas barras sejam reconectadas é iniciado o processo de Restauração. No processo de Restauração os algoritmos atuam para reconectar as barras desconectadas de acordo com as restrições citadas.

No processo de Preparação de ambos os casos foram isoladas 2 barras. Ambos os algoritmos obtiveram a reconexão de todas as barras desconectadas, resultando em 68 barras energizadas. As perdas resultantes de cada configuração são representadas nas Figuras 16 e 17.

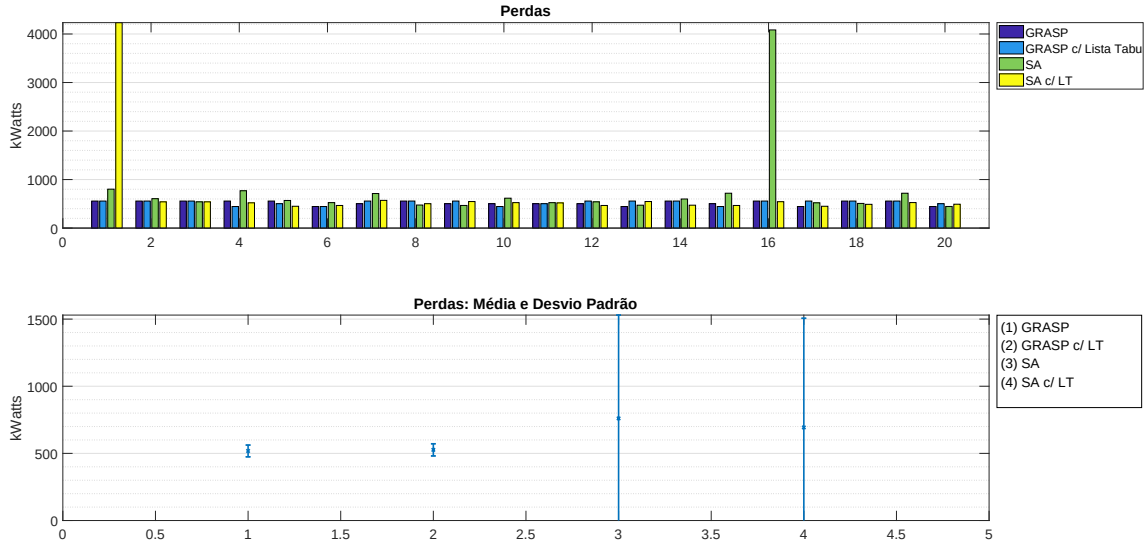

Figura 16: Perdas por iteração - rede de Zidan e El-Saadany (2012)

Na rede de Zidan e El-Saadany (2012), houveram casos em que a reconfiguração não favoreceu ao fator Perda de Potência, visto que esses valores foram consideravelmente

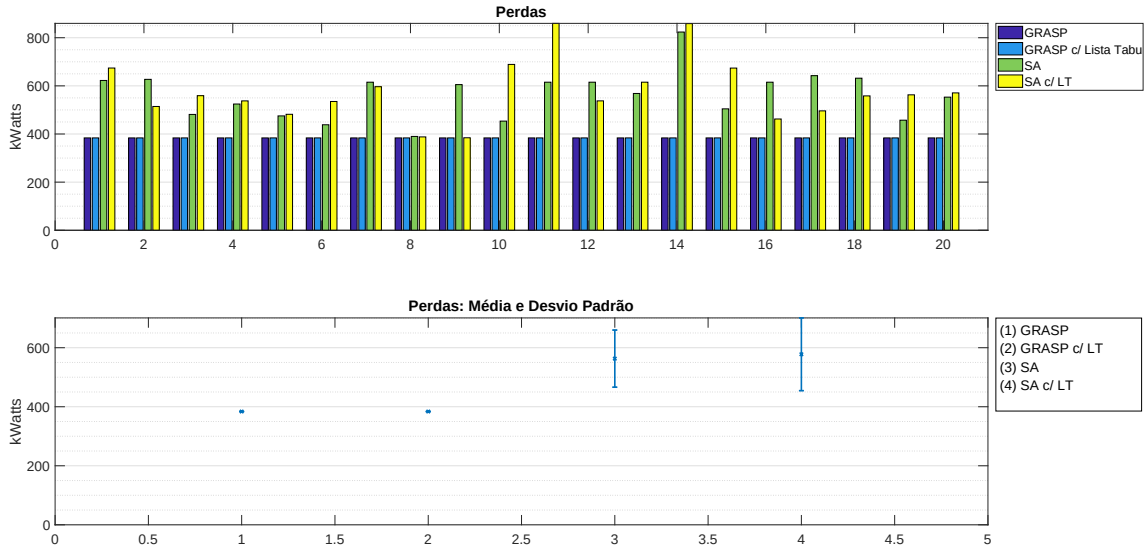

Figura 17: Perdas por iteração - rede de Das (2006)

altos comparados à média entre as amostras dos casos para o mesmo algoritmo. Isso fez com que o desvio padrão fosse significativamente maior para esses algoritmos.

Na rede de Das (2006), não houve uma discrepância tão alta no resultado resultando em um baixo desvio padrão em todos os casos.

As menores tensões registradas são representada nas Figuras 18 e 19.

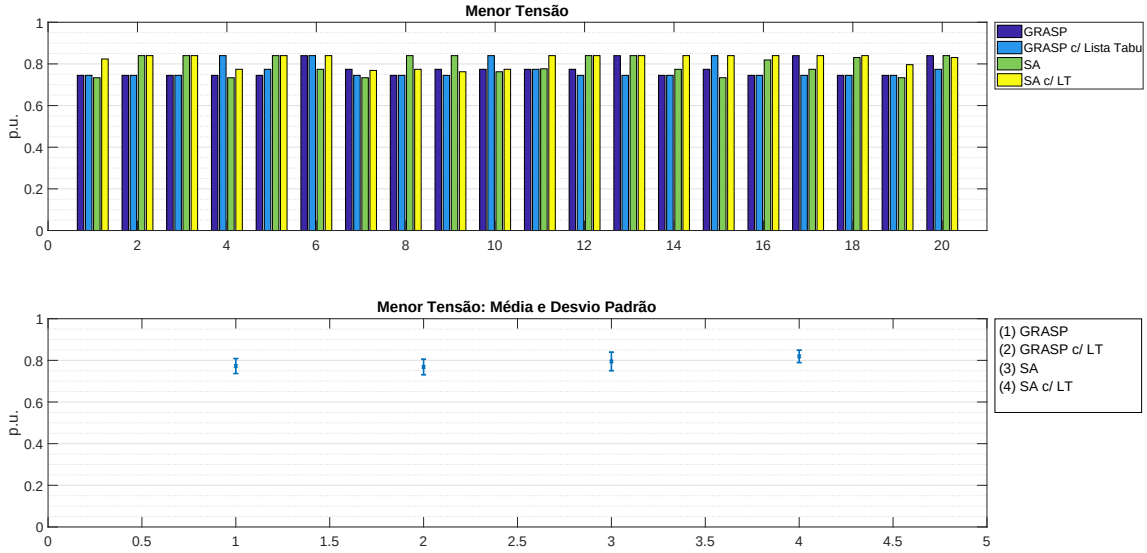

Figura 18: Menor Tensão - rede de Zidan e El-Saadany (2012)

As chaves manobradas são representadas nas Figuras 20 e 21. Esses números de manobras foram obtidas a partir da comparação da posição das chaves originais, configuração inicial da rede, para a configuração gerada pelos algoritmos.

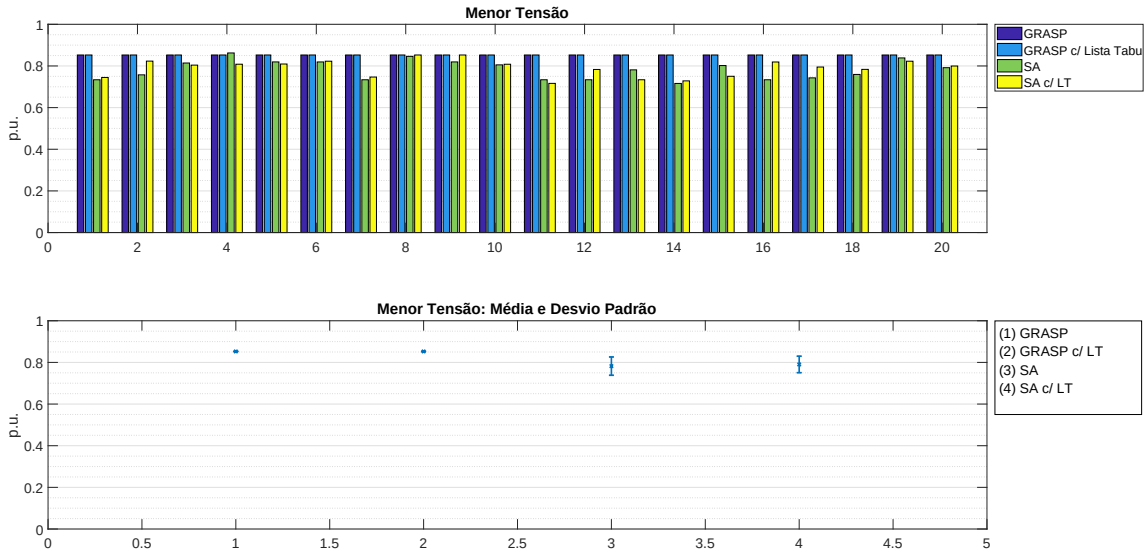

Figura 19: Menor tensão - rede de Das (2006)

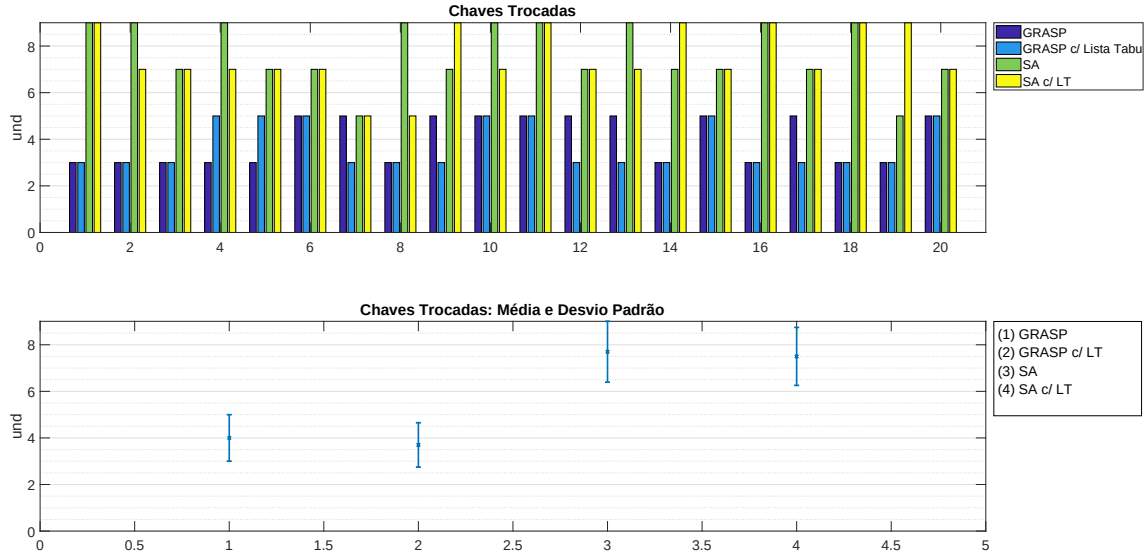

Figura 20: Chaves Manobradas - rede de Zidan e El-Saadany (2012)

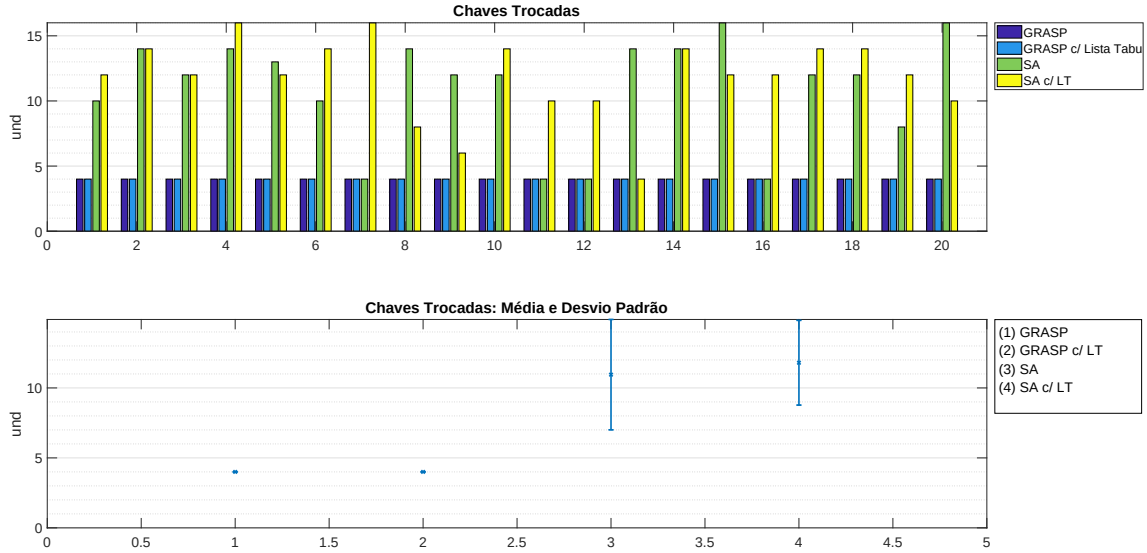

Figura 21: Chaves Manobradas - rede de Das (2006)

5.4 OCORRÊNCIA DE UMA FALTA COM NIVELAMENTO DAS CHAVES

Partindo da configuração original de cada instância, foi aplicada uma falha entre as barras 53 e 54 nas redes de Zidan e El-Saadany (2012) e Das (2006) para o estudo do desempenho dos algoritmos.

Após a ocorrência da falta, inicia-se o procedimento de Preparação, onde os algoritmos localizam o local da falha e a isolam-a. Essa isolação é feita a partir da abertura das chaves mais próximas do local falha adicionando *flags* para essa chaves para que não sejam operadas no processo de Restauração.

Após a Preparação, houve uma área em que as barras ficaram desconectadas, ou seja, sem energia. Para que essas barras sejam reconectadas é iniciado o processo de Restauração. No processo de Restauração os algoritmos atuam para reconectar as barras desconectadas de acordo com as restrições citadas.

Nesse processo de Preparação foi onde ocorreu o nivelamento das chaves, como citado na seção 3.2, partindo das barras desenergizadas. A fim do estudo do comportamento dos algoritmos com tal nivelamento das chaves, variou-se de 1 a 8 o nível máximo das chaves.

No processo de Preparação de ambos os casos foram isoladas 2 barras. Com isso, houve a desenergização de 9 barras nas quais estão fora da falta e que necessitam ser reenergizadas. A figura 22 mostra como ficou o nivelamento feito para a rede de Das (2006) com níveis de 1 a 3.

Ambos os algoritmos obtiveram a reconexão de todas as barras desconectadas, resultando em 68 barras energizadas.

Os gráficos abaixo mostram as médias dos níveis de 1 a 8 representado no eixo horizontal com as médias geradas com a fase de Preparação sem o nivelamento das chaves representado na posição 9 do eixo horizontal.

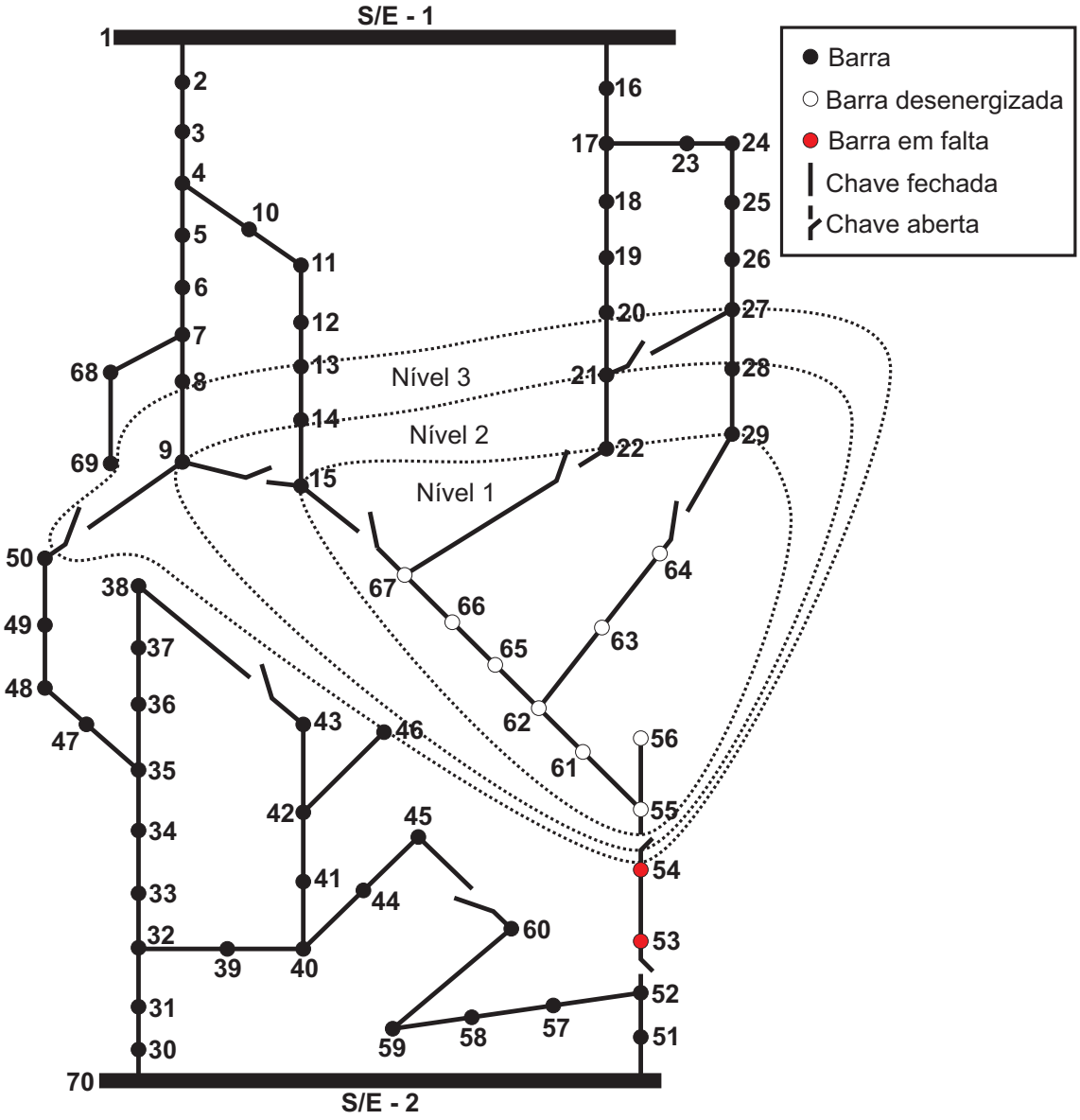

Figura 22: Nivelamento de 1 a 3 - rede de Das (2006)

5.4.1 ALGORITMO GRASP

Nessa configuração é notável que, com o nivelamento, a quantidade de chaves manobradas se mantiveram acima do resultado sem o nivelamento, com exceção do nível 1, em que se teve a menor média. Também houve um considerável aumento no tempo de execução que aumentou conforme se diminuiu o nível máximo das chaves. Em geral, o desvio padrão das amostras foi diminuído após o nivelamento. E houveram casos em que não foi reenergizado todas as barras.

5.4.1.1 REDE DE DAS

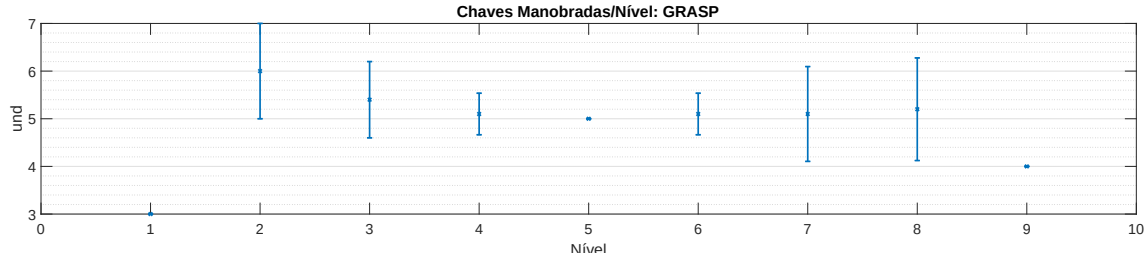

Figura 23: Chaves Manobradas GRASP - rede de Das (2006)

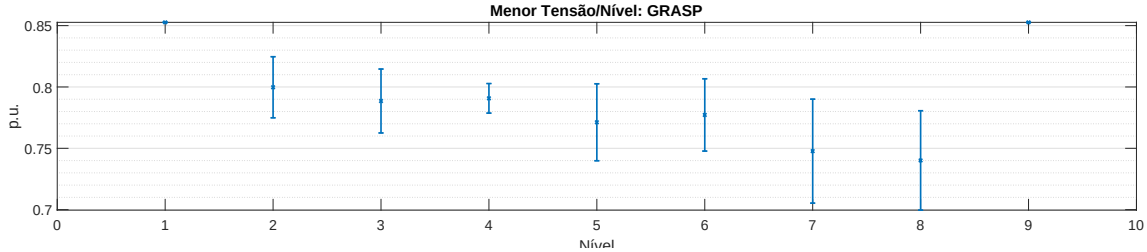

Figura 24: Menor Tensão GRASP - rede de Das (2006)

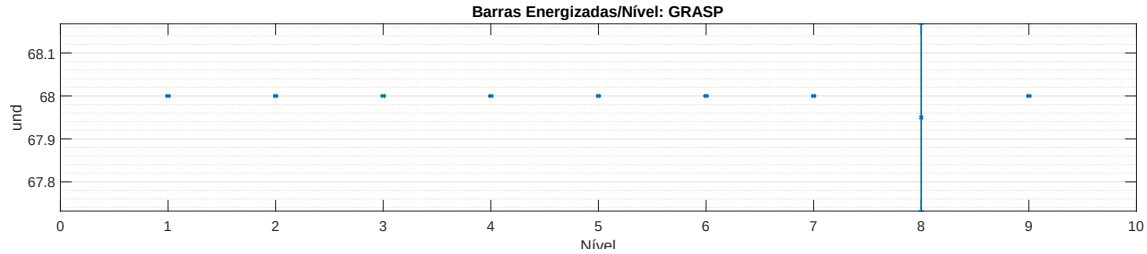

Figura 25: Barras Energizadas GRASP - rede de Das (2006)

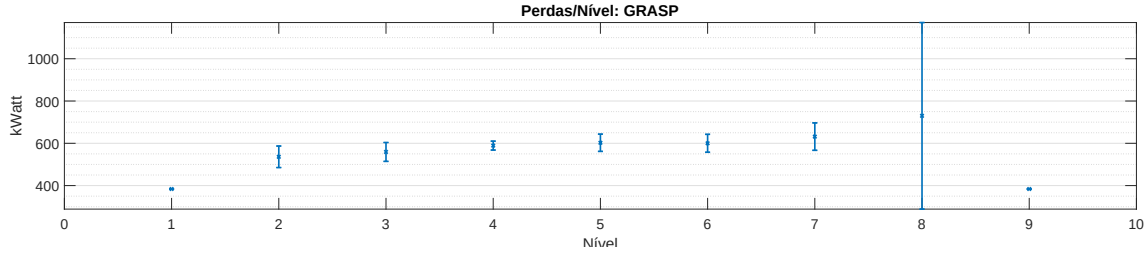

Figura 26: Perdas GRASP - rede de Das (2006)

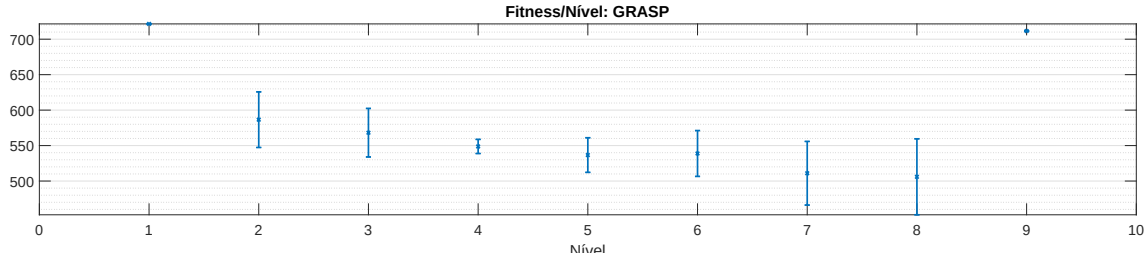

Figura 27: *Fitness* GRASP - rede de Das (2006)

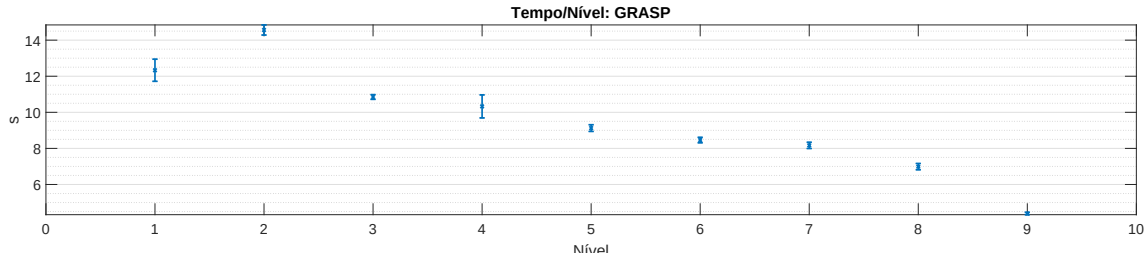

Figura 28: Tempo de execução GRASP - rede de Das (2006)

5.4.1.2 REDE DE ZIDAN E EL-SAADANY

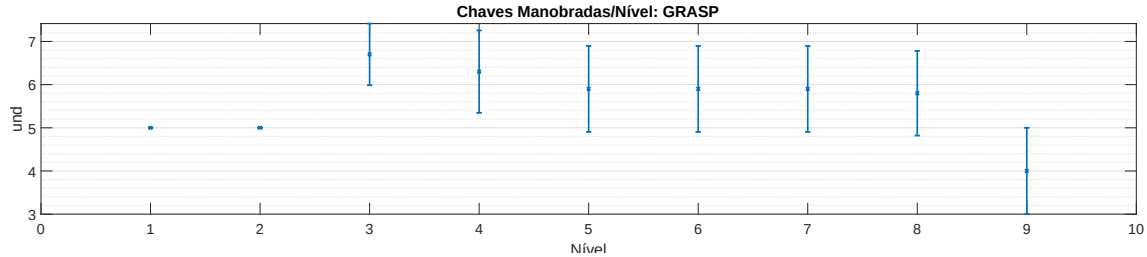

Figura 29: Chaves Manobradas GRASP - rede de Zidan e El-Saadany (2012)

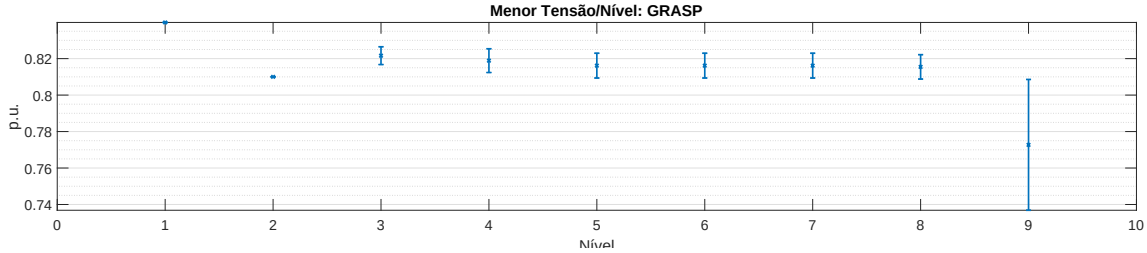

Figura 30: Menor Tensão GRASP - rede de Zidan e El-Saadany (2012)

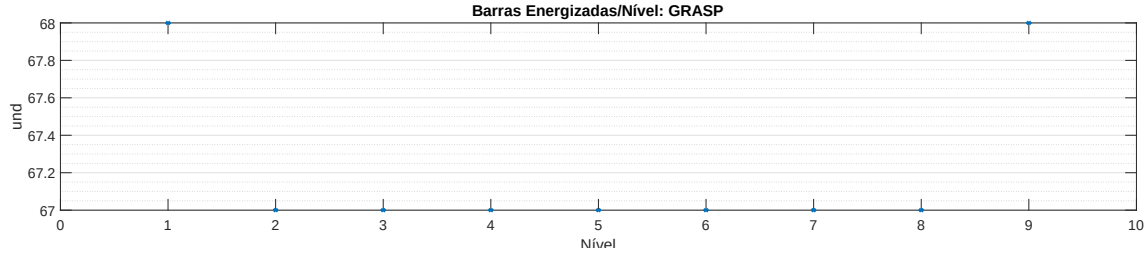

Figura 31: Barras Energizadas GRASP - rede de Zidan e El-Saadany (2012)

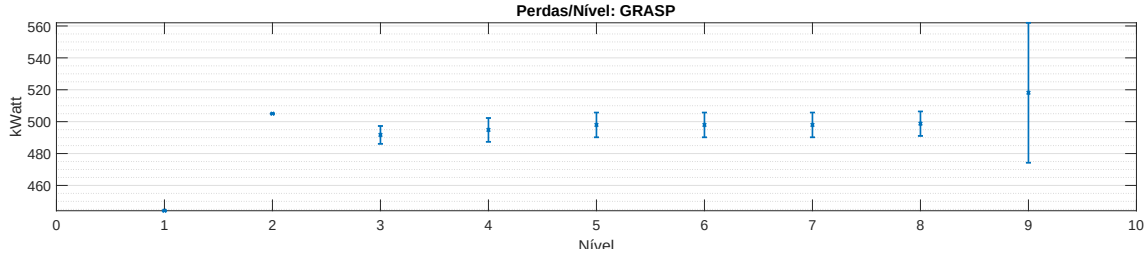

Figura 32: Perdas GRASP - rede de Zidan e El-Saadany (2012)

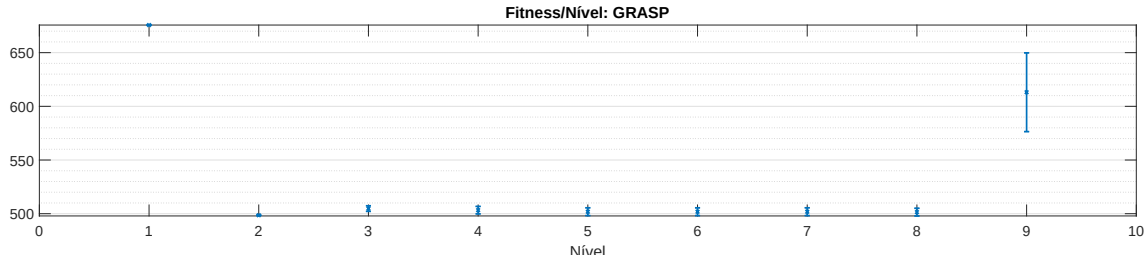

Figura 33: *Fitness* GRASP - rede de Zidan e El-Saadany (2012)

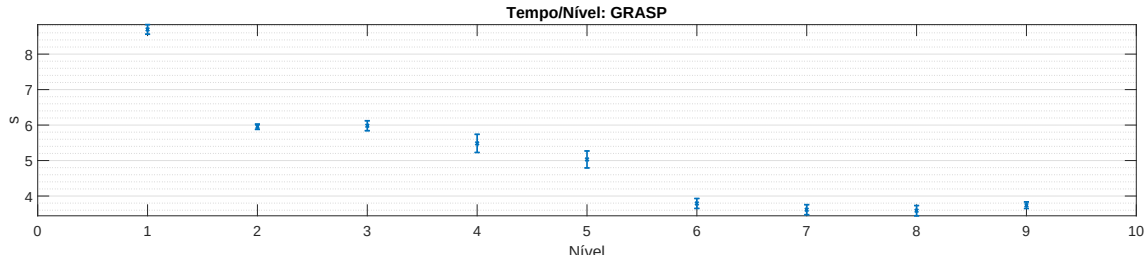

Figura 34: Tempo de execução GRASP - rede de Zidan e El-Saadany (2012)

5.4.2 ALGORITMO GRASP COM LISTA TABU

Ao inserir a Lista Tabu nos algoritmos, as médias tenderam a permanecer no mesmo padrão comparado ao método padrão. E ainda houve um leve aumento no desvio padrão de cada média, o que pode ser resultado da aplicação da Lista Tabu, que fez com que o algoritmo explorasse uma maior área.

5.4.2.1 REDE DE DAS

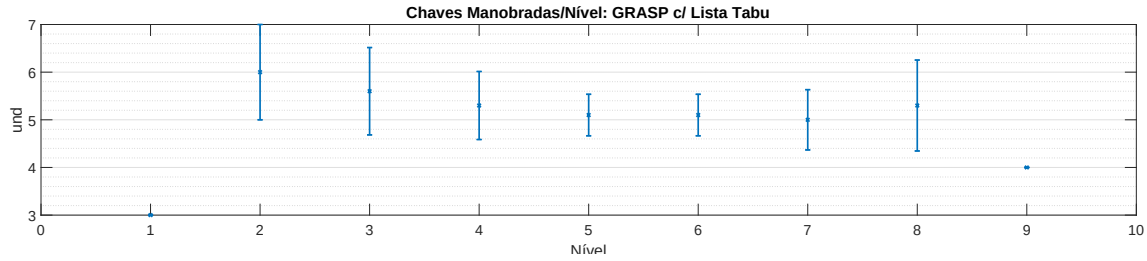

Figura 35: Chaves Manobradas GRASP c/ Lista Tabu - rede de Das (2006)

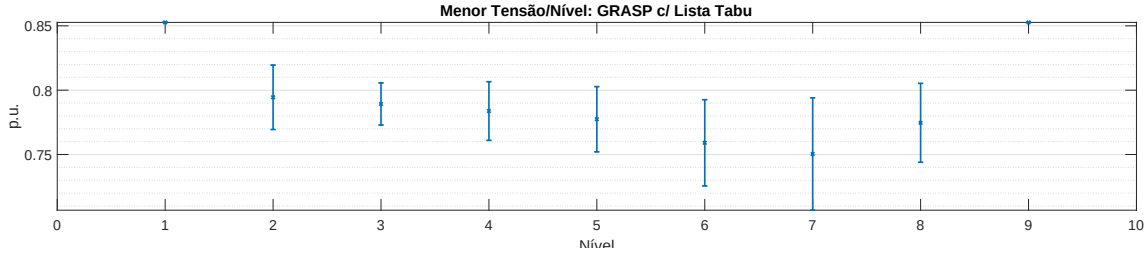

Figura 36: Menor Tensão GRASP c/ Lista Tabu - rede de Das (2006)

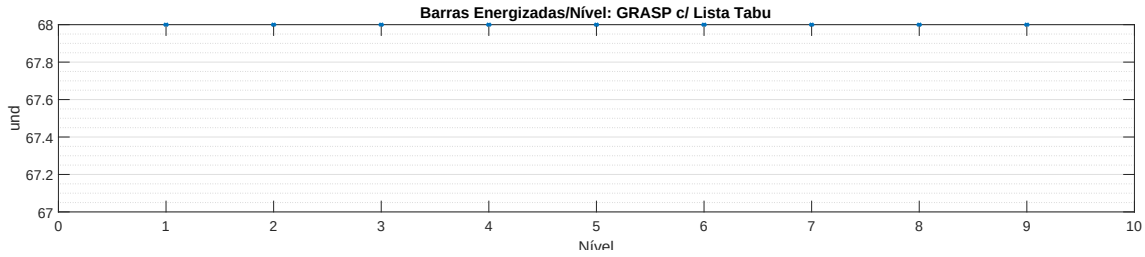

Figura 37: Barras Energizadas GRASP c/ Lista Tabu - rede de Das (2006)

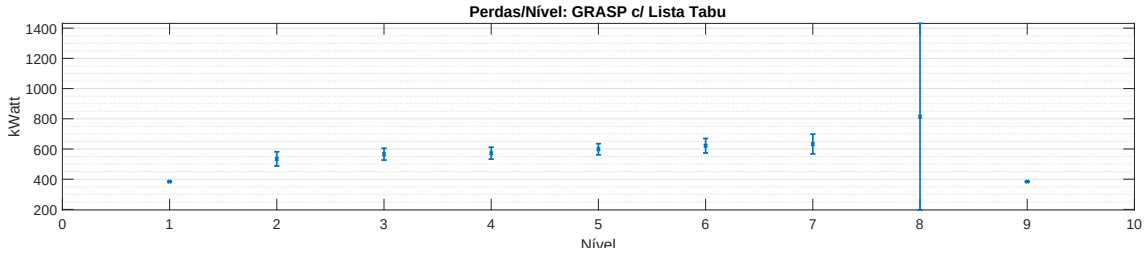

Figura 38: Perdas GRASP c/ Lista Tabu - rede de Das (2006)

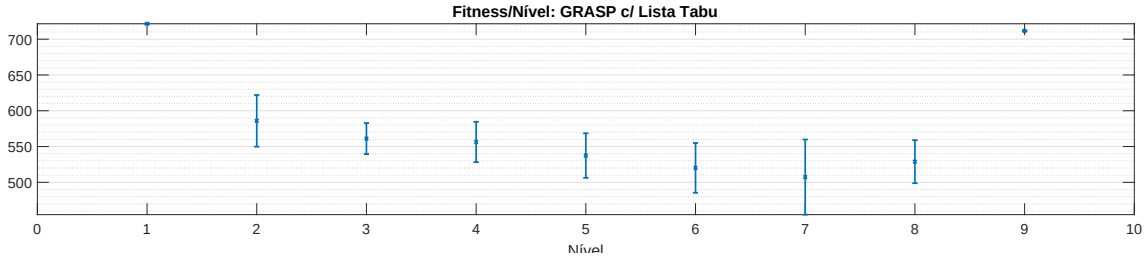

Figura 39: *Fitness* GRASP c/ Lista Tabu - rede de Das (2006)

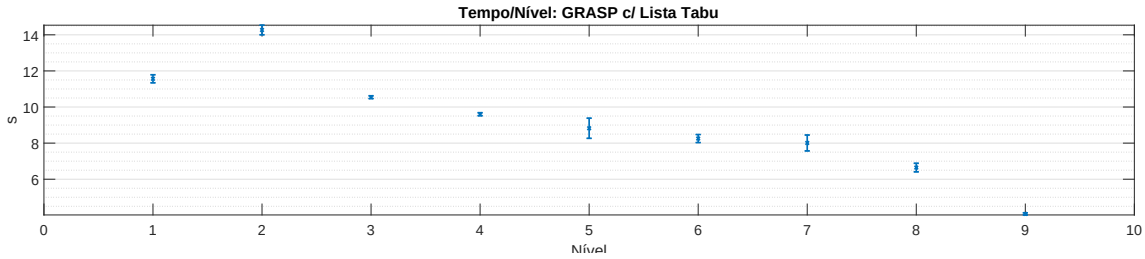

Figura 40: Tempo de execução GRASP c/ Lista Tabu - rede de Das (2006)

5.4.2.2 REDE DE ZIDAN E EL-SAADANY

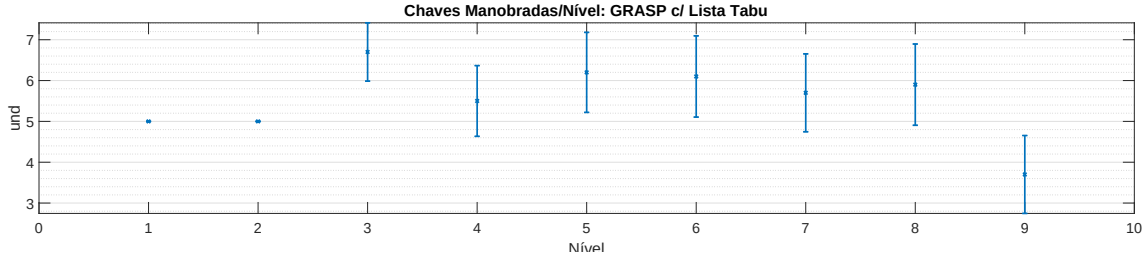

Figura 41: Chaves Manobradas GRASP c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

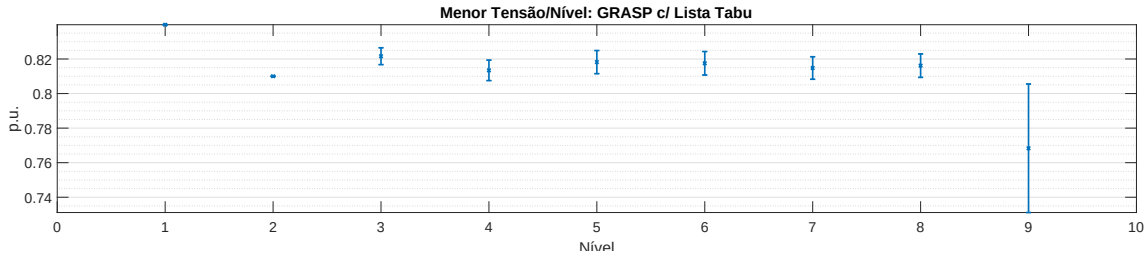

Figura 42: Menor Tensão GRASP c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

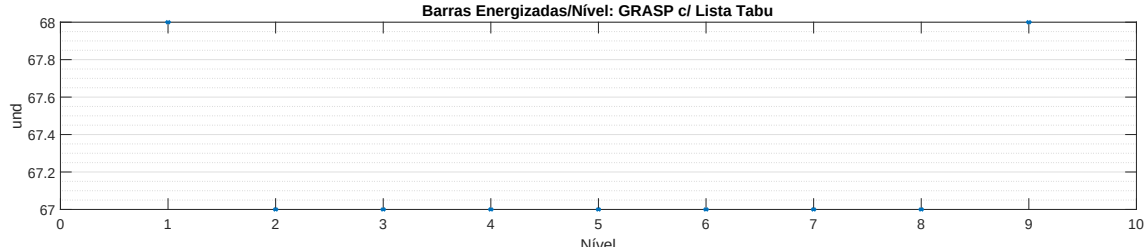

Figura 43: Barras Energizadas GRASP c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

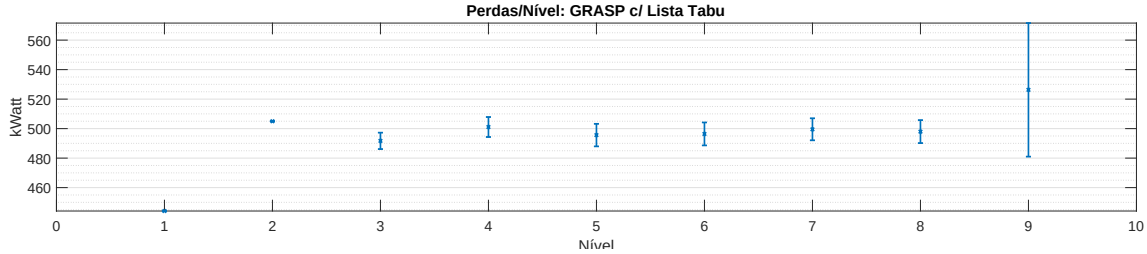

Figura 44: Perdas GRASP c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

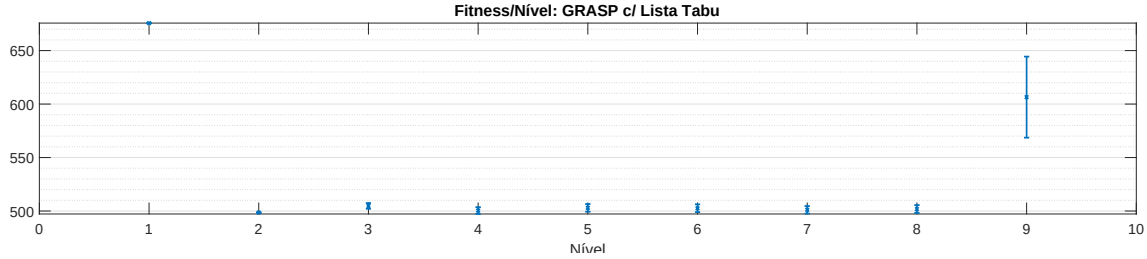

Figura 45: *Fitness* GRASP c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

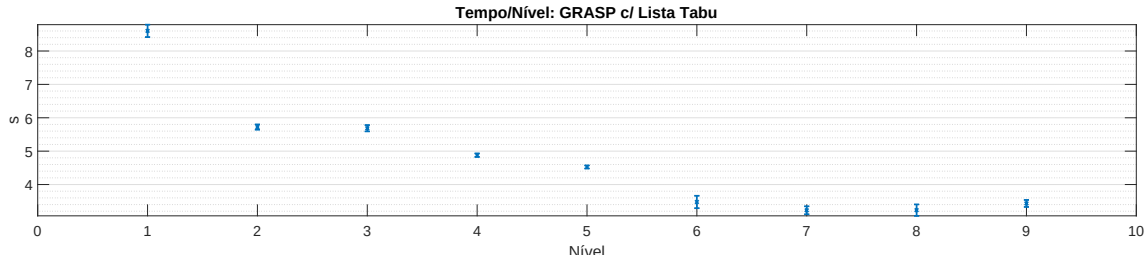

Figura 46: Tempo de execução GRASP c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

5.4.3 ALGORITMO *SIMULATED ANNEALING*

Nessa configuração houve uma visível melhora com o nivelamento, em que, com a diminuição do nível máximo, ocorreu a diminuição no desvio padrão e melhora nos

resultados como aumento da tensão mínima e do *fitness* e diminuição da quantidade de chaves manobradas, das perdas e do tempo de execução.

5.4.3.1 REDE DE DAS

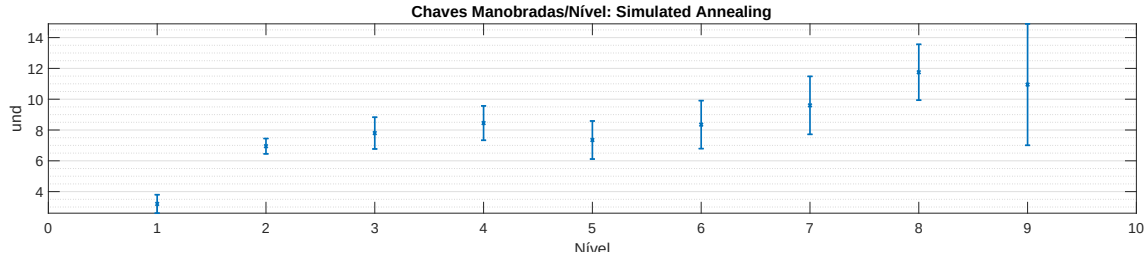

Figura 47: Chaves Manobradas *Simulated Annealing* - rede de Das (2006)

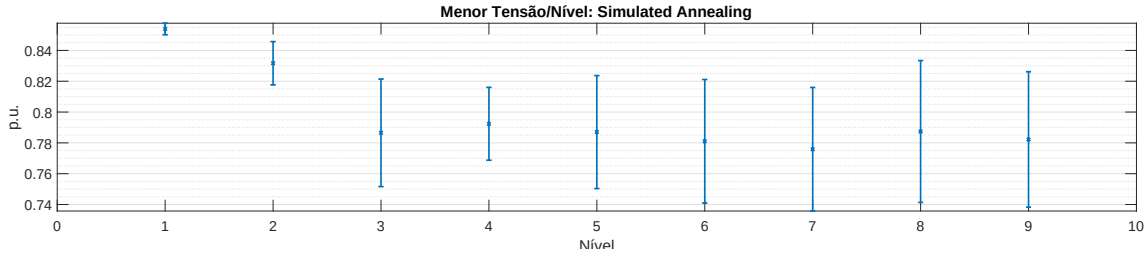

Figura 48: Menor Tensão *Simulated Annealing* - rede de Das (2006)

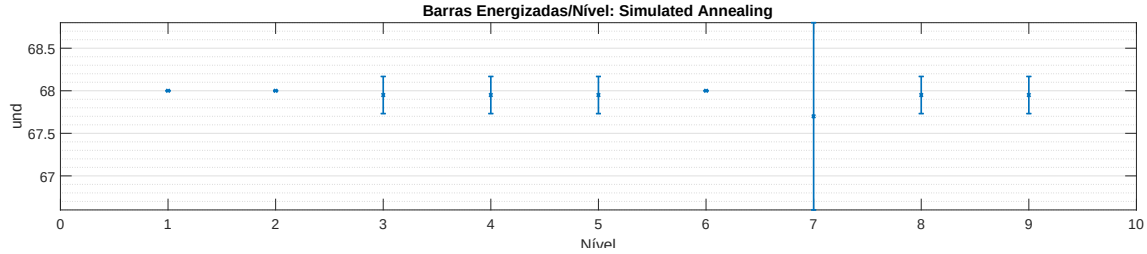

Figura 49: Barras Energizadas *Simulated Annealing* - rede de Das (2006)

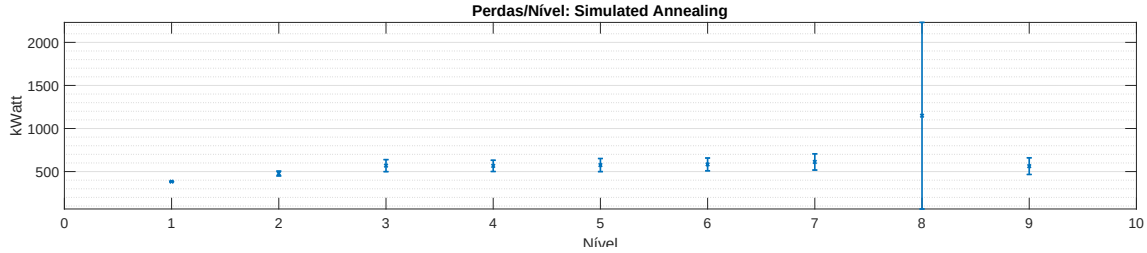

Figura 50: Perdas *Simulated Annealing* - rede de Das (2006)

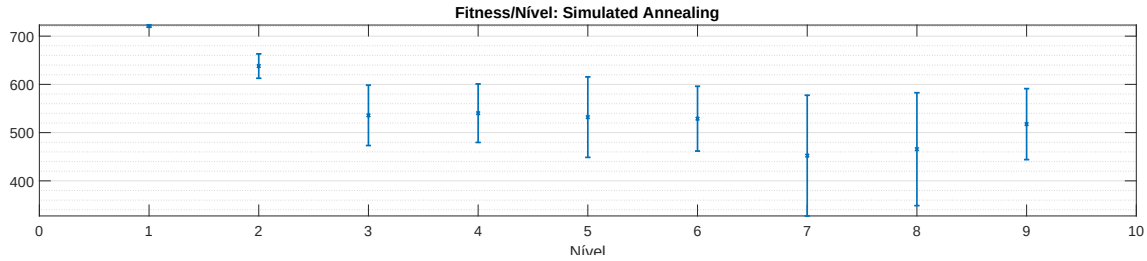

Figura 51: *Fitness Simulated Annealing* - rede de Das (2006)

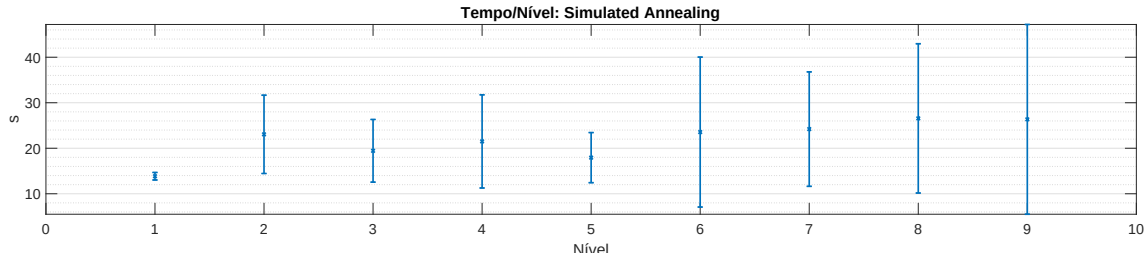

Figura 52: Tempo de execução *Simulated Annealing* - rede de Das (2006)

5.4.3.2 REDE DE ZIDAN E EL-SAADANY

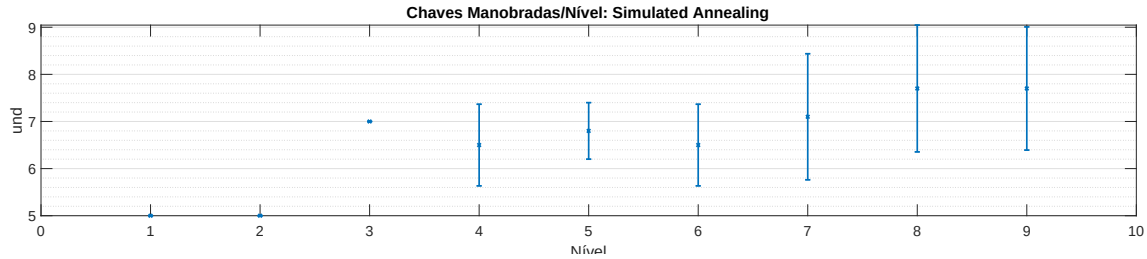

Figura 53: Chaves Manobradas *Simulated Annealing* - rede de Zidan e El-Saadany (2012)

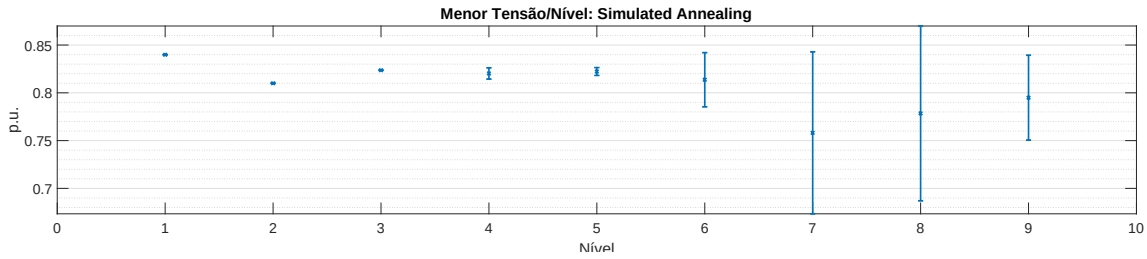

Figura 54: Menor Tensão *Simulated Annealing* - rede de Zidan e El-Saadany (2012)

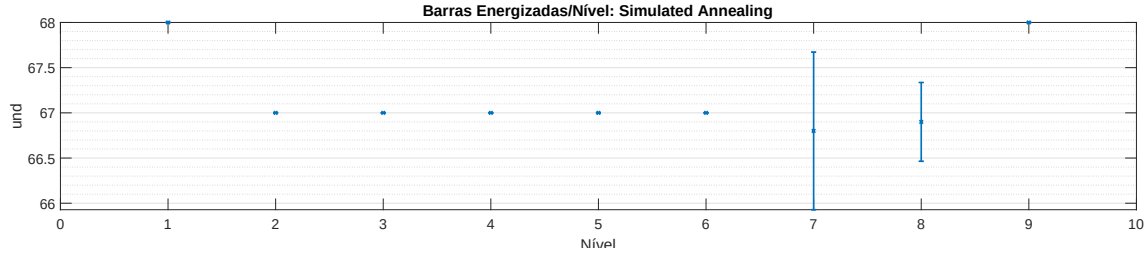

Figura 55: Barras Energizadas *Simulated Annealing* - rede de Zidan e El-Saadany (2012)

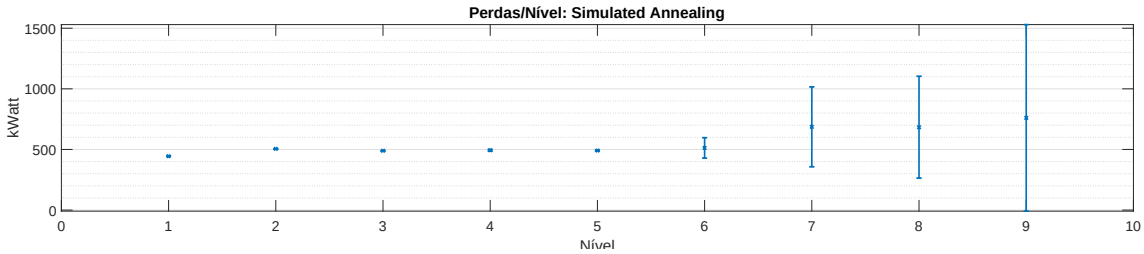

Figura 56: Perdas *Simulated Annealing* - rede de Zidan e El-Saadany (2012)

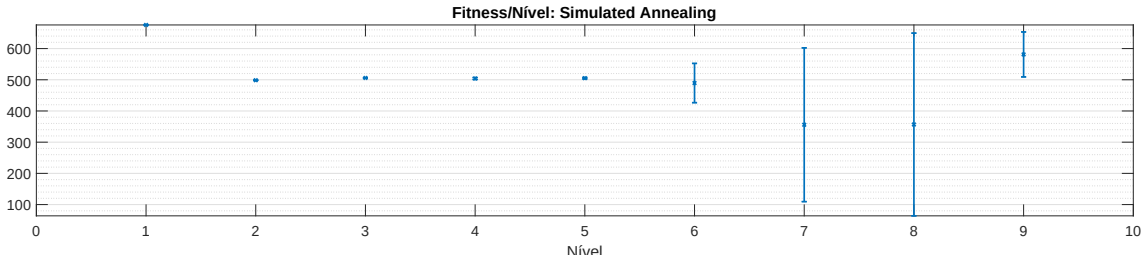

Figura 57: *Fitness Simulated Annealing* - rede de Zidan e El-Saadany (2012)

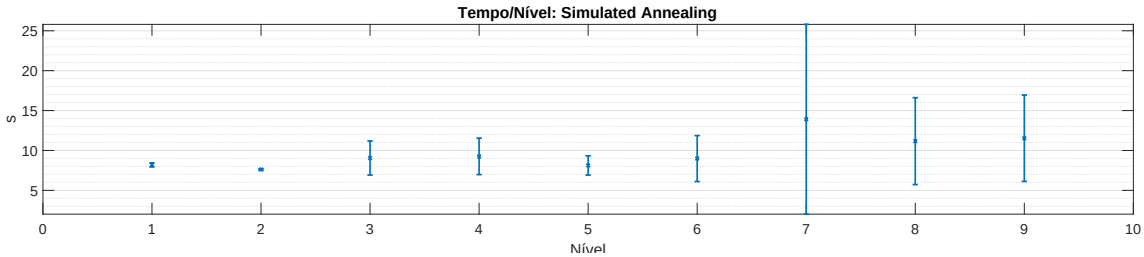

Figura 58: Tempo de execução *Simulated Annealing* - rede de Zidan e El-Saadany (2012)

5.4.4 ALGORITMO *SIMULATED ANNEALING* COM LISTA TABU

Com a inserção da Lista Tabu, as médias ficaram semelhantes ao método convencional e ainda com um aumento no desvio padrão.

5.4.4.1 REDE DE DAS

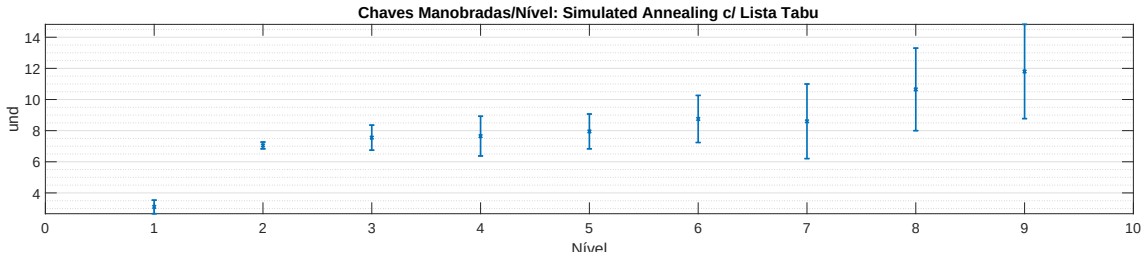

Figura 59: Chaves Manobradas *Simulated Annealing* c/ Lista Tabu - rede de Das (2006)

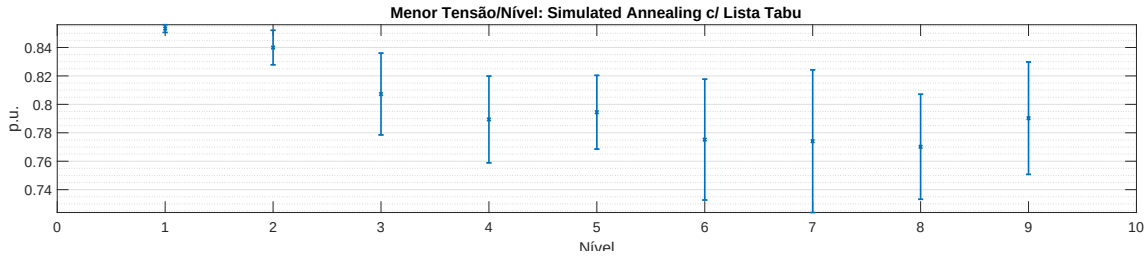

Figura 60: Menor Tensão *Simulated Annealing* c/ Lista Tabu - rede de Das (2006)

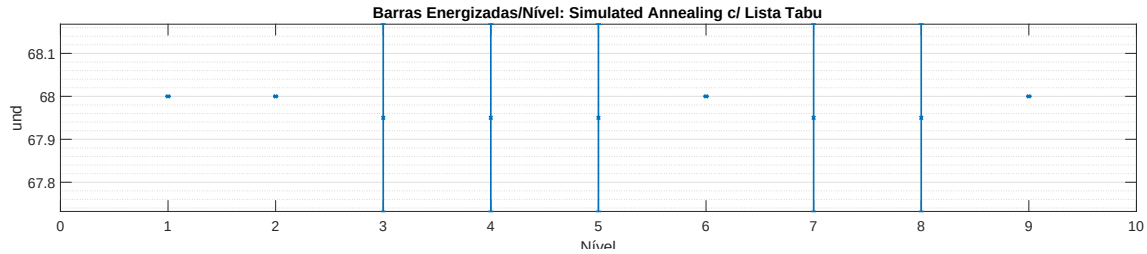

Figura 61: Barras Energizadas *Simulated Annealing* c/ Lista Tabu - rede de Das (2006)

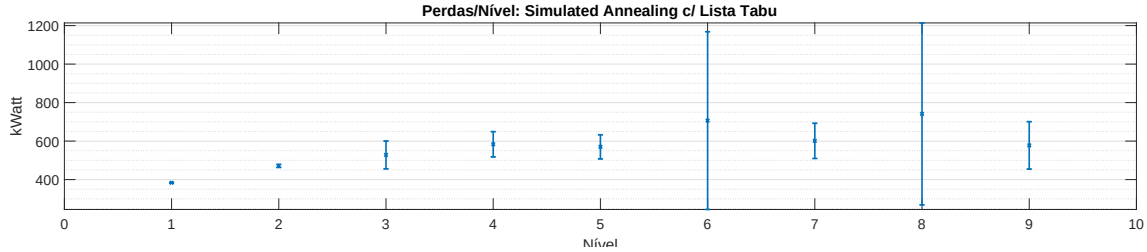

Figura 62: Perdas *Simulated Annealing* c/ Lista Tabu - rede de Das (2006)

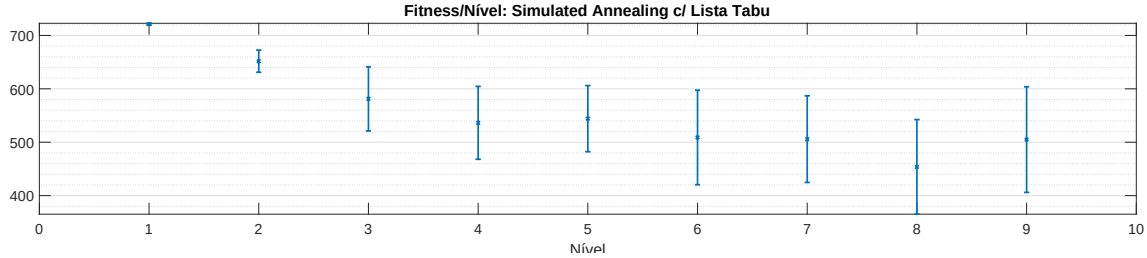

Figura 63: *Fitness Simulated Annealing* c/ Lista Tabu - rede de Das (2006)

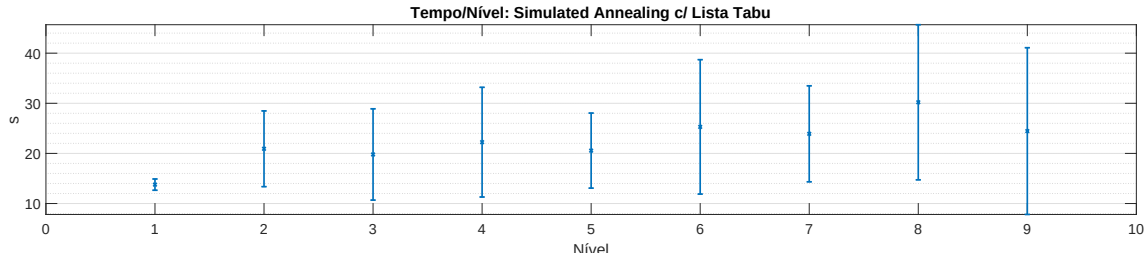

Figura 64: Tempo de execução *Simulated Annealing* c/ Lista Tabu - rede de Das (2006)

5.4.4.2 REDE DE ZIDAN E EL-SAADANY

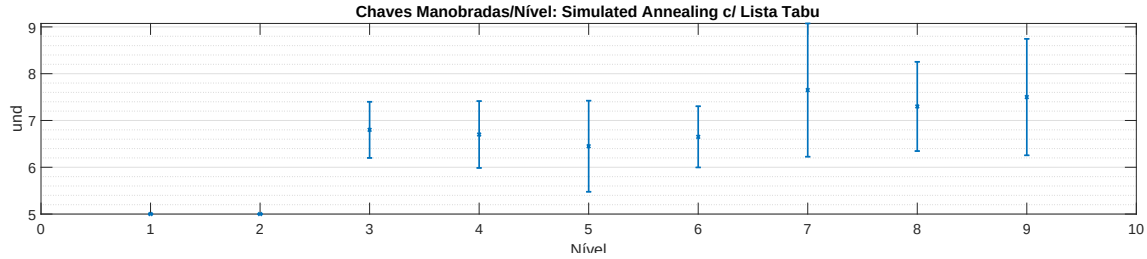

Figura 65: Chaves Manobradas *Simulated Annealing* c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

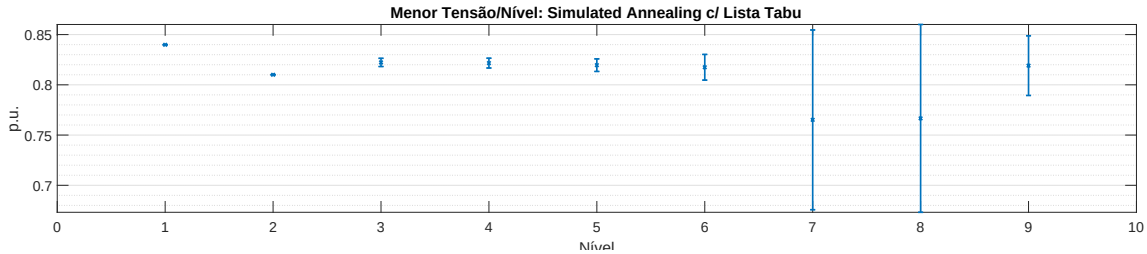

Figura 66: Menor Tensão *Simulated Annealing* c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

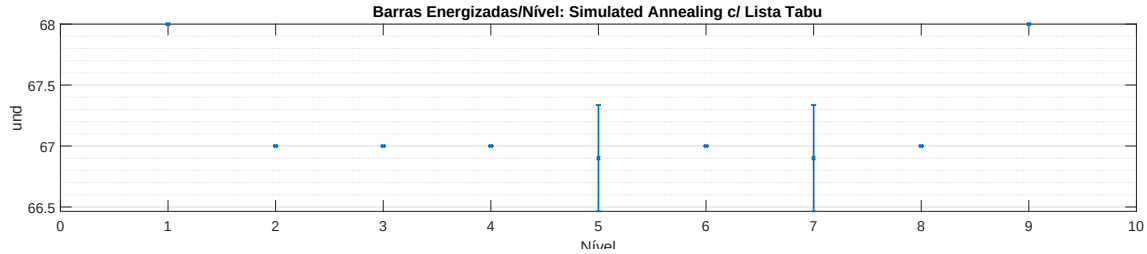

Figura 67: Barras Energizadas *Simulated Annealing* c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

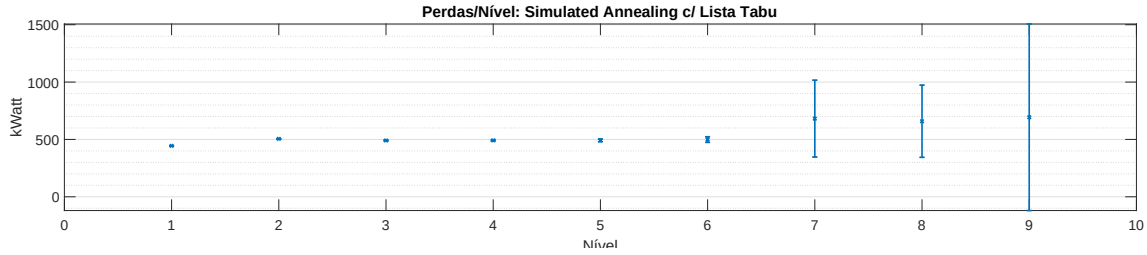

Figura 68: Perdas *Simulated Annealing* c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

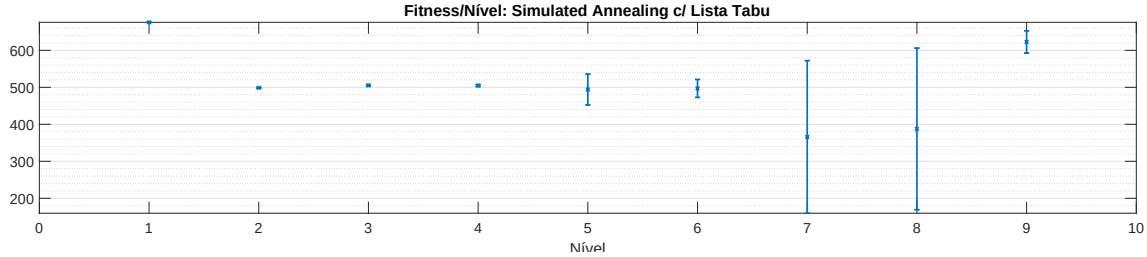

Figura 69: *Fitness Simulated Annealing* c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

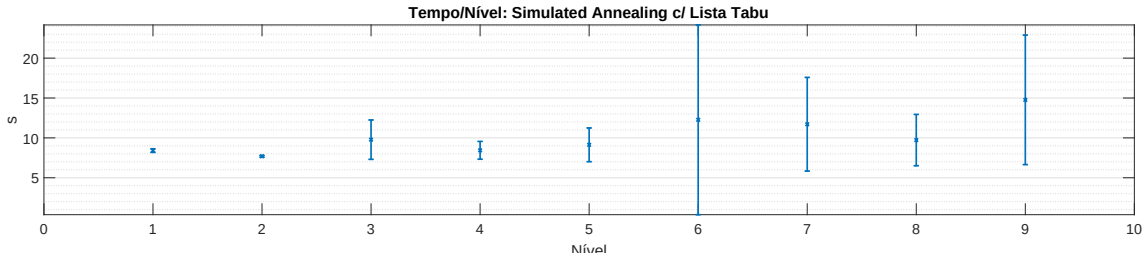

Figura 70: Tempo de execução *Simulated Annealing* c/ Lista Tabu - rede de Zidan e El-Saadany (2012)

Diante do apresentado, pode-se notar que a estratégia de otimização implementada atuou nos resultados de formas diferentes em cada parâmetro.

Ao comparar os resultados obtidos do algoritmo GRASP, pode-se notar que no nível 1 resultou em uma média de 3 chaves manobradas enquanto essa média sem a estratégia de otimização era de 4. Entretanto houve uma ligeira desvantagem para outros parâmetros e níveis. Ao aplicar a Lista Tabu nesse no GRASP, os resultados foram semelhantes aos resultados do GRASP convencional.

Ao analisar os resultados obtidos do *Simulated Annealing*, pôde-se notar que houve uma melhora significativa em várias variáveis do sistema, como minimização do número de manobras das chaves, minimização da perda de potência e maximização da menor tensão do sistema. Ao aplicar a Lista Tabu no SA, os resultados seguiram os padrões dos resultados do SA com uma minimização nos desvios padrão.

Com a variação dos níveis foi notado que houve um aumento no desvio padrão conforme se aumentou o nível, além da diminuição do tempo de execução ao diminuir o nível máximo, especificamente no SA.

6 CONCLUSÃO

Contudo, essa estratégia de otimização pode ser ainda mais refinada com o equacionamento dos critérios de parada e o equacionamento para estipular o nível máximo a ser considerado a partir de variáveis como quantidade de barras desenergizadas, prioridades das barras desenergizadas e quantidade de chaves a ser manobradas na área desenergizada. Essa técnica mostrou-se mais eficiente aplicada ao SA comparado à sua aplicação ao GRASP.

O problema de *self-healing* apresenta um vasto campo de aspectos em que se podem ser analisadas e estudadas. Além dos aspectos apresentados nesse trabalho, restam ainda várias outras janelas a serem exploradas, como, modelagem de um sistema equivalente para simplificação/diminuição do problema; utilização de outras heurísticas como a colônia de formigas; e até mesmo no aspecto físico da rede como sensores e acionadores para a construção de uma rede autônoma.

REFERÊNCIAS

AGENCY, International Energy. **Key World Energy Statistics**. [S.l.]: International Energy Agency, 2008.

ANDERSSON, Goran. **Power System Analysis**. [S.l.: s.n.], 2012.

BLUM, Christian; ROLI, Andrea. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 35, n. 3, p. 268–308, set. 2003. ISSN 0360-0300.

CALIFORNIA, Berkeley. Computer Science University of; KARP, R.M. **Reducibility Among Combinatorial Problems**. [s.n.], 1972. Disponível em: <https://books.google-.com.br/books?id=Pf3FXwAACAAJ>.

CARPANETO, Enrico; CHICCO, Gianfranco. Distribution system minimum loss reconfiguration in the hyper-cube ant colony optimization framework. **Electric Power Systems Research**, v. 78, n. 12, p. 2037 – 2045, 2008. ISSN 0378-7796. Special Issue Papers Presented at the 6th World Energy System Conference6th World Energy System Conference. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0378779608001909>.

CORMEN, T.H.; LEISERSON, C.E.; RIVEST, R.L.; STEIN, C. **Algoritmos: teoria e prática**. [S.l.]: CAMPUS - RJ, 2002. ISBN 9788535209266.

DAS, D. Reconfiguration of distribution system using fuzzy multi-objective approach. **International Journal of Electrical Power & Energy Systems**, Elsevier, v. 28, n. 5, p. 331–338, 2006.

DELGADO-GOMES, V.; MARTINS, J. F.; LIMA, C.; BORZA, P. N. Smart grid security issues. In: **2015 9th International Conference on Compatibility and Power Electronics (CPE)**. [S.l.: s.n.], 2015. p. 534–538. ISSN 2166-9538.

DEO, N. **Graph Theory with Applications to Engineering and Computer Science**. Dover Publications, 2016. (Dover Books on Mathematics). ISBN 9780486807935. Disponível em: <https://books.google.com.br/books?id=uk1KDAAAQBAJ>.

FERREIRA, Lucas Roberto; SIEBERT, Luciano Cavalcante; AYALA, Helon; AOKI, Alexandre Rasi; DIREITO, Luiz Carlos Menezes. Solução do problema de self-healing para redes de distribuição radiais através de otimização via algoritmo genético. **Simpósio Bras. Automação Intel**, 2013.

FINK, L. H.; LIOU, Kan-Lee; LIU, Chen-Ching. From generic restoration actions to specific restoration strategies. **IEEE Transactions on Power Systems**, v. 10, n. 2, p. 745–752, May 1995. ISSN 0885-8950.

GAREY, M.R.; JOHNSON, D.S. **Computers and Intractability: A Guide to the Theory of NP-completeness**. [S.l.]: W. H. Freeman, 1979. (Books in mathematical series). ISBN 9780716710448.

GLOVER, F.W.; KOCHENBERGER, G.A. **Handbook of Metaheuristics**. Springer US, 2006. (International Series in Operations Research & Management Science). ISBN 9780306480560. Disponível em: <https://books.google.com.br/books?id=P-HpBwAAQBAJ>.

GOLDBARG, M.C.; LUNA, H.P.L. **Otimização combinatária e programação linear: modelos e algoritmos**. [S.l.]: Campus, 2000. ISBN 9788535205411.

HARRIS, A. Smart grid thinking - [power super grid]. **Engineering Technology**, v. 4, n. 9, p. 46–49, May 2009. ISSN 1750-9637.

HOPCROFT, J.E.; ULLMAN, J.D.; MOTWANI, R. **Introdução a teoria dos autômatos, linguagens: e computação**. CAMPUS - RJ, 2002. ISBN 9788535210729. Disponível em: <https://books.google.com.br/books?id=jTKQPgAACAAJ>.

JIA, D.; MENG, X.; SONG, X. **Study on technology system of self-healing control in smart distribution grid**. [S.l.: s.n.], 2011. 26-30 p.

KAGAN, N.; OLIVEIRA, C.C.B. de; KAGAN, H. **Métodos de otimização aplicados a sistemas elétricos de potência**. [S.l.]: Blucher, 2009. ISBN 9788521204725.

KIRKPATRICK C. D. GELATT, M. P. Vecchi S. Optimization by simulated annealing. **Science**, American Association for the Advancement of Science, v. 220, n. 4598, p. 671–680, 1983. ISSN 00368075, 10959203.

KUMAR, Y.; DAS, B.; SHARMA, J. Multiobjective, multiconstraint service restoration of electric power distribution system with priority customers. **IEEE Transactions on Power Delivery**, v. 23, n. 1, p. 261–270, Jan 2008. ISSN 0885-8977.

LI, F.; QIAO, W.; SUN, H.; WAN, H.; WANG, J.; XIA, Y.; XU, Z.; ZHANG, P. **Smart Transmission Grid: Vision and Framework**. [S.l.: s.n.], 2010. 168-177 p. ISSN 1949-3053.

LUGER, George F. **Inteligência artificial George F. Luger: tradução Daniel Vieira; revisão técnica Andréa**. [S.l.]: Pearson Education do Brasil Ltda, 2013. ISBN 9788581435503.

MAMO, X.; MALLET, S.; COSTE, T.; GRENARD, S. Distribution automation: The cornerstone for smart grid development strategy. In: **2009 IEEE Power Energy Society General Meeting**. [S.l.: s.n.], 2009. p. 1–6. ISSN 1932-5517.

MCDERMOTT, T. E.; DREZGA, I.; BROADWATER, R. P. A heuristic nonlinear constructive method for distribution system reconfiguration. **IEEE Transactions on Power Systems**, v. 14, n. 2, p. 478–483, May 1999. ISSN 0885-8950.

MICHALEWICZ, Z.; FOGEL, D.B. **How to Solve It: Modern Heuristics**. [S.l.]: Springer Berlin Heidelberg, 2013. ISBN 9783662078075.

MOMOH, J. A. **Smart grid design for efficient and flexible power networks operation and control**. [S.l.: s.n.], 2009. 1-8 p.

NORVIG, P.; RUSSELL, S. **Inteligência Artificial: Tradução da 3a Edição**. Elsevier Brasil, 2015. ISBN 9788535251418. Disponível em: <https://books.google.com.br/books?id=BsNeAwAAQBAJ>.

PAPADIMITRIOU, C. H.; STEIGLITZ, K. **Combinatorial Optimization: Algorithms and Complexity**. [S.l.]: Dover Publications, 1998. (Dover Books on Computer Science Series). ISBN 9780486402581.

PEREZ-GUERRERO, R.; HEYDT, G. T.; JACK, N. J.; KEEL, B. K.; CASTELHANO, A. R. **Optimal Restoration of Distribution Systems Using Dynamic Programming**. [S.l.: s.n.], 2008. 1589-1596 p. ISSN 0885-8977.

RESENDE, Mauricio G. C.; FEO, Thomas A. **Greedy Randomized Adaptative Search Procedures**. Boston, MA, USA: Kluwer Academic, 1995. 109–133 p.

SIPSER, M. **Introdução à teoria da computação**. THOMSON PIONEIRA, 2007. ISBN 9788522104994. Disponível em: <https://books.google.com.br-/books?id=aS2aPgAACAAJ>.

SOUZA, Marcone Jamilson Freitas. **Inteligência Computacional para Otimização**. [S.l.]: Universidade Federal de Ouro Preto, 2009.

SOUZA, S. S. F.; ROMERO, R.; PEREIRA, J.; SARAIVA, J. T. Distribution system reconfiguration with variable demands using the opt-ainet algorithm. In: **2016 13th International Conference on the European Energy Market (EEM)**. [S.l.: s.n.], 2016. p. 1–5.

STEVENSON, W.D. **Elements of power system analysis**. McGraw-Hill, 1982. (McGraw-Hill series in electrical engineering: Power and energy). ISBN 9780070612792. Disponível em: <https://books.google.com.br/books?id=QOVSAAAAMAAJ>.

TALBI, E.G. **Metaheuristics: From Design to Implementation**. Wiley, 2009. (Wiley Series on Parallel and Distributed Computing). ISBN 9780470496909. Disponível em: <https://books.google.com.br/books?id=SIsa6zi5XV8C>.

TAO, L.; ZHAO, Y. C.; THULASIRAMAN, K.; SWAMY, M. N. S. An efficient tabu search algorithm for graph bisectioning. In: **[1991] Proceedings. First Great Lakes Symposium on VLSI**. [S.l.: s.n.], 1991. p. 92–95.

TOSCANI, L.V.; VELOSO, P.A.S. **Complexidade de Algoritmos: Série Livros Didáticos Informática UFRGS - Vol. 13**. Bookman, 2009. ISBN 9788540701397. Disponível em: <https://books.google.com.br/books?id=Ond1ls37VHwC>.

ZIDAN, A.; EL-SAADANY, E. F. **A Cooperative Multiagent Framework for Self-Healing Mechanisms in Distribution Systems**. [S.l.: s.n.], 2012. 1525-1539 p. ISSN 1949-3053.