UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

PROGRAMA DE PÓS GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RONNEESLEY MOURA TELES

Geração de Modelos Sintéticos de Topologia de Sistemas de Distribuição de Energia Elétrica

Goiânia
2022

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[ ] Dissertação [X] Tese [ ] Outro*:_____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Ronneesley Moura Teles

3. Título do trabalho

Geração de modelos sintéticos de topologia de sistemas de distribuição de energia elétrica

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [x] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;

- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Telma Woerle De Lima Soares**, **Professora do Magistério Superior**, em 20/06/2022, às 16:51, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **RONNEESLEY MOURA TELES**, **Discente**, em 20/06/2022, às 17:06, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2988722** e o código CRC **4EE61E41**.

Referência: Processo nº 23070.020918/2022-61 SEI nº 2988722

RONNEESLEY MOURA TELES

Geração de Modelos Sintéticos de Topologia de Sistemas de Distribuição de Energia Elétrica

Tese apresentada ao Programa de Pós–Graduação em Ciência da Computação, do Instituto de Informática, da Universidade Federal de Goiás, como requisito para obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação.

Linha de pesquisa: Sistemas Inteligentes e Aplicações.

Orientadora: Profa. Dra. Telma Woerle de Lima Soares

Co-Orientador: Dr. Marcos Henrique Marçal Camillo

Goiânia
2022

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Teles, Ronneesley Moura
Geração de Modelos Sintéticos de Topologia de Sistemas de Distribuição de Energia Elétrica [manuscrito] / Ronneesley Moura Teles. - 2022.
145, CXLV f.: il.

Orientador: Profa. Dra. Telma Woerle de Lima Soares; co orientador Dr. Marcos Henrique Marçal Camillo.
Tese (Doutorado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação, Goiânia, 2022.
Bibliografia.
Inclui siglas, símbolos.

1. Redes Elétricas. 2. Geração. 3. Algoritmos. 4. Computação Evolutiva. I. Soares, Telma Woerle de Lima, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

ATA DE DEFESA DE TESE

Ata nº **05** da sessão de Defesa de Tese de **Ronneesley Moura Teles**, que confere o título de Doutor em Ciência da Computação, na área de concentração em Ciência da Computação.

Aos vinte e três dias do mês de maio de dois mil e vinte e dois, a partir das oito e trinta horas, via sistema de webconferência da RNP, realizou-se a sessão pública de Defesa de Tese intitulada **"Geração da Topologia de Sistemas de Distribuição de Energia Elétrica Sintéticos"**. Os trabalhos foram instalados pela Orientadora, Professora Doutora Telma Woerle de Lima Soares (INF/UFG), com a participação dos demais membros da Banca Examinadora: Professor Doutor Marcos Henrique Marçal Camillo (Copel Holding), coorientador; Professor Doutor Rodrigo Zempulski Fanucchi (Copel-DISP), membro titular externo; Professor Doutor João Bosco Augusto London Junior (EESC/USP), membro titular externo; Professor Doutor Arlindo Rodrigues Galvão Filho (INF/UFG), membro titular interno; Professor Doutor Flavio Henrique Teles Vieira (EMC/UFG), membro titular interno. A realização da banca ocorreu por meio de videoconferência. Durante a arguição os membros da banca não fizeram sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Tese, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pela Professora Doutora Telma Woerle de Lima Soares, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos vinte e três dias do mês de maio de dois mil e vinte e dois.

TÍTULO SUGERIDO PELA BANCA

Geração de modelos sintéticos de topologia de sistemas de distribuição de energia elétrica.

Documento assinado eletronicamente por **Telma Woerle De Lima Soares**, **Professora do Magistério Superior**, em 23/05/2022, às 12:15, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **RODRIGO ZEMPULSKI FANUCCHI**, **Usuário Externo**, em 23/05/2022, às 12:16, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **João Bosco Augusto London Junior**, **Usuário Externo**, em 23/05/2022, às 12:16, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Arlindo Rodrigues Galvão Filho**, **Professor do Magistério Superior**, em 23/05/2022, às 12:16, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Marcos Henrique Marçal Camillo**, **Usuário Externo**, em 23/05/2022, às 12:16, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **RONNEESLEY MOURA TELES**, **Discente**, em 23/05/2022, às 12:19, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Flavio Henrique Teles Vieira**, **Professor do Magistério Superior**, em 23/05/2022, às 13:04, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2858929** e o código CRC **99EFE81C**.

Referência: Processo nº 23070.020918/2022-61 SEI nº 2858929

Ronneesley Moura Teles

Graduado em Sistemas de Informação na Universidade Salgado de Oliveira, Administração pela Universidade Paulista e Agronomia pelo Instituto Federal Goiâno, é especialista em Tecnologia da Informação e Negócios Eletrônicos pela Universidade Salgado de Oliveira, possui Mestrado em Ciência da Computação pela Universidade Federal de Goiás. Atuou como programador no Tribunal de Justiça de Goiás e atualmente é professor de computação pelo Instituto Federal Goiâno campus Ceres.

Dedico este trabalho a todas as pessoas de bem que anseiam que próximo atinja seu potencial e sonham por um mundo melhor, mais justo, igualitário e com oportunidade de crescimento para todos.

Em especial, decido este trabalho a minha família, que soube lidar com todas as consequências deste estudo, principalmente, a minha ausência em muitos momentos.

Do fundo do coração à minha esposa, Deusirene da Silva Sousa Teles, e minha filha, Rayanne Sousa Teles, por me apoiarem durante todos os momentos deste percurso.

A minha mãe, Maria Vilani Pereira de Moura, e ao meu irmão, Ronneery Moura Teles, pelas horas de debate sobre o tema e a importância da educação e estudo nesse país.

Agradecimentos

Agradeço a minha orientadora, Telma Woerle de Lima Soares, pela paciência em lidar com os meus questionamentos e por guiar a mim e este trabalho.

Agradeço ao Instituto Federal Goiano Campus Ceres pela oportunidade de fazer este estudo.

Agradeço a todos os pesquisadores que me antecederam neste tema permitindo que este trabalho fosse realizado.

Por fim, agradeço a Companhia Paranaense de Energia por fomentar pesquisas relacionadas ao tema deste trabalho através dos P&Ds 2866-0272/2013, 2866-0484/2017 e 2866-0504/2018.

Sábio é aquele que conhece os limites da própria ignorância.

Socrates,

.

Resumo

Teles, Ronneesley Moura. **Geração de Modelos Sintéticos de Topologia de Sistemas de Distribuição de Energia Elétrica**. Goiânia, 2022. 146p. Tese de Doutorado Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

O desenvolvimento de algoritmos para resolução de problemas relacionados às redes elétricas sempre esbarrou no acesso aos dados, uma vez que, por questões de segurança e sigilo, os analistas não conseguem obter dados reais destas redes. Este trabalho surge a fim de mitigar esse problema, permitindo que estes pesquisadores gerem redes sintéticas com características próximas às redes reais. Para isso, foram desenvolvidos algoritmos de geração de florestas com várias árvores enraizadas com base nas características de redes elétricas reais. Estes algoritmos são guiados durante o processo de busca através das distribuições topológicas desejadas, permitindo que sejam usadas características de redes reais de qualquer estado brasileiro ou regiões do mundo. Este estudo utilizou como base as redes elétricas do estado do Paraná. Cada algoritmo proposto foi estudado em relação as suas tendências de geração de árvores permitindo uma melhor compreensão do seu comportamento. Estes algoritmos foram empregados na geração de redes elétricas através de um processo de computação evolutiva multiobjetivo, usando o método NSGA-II, e resultaram em redes com distribuição de graus, número de barras por alimentador e número de folhas por alimentador similares as redes reais estudadas. Além disso, determinou-se um método de posicionamento dos consumidores e foi desenvolvido um algoritmo genético mono-objetivo para o posicionamento ideal das chaves normalmente fechadas e normalmente abertas. Este foi capaz de avaliar faltas em todas as barras de uma rede elétrica em uma única busca em profundidade através do uso de técnicas de programação dinâmica. Ao utilizar os algoritmos e metodologia desenvolvidos nesta tese o pesquisador terá uma topologia contendo alimentadores, barras, trechos, chaves normalmente abertas e normalmente fechadas, e número de consumidores nas barras à sua disposição para seus estudos.

Palavras–chave

Redes Elétricas, Geração, Algoritmos, Computação Evolutiva

Abstract

Teles, Ronneesley Moura. **Topology Generation of Synthetic Electric Power Distribution Systems**. Goiânia, 2022. 146p. PhD. Thesis Relatório de Graduação. Instituto de Informática, Universidade Federal de Goiás.

The development of algorithms to solve problems related to electrical networks has always faced access to data, since, for reasons of security and secrecy, analysts are unable to obtain real data from these networks. This work arises in order to mitigate this problem, allowing these researchers to generate synthetic networks with characteristics close to real networks. For this, algorithms were developed to generate forests with several rooted trees based on the characteristics of real electrical networks. These algorithms are guided during the search process through the desired topological distributions, allowing the use of characteristics of real networks from any Brazilian state or regions of the world. This study was based on the electrical networks of the state of Paraná. Each proposed algorithm was studied in relation to its tree generation trends allowing a better understanding of its behavior. These algorithms were employed in the generation of electrical networks through a multiobjective evolutionary computation process, using the NSGA-II method, and resulted in networks with degree distribution, number of buses per feeder and number of leaves per feeder similar to the real networks studied. In addition, a method for positioning consumers was determined and a single-objective genetic algorithm was developed for the ideal positioning of normally closed and normally open switches. It was able to evaluate faults in all the buses of an electrical network in a single depth search through the use of dynamic programming techniques. By using the algorithms and methodology developed in this thesis, the researcher will have a topology containing feeders, buses, sections, normally open and normally closed switches, and the number of consumers in the buses at his disposal for his studies.

Keywords

Electrical networks, Generation, Algorithms, Evolutionary Computing

Siglas

AG Algoritmo Genético. 43, 44, 97, 98, 100, 104, 133, 134, 137, 138
ANOVA análise de variância. 102

BIC Bayesian Information Criterion. 30, 31

CE Computação Evolutiva. 22, 42–44, 46, 49
COPEL Companhia Paranaense de Energia. 20
CSA Clonal Selection Algorithm. 35, 36
CV Characteristic Vector Encoding. 46, 47

DIC Delineamento Inteiramente Casualizado. 102, 103, 107, 109, 110, 115
DSCRG Dual-stage constructed random graph. 26–28, 46

F1 primeira fronteira de Pareto. 96, 127

GMM modelo de mistura gaussiana. 30, 31
GNLG Geographical Network Learner and Generator. 30, 46
GRASP Busca Gulosa Aleatorizada e Adaptativa. 44
gSDE gerador de SDE. 134, 135, 137

IEEE Instituto de Engenheiros Eletricistas e Eletrônicos. 19, 23
IMEG indivíduo de menor erro geral. 120, 121, 127, 128

LNB Link and Node Biased Encoding. 46, 47

MST árvore geradora mínima. 30, 47, 49

NPE node-depth phylogenetic-based encoding. 47–49, 55, 77, 81–85, 87, 91, 93, 110, 120, 137
NSGA-II Non-Dominated Sorting Genetic Algorithm II. 44–46, 49, 78, 95, 96, 102, 104, 127, 137, 138

P&D Pesquisa e Desenvolvimento. 50
PDF função de densidade de probabilidade. 24, 33
PMX Partially Mapped Crossover. 101, 102

PSPR Probabilistic Subtree Pruning and Regrafting. 91–94, 96, 103–105, 117, 118, 138

PSPRN Probabilistic Subtree Pruning and Regrafting on NPE. 93, 94

PTBR Probabilistic Tree Bisection and Reconnection. 91, 92, 94, 96, 103–105, 117, 118, 138

PTBRN Probabilistic Tree Bisection and Reconnection on NPE. 94

RNP Representação Nó-Profundidade. 47, 48, 90

SDE Sistema de Distribuição de Energia Elétrica. 18–21, 26, 49, 50, 54, 56, 57, 60, 61, 63, 75–77, 89, 90, 95, 97, 104, 107, 121, 125, 134, 135, 137–139

SDNG Spatially Distributed Nodes Generator. 30

SPR Subtree Pruning and Regrafting. 47, 91, 92, 96, 104, 117, 118, 137

SPRN Subtree Pruning and Regrafting on NPE. 47–49, 77, 91, 94, 118, 126

STE Sistema de Transmissão de Energia Elétrica. 23, 24, 26

TBR Tree Bisection and Reconnection. 47, 91, 92, 96, 104, 117, 118, 137

TBRN Tree Bisection and Reconnection on NPE. 47–49, 77, 91, 92, 118, 126

TWST Tunable Weight Spanning Tree. 30, 31

UFG Universidade Federal de Goiás. 20

Lista de Símbolos

$C(G)$ coeficiente de clusterização. 24, 32

$C(R)$ coeficiente de clusterização de um grafo aleatório. 24

R^2 coeficiente de determinação. 69, 96, 112

$W(\mathbb{T})$ Bus Type Entropy. 33, 39

$\langle k \rangle$ Grau médio dos vértices. 23–25, 27, 28, 32, 41

$\langle l \rangle$ Comprimento médio de um caminho. 23

ρ coeficiente de correlação de Pearson dos graus dos vértices adjacentes. 24

$\underline{k}$ vetor dos graus dos vértices. 23

m número de arestas. 23, 24, 33, 35, 36, 40–42, 57, 81

n número de vértices. 23–25, 27, 28, 33, 35–37, 40–42, 47, 57, 81, 94, 95

$r(k_i > \overline{k})$ proporção de graus maiores que o grau médio de uma aresta aleatória. 23

r coeficiente de correlação de Pearson. 69, 71–73, 96, 97, 112

Sumário

1 Introdução **18**
1.1 Objetivos 19
1.2 Contribuições realizadas 20
1.3 Organização 20

2 Fundamentação Teórica **22**
2.1 Geração de redes elétricas 22
2.2 Geração de topologias de redes elétricas 25
2.2.1 Dual-stage constructed random graph (DSCRG) 26
2.2.2 Geographical Network Learner and Generator (GNLG) 30
2.3 Atribuição de dados elétricos 32
2.3.1 Tipos de barras 32
2.3.2 Geração e carga 37
2.3.3 Outras informações relevantes 40
2.4 Grafos aleatórios 40
2.4.1 Erdős e Rényi 40
2.4.2 Watts e Strogatz 41
2.4.3 Albert e Barabási 41
2.4.4 GraphCuisine 42
2.5 Computação Evolutiva 43
2.6 Geração de árvores com computação evolutiva 46
2.7 Considerações finais 49

3 Sistemas de Distribuição de Energia Elétrica e Análise Estatística **50**
3.1 Elementos de um SDE 50
3.1.1 Postes 50
3.1.2 Trechos 51
3.1.3 Chaves 52
3.1.4 Postos transformadores 52
3.1.5 Banco de Capacitores 53
3.1.6 Reguladores de tensão 53
3.1.7 Alimentadores 54
3.2 A modelagem do SDE 54
3.3 Análise estatística dos SDEs 56
3.3.1 Análise descritiva 57
Análises do estado do Paraná 57
Análises segmentadas por tipo de alimentador 62
3.3.2 Análise de regressão 69

Número de consumidores pelo número de barras em um alimentador 69
Determinação do número de consumidores residenciais, comerciais e industriais em alimentadores urbanos 71
Número de chaves pelo tamanho do alimentador 72
Determinação da potência 73

4 Gerador de SDE (gSDE) **75**
4.1 Geração da topologia 76
4.1.1 Estratégia da busca de topologias 76
4.1.2 Busca e função de *fitness* 77
4.1.3 Inicialização 79
Algoritmo de inicialização 79
Algoritmo de construção de árvores 81
Geração de árvores com grau máximo 84
Geração de árvores com profundidade máxima 86
Geração de árvore por composição de floresta em representação de grafos 88
4.1.4 Operadores de *crossover* 89
Crossover por tamanho da árvore 89
Crossover por *locus* 90
4.1.5 Mutação 91
SPR e TBR mantendo a raiz do alimentador com grau 1 91
Operadores probabilísticos de poda 92
4.1.6 Visão geral do gerador de topologias 95
4.2 Posicionamento dos consumidores 96
4.3 Posicionamento das chaves 97
4.3.1 Cálculo do *fitness* 98
4.3.2 Representação 100
4.3.3 Inicialização, *crossover* e mutação 101
4.4 Experimentos 102
4.4.1 Amostrador não tendencioso no espaço de busca 102
4.4.2 Direcionamento correto da tendência da inicialização 103
4.4.3 Tendência dos operadores de geração de árvores 103
4.4.4 Observação do tempo de geração de árvores com o NPE 103
4.4.5 Inserção de tendência nos operadores PSPR e PTBR 103
4.4.6 Busca topológica 104
4.4.7 Comparação do posicionamento das chaves na rede elétrica real 104

5 Resultados e Discussão **105**
5.1 Análise do espaço de busca 105
5.2 Amostrador não tendencioso 107
5.3 A tendência da inicialização 109
5.4 Análise de tendência dos operadores de geração de árvores 110
5.4.1 Algoritmo de construção de árvores 110
5.4.2 Algoritmo de construção de árvores com grau máximo 113
5.4.3 Algoritmo de construção de árvores com profundidade máxima 114
5.4.4 Tendência na geração de árvores por composição de floresta 115
5.5 Inserção de tendência com o PSPR e PTBR 117
5.6 Avaliação do processo de busca topológica do gSDE 118

5.7 Busca sem o número de barras por alimentador como objetivo 124
5.8 Comparação das configurações com 2 e 3 objetivos 130
5.9 Posicionamento de chaves 133
5.10 Comparação do posicionamento das chaves 134

6 Conclusões e trabalhos futuros **137**
6.1 Contribuições realizadas 138
6.2 Trabalhos futuros 138

Referências Bibliográficas **140**

CAPÍTULO 1

Introdução

Os Sistemas de Distribuição de Energia Elétrica (SDEs) se tornaram essenciais para as atividades modernas, uma vez que, grande parte da produção humana de bens e serviços demandam deles. No entanto, a manutenção desses sistemas em perfeito funcionamento exige, cada vez mais, melhorias para o aprimoramento da qualidade do serviço. No entanto, os pesquisadores desta área esbarram em problemas que acabam por adiar a evolução tecnológica, são eles: acesso aos dados e reprodutibilidade dos resultados.

Mesmo lidando com problemas diferentes, todos os pesquisadores precisam primeiramente da topologia e dos metadados do problema do SDE em pesquisa. E, para isso, no Brasil, é preciso acessar um dos bancos de dados privados de uma das 63 concessionárias de energia [2]. Como esses dados são e devem ser protegidos visando a segurança da rede e privacidade dos usuários, seu acesso se torna muito difícil ou praticamente nulo, mesmo quando a finalidade é o desenvolvimento de algoritmos para resolução dos problemas inerentes.

Poucos são aqueles que conseguem este acesso, após muito tempo de esforço, e desenvolvem suas pesquisas, tais como: a detecção de vulnerabilidades da rede [16], a descentralização dos centros de produção [28, p. 428], a detecção e correção de problemas elétricos [38] e o reestabelecimento da energia através de novos roteamentos [20, 19]. Apesar de poucos trabalhos existentes na literatura, os mesmos mostram a importância das pesquisas na área, a permissão de acesso para os pesquisadores e, principalmente, a inovação tecnológica que acontece quando estas pesquisas são concluídas.

Apesar de tudo, mesmo aqueles pesquisadores que conseguem superar essa primeira barreira, acabam esbarrando, mais tarde, em um segundo problema que é a reprodutibilidade dos resultados de sua pesquisa. A reprodutibilidade é uma etapa importante para aumentar a confiabilidade do produto da pesquisa, mas devido ao sigilo dos dados, os pesquisadores que poderiam validar os resultados acabam por não o fazer.

Essa dificuldade de acesso à modelos de sistemas elétricos reais, não é uma realidade exclusiva do Brasil. Assim, para transpor esta dificuldade, alguns pesquisadores e instituições tem dedicado seus estudos na geração de modelos sintéticos que tentam

reproduzir as características dos sistemas elétricos de distribuição e/ou transmissão reais. Estes esforços podem ser divididos em dois métodos: a geração manual de modelos, como os casos de teste do Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) [51], e a geração automática de modelos [64, 31, 54].

Ambos os métodos apresentam entraves próprios, sendo difícil determinar qual é o melhor deles. Enquanto os maiores modelos criados manualmente apresentam poucos milhares de barras, os gerados automaticamente podem conter milhões, se assimilando aos modelos reais. No entanto, enquanto os modelos criados manualmente podem preservar várias características estatísticas, os gerados automaticamente apresentam poucos destes atributos. É importante observar que, segundo Schneider et al [51] vários modelos da IEEE foram criados com o propósito de pôr à prova os sistemas que calculam o fluxo de potência, e não com o objetivo de representar fielmente a estrutura de uma SDE real.

Vale ressaltar que, os sistemas elétricos de distribuição e/ou transmissão são redes complexas constituídas de diversos equipamentos, levando a várias condições elétricas que devem ser meticulosamente analisadas. E, por isso, é importante avaliar a eficácia e eficiência de algoritmos desenvolvidos em ambientes reais ou similares a eles, pois não considerar todos os aspectos dos sistemas elétricos reais pode resultar em prejuízos que podem abranger desde queima de equipamentos até mesmo perdas de vidas [62, p. 28].

É justamente na geração automática de modelos de SDE que este trabalho se encontra. Atualmente, as melhores soluções neste segmento consistem na análise das distribuições estatísticas de variáveis destes modelos sintéticos, como os modelos da IEEE, e reprodução em modelos aleatórios de maior escala, observando os aspectos topológicos e elétricos [64, 62, 28]. No entanto, a grande maioria destes algoritmos são fortemente estocásticos e observam apenas o grau médio dos vértices de uma rede elétrica [31]. Assim vários aspectos são desconsiderados, como o número de folhas, excentricidade, raio, entre outros. Até mesmo as próprias variações de grau entre os vértices acabam se perdendo neste processo.

1.1 Objetivos

O objetivo geral deste trabalho é a criação de uma nova abordagem de geração automática de redes elétricas através de técnicas de inteligência artificial afim de criar cenários realistas que podem ser empregados para resolução de diversos problemas relacionados a essas redes. Para que esse objetivo seja alcançado, os objetivos específicos a seguir foram definidos:

1. Caracterizar, quanto a topologia, os modelos de larga escala com acesso restrito;

2. Identificar o estado da arte da geração de árvores utilizando computação evolutiva;
3. Gerar árvores com características topológicas similares às dos SDEs utilizando computação evolutiva;
4. Caracterizar chaves seccionadoras e utilizar este resultado na criação de chaves que liguem barras em uma árvore e até mesmo árvores diferentes;
5. Avaliar quantitativamente a similaridade dos modelos gerados pelo sistema com as características desejadas.

Nas análises realizadas nesta tese foram consideradas condições ideais com fornecimento de energia sem limitações elétricas. Portanto, não fez parte deste trabalho a geração de parâmetros elétricos da rede como: bitola e resistência de cabos, voltagem dos geradores, carga consumida por hora do dia, etc.

1.2 Contribuições realizadas

Como resultado deste trabalho obteve-se um conjunto de ferramentas capaz de gerar modelos sintéticos de SDE similares aos sistemas elétricos reais. Como consequência, é possível obter um ambiente realista e propício para a experimentação de novas soluções para os problemas relacionados aos sistemas elétricos de distribuição.

Deste modo, foi desenvolvida uma nova forma guiada de geração de grafos que pode ser utilizada em várias áreas, incluindo na síntese de SDEs. Além disso, foi criada uma metodologia de alocação de consumidores para que fosse possível a alocação das chaves visando minimizar o número de consumidores sem energia após o isolamento da falha na rede elétrica.

Como resultado secundário dos objetivos propostos obteve-se novos operadores para geração de árvores de grafos em metaheurísticas. Outra contribuição foi a criação de métodos específicos para composição de sistema de distribuição de energia elétrica a partir de várias árvores enraizadas.

Todas estas contribuições unidas, criam um ambiente propício para pesquisas e comparação de resultados entre pesquisadores, uma vez que, estes poderão comparar seus resultados em uma mesma situação. Isto, também contribui largamente para os estudos realizados nos projetos de pesquisa e desenvolvimento da Companhia Paranaense de Energia (COPEL) em parceria com a Universidade Federal de Goiás (UFG).

1.3 Organização

Este capítulo expôs a motivação, os objetivos e um resumo das contribuições deste trabalho. O capítulo 2 descreve as principais técnicas e os métodos para geração

de redes elétricas, além de apresentar as representações e os operadores relacionados a geração de árvores com computação evolutiva. O capítulo 3 apresenta os elementos de um SDE, a modelagem computacional utilizada neste trabalho e a análise estatística do SDE do estado do Paraná cujas características serviram de alicerce para as buscas. O capítulo 4 apresenta a metodologia e as ferramentas utilizadas para geração das redes sintéticas e os operadores propostos. Além disso, apresenta o plano dos experimentos que permitiu afirmar propriedades sobre os operadores propostos. O capítulo 5 apresenta os resultados e a discussão dos experimentos realizados neste trabalho. E, por fim, o capítulo 6 apresenta as conclusões deste trabalho e sugestões de trabalhos futuros relacionados ao tema.

CAPÍTULO 2

Fundamentação Teórica

Este capítulo apresenta os conceitos existentes sobre a geração de redes elétricas e, para isto, os trabalhos relacionados a este tema foram agrupados em duas categorias. A mais específica abrange a geração de topologias, incluindo a representação das redes elétricas em grafos, e os dados elétricos, como impedância, potência ativa e reativa, magnitude e ângulo de tensão, totalizando três seções 2.1, 2.2 e 2.3. Na seção 2.1 é apresentado o algoritmo *RT-nested-smallworld* capaz de gerar a estrutura da rede e seus parâmetros elétricos. Já a seção 2.2 apresenta algoritmos capazes de gerar somente a topologia da rede. Por fim, a seção 2.3 mostra como o grupo responsável pelo *RT-nested-smallworld* estudou e realizou a atribuição dos parâmetros elétricos em seu algoritmo.

Já a categoria mais geral envolve a geração de grafos com características desejadas compondo três seções. Na seção 2.4 são apresentados os métodos tradicionais para criação de grafos aleatórios. Na seção 2.5 são apresentados os principais conceitos de computação evolutiva e otimização multiobjetivo utilizados neste trabalho. Na seção 2.6 são apresentadas técnicas de Computação Evolutiva (CE) empregadas para gerar grafos e árvores. Por fim, na seção 2.7 são apresentadas as considerações finais.

2.1 Geração de redes elétricas

Baseado no entendimento que uma rede elétrica é composta tanto por sua topologia quanto por seus dados elétricos, apenas o algoritmo *RT-nested-smallworld* apresentou síntese simultânea de ambas informações. Este algoritmo foi criado devido ao problema da indisponibilidade de dados de redes elétricas de transmissão pelos pesquisadores Wang, Thomas e Scaglione [64].

A premissa fundamental deste trabalho é que conhecer as distribuições de cada característica das redes é o passo fundamental para a sua geração. Por este motivo grande parte do processo de aleatorização foi baseado nos resultados de estudos estatísticos.

O grupo de pesquisa do *RT-nested-smallworld* define a rede elétrica usando uma matriz de incidência [64, 61, 62, 11], de m arestas por n vértices, conforme a equação 2-1. Embora seja uma matriz de incidência, sua definição difere da encontrada na literatura

como em Nicoletti e Hruschka Jr. [39], em que a matriz de incidência possui dimensões de n vértices por m arestas, conforme a equação 2-2.

$$A(i,j)_{RT} = \begin{cases} 1 & \text{Se a } i\text{-ésima aresta inicia do } j\text{-ésimo vértice} \\ -1 & \text{Se a } i\text{-ésima aresta termina no } j\text{-ésimo vértice} \\ 0 & \text{Outros casos} \end{cases} \quad (2\text{-}1)$$

$$A(i,j) = \begin{cases} 0 & \text{se } v_i \text{ não for uma extremidade de } e_j \\ 1 & \text{se } v_i \text{ for uma extremidade de uma aresta não-loop } e_j \\ 2 & \text{se } v_i \text{ for uma extremidade de um } \textit{loop}\ e_j \end{cases} \quad (2\text{-}2)$$

Onde, v_i representa o i-ésimo vértice e_j a j-ésima aresta. Embora muito parecidas, as duas definições mudam as dimensões da matriz e seus valores. Como consequência, a obtenção de qualquer coeficiente é diferente das definições tradicionais.

Os estudos do grupo utilizaram como base das análises os modelos de Sistemas de Transmissão de Energia Elétrica (STEs) como os da IEEE. Suas pesquisas tiveram como objetivo entender o relacionamento do número de vértices (n) e do número de arestas (m) com os seguintes coeficientes [64, 62]:

- **Comprimento médio de um caminho ($\langle l \rangle$)**: onde l_{ij} é o número de saltos do menor caminho do vértice i para o j.

$$\langle l \rangle = \frac{2}{n \times (n-1)} \times \sum_{(i,j)} l_{ij} \quad (2\text{-}3)$$

- **Grau médio dos vértices ($\langle k \rangle$)**: onde k_i é o grau o i-ésimo vértice.

$$\langle k \rangle = \frac{1}{n} \sum_i k_i \quad (2\text{-}4)$$

- **O vetor dos graus dos vértices ($\underline{k}$)**:

$$\underline{k} = [k_1, k_2, k_3, \cdots, k_n] \quad (2\text{-}5)$$

- **A proporção de graus maiores que o grau médio de uma aresta aleatória ($r(k_i > \overline{k})$)**:

$$\overline{k} = (2m)^{-1} \sum_{(i,j)} (k_i + k_j) = (2m)^{-1} \sum_i (k_i)^2 \quad (2\text{-}6)$$

$$r(k_i > \overline{k}) = \frac{||k_i : k_i > \overline{k}||}{n} \quad (2\text{-}7)$$

- **O coeficiente de clusterização ($C(G)$)**: onde $\lambda G(i)$ é o número de arestas entre o i-ésimo vértice e seus vizinhos e $\tau G(i)$ é o número arestas que poderiam existir entre o i-ésimo vértice e seus vizinhos.

$$C(G) = \frac{1}{n}\sum_{i=1}^{n} C_i \tag{2-8}$$

$$C_i = \frac{\lambda G(i)}{\tau G(i)} \tag{2-9}$$

- **O coeficiente de clusterização de um grafo aleatório ($C(R)$)**: é a probabilidade de selecionar aleatoriamente m arestas entre todas as possíveis.

$$C(R) = \frac{2m}{n \times (n-1)} = \frac{\langle k \rangle}{n-1} \tag{2-10}$$

- **O coeficiente de correlação de Pearson dos graus dos vértices adjacentes (ρ)**: determina a correlação entre o grau dos dois vértices adjacentes existentes em uma aresta.

$$\rho = \frac{\sum_{(i,j)}(k_i - \bar{k}) \times (k_j - \bar{k})}{\sqrt{\sum_{(i,j)}(k_i - \bar{k})^2 \times \sum_{(i,j)}(k_j - \bar{k})^2}} \tag{2-11}$$

Wang, Thomas e Scaglione [62] notaram que o grau médio dos vértices ($\langle k \rangle$) não aumenta conforme o tamanho do grafo (n e m) aumenta e concluíram que os grafos de redes elétricas são esparsos ficando com grau médio entre 2,5 e 3,0. Quanto ao coeficiente de clusterização, observaram que os valores para os modelos sintéticos são bem superiores aos encontrados em grafos aleatórios, criados pelo algoritmo de Erdős e Rényi [24], de mesmo tamanho. Além disto, observaram que a função de densidade de probabilidade (PDF) dos graus dos vértices quando aplicada uma transformação logarítmica $log(p(k))$ se aproxima de uma linha na qual $p(k) = exp(-k/\gamma)/\gamma$ é a probabilidade de um vértice estar ligado a outros k vértices, dado que $\gamma = \langle k \rangle$ [53].

Segundo Solé et al. [53] a distribuição dos graus dos STEs europeus é exponencial. No entanto, Wang, Scaglione e Thomas [62] notaram que a cauda desta mesma distribuição para os modelos sintéticos se assemelha da distribuição geométrica, observando que, para vértices com grau menor ou igual a 3, esta associação não é válida. Portanto, eles representaram esta distribuição como uma distribuição mista através da junção de uma distribuição geométrica truncada com uma distribuição discreta irregular.

Após estas observações, a solução dada consistiu em dividir o problema em duas partes, a primeira consiste na geração da topologia e a segunda em atribuir dados elétricos à topologia gerada. A geração de topologias é feita em três etapas:

1. **Criação dos vértices**: posicionamento de n vértices dentro de uma área fixada utilizando uma distribuição aleatória (uniforme ou Poisson);
2. **Criação de arestas**: são criadas arestas para cada vértice respeitando um grau médio desejado $\langle k \rangle$. Dentre os $n-1$ possíveis vértices vizinhos são considerados somente aqueles cujo comprimento esteja dentro do limite de distância mínima e máxima esperada, ou seja, $d_{min} \leq d \leq d_{max}$;
3. **Verificação de conectividade**: se o grafo é conexo então a busca é encerrada, caso contrário volta ao passo 1.

A atribuição dos dados elétricos também foi dividida em três etapas:

1. **Atribuição de impedância às linhas de transmissão**: assume-se que a impedância de cada linha de transmissão é proporcional ao seu comprimento adicionando-se uma pequena variação aleatória $Z_{pr} = Z_0 \cdot L_{line} + \varepsilon_Z$, onde $Z_0 = r + jx$ é a impedância por unidade de comprimento e ε_Z é uma variável aleatória limitada por $[-\varepsilon_{Z_0}, +\varepsilon_{Z_0}]$;
2. **Atribuição das cargas**: para as cargas de cada barra $(P_l + jQ_l)$, valores da potência ativa e reativa são amostrados uniformemente entre os limites $[P_{lmin}, P_{lmax}]$ e $[Q_{lmin}, Q_{lmax}]$;
3. **Determinação dos geradores**: é utilizada a probabilidade p_{gen} para escolha das barras que serão consideradas geradoras. A voltagem E de cada um destes geradores é escolhida uniformemente dentro dos limites $[E_{min}, E_{max}]$ e seu ângulo δ é escolhido em uma distribuição normal de média m_δ e desvio-padrão de σ_δ. O primeiro gerador escolhido é definido como barramento de folga com magnitude de tensão M escolhida uniformemente entre $[M_{min}, M_{max}]$ e impedância entre o limite $[X_{dmin}, X_{dmax}]$.

Ao longo do tempo novos trabalhos envolvendo estudos relacionados às características elétricas foram adicionando seus resultados ao algoritmo. Estes serão mencionados e detalhados na seção 2.3.

2.2 Geração de topologias de redes elétricas

Esta seção apresenta os trabalhos relacionados a geração de topologias similares às observadas em redes elétricas reais. Neste trabalho, o termo topologia será utilizado para referenciar o posicionamento dos elementos de uma rede elétrica e suas respectivas ligações por meio de grafos.

2.2.1 Dual-stage constructed random graph (DSCRG)

O Dual-stage constructed random graph (DSCRG) é um algoritmo proposto por Ma, Yu e Zhao [31] para geração de STEs que objetivou resolver algumas deficiências do *RT-nested-smallworld*. Segundo estes pesquisadores os algoritmos tradicionais para geração de grafos como Erdös-Rényi [24], Watts e Strogatz [65] e até mesmo o *RT-nested-smallworld* sofrem de várias deficiências:

1. As topologias geradas não são necessariamente conexas[1];
2. O grau médio dos vértices não é precisamente controlado;
3. As topologias geradas diferem quanto a distribuição de graus dos vértices em relação as redes elétricas reais;
4. O número de componentes conexos nas topologias geradas é sempre igual a 1.

De acordo com eles, as redes elétricas são criadas com topologias do tipo árvore e com o tempo novas linhas e chaves são adicionadas para incrementar a confiabilidade dos SDEs. Neste sentido, o DSCRG é apresentado como um algoritmo que gera uma árvore (algoritmo 2.1) e adiciona novas arestas para atingir o grau médio desejado para os vértices (algoritmo 2.2).

Algoritmo 2.1: $DSCRG(n, \langle k \rangle)$, adaptado de Ma, Yu e Zhao [31].

Entrada: número de vértices desejados n, grau médio dos vértices desejado $\langle k \rangle$.

Saída: matriz de adjacências $A(n \times n)$ e $\langle k \rangle$ grau médio.

$A \leftarrow$ matriz inteira de $n \times n$ preenchida com zeros
$x \leftarrow rand(1,n)$ /* Sorteia um vértice */
$X \leftarrow \{x\}$ /* Vértices na árvore */
$Y \leftarrow \{1,2,...,n\} - X$ /* Vértices a serem adicionados */
enquanto $Y \neq \varnothing$ **faça**
    $x \leftarrow rand(X)$
    $y \leftarrow rand(Y)$ /* Sorteia um vértice de cada conjunto */
    $A(x,y) \leftarrow A(y,x) \leftarrow 1$ /* Conecta x e y */
    $Y \leftarrow Y - \{y\}$
    $X \leftarrow X \cup \{y\}$
fim
return $DSCRG - etapa2(A, n, \langle k \rangle)$

[1] Em um grafo conexo é possível ir de qualquer vértice do grafo para qualquer outro através das arestas do grafo [39].

O algoritmo 2.1 inicializa uma matriz de adjacência A na linha 1. Em seguida, nas linhas 2 a 4, um dos vértices é selecionado (x) para compor a árvore, separando os vértices que já estão na árvore (X) dos que ainda não estão (Y). Nas linhas 5 a 11, são selecionados dois vértices (linhas 6 e 7), um que já está na árvore (x) e outro que não (y), ambos são ligados por uma aresta (linha 8) e os conjuntos X e Y são atualizados (linhas 9 e 10). Este processo garante que não haja ciclos, o grafo seja conexo e que existam apenas $n-1$ arestas, formando a árvore desejada.

Algoritmo 2.2: $DSCRG-etapa2(A,n,\langle k\rangle)$, adaptado de Ma, Yu e Zhao [31].

Entrada: matriz de adjacências A, número de vértices desejados n, grau médio dos vértices desejado $\langle k\rangle$.

Saída: matriz de adjacências $A(n\times n)$ com n vértices e $\langle k\rangle$ grau médio.

```
1  S ← ⟨k⟩·n/2 − (n − 1)                    /* Quantidade de novas arestas */
2  R ← n·(n−1)/2 − (n − 1)                  /* Número máximo de arestas opcionais */
3  i ← 1, j ← 2                             /* i e j representam diferentes vértices */
4  repita
5      se A(i, j) = 0 então
6          θ ← rand()                       /* Número aleatório entre [0,1] */
7          se θ < S/R então
8              A(i, j) ← A(j, i) ← 1        /* Conecta os vértices i e j */
9              S ← S − 1
10         fim
11         R ← R − 1
12         se S = 0 então
13             return A
14         fim
15     fim
16     i ← i + 1, j ← j + 1
17 até i = n − 1
18 return A
```

A segunda etapa do método DSCRG (algoritmo 2.2), responsável por adicionar o número correto de arestas na árvore para obter um grafo com grau médio $\langle k\rangle$, inicia no último comando do algoritmo 2.1. Dado que a árvore já possui $n-1$ arestas e que um grafo simples de grau médio $\langle k\rangle$ possui $\frac{\langle k\rangle\cdot n}{2}$ arestas, o valor $S=\frac{\langle k\rangle\cdot n}{2}-(n-1)$ é calculado e representa exatamente a quantidade de arestas a serem adicionadas para atingir este objetivo. De modo semelhante, também é calculado R que é o número máximo de arestas

que podem ser adicionadas, através da diferença do número de arestas do grafo completo equivalente $K_n = \frac{n\cdot(n-1)}{2}$ menos o número de arestas que a árvore já possui $(n-1)$.

Os valores de S e R são fundamentais para o algoritmo 2.2. Basicamente o algoritmo adiciona uma aresta aos vértices não adjacentes i e j com probabilidade $\frac{S}{R}$. A linha (9) controla quantas arestas ainda devem ser adicionadas (S) e a linha (11) quantas arestas estão disponíveis (R). O algoritmo termina quando não há mais arestas a adicionar ($S = 0$) e a matriz de adjacência final é retornada.

Os algoritmos 2.1 e 2.2 geram apenas um grafo conexo de grau médio $\langle k \rangle$. Para a geração de um grafo com vários componentes conexos, Ma, Yu e Zhao [31] expandiram o DSCRG para que ele gerasse um grafo com n vértices com c componentes conexos por meio do algoritmo 2.3. Neste n_i é o número de vértices de cada componente conexo, onde $\sum_{i=1}^{c} n_i = n$.

O primeiro passo do algoritmo 2.3 consiste em determinar o número de vértices de cada componente n_i, sabendo:

$$\begin{cases} n = \sum_{i=1}^{c} n_i \quad i = 1,2,...,c \\ n_i \geq 2 \end{cases} \tag{2-12}$$

A fim de obter o grau médio $\langle k \rangle$ desejado, o algoritmo tem como meta que cada um dos c componentes tenham o mesmo grau médio $\langle k \rangle$. No entanto, quando $n_i - 1 < \langle k \rangle$ é impossível atingir o valor desejado. Assim, quando esta condição ocorre, a estratégia adotada foi gerar grafos completos para que estes chegassem no valor mais próximo do objetivo. Deste modo, ao ordenar os grafos pelo número de vértices n_i tem-se:

$$\overbrace{G_1, G_2, G_3, ..., G_j}^{\text{Grafos completos}}, \underbrace{G_{j+1}, ..., G_c}_{\text{Grau médio } \langle k \rangle'} \tag{2-13}$$

Nota-se que, os grafos $[j+1, c]$ da sequência da equação 2-13 possuem grau médio $\langle k \rangle'$. Este valor é superior ao $\langle k \rangle$ inicial para compensar a falta de arestas nos grafos anteriores.

Algoritmo 2.3: $DSCRG-generalizado(n,c,\langle k\rangle)$, adaptado de Ma, Yu e Zhao [31].

Entrada: número de vértices desejados n, número de componentes conexos c, grau médio dos vértices desejado $\langle k\rangle$.

Saída: matriz de adjacência $A(n\times n)$ do grafo com n vértices com grau médio mais próximo de $\langle k\rangle$

Aleatoriza n_i observando que $n=\sum_{i=1}^{c} n_i$ e $n_i \geq 2$
ordenar(n_i) /* Ordem crescente n_i */
$j^{(0)} \leftarrow 0$ /* Quantidade de grafos completos */
$l \leftarrow 1$ /* Número de iterações */
$\langle k^{(l)}\rangle \leftarrow \langle k\rangle$
repita
  $i \leftarrow j^{(j-1)}+1$
  $j^{(l)} \leftarrow 0$
  repita
    se $n_i < \langle k^{(l)}\rangle + 1$ **então**
      $j^{(l)} \leftarrow i$
      $i \leftarrow i+1$
    senão
      pare
    fim
  até $i > c$
  se $j^{(l)} > 0$ **então**
    $e^{(l)} \leftarrow e^{(l-1)} - \sum_{m=1}^{j^{(l)}} \frac{n_m(n_m-1)}{2}$ /* Arestas restantes */
    $n^{(l)} \leftarrow n^{(l-1)} - \sum_{m=1}^{j^{(l)}} n_m$ /* Vértices restantes */
    $\langle k^{(l)}\rangle \leftarrow \frac{2e^{(l)}}{n^{(l)}}$ /* Grau médio para o restante do grafo */
    $l \leftarrow l+1$
  senão
    pare
  fim
até *falso*
para $m \leftarrow 1$ ***até*** $j^{(l-1)}$ **faça**
  $A_m \leftarrow matriz\_adjacencia(K_{n_m})$
fim
para $m \leftarrow j^{(l-1)}+1$ ***até*** c **faça**
  $A_m \leftarrow DSCRG(n_m, \langle k^{(l-1)}\rangle)$
fim
return *Junção* A_1, A_2, A_3, ..., A_c

2.2.2 Geographical Network Learner and Generator (GNLG)

De acordo com Soltan e Zussman [54], a grande parte dos algoritmos de geração de grafos não considera a distribuição espacial das barras. Segundo eles, esta é uma informação importante, pois está correlacionada ao comprimento das linhas de transmissão existentes entre elas.

O Geographical Network Learner and Generator (GNLG), apresentado no algoritmo 2.4, é composto de três algoritmos: Spatially Distributed Nodes Generator (SDNG), Tunable Weight Spanning Tree (TWST) e um algoritmo responsável por adicionar arestas a árvore criada pelo TWST.

Algoritmo 2.4: $GNLG(G, \{p_i\}_{i=1}^n, \kappa, \alpha, \beta, \gamma, N)$, traduzido de Soltan e Zussman [54].

Entrada: grafo de referência G, posições geográficas de cada barra $\{p_i\}_{i=1}^n$, parâmetros das distribuições de probabilidade $\kappa, \alpha, \beta, \gamma > 0$, número de vizinhos mais próximos considerados $N \in \mathbb{N}$.

Saída: Grafo gerado G'

1 $\{p'_i\}_{i=1}^n \leftarrow SDNG(G, \{p_i\}_{i=1}^n)$
2 $TWST(n, \{p'_i\}_{i=1}^n, \kappa)$
3 $Reforco(n, m, \{p'_i\}_{i=1}^n, \alpha, \beta, \gamma, \eta, N)$
4 **return** G'

A premissa básica do SDNG apresentada no algoritmo 2.5 é que as posições das barras estão relacionadas ao tamanho da população e características geográficas. Deste modo os vértices são agrupados em c aglomerados, determinados pelo método Bayesian Information Criterion (BIC), e, em seguida, é criado um modelo de mistura gaussiana (GMM) [44].

O algoritmo 2.6 é responsável pela criação de arestas entre os vértices criados pelo SDNG. Primeiramente, os vértices são ordenados, linhas 3 a 7, de acordo com a distância euclidiana entre eles, simbolizada por $||p_i - p_j||$. Em seguida, nas linhas 8 a 10 um vértice é conectado ao vizinho mais próximo $\sigma(j^*)$ de modo que $j^* < i$. O resultado do processo é uma árvore T cujo peso depende da ordenação dos vértices que é influenciada pelo parâmetro κ. Quanto maior este parâmetro, mais T se assemelha a árvore geradora mínima (MST) para os vértices escolhidos.

Algoritmo 2.5: $SDNG(G, \{p_i\}_{i=1}^n)$, traduzido de Soltan e Zussman [54].

Entrada: grafo de referência G, posições geográficas de cada barra $\{p_i\}_{i=1}^n$.

Saída: distribuição especial similar a original $\{p'_i\}_{i=1}^n$

1 Ajustar um modelo GMM para $\{p_i\}_{i=1}^n$ para agrupá-los em c clusters que maximize o BIC

2 Para todo $i = 1, \cdots, n$ amostre z_i da distribuição de probabilidade categórica π obtida do GMM

3 Para todo i amostre p'_i da distribuição de probabilidade $\mathcal{N}(\mu_{z_i}, \sum_{z_i})$ obtida do GMM

4 **return** $\{p'_i\}_{i=1}^n$

Algoritmo 2.6: $TWST(n, \{p'_i\}_{i=1}^n, \kappa)$, traduzido de Soltan e Zussman [54].

Entrada: número de vértices n, distribuição espacial do modelo ajustado $\{p'_i\}_{i=1}^n$, parâmetro da distribuição κ.

Saída:

1 $A \leftarrow \{1, \cdots, n\}$

2 $\sigma \leftarrow$ vetor vazio com n elementos

3 **para** $i \leftarrow 1$ ***até*** n **faça**

4 Obtenha um vértice de A de modo que a probabilidade de obter o vértice j seja $\frac{||p'_j - \overline{p}'||^{-\kappa}}{\sum_{a \in A} ||p'_a - \overline{p}'||^{-\kappa}}$

5 $\sigma(i) \leftarrow j$

6 $A \leftarrow A - \{j\}$

7 **fim**

8 **para** $i \leftarrow 2$ ***até*** n **faça**

9 Conecte o vértice $\sigma(i)$ ao $\sigma(j^*)$ de modo que $j^* \leftarrow argmin_{j<i} ||p'_{\sigma(i)} - p'_{\sigma(j)}||$

10 **fim**

O algoritmo 2.7 objetiva aumentar a robustez da árvore T gerada pelo TWST. Para isto são adicionadas novas arestas transformando-a em um grafo. As arestas são adicionadas para atingir o comprimento do caminho médio e o coeficiente de clusterização médio desejado. As premissas deste algoritmo são: (1) a distribuição dos graus é geométrica com poucos vértices de grau 1 e 2; (2) redes elétricas não possuem linhas longas; (3) vértices em uma área densa possuem graus maiores.

Algoritmo 2.7: $Reforco(n, m, \{p'_i\}_{i=1}^{n}, \alpha, \beta, \gamma, \eta, N)$, traduzido de [54].

Entrada: número de vértices n, número de arestas m, distribuição espacial do modelo ajustado $\{p'_i\}_{i=1}^{n}$, parâmetros de ajuste das distribuições $\alpha, \beta, \gamma, \eta$, número de vizinhos mais próximos considerados N.

Saída:

Para cada vértice i calcule ρ_i (distância média de i para seus N vizinhos)
para $contador \leftarrow 1$ ***até*** $m - n + 1$ **faça**
  se *rede grande* **então**
    Para todos os vértices com grau menor que 3 amostre o vértice i com probabilidade $\propto \rho_i'^{-\alpha}$
  senão
    Amostre o vértice i com probabilidade $\propto d_i'^{-\eta} \rho_i'^{-\alpha}$
  fim
  Conecte o vértice i com o j amostrado de todos os outros vértices com probabilidade $\propto ||p'_i - p'_j||^{-\beta} d_j'^{\gamma}$
fim

2.3 Atribuição de dados elétricos

Uma terceira categoria de trabalhos sobre geração de redes elétricas consiste em atribuir valores elétricos a partir de uma topologia preexistente.

2.3.1 Tipos de barras

Em Wang, Elyas e Thomas [59], criadores do *RT-nested-smallworld* (seção 2.1), perceberam que os tipos de barras podiam ser agrupados em três categorias: geradores (G), barras de carga (L) e barras de conexão (C). Ao observarem os modelos IEEE-30, IEEE-57, IEEE-118, IEEE-300 e NYISO, perceberam que de 10 a 40% das barras são geradoras (G), de 40 a 60% são barras de carga (L) e de 10 a 20% são barras de conexão (C) [63, 59].

A tabela 2.1 apresenta o grau médio geral ($\langle k \rangle$) e o coeficiente de clusterização ($C(G)$) por tipo de barra ($\langle k \rangle_G$, $\langle k \rangle_L$ e $\langle k \rangle_C$). É possível notar que ambos coeficientes são dependentes dos tipos de barra.

Para estudar a distribuição destas barras Wang, Elyas e Thomas [59] atribuíram os valores 1, 2 e 3 para as barras do tipo G, L e C. Assim, foi criado um vetor $\mathbb{T}$ com n elementos para armazenar os tipos de cada barra. Seguindo esta estratégia, as ligações $\mathbb{L}$ formaram um vetor de m elementos que são mapeados em tipos $\{GG, GL, GC, LL, LC, CC\}$

recebendo os respectivos valores $\{1,2,3,4,5,6\}$. Com base nestes conceitos foi definida a métrica Bus Type Entropy ($W(\mathbb{T})$):

$$W(\mathbb{T}) = -\sum_{i=1}^{n} \log(r_{\mathbb{T}_i}) - \sum_{j=1}^{m} \log(R_{\mathbb{L}_j}) \tag{2-14}$$

onde n_k e m_k representam o número de barras e linhas do tipo k, $r_{\mathbb{T}_i} = \frac{n_{\mathbb{T}_i}}{n}$ e $R_{\mathbb{L}_j} = \frac{m_{\mathbb{L}_j}}{m}$ são proporções destes tipos de barras e linhas.

Tabela 2.1: *Grau médio do vértice e coeficiente de clusterização pelo tipo de barra. Fonte: Wang e Thomas [63].*

Modelo	$\langle k \rangle$	$\langle k \rangle_G$	$\langle k \rangle_L$	$\langle k \rangle_C$	C_{all}	C_G	C_L	C_C
IEEE-30	2,73	2,00	2,61	3,83	0,2348	0,1944	0,2537	0,2183
IEEE-57	2,74	3,86	2,54	2,67	0,1222	0,1524	0,1352	0,0778
IEEE-118	3,03	3,56	2,44	3,40	0,1651	0,1607	0,1969	0,0167
IEEE-300	2,73	1,96	2,88	3,15	0,0856	0,1227	0,0895	0,0364
NYISO	4,47	4,57	5,01	3,33	0,2134	0,2693	0,2489	0,0688
WECC	2,67	-	-	-	0,0801	-	-	-

Segundo Wang, Elyas e Thomas [59] a equação 2-14 pode ser reescrita como $W_1(\mathbb{T})$ e possui duas variantes $W_2(\mathbb{T})$ e $W_3(\mathbb{T})$:

$$W_1(\mathbb{T}) = -\sum_{k=1}^{3} \log(r_k) \times n_k - \sum_{k=1}^{6} \log(R_k) \times m_k \tag{2-15}$$

$$W_2(\mathbb{T}) = -\sum_{k=1}^{3} \log(r_k) - \sum_{k=1}^{6} \log(R_k) \tag{2-16}$$

$$W_3(\mathbb{T}) = -\sum_{k=1}^{3} \log(r_k) \times \frac{1}{n_k} - \sum_{k=1}^{6} \log(R_k) \times \frac{1}{m_k} \tag{2-17}$$

Com o objetivo de avaliar os descritores utilizaram o método de Monte Carlo a fim de entender o posicionamento da entropia $W^* = W(\mathbb{T}^*)$, onde $\mathbb{T}^*$ é o tipo de barra de uma rede real, de uma rede real em relação a redes aleatórias. Para isto gerou-se uma quantidade massiva de dados através da permutação de $\mathbb{T}^*$.

Estas permutações mantém a quantidade de cada tipo de barra e cria um espaço amostral de tamanho $\hat{n} = \frac{n!}{n_G! n_L! n_C!}$. Como este número é muito grande, define-se então um valor k^{max} de amostras suficientemente grande para que o experimento tenha força estatística suficiente.

Nos estudos foram utilizados os modelos sintéticos NYISO e IEEE 300 e $k^{max} = 10.000$. A figura 2.1 exibe a PDF para a métrica *Bus Type Entropy* $W(\widetilde{\mathbb{T}})$, onde

$W(\widetilde{\mathbb{T}})$ é o conjunto de aleatorizações da rede real, o ajuste da distribuição normal e a posição do descritor da rede real marcado com uma estrela vermelha.

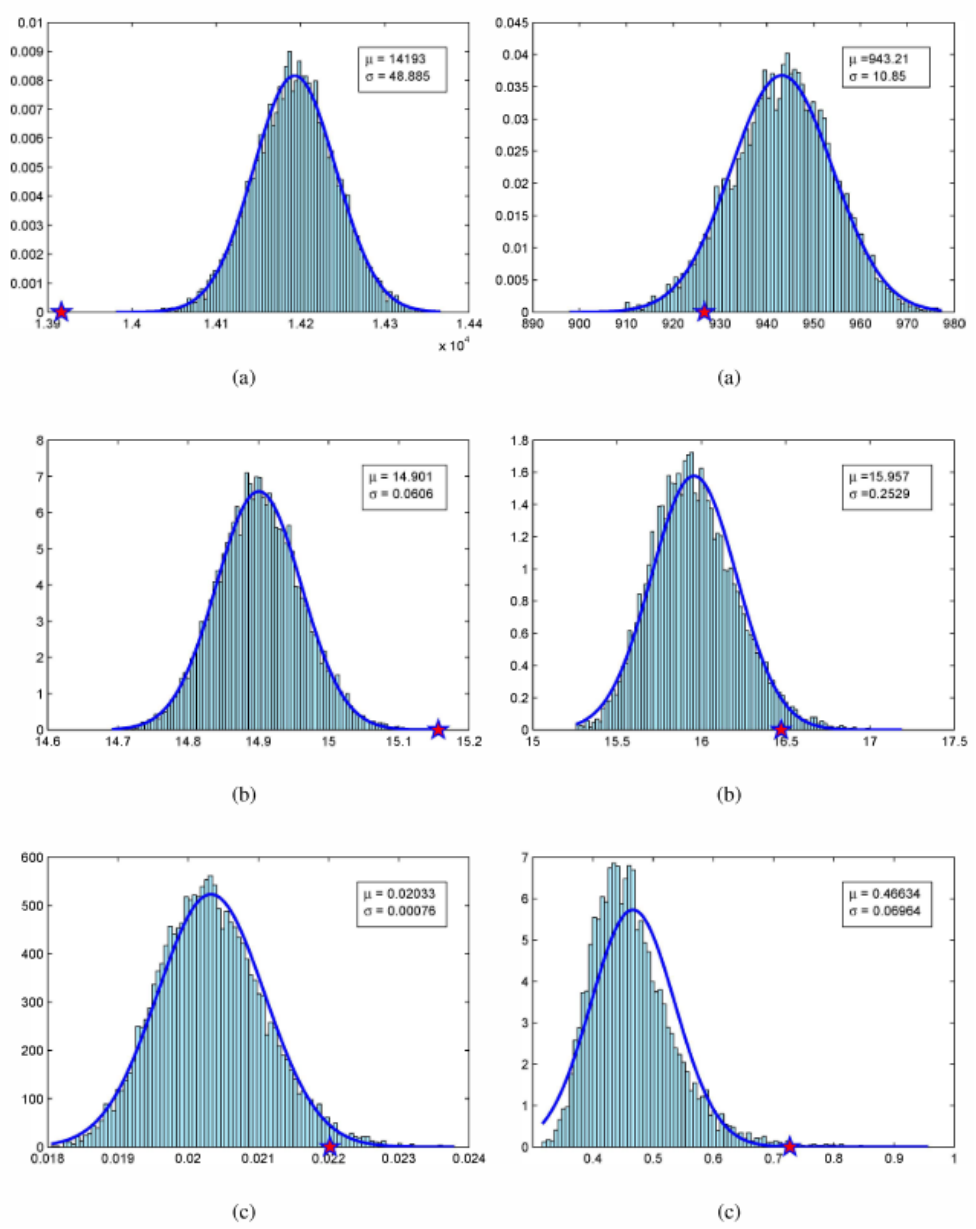


Figura 2.1: *PDF e ajuste da distribuição normal na amostragem aleatória para NYISO (coluna esquerda) e IEEE 300 (coluna direita). (a) $W_1(\tilde{T})$; (b) $W_2(\tilde{T})$; (c) $W_3(\tilde{T})$. Fonte: Wang, Elyas e Thomas [59].*

Segundo Elyas e Wang [21] este posicionamento é classificado em três grupos de acordo com a posição e o ajuste da curva normal sobre o experimento:

a) $|W^* - \mu| > 3\sigma$: o valor computado para rede elétrica W^* é diferente de 99,9% dos dados aleatórios;

b) $\sigma < |W^* - \mu| \leq 3\sigma$: o valor computado está em um intervalo intermediário, servindo ainda como guia limitado para uma busca do tipo de barra adequado;

c) $|W^* - \mu| \leq \sigma$: o valor computado não difere da maioria dos dados. Consequentemente uma busca guiada por este índice falhará em distinguir entre o tipo de barra de uma rede realística e uma rede aleatória.

Em suas observações Elyas e Wang [21] notaram apenas casos (a) e (b) utilizando as redes elétricas IEEE 3000, NYISO e MPC com os coeficientes W_1, W_2 e W_3. Ao observar um ajuste adequado da distribuição normal propôs o cálculo de distância normalizada como um índice para as análises:

$$d = \frac{|W^* - \mu|}{\sigma} \tag{2-18}$$

Em seguida, calcularam esta distância para cada rede elétrica estudada conforme a tabela 2.2. Através desta tabela é possível observar que à medida que o tamanho da rede aumenta, a distância relativa d também tende a incrementar, especialmente, em W_2 e W_3.

Tabela 2.2: *Distância normalizada para a entropia original W^* em redes elétricas. Fonte: Elyas e Wang [21].*

Modelo sintético	(n, m)	d_{W_1}	d_{W_2}	d_{W_3}
IEEE-300	(300, 409)	1,48	1,96	3,76
NYISO	(2935, 6567)	5,78	28,72	2,42
MPC	(5633, 7053)	16,71	33,30	177,48

Com base nestas observações, foi proposto uma busca usando o Clonal Selection Algorithm (CSA) a fim de encontrar a melhor atribuição de tipo de barras a uma rede sintética:

1. Inicializa a população com P indivíduos, onde cada indivíduo é um vetor $\mathbb{T}$ com n elementos cujo valor é gerado aleatoriamente podendo ser 1, 2 ou 3;
2. Avalia os indivíduos usando $W(\mathbb{T})$ como função de adaptação e a função de afinidade definida por Mahanty e Gupta [32];
3. Ordena as soluções de acordo com a função de afinidade seguindo a função de entropia escolhida W_1, W_2 e W_3;
4. Seleciona as n primeiras soluções com base na ordenação e realiza cópias, onde o número de cópias $nc_i = \lceil \frac{\beta N}{i} \rceil$, no qual β é uma constante que define a taxa de cópia e i é a posição da solução. O número total de soluções na próxima geração é dado por $NC = \sum_{i=1}^{n} \lceil \frac{\beta N}{i} \rceil$;
5. Realiza uma mutação nas soluções com base na entropia $W(\mathbb{T})$. Assim soluções com mais afinidade devem ser modificadas com menor frequência que as soluções com baixa afinidade;
6. Avalia a população e seleciona as primeiras m soluções com maior afinidade para formar a população da geração seguinte, onde $m < P$;

7. Gera $p = P - m$ novas soluções para a próxima geração através de um processo aleatório para aprimorar a diversidade do CSA;
8. Volta ao passo 2 e repete o processo até atingir o critério de convergência. Quando o atinge, a melhor solução é retornada no fim do processo.

Em seus experimentos Elyas e Wang [21] perceberam uma redução consistente da função objetivo ($minError = |W^* - W(\mathbb{T})|$) nas duas redes elétricas analisadas, IEEE-300 e NYISO, para os três coeficientes de entropia W_1, W_2 e W_3 atingindo valores muito próximos de zero.

Em trabalhos seguintes foi proposto um novo coeficiente de entropia [60, 22]:

$$W_0(\mathbb{T}) \quad = \quad -\sum_{k=1}^{3} r_k \times \log(r_k) - \sum_{k=1}^{6} R_k \times \log(R_k) \tag{2-19}$$

E desta vez avaliaram mais redes elétricas IEEE 30, 57, 118 e 300, NYISO 2935, ERCOT 5633, PEG 9241 e 13659 e WECC 16994. A tabela 2.3 apresenta os coeficientes de entropia W_0 e W_1 além das distâncias normalizadas d para as redes estudadas.

Tabela 2.3: *Distâncias d e entropias W_0 e W_1 para os modelos sintéticos. Fonte: Elyas e Wang [22].*

Modelo	n	m	$d_{W_{0(\mathbb{T})}}$	$d_{W_{1(\mathbb{T})}}$	$W_{0(\mathbb{T})}$ $\mu/\sigma/W_0^*$	$W_{1(\mathbb{T})}$ $\mu/\sigma/W_1^*$
IEEE-30	30	41	1,22	1,31	2,38/0,09/2,49	87,02/3,8/92
IEEE-57	57	78	2,24	2,28	2,31/0,058/2,44	161,4/4,63/172
IEEE-118	118	179	0,22	0,08	2,34/0,045/2,35	364,39/8,13/365,04
IEEE-300	300	409	-1,53	-1,48	2,57/0,026/2,53	943,21/10,58/927,5
NYISO-2935	2935	6567	-5,71	-5,79	2,74/0,007/2,70	14193/48,8/13910
ERCOT-5633	5633	7053	-16,25	-16,71	2,36/0,008/2,23	15372/56,47/14428
PEG-9241	9241	16049	-44,5	-52,44	2,84/0,006/2,58	34367/113,40/28441
PEG-13659	13659	20467	-80,1	-248,04	2,80/0,003/2,56	50075/67,26/33392
WECCC-16994	16994	21539	-114,7	-350,3	2,72/0,0034/2,33	53813/74,33/27775

Na tabela 2.3 pode ser observado que conforme o tamanho da rede (n) aumenta menor é o valor de d para ambos coeficientes de entropia. Isolando a variável W^* na equação 2-18 e assumindo que d é dependente do tamanho da rede, tem-se:

$$W^* \quad = \quad \mu + \sigma \cdot d_W(n) \tag{2-20}$$

Com esta equação pode-se estimar um valor para W^* sem a necessidade de possuir os dados de uma rede elétrica. Esta relação foi calculada através de uma regressão em Wang, Elyas e Thomas [60], utilizando as redes IEEE 30, 57, 118, 300, NYISO-

2935, ERCOT-5633 e WECC-16994, e posteriormente em Elyas e Wang [22], incluindo as redes PEG-9242 e PEG-13659, cada regressão obteve coeficientes diferentes à medida que novas redes foram inclusas:

$$d_{W_0}(n) = \begin{cases} -1,721 \log n + 8 & \log n \leq 8 \\ -6,003 \times 10^{-14} (\log n)^{15,48} & \log n > 8 \end{cases} \quad (2\text{-}21)$$

$$d_{W_1}(n) = \begin{cases} -1,748 \log n + 8,276 & \log n \leq 8 \\ -6,053 \times 10^{-22} (\log n)^{24,1} & \log n > 8 \end{cases} \quad (2\text{-}22)$$

2.3.2 Geração e carga

Em Wang e Elyas [58] foi analisada a dependência da geração e da carga de energia com o número de vértices de uma rede, o resultado destas regressões são apresentados na figura 2.2 para os 14 modelos sintéticos[2], na qual ambos os eixos estão em escala logarítmica. Observou-se que a relação entre geração/carga ficou entre 1,5 e reduz conforme o número de vértices aumenta até atingir 1,0.

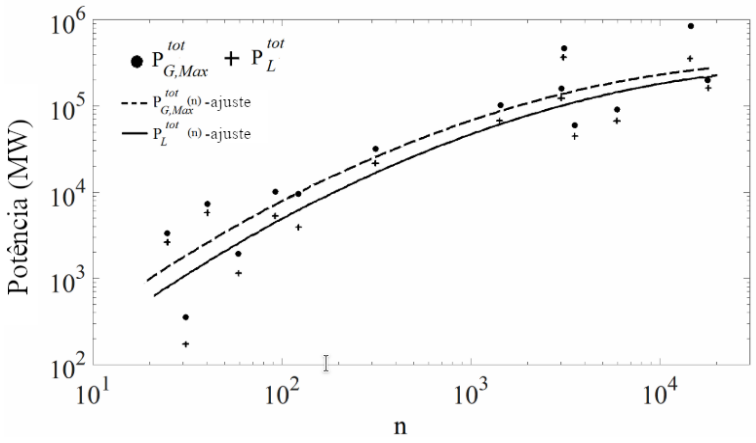


Figura 2.2: *Regressão da geração e carga total em função do número de barras da rede. Fonte: Wang e Elyas [58].*

A dependência da geração e carga em relação ao número de vértices foi:

$$\log P^{tot}_{G,max}(n) = -0,21 \cdot (\log n)^2 + 2,06 \cdot \log n + 0,66 \quad (2\text{-}23)$$

$$\log P^{tot}_{L}(n) = -0,20 \cdot (\log n)^2 + 1,98 \cdot \log n + 0,58 \quad (2\text{-}24)$$

Adicionalmente observou-se que a capacidade de geração e carga seguem uma distribuição exponencial para as redes elétricas PEGASE-13659, WECC-16994 e

[2] IEEE 24, 30, 57, 118, 300, NE-39, PEGASE 89, 1354, 2869, 13659, NYISO-2935, Polish-3375, ERCOT-5633 e WECC-16994.

NYISO-2935. Cerca de 99% dos geradores nestas redes seguem uma distribuição exponencial e apenas 1% possuem altas capacidades.

Após este estudo, Elyas, Wang e Thomas [23] desenvolveram uma metodologia que envolve duas matrizes normalizadas em relação ao grau dos vértices para estimativa da geração e carga nas barras da rede. Estas normalizações acontecem em função do grau, geração e carga máxima:

$$\overline{P_{g_n}^{Max}} = \frac{P_{g_n}^{Max}}{\max\limits_i P_{g_i}^{Max}} \quad (2\text{-}25) \qquad \overline{k_n} = \frac{k_n}{\max\limits_i k_i} \quad (2\text{-}26) \qquad \overline{P_{L_n}} = \frac{P_{L_n}}{\max\limits_i P_{L_i}} \quad (2\text{-}27)$$

A matriz normalizada da máxima potência ativa da rede WECC-16994 é apresenta na 2.3. Ao todo são 13 classes de grau e de potência com 169 valores em frequência relativa no intervalo $[0,1]$.

$\overline{P_{g_n}^{Max}}$ \ $\overline{k_n}$	0.00–0.01	0.01–0.03	0.03–0.06	0.06–0.1	0.1–0.15	0.15–0.21	0.21–0.28	0.28–0.36	0.36–0.45	0.45–0.55	0.55–0.66	0.66–0.78	0.78–1.00	Marginal Prob
1.00–0.78	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.002
0.78–0.66	0.000	0.000	0.000	0.001	0.000	0.000	0.001	0.001	0.001	0.000	0.000	0.000	0.000	0.004
0.66–0.55	0.000	0.000	0.001	0.000	0.000	0.000	0.001	0.001	0.000	0.000	0.000	0.000	0.000	0.003
0.55–0.45	0.000	0.001	0.000	0.004	0.008	0.000	0.002	0.001	0.000	0.001	0.000	0.001	0.000	0.018
0.45–0.36	0.006	0.002	0.000	0.009	0.008	0.003	0.003	0.002	0.000	0.000	0.000	0.000	0.000	0.034
0.36–0.28	0.003	0.011	0.012	0.017	0.013	0.007	0.003	0.002	0.000	0.001	0.000	0.000	0.000	0.072
0.28–0.21	0.009	0.024	0.016	0.024	0.013	0.004	0.003	0.001	0.000	0.000	0.000	0.000	0.001	0.097
0.21–0.15	0.025	0.027	0.016	0.013	0.009	0.002	0.002	0.000	0.000	0.000	0.000	0.000	0.001	0.097
0.15–0.1	0.027	0.031	0.010	0.010	0.005	0.004	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.088
0.1–0.06	0.033	0.017	0.003	0.003	0.005	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.063
0.06–0.03	0.090	0.030	0.01	0.008	0.001	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.151
0.03–0.01	0.082	0.140	0.070	0.04	0.010	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.360
0.01–0.00	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Marginal Prob	0.283	0.291	0.147	0.141	0.077	0.022	0.017	0.008	0.001	0.003	0.000	0.001	0.002	1.000

Figura 2.3: *Matriz normalizada em função da geração e grau do vértice para o WECC-16994. Fonte: Elyas, Wang e Thomas [23].*

A figura 2.4 apresenta o fluxograma utilizado para atribuir valores de geração de energia que com poucas modificações pode ser utilizado para atribuição de carga. A seguir a descrição dos passos do algoritmo:

1. Estimativa da capacidade de geração total usando 2-23;
2. Geração aleatoriamente correta das capacidades de geração $[P_{g_n}^{Max}]_{1\times N_g}$, onde N_g é o número de geradores. Nota-se que 99% das capacidades seguem uma distribuição exponencial enquanto que as demais possuem valores entre 2 a 3 vezes maiores;
3. Dimensiona a capacidade de geração se $\sum_{i=1}^{N_g} P_{g_n}^{Max} > 1,05P^{tot}$ para garantir que a capacidade de geração está dentro de $P_g^{tot}(n)$, utilizando a equação:

$$[P_{g_n}^{Max}]'_{1\times N_g} = [P_{g_n}^{Max}]_{1\times N_g} \times \frac{P_g^{tot}}{\sum_{n=1}^{N_g} P_{g_n}^{Max}} \quad (2\text{-}28)$$

onde $[P_{g_n}^{Max}]'_{1\times N_g}$ são as capacidades de geração atualizadas;

4. Calcula o grau de cada barra de geração;
5. Normaliza o grau de cada barra e a potência usando as equações 2-25 e 2-26;
6. Verifica a compatibilidade com a matriz apresentada na figura 2.3 e reordena os segmentos incompatíveis;
7. Atribui as capacidades de geração às barras geradoras em relação a seus graus;
8. Converte os valores normalizados para os valores reais.

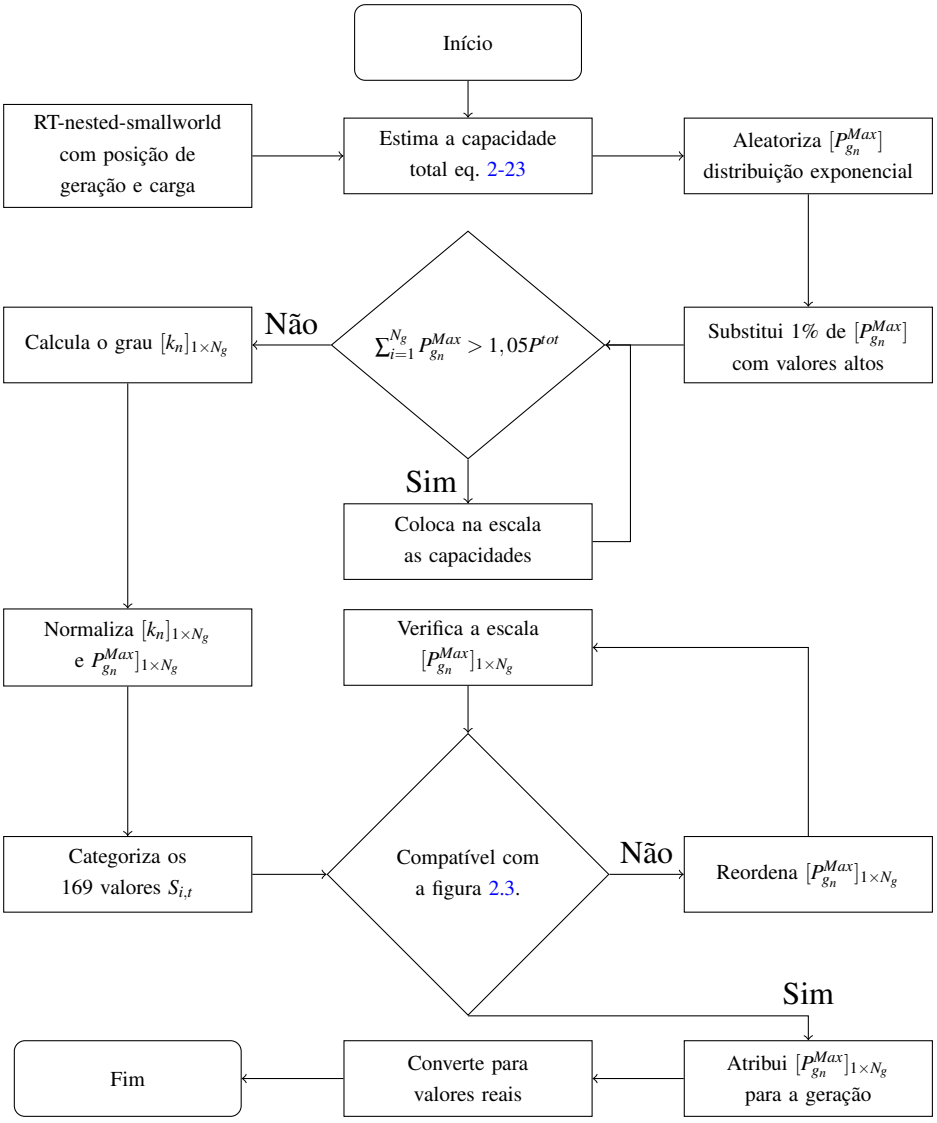


Figura 2.4: *Fluxograma do algoritmo de atribuição aleatória da capacidade de geração. Adaptado de: Elyas, Wang e Thomas [23].*

Devemos observar que mesmo sendo boas iniciativas de atribuição dos dados elétricos, a determinação dos tipos de barras, apresentado na seção 2.3.1, se baseou puramente na métrica Bus Type Entropy ($W(\mathbb{T})$), que por sua vez, considera apenas a proporção entre os tipos de barras e suas ligações, desconsiderando as relações topológicas

entre esses elementos. Apenas a atribuição de geração e carga, apresentada na seção 2.3.2, levou em consideração o grau dos vértices das barras. Neste sentido, estudos que considerem o posicionamento das barras de carga e dependências mais profundas da carga com a topologia são necessários.

2.3.3 Outras informações relevantes

Além dos dados já mencionados, as redes elétricas podem possuir chaves seccionadoras, reguladores de tensão e bancos de capacitor. Estes elementos são parte integrante de um dos principais problemas em sistemas de distribuição de energia elétrica, que é o restabelecimento do fornecimento de energia para regiões a jusante de uma falta [49, 42, 50, 6, 8, 48, 35].

É importante observar que a cada barra estão vinculadas informações sobre as unidades consumidoras, tais como: tipo de usuário, grupo (A e B) e subgrupo (A1, A2, A3, A3a, A4, AS, B1, B2, B3, B4), classe (residencial, industrial, comercial, rural, poder público, iluminação pública, serviço público e consumo próprio) e subclasse de acordo com a Resolução Normativa n° 414 [1]. No entanto, dados relacionados aos consumidores de uma rede são ainda mais sigilosos do que a própria topologia.

A respeito das linhas elétricas é imprescindível conhecer o material constituinte, diâmetro e o comprimento do cabo, pois juntos permitem o cálculo da ampacidade, perda de carga, impedância e admitância. Adicionalmente, é essencial conhecer a voltagem e a corrente elétrica em cada barra e, para isto, é utilizado um procedimento de fluxo de potência [52]. Se a topologia da rede for radial é comum a utilização do método de varredura direta/inversa [10, 52, 37].

2.4 Grafos aleatórios

Esta seção apresenta os algoritmos amplamente utilizados na geração de grafos. Estes algoritmos se tornaram ao longo do tempo base para criação de novos métodos além de serem guia para avaliação do comportamento de novas propostas.

2.4.1 Erdős e Rényi

Em 1959, Erdős e Rényi [24] estudaram a conectividade de grafos aleatórios. Para isto propuseram um algoritmo capaz de gerar grafos simples[3] com n vértices e m arestas que ficou amplamente conhecido por sua simplicidade escolhendo m arestas das $\binom{n}{2}$ possíveis.

[3] Entende-se por grafo simples aqueles que não tem *loops* e não tem arestas paralelas [39].

A principal característica deste método é sua uniformidade, ou seja, todos os grafos com n vértices e m arestas têm a mesma probabilidade de aparecerem como resultado final do processo. Isto o tornou a base de comparação para muitos outros algoritmos.

2.4.2 Watts e Strogatz

Em 1998, Watts e Strogatz [65], influenciados por Travers e Milgram [55] a respeito das redes de contatos sociais, desenvolveram um algoritmo de geração de grafos simples com n vértices e $\frac{n \cdot \langle k \rangle}{2}$ arestas, onde $\langle k \rangle$ representa o número de arestas em cada vértice na inicialização. O algoritmo consiste em inicializar uma grade regular em forma de círculo com n vértices onde cada um possui $\langle k \rangle$ arestas conforme apresentado na imagem mais à esquerda da figura 2.5. Em seguida, os vértices do grafo são percorridos em sentido horário de modo que cada aresta seja visitada e submetida a modificação de seu vértice oposto com probabilidade p desde que a ligação com o novo vértice-extremidade não forme um *loop* ou uma aresta paralela.

Segundo Watts e Strogatz [65], a medida que p aumenta de 0 até 1 saímos de um grafo totalmente estruturado para um grafo totalmente aleatório. No meio deste caminho são encontrados grafos com características de "mundo-pequeno", ou seja, grafos com alto coeficiente de clusterização e com pequeno comprimento médio de caminho.

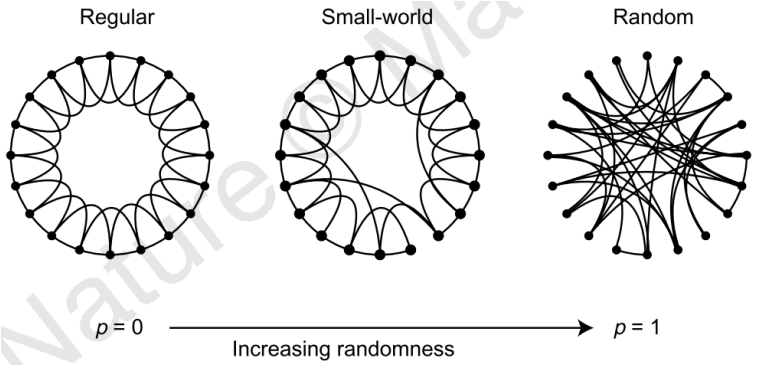


Figura 2.5: *Religação das arestas de uma grade a um grafo totalmente aleatório. Fonte: [65].*

2.4.3 Albert e Barabási

Em 2002, Albert e Barabási [3] propuseram um algoritmo que cria um grafo com n vértices e $\frac{n \cdot m}{2}$ arestas, onde m é o número de arestas ligadas a cada vértice, similar a Watts e Strogatz [65]. No entanto, a criação de arestas é feita de modo que os vértices com maior grau tenham maior probabilidade de serem escolhidos assim como acontece com muitas redes reais, como a Internet.

Neste sentido, eles inicializam um grafo com m_0 vértices no qual serão adicionados $n - m_0$ vértices simulando o crescimento da rede. Cada vértice adicionado será ligado a m outros vértices com probabilidade $\frac{k_i}{\sum k_j}$, onde k_i é o grau do i-ésimo vértice, ou seja, quanto maior o grau de um vértice, maior é a probabilidade de ser escolhido. O número de arestas adicionadas a cada etapa é restrito a $m \leq m_0$, no entanto é frequente a escolha de $m = m_0$.

2.4.4 GraphCuisine

O GraphCuisine, criado por Bach et al. [4], é uma das poucas aplicações que utilizam Computação Evolutiva (CE) para geração de grafos. Eles propõem uma representação indireta utilizando 12 geradores de grafos. O grafo final é formado pela ação destes geradores, que leem seus parâmetros no cromossomo e agem na saída do gerador anterior.

Estes geradores possuem funções específicas de adicionar vértices ou arestas. Desta forma, o processo evolutivo tenta encontrar a melhor parametrização dos geradores para atingir o objetivo definido pelo usuário.

Cada gerador possui entre 2 a 5 parâmetros em sua configuração e são agrupados em duas categorias: os criadores de padrões e os geradores de ruídos. Entre os criadores de padrões estão formação de: caminhos, ciclos, estrelas e *clusters*. Já entre os geradores de ruídos estão: inserção ou remoção aleatória de arestas e vértices, adição de arestas para atingir determinadas distribuições de grau (exponencial e logarítmica), adição de arestas de acordo com o modelo de Eppstein e remoção de vértices de grau zero.

A adaptação dos indivíduos é medida através da distância entre as métricas do grafo gerado e as usadas como objetivo. As métricas utilizadas foram: quantidade de vértices e arestas, densidade, diâmetro, número de *clusters*, número de componentes conexos, grau médio e coeficiente de clusterização.

Segundo os autores existem duas vantagens em utilizar um conjunto de parâmetros dos geradores como genoma: (1) o tamanho do genoma é constante independentemente do tamanho do grafo gerado; (2) as métricas são independentes dos parâmetros, permitindo facilmente a inclusão de novas métricas e novos geradores. A figura 2.6 apresenta o processo de criação de um grafo no GraphCuisine.

Cada cromossomo é dividido em três blocos. O primeiro constitui a semente de geração de números aleatórios, o segundo e o terceiro blocos representam, respectivamente, os geradores de padrão e de ruído.

Os dois primeiros genes de cada gerador armazenam a mesma informação. O primeiro serve para especificar se o gerador está ativo ou não e o segundo gene especifica a ordem de execução do gerador, como pode ser visto na figura 2.6. Os demais genes

são parâmetros próprios de cada gerador. Exemplos destes parâmetros próprios são: o percentual de vértices ou arestas que devem ser removidos, ou o número de vértices ou arestas que o grafo deve ter.

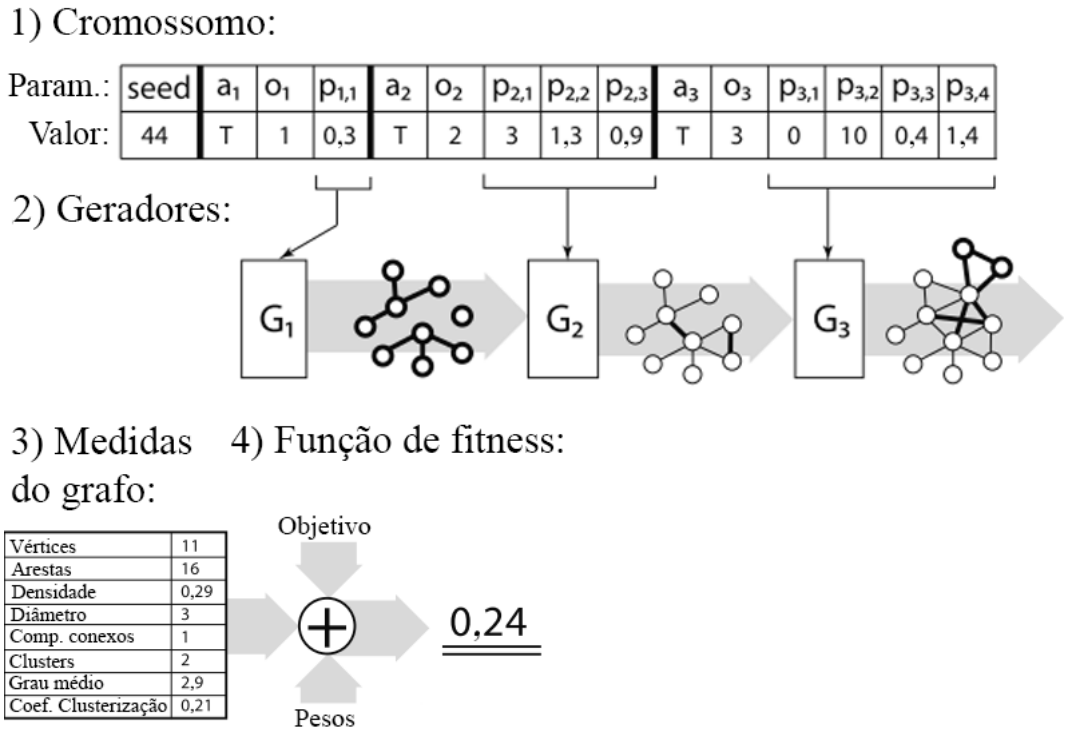


Figura 2.6: *Processo de criação de um grafo no GraphCuisine. (1) Parâmetros codificados nos cromossomos; (2) aplicação dos geradores; (3) extração das métricas; (4) cálculo do valor da adaptação. Adaptado de: Bach et al. [4].*

2.5 Computação Evolutiva

A Computação Evolutiva (CE) se firmou em 1960 como uma área de construção de heurísticas inspiradas nos mecanismos de adaptação dos seres vivos. Embora tenha surgido através da observação destes mecanismos, o pensamento moderno entende que este fator é opcional [27].

Um dos pontos fortes desta área reside no fato de que agentes com ações simples poderem produzir soluções complexas e, além disso, permitirem o uso do paralelismo computacional. Tais características viabilizam a resolução de problemas em que o tempo necessário é demasiadamente grande ou quando o problema é difícil de resolver.

Uma das primeiras metáforas da CE foram os Algoritmos Genéticos (AGs). Estes utilizaram os conceitos da teoria da evolução de Darwin na qual os indivíduos mais bem

adaptados ao ambiente tendem a propagar seus genes nas gerações seguintes em maior quantidade.

Nos AGs a resolução de problemas é feita representando cada solução como um indivíduo de uma população. Estes são submetidos aos operadores de *crossover* e mutação para gerarem suas proles e possuírem variabilidade genética. Durante o processo de seleção dos genitores, os melhores indivíduos possuem maiores chances de reprodução mantendo assim a analogia da seleção natural.

Embora os AGs sejam apresentados como um dos maiores destaques da CE, ela também é constituída de vários algoritmos baseados nos fundamentos das metáforas biológicas, tais como: Programação Genética, Colônia de Formigas, Algoritmos Imunoinspirados, Evolução Diferencial, Estimação da Distribuição, Recozimento Simulado, Busca Tabu, Busca Gulosa Aleatorizada e Adaptativa (GRASP).

Neste trabalho utilizaremos o Non-Dominated Sorting Genetic Algorithm II (NSGA-II) [15]. Este é um método de busca de soluções multiobjetivo, de segunda geração da CE, que utiliza seleção pela dominância de Pareto.

Quando são tratados problemas mono-objetivo, a comparação entre dois indivíduos se resume a comparação do valor resultante da função de adaptação entre estes indivíduos. Assim, é possível dizer que um indivíduo é melhor ou pior que outro. Consequentemente, é possível colocar os indivíduos em ordem de adaptação.

No entanto, quando o problema é multiobjetivo esta classificação dos melhores indivíduos não é tão simples assim. Nestes casos, ao invés de uma única função-objetivo, existe um vetor de funções-objetivo:

$$f(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_p(x) \end{bmatrix} \tag{2-29}$$

Consequentemente, no lugar de um número escalar que descreve a adaptação de um indivíduo, existe um vetor que o qualifica. Esta simples modificação muda completamente como é tratado a busca e o elemento ótimo.

Quando se tem dois vetores v_1 e v_2, nem sempre todos os elementos de v_1 são menores que os de v_2. Geralmente, existem alguns elementos de v_1 que são menores que os respectivos elementos de v_2 e o restante dos elementos de v_1 são maiores ou iguais aos de v_2. Para exemplificar essa questão, considere os vetores:

$$v_1 = \begin{bmatrix} 1 \\ 6 \\ 3 \end{bmatrix}, v_2 = \begin{bmatrix} 3 \\ 5 \\ 8 \end{bmatrix}, v_3 = \begin{bmatrix} 2 \\ 4 \\ 3 \end{bmatrix} \tag{2-30}$$

Note que todos os elementos de v_3 são menores que os de v_2, então em uma busca de minimização multiobjetivo seria seguro dizer que a solução que proporcionou a adaptação v_3 é melhor que a solução que obteve v_2. No entanto, ao realizar esta mesma comparação entre v_1 e v_2 não é possível dizer qual deles é o melhor, uma vez que, apenas dois elementos de v_1 são menores que seus respectivos elementos em v_2, já o segundo elemento de v_1 é maior que seu correspondente em v_2. Deste modo surge o operador $\prec$ para esta comparação de vetores, assim pode ser escrito $v_3 \prec v_2$.

Neste contexto, surgem os conceitos de dominância, solução eficiente, conjunto de Pareto ótimo e fronteira de Pareto [27]:

- **Dominância**: sejam dois vetores numéricos x_a e x_b que representam o valor das funções-objetivo dos indivíduos de A e B. Se $x_a \prec x_b$, diz-se que A domina B.
- **Solução eficiente**: a solução dada pelo indivíduo A é dita como solução eficiente, se não houver nenhuma outra solução factível que a domine.
- **Conjunto de Pareto-Ótimo**: o conjunto $\mathcal{P}$ de todas as soluções eficientes do problema;
- **Fronteira Pareto-Ótima**: a imagem do conjunto $\mathcal{P}$ de todos os pontos eficientes no espaço de objetivos.

O NSGA-II surge como um processo evolutivo que apresentou o algoritmo *Fast Nondominated Sorting* que ordena as fronteiras de Pareto com complexidade $O(MN^2)$, onde M é o número de objetivos e N é o tamanho da população.

A figura 2.7 apresenta uma visão geral do processo evolutivo no NSGA-II. Primeiramente, uma população inicial P_0 é criada. Sua prole (Q_0) é feita, através das operações seleção, recombinação e mutação. Em qualquer geração t, os genitores P_t e a prole Q_t são unidos na população, formando R_t de tamanho $2N$.

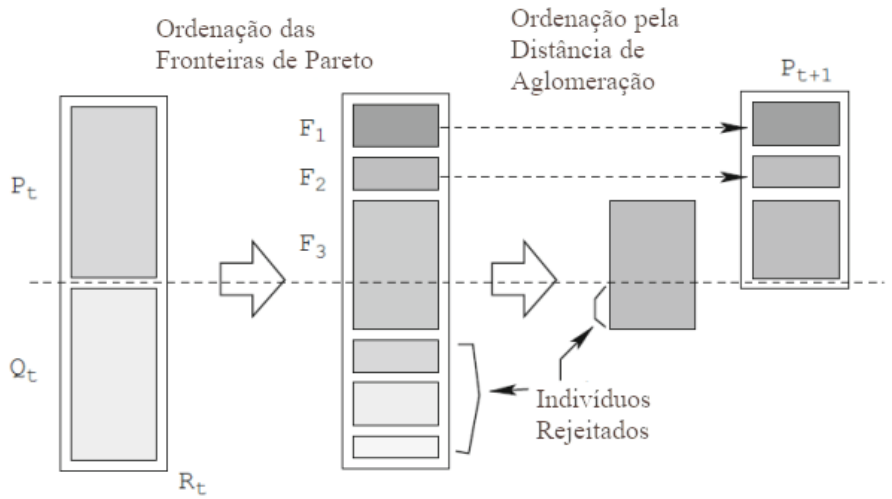


Figura 2.7: *Procedimento do NSGA-II. Fonte: Deb et al. [15].*

Como na próxima geração, a população (P_{t+1}) terá somente N indivíduos, a ordenação das fronteiras é utilizada para selecionar aqueles que farão parte do próximo ciclo. Enquanto couber todos os indivíduos de uma fronteira, a partir da primeira, em P_{t+1} eles serão adicionados, assim como acontece com as fronteiras F1 e F2 na figura 2.7. Quando uma fronteira F_l não couber completamente, então é utilizado uma classificação de aglomeração para escolher os indivíduos mais espalhados no espaço de busca.

É importante notar que o NSGA-II é um processo elitista, pois as melhores soluções de P_t permanecerão na próxima geração se elas ainda forem as melhores soluções após a geração da prole Q_t.

2.6 Geração de árvores com computação evolutiva

Esta seção apresenta os principais métodos e representações para geração de árvores usando computação evolutiva porque o método proposto neste trabalho cria uma floresta e depois adiciona arestas para conectar cada árvore agindo de modo parecido ao DSCRG [31] (seção 2.2.1) e ao GNLG [54] (seção 2.2.2).

De acordo com Rothlauf [45], as principais representações utilizadas na CE para árvores são: números Prüfer, Characteristic Vector Encoding (CV), Link and Node Biased Encoding (LNB) [40, 41], NetKey [47], NetDir e codificação Edge-Set [43].

Estas representações podem ser classificadas em diretas e indiretas. Nas representações diretas, o problema é codificado em sua forma mais intuitiva enquanto que nas representações indiretas ocorre um mapeamento genótipo-fenótipo, necessitando de um decodificador [46]. Assim, são consideradas diretas NetDir e Edge-Set, e as indiretas são os números Prüfer, CV, LNB e NetKey.

O NetDir apresenta um operador de mutação e *crossover* próprio. Na mutação, ele desconecta uma árvore através da remoção de uma aresta. E, em seguida, reconecta as duas árvores formadas inserindo uma nova aresta entre um vértice de cada uma delas. Este processo garante uma distância fenotípica mínima.

Já seu operador de *crossover* cria uma partição do conjunto de vértices em dois conjuntos, criando quatro subgrafos induzidos por estes vértices, dois para cada genitor. Em seguida, cada uma das duas proles, recebe um subgrafo de cada genitor para formar o conjunto de vértices original. Posteriormente, são inseridas arestas para reconectar o grafo, formando uma árvore, de modo que somente são permitidas as arestas que já estavam presentes nos genitores.

O Edge-Set se baseia em três algoritmos PrimRST, RandWalkRST e KruskalRST. Estes podem ser utilizados para inicialização e *crossover*. O KruskalRST é o mais recomendável quando não se quer viés a árvores do tipo estrela. Ele é similar ao algoritmo de Kruskal [30], porém a aresta que compõe a árvore é escolhida aleatoriamente.

No mecanismo de *crossover*, o Edge-Set realiza a união das arestas dos dois genitores e utiliza um dos três algoritmos para escolher as arestas finais. Seu operador de mutação se assemelha ao NetDir, no sentido em que remove uma aresta e depois reconecta as duas árvores formadas, mas também há uma variante em que ele adiciona uma aresta, criando um ciclo no grafo, e depois remove uma aresta do ciclo formado.

Os números Prüfer consistem em um código de $n-2$ números de 1 a n que representam todas as árvores de n vértices conforme o teorema de Cayley [9]. O CV mapeia todas as possíveis arestas do grafo no cromossomo, recebendo valor 0 e 1 para as arestas ausentes e presentes. Por esse motivo, o CV sofre em garantir que o grafo formado seja uma árvore, necessitando de mecanismos de correções.

O LNB utiliza uma matriz, um vetor e dois pesos para determinar a distâncias entre os vértices. Com base nessas distâncias, calcula-se a MST. Por fim, o NetKey, surge como uma solução para a CV, no qual os valores do gene são pesos assim como na LNB. O NetKey define uma decodificação do cromossomo de forma que o grafo final seja uma árvore.

Apesar das várias representações supracitadas, as representações por Nó-Profundidade tem apresentado resultados relevantes do ponto de vista de qualidade da solução e desempenho computacional [18, 13, 17]. Além disso, essas representações foram utilizadas em uma série de trabalhos na área de sistemas de potência [6, 8]. Assim, essa representação mostra-se uma forma mais intuitiva para codificar florestas para o objetivo deste trabalho.

As Representações Nó-Profundidade (RNPs), inicialmente proposta por Delbem et al. [18], é uma forma de representação indireta de árvores. A versão mais recente das RNPs é a proposta por Lima et al. [13], a node-depth phylogenetic-based encoding (NPE) e os conceitos serão apresentados com base nela, mas sendo replicáveis nas demais RNPs.

Fundamentalmente, a RNP é um vetor que retém a profundidade e o rótulo do vértice da árvore. E para sua construção é realizada uma busca em profundidade. A figura 2.8 apresenta a codificação de uma árvore, com raiz no vértice 1. Por este motivo, este vértice tem profundidade 0. Em seguida, o vértice 2 tem profundidade 1, o vértice 8 tem profundidade 2 e assim por diante.

As RNPs possuem dois operadores principais: o Subtree Pruning and Regrafting on NPE (SPRN) e o Tree Bisection and Reconnection on NPE (TBRN), fundamentados nas definições de Subtree Pruning and Regrafting (SPR) e Tree Bisection and Reconnection (TBR) [25].

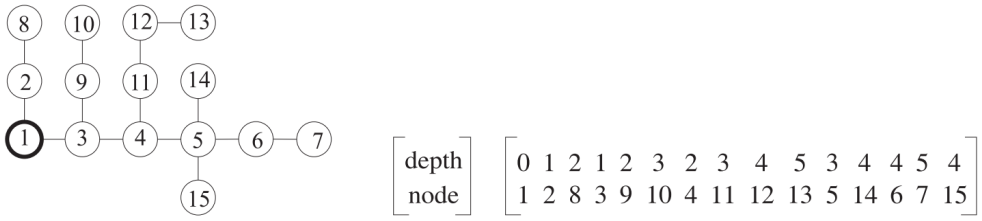
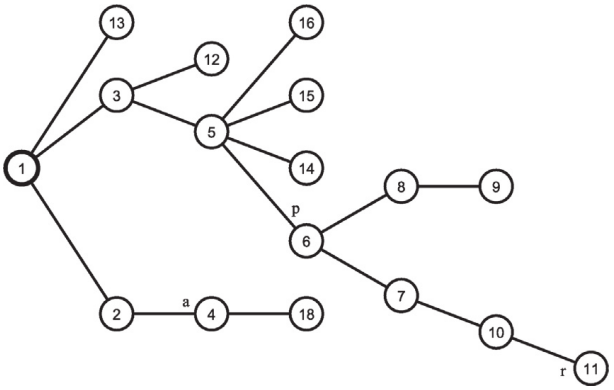
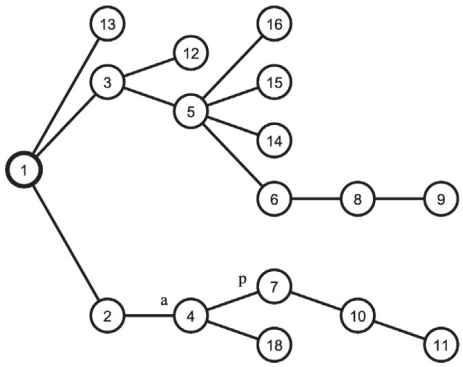


Figura 2.8: *Representação do NPE. Fonte: Lima et al. [13].*

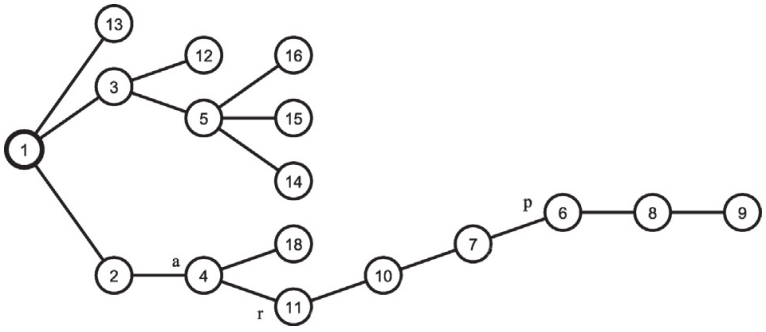
A figura 2.9(a) apresenta uma árvore com três vértices escolhidos a, p e r para aplicação destes operadores. A figura 2.9(b) apresenta a aplicação do SPRN. Este operador poda a subárvore do vértice p e a adiciona no vértice a, nota-se que apenas os vértices a e p constituem entrada deste operador. Já a figura 2.9(c) apresenta a aplicação do TBRN. Nele, acontece a mesma poda ocorrida no SPRN no vértice p, mas ao invés de adicionar a subárvore do vértice p da mesma forma como foi podada, assim como acontece no SPRN, ele reinsere a subárvore a partir do vértice r.



(a) *Árvore com os vértices a, p e r escolhidos.*



(b) *Árvore após a aplicação do SPRN.*



(c) *Árvore após a aplicação do TBRN.*

Figura 2.9: *Árvore e ação dos operadores da RNP. (a) árvore antes de aplicação; (b) ação do SPRN; (c) ação do TBRN. Fonte: Lima et al. [13].*

Em Lima et al. [13] é apresentado um algoritmo de decodificação com complexidade polinomial: $O(|N|)$, onde $|N|$ é o tamanho do genótipo. Além disto, apresentam

propriedades do NPE quanto a localidade, a factibilidade, a complexidade e os vieses dos operadores. Utilizando a distância de Hamming o estudo demonstrou que a medida d_m, proposta por Rothlauf [45], é igual a zero. Indicando que, todos os genótipos vizinhos correspondem a fenótipos vizinhos, consequentemente, a localidade é perfeita. Uma observação relevante é que os operadores SPRN e TBRN só geram soluções factíveis, uma vez que só é possível inserir um gene de profundidade d se houver um vértice pai de profundidade $d-1$. Além disto, a complexidade de ambos operadores, SPRN e TBRN, foi determinada em $O(|N|)$. E, em relação ao viés, nenhum dos operadores apresentou tendência quando comparados à distância em relação a uma MST e a árvores estrela.

2.7 Considerações finais

Neste trabalho escolhemos a CE para a geração de Sistemas de Distribuição de Energia Elétrica (SDEs) sintéticos porque o espaço de busca é muito grande, os objetivos são complexos e conflitantes, e é neste campo que a CE tem se destacado. Assim, espera-se obter resultados adequados em um tempo computacional viável.

Entre os processos evolutivos, optou-se pelo NSGA-II por causa da qualidade dos resultados obtidos por este algoritmo em diversos trabalhos científicos [12, 50, 34, 33]. Já a representação NPE foi escolhida por possuir um excelente desempenho computacional e por ser amplamente utilizada em problemas que envolvem SDEs [13, 7, 8, 6, 37, 35, 17, 50, 48].

Deste modo, o próximo capítulo apresenta a proposta deste trabalho para a geração de SDEs sintéticas utilizando estas tecnologias junto com novas metodologias para definição da estrutura das redes elétricas e posicionamento dos consumidores e chaves.

CAPÍTULO 3

Sistemas de Distribuição de Energia Elétrica e Análise Estatística

Neste capítulo será descrita a modelagem de um SDE e as análises estatísticas para o entendimento das conexões entre os componentes de uma rede elétrica. Em busca deste entendimento de como os componentes de uma SDE são organizados e conectados uns aos outros, foi realizado um estudo prévio da topologia do SDE da cidade de Londrina-PR utilizada no projeto de Pesquisa e Desenvolvimento (P&D) Aneel 2866-0272/2013. Mais tarde, foram analisados os dados topológicos do estado do Paraná durante o projeto de P&D Aneel 2866-0484/2017 e 2866-0504/2018. Esta compreensão da organização dos componentes foi necessária porque as distribuições estatísticas e proporções obtidas deste estudo serviram de objetivo para o processo de geração de SDEs sintéticas.

Este capítulo está estruturado da seguinte forma. A seção 3.1 apresenta os principais elementos de uma SDE utilizados neste trabalho. A seção 3.2 apresenta como os elementos da SDE foram modelados computacionalmente. A seção 3.3 apresenta as análises estatísticas realizadas com base nos dados do estado do Paraná.

3.1 Elementos de um SDE

Embora existam diversos componentes em uma SDE incluindo diferentes fabricantes e materiais de fabricação, estes componentes podem ser agrupados quanto a sua aplicação na rede elétrica da seguinte forma: postes, trechos, chaves, postos transformadores, capacitores, reguladores de tensão e alimentadores. As seções a seguir apresentam estes componentes instalados em SDEs reais.

3.1.1 Postes

Os postes são estruturas da rede aérea que servem de suporte aos trechos (seção 3.1.2) e nos quais são instalados diversos elementos de um SDE, tais como chaves (seção

3.1.3) e postos transformadores (seção 3.1.4), além da própria iluminação pública, cabos da rede de Internet e televisão. A figura 3.1 apresenta um poste com dois circuitos de alimentadores passando por ele.

Figura 3.1: *Poste com 4 trechos primários para outros postes.*

3.1.2 Trechos

Os trechos, apresentados na figura 3.2, são condutores que interligam dois postes. Os trechos primários estão localizados em sua parte superior do poste enquanto que os trechos secundários estão localizados na parte intermediária do poste. A diferença entre estes condutores pode ocorrer de várias formas sendo as principais o material de composição e bitola do cabo. Além disso, nos trechos primários passam corrente elétrica em uma tensão de 13,8kV ou 34,5kV [5], enquanto que nos trechos secundários essa tensão varia de 220/127V ou 380/220V [36].

Figura 3.2: *Trechos primários e secundários.*

3.1.3 Chaves

As chaves (figura 3.3) são dispositivos capazes de estabelecer ou interromper o fluxo da corrente elétrica. Elas são classificadas em relação a sua posição original no circuito, sendo normalmente abertas ou normalmente fechadas. Quando a configuração original determina que a chave deve permanecer aberta, tem-se uma chave normalmente aberta (NA), já quando a configuração original determina que a chave deve permanecer fechada, tem-se uma chave normalmente fechada (NF) [5].

(a) *Chave seccionadora unipolar fechada.* (b) *Chave de proteção fusível fechada.*

Figura 3.3: *Chaves.*

3.1.4 Postos transformadores

Os postos transformadores (figura 3.4) reduzem a tensão presente nos trechos primários, 13,8kV ou 34,5kV, para a tensão dos trechos secundários, 220/127V ou 380/220V, que é requerida pela maioria dos consumidores finais [5]. São nos postos transformadores que é calculada a carga demandada pelo sistema para alimentar o circuito secundário associado.

Figura 3.4: *Posto transformador.*

3.1.5 Banco de Capacitores

Os bancos de capacitores (figura 3.5) são fontes de energia reativa utilizados para redução de perdas de energia, correção dos perfis de tensão, controle dos fluxos de potência, melhoria do fator de potência, aumento da capacidade dos sistemas e dependendo do local de instalação ajudam na regulação de tensão e redução da corrente que circula à montante nos condutores [5, 66].

Figura 3.5: *Banco de capacitores.*

3.1.6 Reguladores de tensão

Os reguladores de tensão (figura 3.6) são autotransformadores utilizados para elevar ou abaixar a tensão em um intervalo de 10% nas barras à jusante da instalação, podendo operar de modo manual ou automático. Geralmente os reguladores de tensão são utilizados para elevar a tensão à jusante [5, 66].

Figura 3.6: *Regulador de tensão.*

3.1.7 Alimentadores

Um alimentador é o conjunto de componentes elétricos que estão interligados em um mesmo circuito incluindo todos os elementos anteriormente citados. Desta forma pode-se dizer que um alimentador é formado por postes, trechos, chaves, capacitores, reguladores de tensão, postos transformadores, etc. Seu início se dá na subestação que abaixa a tensão vinda da linha de transmissão, como apresentado na figura 3.7.

Figura 3.7: *Início de alimentadores.*

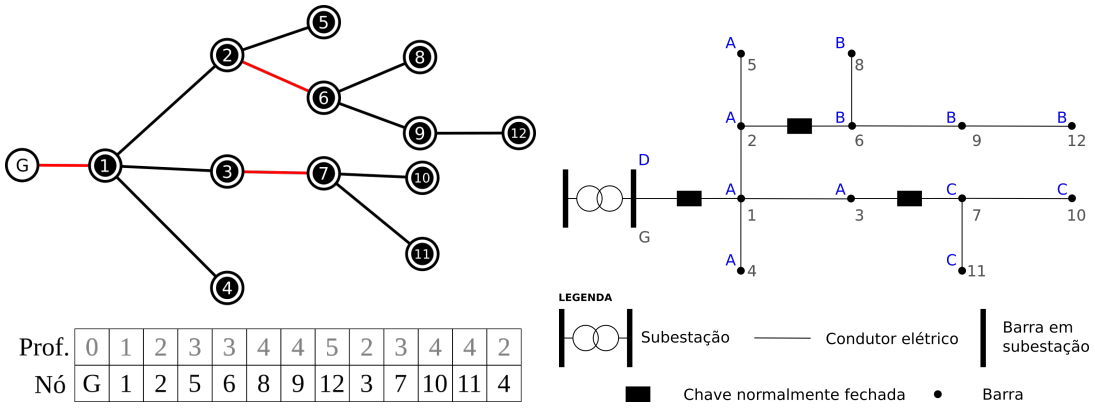
3.2 A modelagem do SDE

O modelo barra-linha é a forma como os grafos dos SDEs são representados neste projeto. Ele é uma expansão de como as ligações físicas dos postes acontecem e permite o entendimento do fluxo elétrico da rede em pontos que possuem chaves e outros equipamentos elétricos. Na representação de grafos, as barras de carga ou de passagem se tornam vértices do grafo. Já as chaves, os trechos primários e os reguladores de tensão são representados como arestas. E, por fim, as informações do transformador da subestação são armazenadas como atributos do vértice raiz do alimentador e os transformadores da rede primária para a secundária são representados como barras de carga (figura 3.9).

Uma definição importante na modelagem dos SDEs é o setor. Neste trabalho, um setor é a unidade mínima de manobra dos SDEs, ou seja, todos os nós contidos no setor sempre terão o mesmo estado: energizado ou não energizado. Do ponto de vista de grafos, eles são árvores enraizadas que podem possuir chaves nas folhas ou no vértice raiz. A figura 3.8 apresenta um alimentador representado do ponto de vista de grafos

e usando notações da engenharia elétrica. Na figura 3.8(a) as arestas em vermelho são as chaves normalmente fechadas e as arestas em preto são trechos primários. A partir da definição de setor apresentada é possível identificar que na figura existem 4 setores. Estes são constituídos pelos nós: $A = \{1,2,3,4,5\}, B = \{6,8,9,12\}, C = \{7,10,11\}, D = \{G\}$. A figura 3.8(b) apresenta o diagrama elétrico do alimentador onde as chaves são representadas como retângulos e as barras como pontos de conexão.

Nota-se que os trechos primários, representados pelas arestas, são componentes dos setores, por exemplo o trecho $(1,2)$ pertence ao setor A e o $(7,10)$ está contido no setor C. No entanto, as chaves não possuem um setor definido, pois possuem a função de interligar dois setores, como o caso da chave $(3,7)$ que liga os setores A e C.



(a) *Representação de alimentador como grafo e sua NPE.*

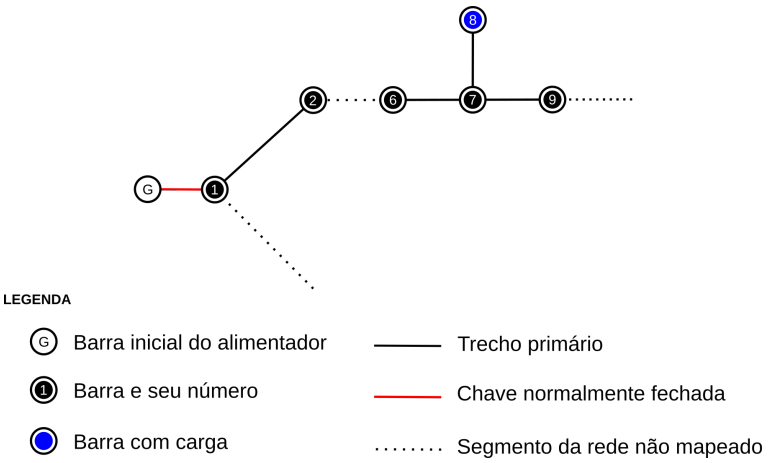
(b) *Representação de um alimentador de um SDE do ponto de vista elétrico.*

Figura 3.8: *Exemplo de alimentador composto por quatro setores do ponto de vista de grafos e do esquema elétrico.*

A figura 3.9 apresenta um cenário real, figura 3.9(a), e seu respectivo mapeamento usando o modelo barra-linha, figura 3.9(b). Neste cenário a energia vem através do circuito da barra inicial do alimentador e chega na barra numerada como 6. Nota-se que este número foi escolhido para simplificar o exemplo, mas na realidade existem vários componentes elétricos antes desta barra que estão representados pela linha pontilhada na modelagem, figura 3.9(b). Da barra 6, a rede primária continua até chegar na barra 7, no qual está instalado um posto transformador. Este posto transformador constitui a barra 8, convertendo a tensão da rede primária para a rede secundária. Além disso, é importante observar que a rede primária continua da barra 7 para uma outra barra numerada como 9, sendo esta barra correspondente a outro poste adjacente ao poste onde está instalado o transformador. Na figura 3.9(a) só é possível visualizar o cabeamento da rede primária saindo da barra 7 em direção a barra 9. Este exemplo de modelagem, mostra como as barras de carga sempre formam vértices considerados folhas no grafo, ou seja, vértices

com apenas uma ligação que não sejam o vértice raiz da árvore, como pode ser visto na figura 3.9(b).

(a) *Cenário real com as barras numeradas.*



(b) *Modelagem do cenário.*

Figura 3.9: *Exemplo modelagem de um posto transformador com bifurcação.*

3.3 Análise estatística dos SDEs

Após a modelagem e carregamento do SDE foi realizada a análise estatística da rede elétrica. Como os dados da cidade de Londrina é uma amostra dos dados do estado do Paraná, os dados de Londrina serão omitidos. Acredita-se que este estudo das propriedades presentes nos SDEs seja um passo fundamental para a síntese de redes elétricas realistas.

Deste modo, foi realizada uma análise descritiva (seção 3.3.1) dos aspectos topológicos da rede elétrica como: o número de barras, o número de folhas e o grau das barras, assim como as correlações entre estas propriedades (seção 3.3.2).

Todas as análises foram realizadas no estado normal das chaves. Como os SDEs radiais formam árvores em seu estado normal de operação, o número de arestas pode ser prontamente obtido pela equação $m = n - 1$. Logo esta propriedade não foi considerada diretamente nas análises.

3.3.1 Análise descritiva

Os estudos a seguir apresentam, na forma de gráficos, as distribuições de frequência do estado do Paraná relacionadas ao número e grau das barras, número de folhas e número de chaves por alimentador e por setor. A definição das classes de cada distribuição ocorreu através da observação dos dados da cidade de Londrina, de modo que 90% da população estivesse entre a primeira e a penúltima classe. Assim, a última classe definiu este limiar até o valor máximo observado.

Na análise descritiva buscou-se entender a magnitude dos dados como número de: alimentadores, barras, folhas, consumidores e carga, do SDE assim como suas proporções e distribuições.

As análises descritivas a seguir estão divididas em duas seções. A primeira apresenta as distribuições de frequência e proporções encontradas para o estado do Paraná. E a segunda apresenta as mesmas análises segmentando os alimentadores em puramente urbanos, puramente rurais e mistos. Considerou-se alimentadores puramente urbanos aqueles cujos pontos significativos eram todos classificados como urbanos. Sendo que os alimentadores puramente rurais seguiram a mesma analogia. Já os alimentadores mistos foram aqueles com a presença de pontos significativos urbanos e rurais.

Análises do estado do Paraná

A figura 3.10 apresenta a frequência relativa dos graus das barras da rede elétrica[1]. Nesta figura, 83,1% das barras do Paraná, possui grau 1 a 2. Já as barras de grau 3 e 4 representaram 16,9%. Além das folhas, também se observou que o grau da raiz do alimentador foi sempre 1.

[1]Nos gráficos a seguir, o intervalo é aberto para a contagem do histograma, com exceção para o último intervalo.

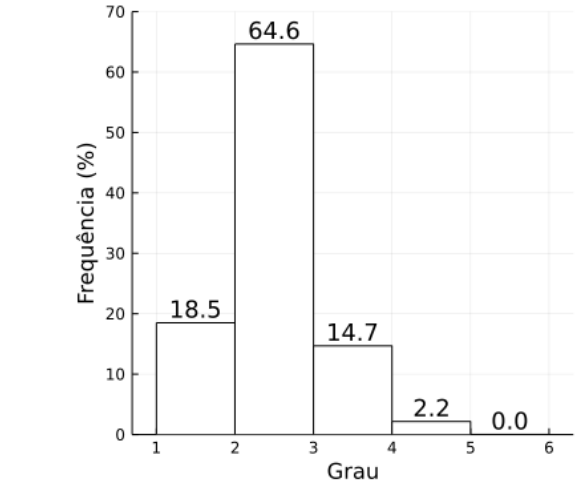


Figura 3.10: *Frequência relativa do grau das barras.*

A figura 3.11 apresenta a frequência relativa do número de barras por alimentador. No Paraná, 33,9% dos alimentadores possuem entre 1.251 a 17.198 barras. Além disso, ocorre uma redução de frequência entre a segunda e quinta classe da distribuição. As cinco primeiras classes para o estado do Paraná representam 66,2% dos alimentadores.

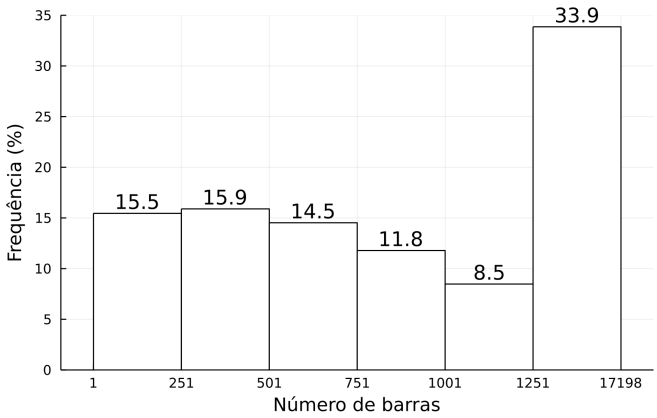


Figura 3.11: *Número de barras na árvore dos alimentadores.*

A figura 3.12 apresenta a distribuição do número de folhas por alimentador. As barras consideradas como folhas são as barras finais da árvore formada pelo alimentador, elas possuem grau 1, ou seja, estão ligadas somente a uma outra barra e são nelas que os consumidores estão localizados, por isso são importantes neste contexto. Esta mesma distribuição para a cidade de Londrina apresentou uma redução entre a primeira e última classe, no entanto, nenhum padrão pode ser observado para o estado do Paraná. Isto, provavelmente se deve a grande variedade de topologias presentes no estado. Nesta distribuição, é possível observar que 57,6% dos alimentadores possuem até 199 folhas.

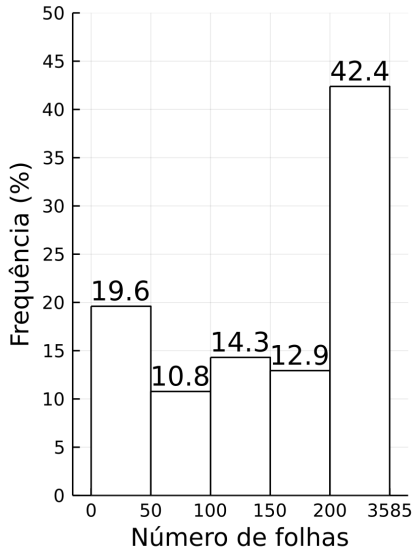


Figura 3.12: *Número de folhas na árvore dos alimentadores.*

A figura 3.13 apresenta a distribuição do número de barras por setor. No estado, 88,8% dos setores possuem de 1 a 40 barras. Além disso, é possível perceber uma redução entre a primeira e penúltima classe.

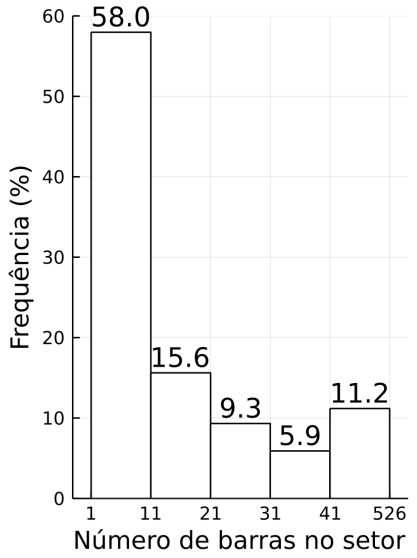


Figura 3.13: *Número de barras por setor.*

A figura 3.14 apresenta a distribuição do número de chaves por setor. No estado os setores com 1 a 5 chaves representam 93,1% dos casos. Nesta distribuição é possível perceber uma redução entre a primeira e penúltima classe, ou seja, setores com muitas chaves são menos frequentes. No entanto há setores com até 36 chaves.

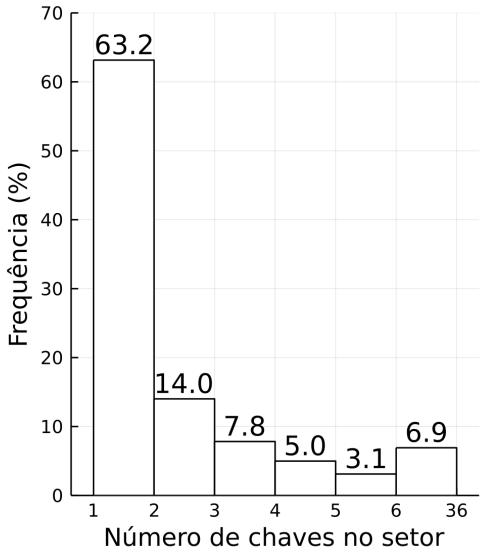


Figura 3.14: *Número de chaves por setor.*

A figura 3.15 apresenta o número de setores por alimentador. Nota-se que não há um padrão de redução ou crescimento claro na distribuição. No entanto, é possível observar uma redução entre a terceira e penúltima classe. Esta redução também foi observada para a cidade de Londrina abrangendo desde a segunda até a penúltima classe. No entanto, no estado a segunda classe apresenta apenas 14,7% dos alimentadores.

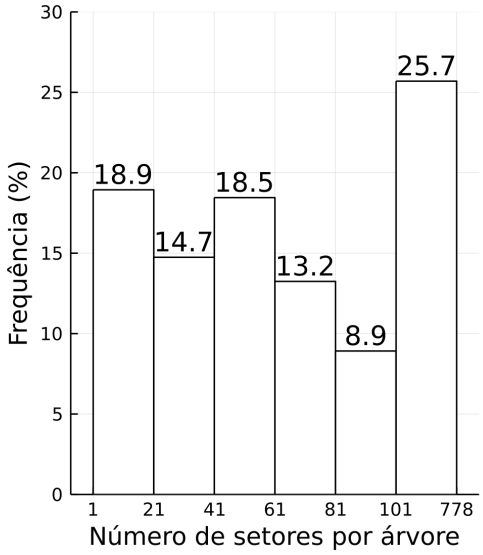


Figura 3.15: *Número de setores por árvore de alimentador.*

A figura 3.16 apresenta a proporção de barras com e sem carga. Observou-se que 13,7% das barras do Paraná possuem carga. Através desde gráfico fica claro que grande parte da infraestrutura de uma SDE consiste em barras para permitir o transporte

da energia elétrica até o consumidor, pois aproximadamente 1 a cada 7 barras possui carga.

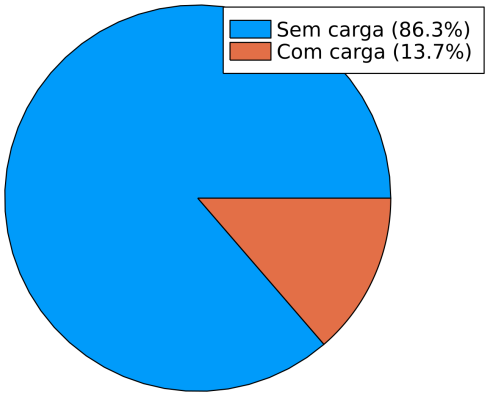


Figura 3.16: *Proporção das barras com e sem carga.*

A figura 3.17 apresenta a distância de uma barra com carga para outra mais próxima. No Paraná, a distância mínima encontrada entre duas barras com carga foi 3, ou seja, entre duas barras com carga quaisquer existe pelo menos duas barras sem carga no caminho. Além disso, as distâncias de 3 a 7 representaram 68,3% dos casos. Observa-se que a ausência de barras com carga possuindo distância de 1 a 2 saltos também se deve ao processo de acumulação de consumidores nos postos transformadores entre as redes primária e secundária.

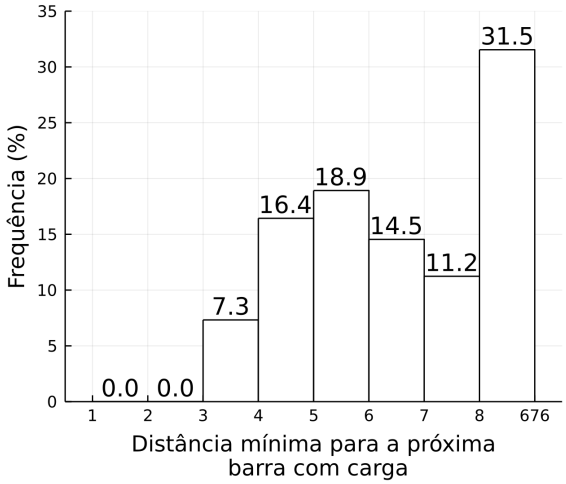


Figura 3.17: *Distância de uma barra com carga para a outra mais próxima.*

A figura 3.18 apresenta o resultado do estudo do posicionamento dos consumidores na SDE. Neste gráfico é possível visualizar que 71,6% das barras folhas possuem consumidores, ou seja, possuem carga. No entanto, 28,4% das barras folhas, não possuem

consumidores. Destas, apenas 3,5%, do total, são folhas contendo uma chave aberta, utilizadas para realização de manobras, e 24,9% não possuem nenhuma chave, utilizadas para ancoragem dos cabos.

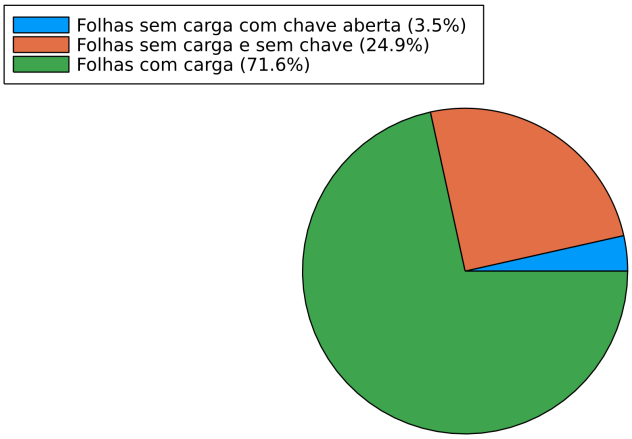


Figura 3.18: *Proporção de folhas com carga, sem carga com chave aberta e sem carga sem chave.*

Análises segmentadas por tipo de alimentador

Nas análises a seguir, foram considerados alimentadores urbanos aqueles que só possuíam postes classificados como urbanos, a mesma lógica foi empregada para os alimentadores rurais. Considerou-se alimentadores mistos aqueles que possuíam postes tanto urbanos quanto rurais. Ao todo foram contemplados nesta análise 2.265 alimentadores do estado do Paraná, sendo 817 urbanos, 55 rurais e 1.393 mistos e suas proporções são apresentadas na figura 3.19.

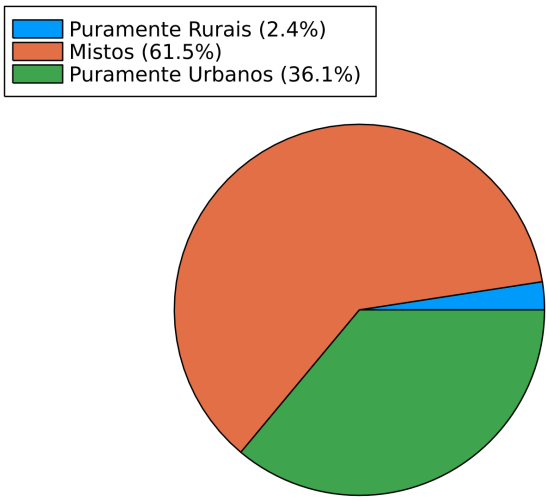


Figura 3.19: *Proporção dos alimentadores quanto ao tipo: urbanos, rurais e mistos.*

A figura 3.20 apresenta a distribuição do grau das barras destes alimentadores. Percebe-se que para os três tipos de alimentadores os graus 1 e 2 apresentam similaridade na frequência, neste caso o grau 1 com valores entre 18,6 e 22,9% das barras e o grau 2 entre 59,4 e 64,7% das barras. A partir do grau 3 ocorre uma redução da proporção das barras, mostrando que tais barras são mais raras nos SDEs estudados. Estas três distribuições se assemelham bastante a distribuição de graus do estado sem segmentação apresentada na figura 3.10.

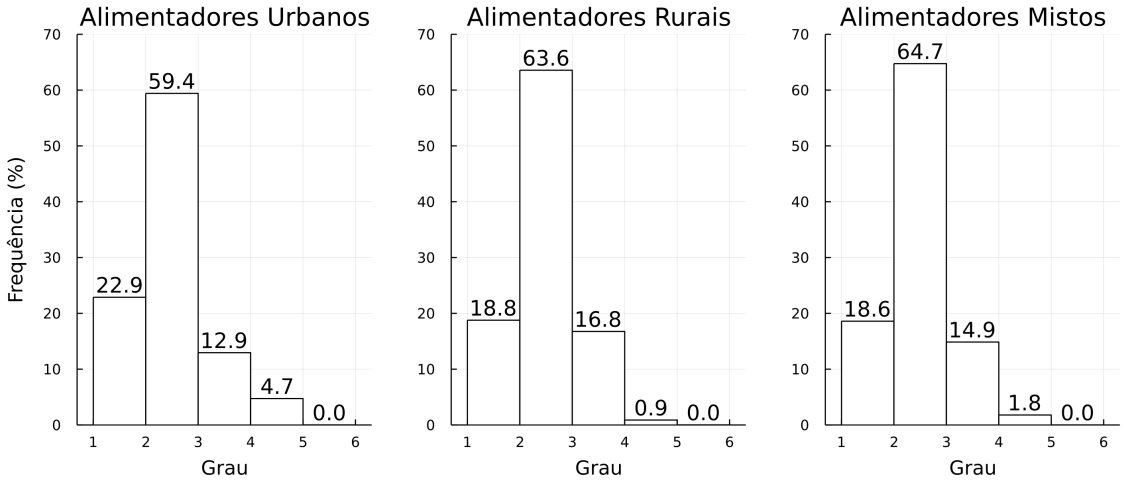


Figura 3.20: *Distribuição do grau das barras dos alimentadores segmentados por tipo: urbanos, rurais e mistos.*

A figura 3.21 apresenta o número de folhas por alimentador. Nela é possível perceber que 43,6% dos alimentadores rurais possuem até 49 folhas enquanto que apenas 30,2% dos alimentadores urbanos estão nessa classe. Além disso, os alimentadores mistos possuem mais folhas que os outros dois tipos, tendo 58,3% dos alimentadores com mais de 200 folhas. Esta grande diferença de frequência entre as classes dos três tipos de alimentadores explica a ausência de padrão da distribuição do estado (figura 3.12).

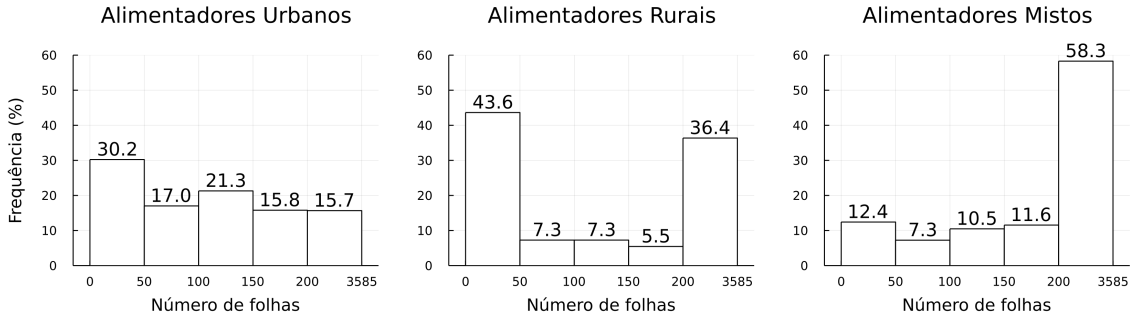


Figura 3.21: *Número de folhas por alimentador segmentados por tipo: urbanos, rurais e mistos.*

A figura 3.22 apresenta o número de barras por alimentador. Nela é possível visualizar que 31% dos alimentadores urbanos e 38,2% dos alimentadores rurais possuem

até 250 barras. Pode-se notar também uma redução forte da frequência à medida que o número de barras aumenta para os alimentadores urbanos e uma redução mais suave para os alimentadores mistos. As análises corroboram o esperado de que alimentadores urbanos possuem um número menor de barras em comparação aos alimentadores classificados como mistos e rurais. Menos de 10% dos alimentadores urbanos possuem mais do que 1.000 barras, esse fato é esperado, pois alimentadores urbanos são curtos e com alta concentração de consumidores numa região menor. Por outro lado, alimentadores mistos e rurais cobrem regiões maiores com as cargas mais distantes e, portanto, com dimensão maior.

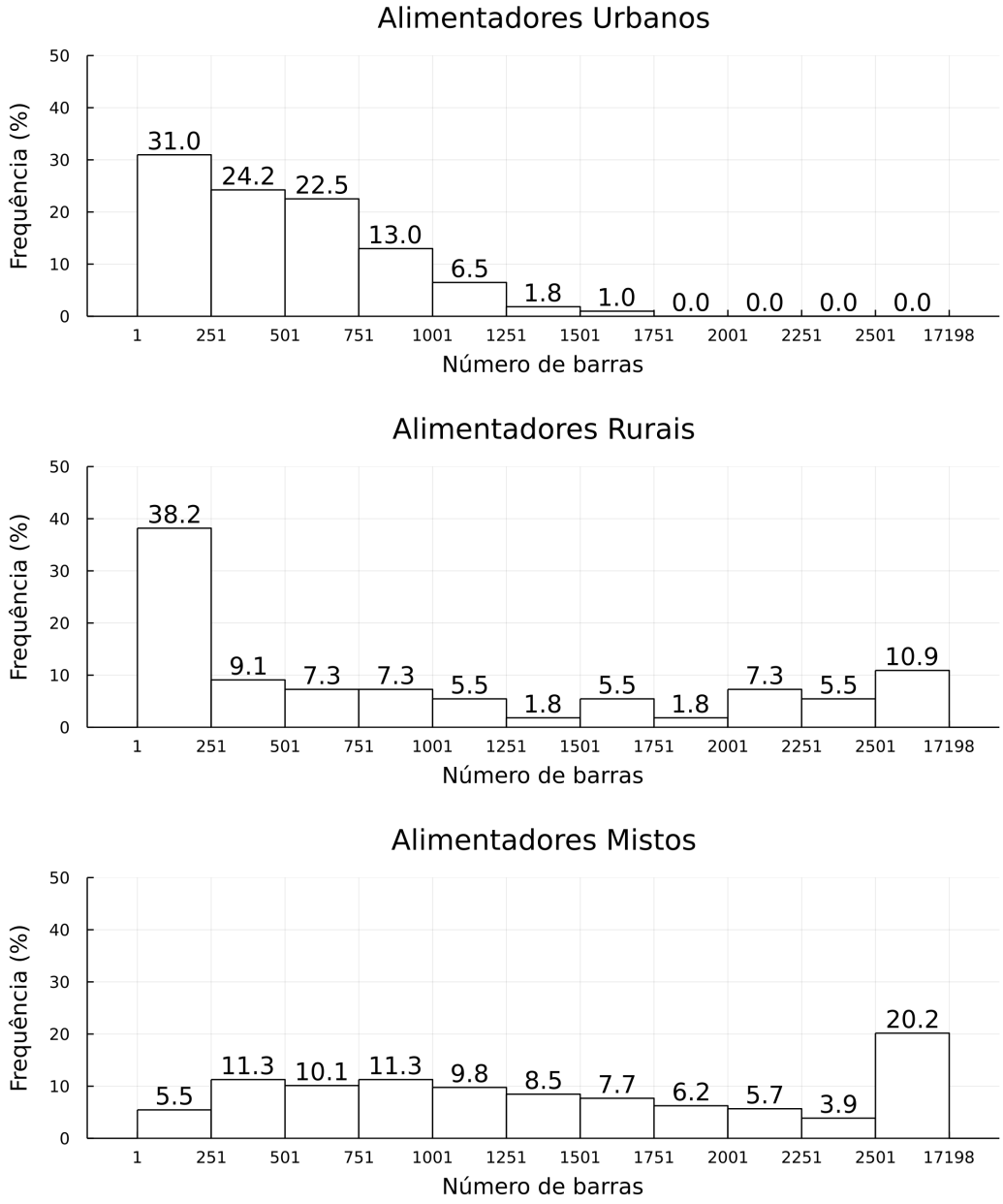


Figura 3.22: *Número de barras por alimentador segmentados por tipo: urbanos, rurais e mistos.*

A figura 3.23 apresenta que mais de 52,7% dos setores possuem até 10 barras independentemente do tipo de alimentador. Em todos os casos ocorre uma redução deste percentual à medida que o número de barras aumenta. É importante notar que, em alimentadores rurais e mistos a cauda, 41 a 526 barras, desta distribuição possui uma proporção quase três vezes maior que a dos alimentadores urbanos. Esse fato se explica devido a distância geográfica coberta por esses tipos de alimentadores com baixa presença de carga, não havendo assim a necessidade da divisão em setores menores.

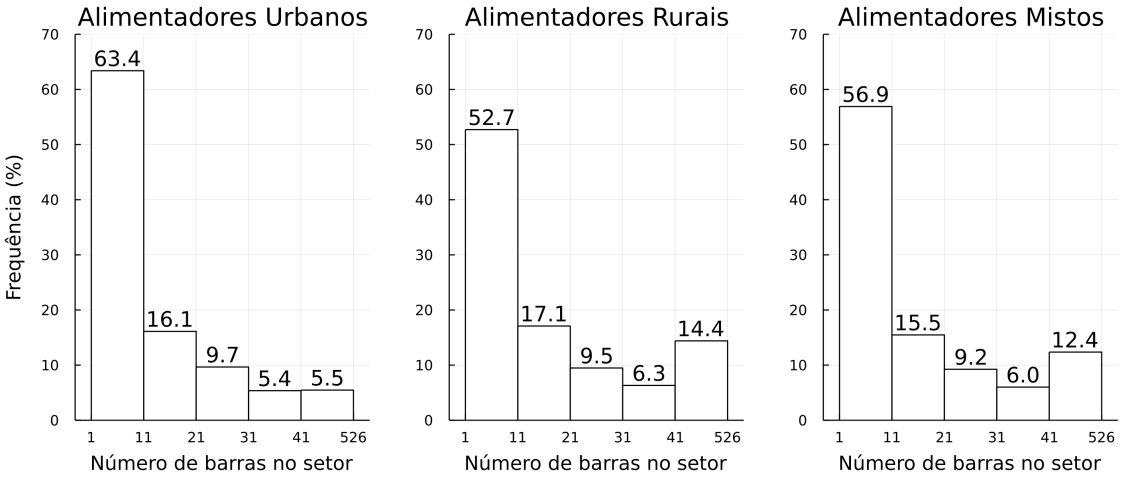


Figura 3.23: *Número de barras por setor com alimentadores segmentados por tipo.*

A figura 3.24 apresenta a análise do número de chaves por setor. Nela é possível notar que mais de 57% dos setores possuem apenas uma chave. O restante da distribuição para os três tipos de alimentadores também se assemelha, possuindo uma redução deste percentual à medida em que o número de chaves aumenta. É importante notar a presença de mais de 6% de setores possuindo entre 6 a 36 chaves independentemente do tipo de alimentador.

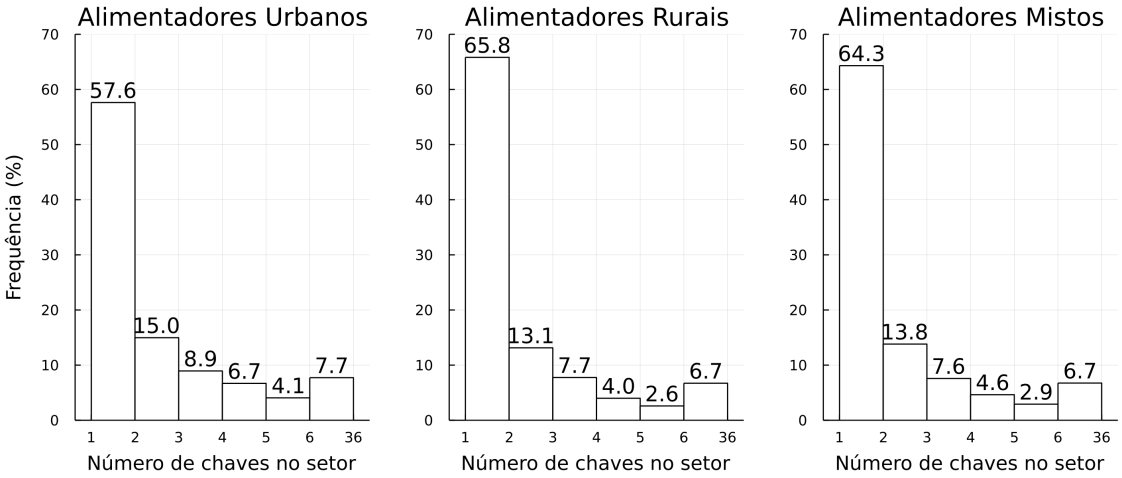


Figura 3.24: *Número de chaves por setor com alimentadores segmentados por tipo.*

A figura 3.25 apresenta o número de setores por alimentador segmentados por tipo de alimentador. Nota-se que 92,4% dos alimentadores urbanos possuem entre 1 a 80 setores. Já os alimentadores rurais possuem 69,1% dos alimentadores na primeira e última classe da distribuição. De forma similar aos alimentadores rurais, 39,6% dos alimentadores mistos possuem entre 101 a 778 setores, no entanto, apenas 11,5% deles possuem entre 1 a 20 setores.

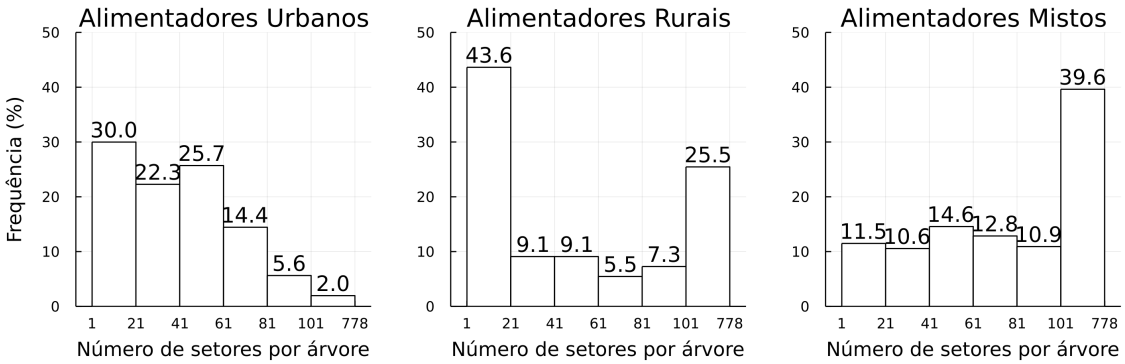


Figura 3.25: *Número de setores por alimentadores segmentados por tipo.*

A figura 3.26 apresenta a proporção de barras com carga dos alimentadores urbanos, rurais e mistos. Nota-se que entre 13,5 a 15% das barras possuem carga nos alimentadores, mostrando um padrão claro para todos os três tipos de alimentadores.

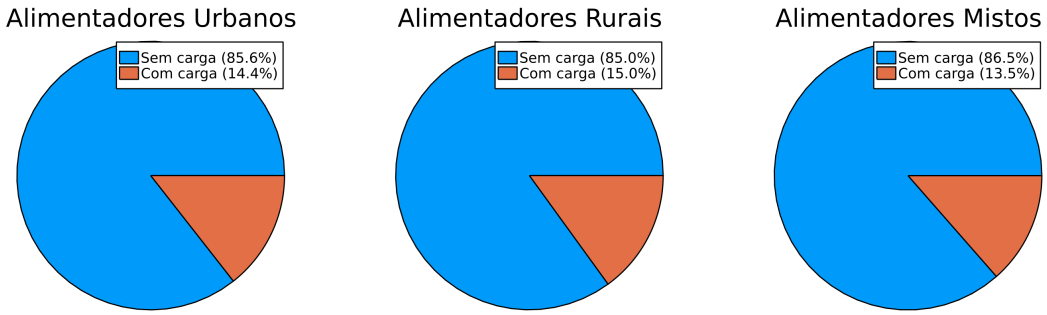


Figura 3.26: *Proporção de barras com e sem carga com alimentadores segmentados por tipo.*

A figura 3.27 apresenta o número de chaves fechadas por tipo de alimentador. Nota-se em todos os casos uma redução do percentual à medida que o número de chaves fechadas aumenta.

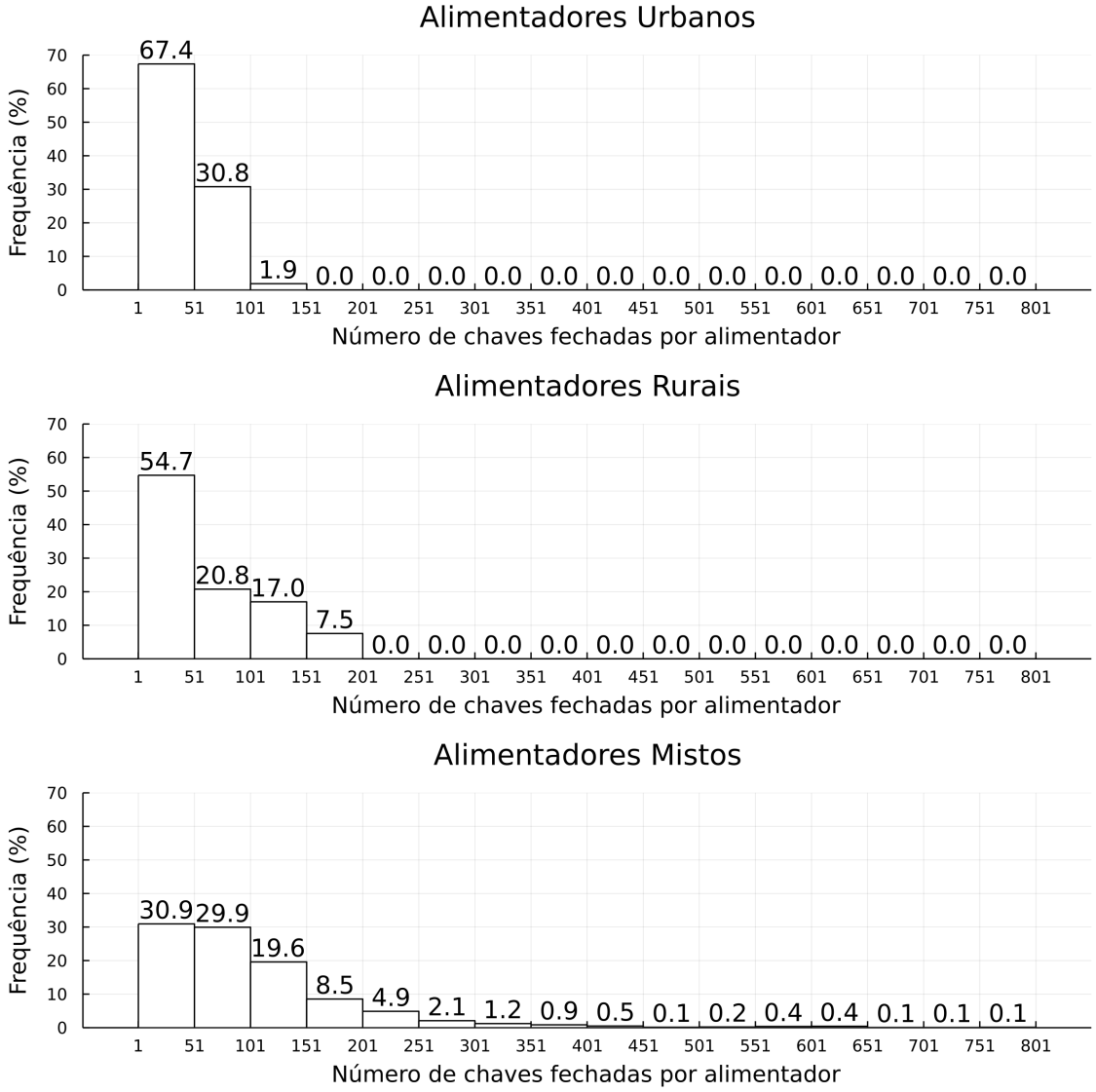


Figura 3.27: *Número de chaves fechadas por alimentadores segmentados por tipo.*

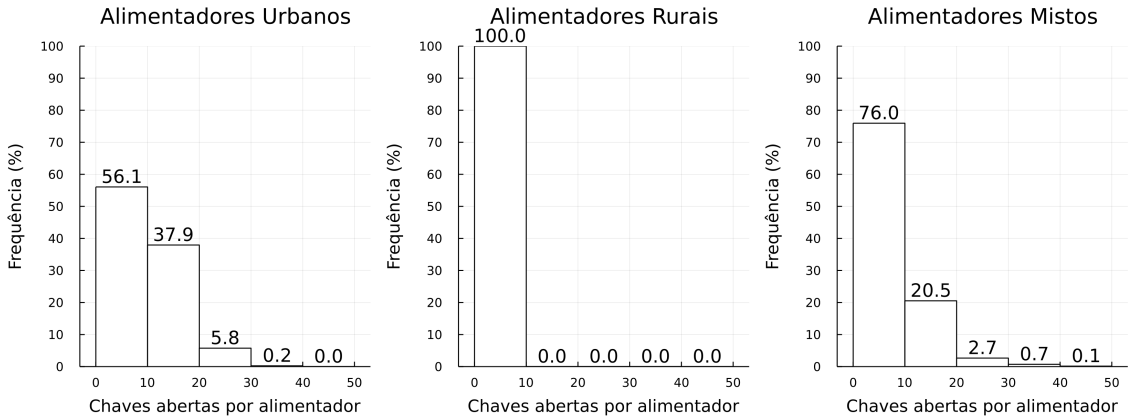


Figura 3.28: *Número de chaves abertas por alimentadores segmentados por tipo.*

A figura 3.28 apresenta o número de chaves abertas por tipo de alimentador. Nota-se uma redução da proporção a medida em que o número de chaves abertas aumenta para os alimentadores urbanos e mistos. Já todos os alimentadores rurais possuem no máximo 9 chaves abertas. A figura 3.29 apresenta o detalhamento deste percentual, subdividindo a classe 0 a 10 dos alimentadores rurais. Deste modo é possível observar que 58,2% dos alimentadores rurais possuem até 2 chaves abertas. Esse resultado é coerente, visto que alimentadores rurais estão presente em áreas remotas e com poucas opções de interconexão com outros alimentadores.

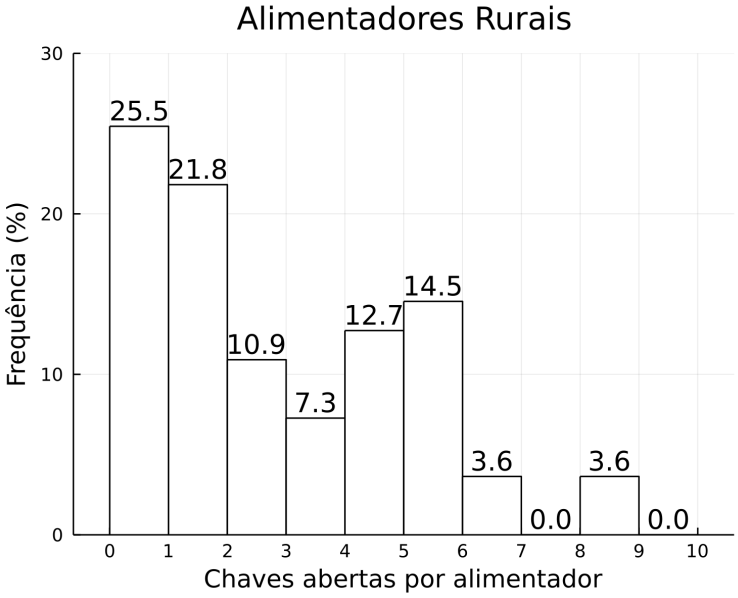


Figura 3.29: *Número de chaves abertas dos alimentadores rurais considerando até 10 chaves.*

Por fim, a tabela 3.1 apresenta as distribuições de frequência estudadas neste trabalho categorizadas em topológicas, posicionamento de chaves e carga. As categorias com a mesma letra são interdependentes, já as categorias com letras diferentes podem ser otimizadas separadamente. Por exemplo, dada uma rede com topologia adequada, pode-se escolher quais arestas serão chaves ou determinar quais barras conterão informações de carga sem interferir na qualidade das métricas relacionadas à topologia. No entanto, ao adicionar uma barra em um alimentador, todas as distribuições topológicas podem ser afetadas.

Tabela 3.1: *Categoria para cada distribuição. T: topológico. P: posicionamento de chaves. C: carga.*

Distribuição de frequência relativa	Categoria
Grau das barras	T
Número de barras por alimentador	T
Número de folhas por alimentador	T
Número de barras por setor	P
Número de chaves por setor	P
Número de setores por alimentador	P
Razão barras com carga pelas barras sem carga	C
Número de barras com carga por alimentador	C
Número de barras sem carga por alimentador	C
Distância da barra com carga para outra mais próxima	C
Distância da barra com carga para uma sem carga mais próxima	C

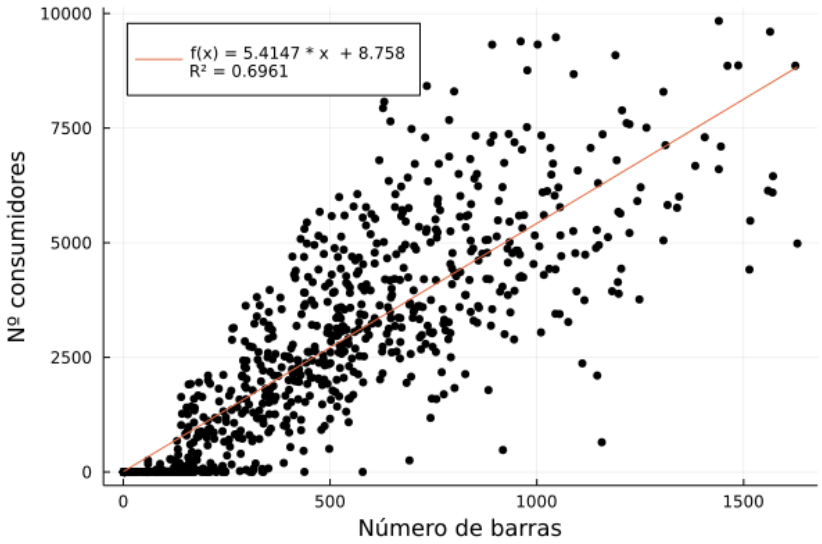
3.3.2 Análise de regressão

Além dos estudos descritivos, também objetivou-se encontrar relações entre o número de consumidores e o tamanho do alimentador, assim como o número de chaves normalmente abertas e normalmente fechadas em função de seu tamanho. Adicionalmente tentou-se estabelecer a relação de potência mensal demandada em função do número de consumidores no alimentador.

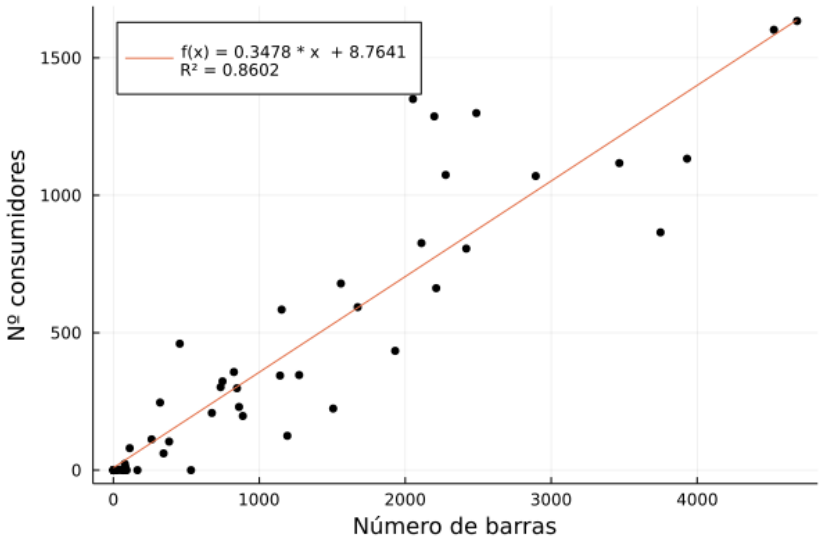
Número de consumidores pelo número de barras em um alimentador

Ao realizar a relação entre o número de consumidores e o número de barras em um alimentador o coeficiente de correlação de Pearson (r) foi igual a $0,21$, indicando uma correlação fraca ($0,20 \leq r < 0,40$).

No entanto, ao realizar essa mesma análise segmentando os alimentadores em urbanos, rurais e mistos é possível perceber relações entre estas duas variáveis. A figura 3.30 apresenta as regressões para os alimentadores urbanos e rurais. Para os alimentadores urbanos, o $r = 0,83$, figura 3.30(a). O coeficiente de correlação de Pearson (r) pode ser obtido através da raiz quadrada do coeficiente de determinação (R^2). Quanto menor R^2 mais fraca é a correlação, sendo que a correlação perfeita $R^2 = 1$ independente se a relação é diretamente ou inversamente proporcional. Assim a correlação entre o número de barras e o número de consumidores para alimentadores urbanos é considerada forte ($0,70 \leq r < 0,90$). Já para os alimentadores rurais, o $r = 0,93$, figura 3.30(b), é considerada muito forte ($r \geq 0,90$). Para os alimentadores mistos, o $r = 0,31$, logo a correlação é considerada fraca.



(a) *Alimentadores urbanos.*



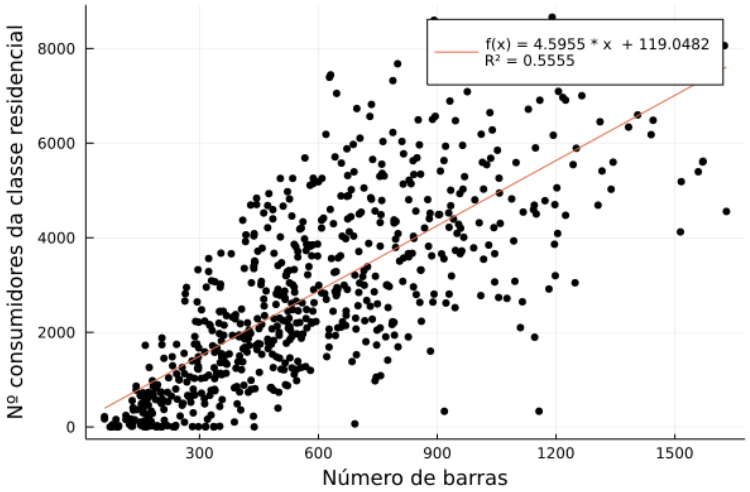
(b) *Alimentadores rurais.*

Figura 3.30: *Regressão do número de consumidores pelo número de barras do alimentador.*

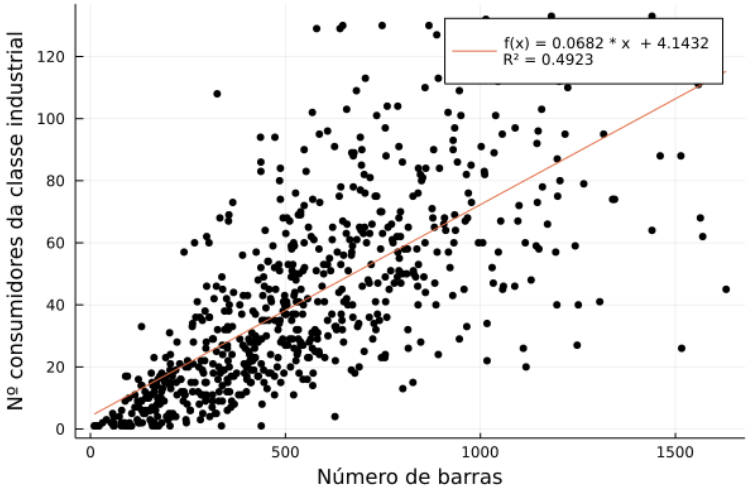
Dado que a correlação do número de consumidores com o número de barras dos alimentadores urbanos é forte e eles representam 36,1% dos dados dos alimentadores optou-se por segmentar os alimentadores focando nos alimentadores urbanos, visto que não é possível determinar o número de consumidores para todos os tipos de alimentadores apenas com o tamanho do alimentador. Deste modo, acredita-se que a inferência do número de consumidores para os alimentadores mistos deve observar outros fatores além do número de barras do alimentador.

Determinação do número de consumidores residenciais, comerciais e industriais em alimentadores urbanos

A figura 3.31 apresenta a regressão do número de consumidores residenciais e industriais pelo número de barras nos alimentadores urbanos. A interpretação do coeficiente de correlação de Pearson (r) para as duas regressões foi considerada forte ($0,70 \leq r < 0,90$), pois $r \geq 0,70$ para as duas regressões, ou seja, é seguro determinar o número de consumidores de cada classe conforme o número de barras do alimentador urbano. No entanto, a correlação para o número de consumidores comerciais encontrada foi fraca ($r = 0,36$). Dado esta correlação fraca para os consumidores comerciais, optou-se por não proceder essa estratificação dos consumidores em residenciais, comerciais e industriais.



(a) *Consumidores residenciais ($r = 0,75$).*



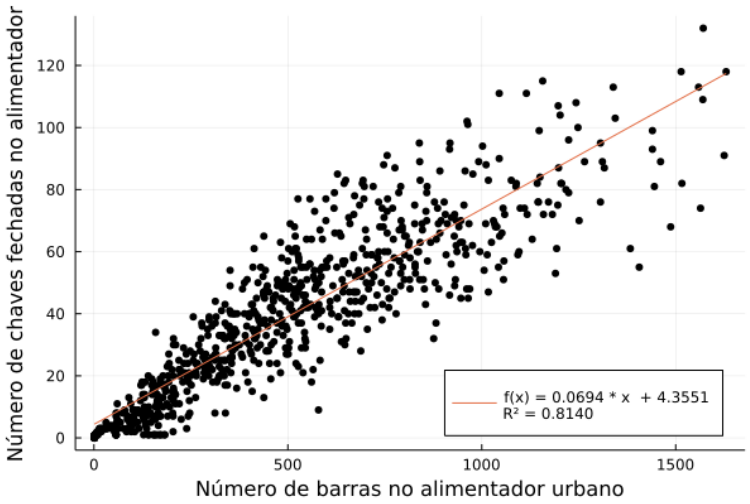
(b) *Consumidores industriais ($r = 0,70$).*

Figura 3.31: *Regressão do número de consumidores pelo número de barras do alimentador urbano.*

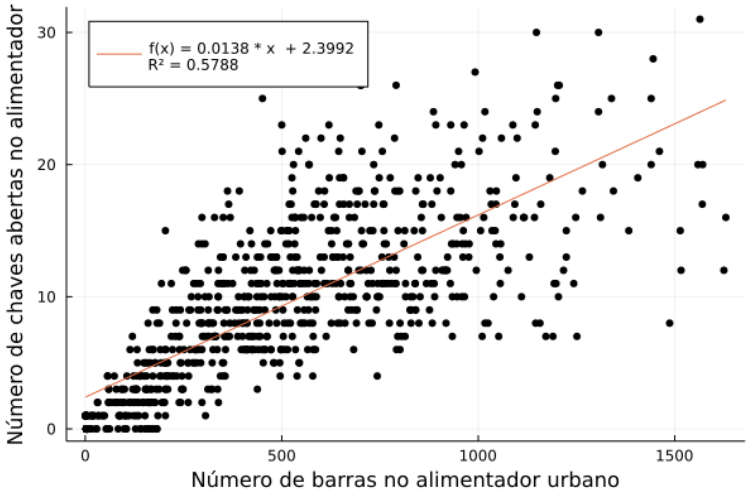
Número de chaves pelo tamanho do alimentador

As figuras 3.32(a) e 3.32(b) apresentam, respectivamente, regressões do número de chaves normalmente fechadas e normalmente abertas em função do número de barras dos alimentadores urbanos. A regressão do número de chaves fechadas, teve $r = 0,90$, logo sua correlação é considerada muito forte ($r \geq 0,90$). Já a regressão do número de chaves abertas, teve $r = 0,76$, resultando em uma correlação forte ($0,70 \leq r < 0,90$).

Em relação ao número de chaves fechadas, foi observado que a primeira aresta, que liga a primeira com a segunda barra, de todos os alimentadores sempre era uma chave normalmente fechada. Isso ocorre devido ao fato de que essas barras representam a ligação do alimentador com o trafo da subestação.



(a) *Chaves normalmente fechadas ($r = 0,90$).*

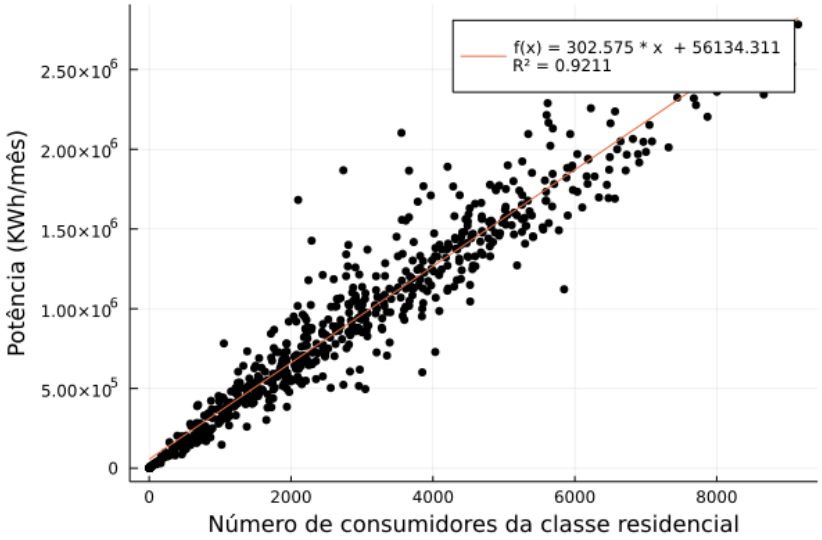


(b) *Chaves normalmente abertas ($r = 0,76$).*

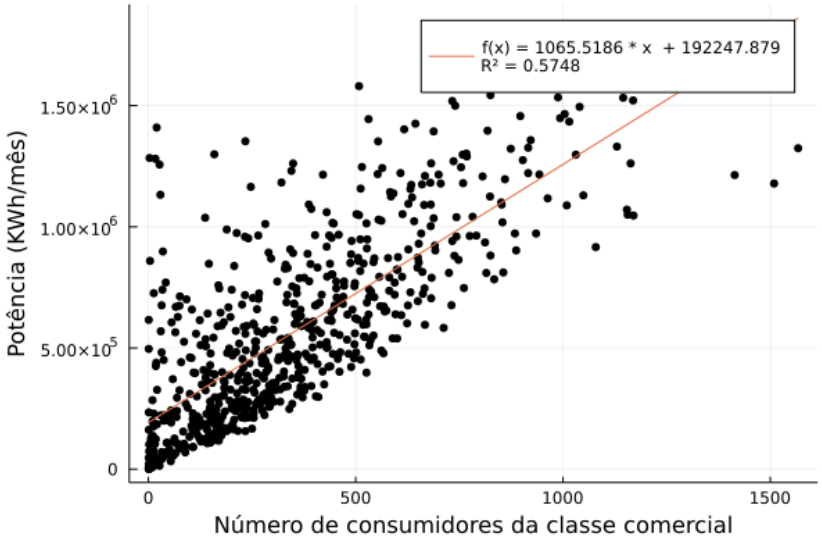
Figura 3.32: *Regressão do número de chaves pelo número de barras no alimentador urbano.*

Determinação da potência

A figura 3.33 apresenta a regressão da potência mensal demandada em função do número de consumidores nos alimentadores urbanos de 13,8 kV. É interessante observar que a interpretação do coeficiente de correlação de Pearson (r) é muito forte ($r \geq 0,90$) para os consumidores residenciais, figura 3.33(a), e forte ($0,70 \leq r < 0,90$) para os consumidores da classe comercial, na figura 3.33(b). No entanto, para os consumidores industriais $r = 0,29$, resultando em uma correlação fraca ($0,20 \leq r < 0,40$).



(a) *Consumidores residenciais ($r = 0,96$).*



(b) *Consumidores comerciais ($r = 0,76$).*

Figura 3.33: *Regressão da potência consumida pelo número de barras do alimentador urbano.*

Devido a correlação fraca do número de consumidores comerciais em função do

número de barras do alimentador inviabilizando essa estratificação e a correlação fraca entre a potência demandada e o número de consumidores industriais optou-se por não diferenciar os consumidores em classes e nem determinar a potência demandada para cada uma delas.

CAPÍTULO 4

Gerador de SDE (gSDE)

Este capítulo apresenta a metodologia, as ferramentas e os operadores propostos para a criação dos SDEs sintéticos desta tese. A metodologia utilizada para este propósito consiste em dividir o problema em três etapas: 1) a geração de topologia; 2) o posicionamento de consumidores; e 3) o posicionamento de chaves. A figura 4.1 apresenta a metodologia de geração de SDEs proposta neste trabalho. Na geração de topologia, é gerada uma floresta cujas árvores constituem as topologias de cada alimentador. No posicionamento dos consumidores, são escolhidas as barras que conterão consumidores e sua respectiva quantidade. No posicionamento das chaves, são posicionadas as chaves normalmente abertas e normalmente fechadas que permitem o redirecionamento de carga entre os alimentadores e as manobras de manutenção do SDE.

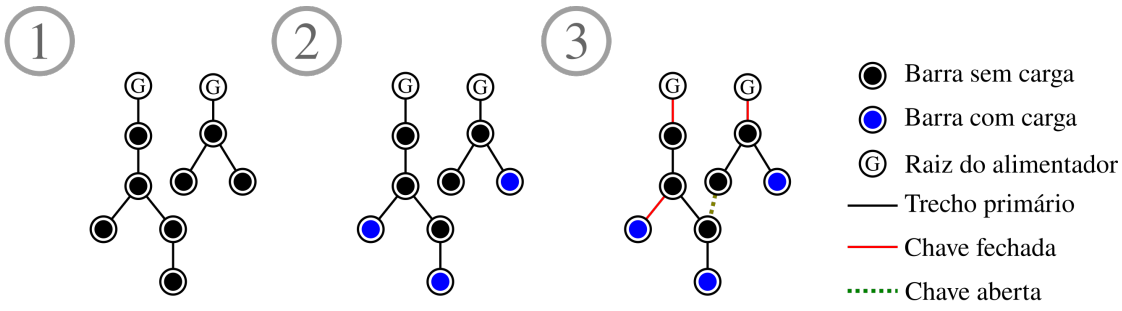


Figura 4.1: *Metodologia de geração de SDEs. 1) geração da topologia da árvore de cada alimentador; 2) posicionamento dos consumidores; 3) posicionamento das chaves normalmente abertas e normalmente fechadas.*

A geração de topologia utilizou um processo evolutivo multiobjetivo, o posicionamento dos consumidores utilizou regressões extraídas da análise estatística e o posicionamento das chaves utilizou um processo evolutivo mono-objetivo. Assim, para o funcionamento desta metodologia foi necessário o desenvolvimento de operadores de inicialização, mutação e *crossover* para os processos evolutivos. Além disto, todas estas informações adicionadas aos grafos gerados tiveram como base os estudos descritivos de SDEs reais (capítulo 3).

Este capítulo está estruturado da seguinte forma. A seção 4.1 apresenta a estratégia de geração da topologia dos SDEs sintéticos. A seção 4.2 apresenta a metodologia de posicionamento dos consumidores na topologia gerada. A seção 4.3 apresenta a estratégia de posicionamento das chaves dos SDEs gerados. Por fim, na seção 4.4 são apresentados os experimentos para avaliação da metodologia e dos operadores propostos.

4.1 Geração da topologia

Esta seção apresenta detalhadamente todas as estratégias e funções criadas para a busca topológica. Além disso, essa seção descreve desde a estratégia de busca de topologias até os operadores desenvolvidos e os utilizados no processo de busca.

Neste contexto a topologia do SDE é entendida como um grafo cujos vértices não são rotulados. Cada vértice do grafo constitui uma barra do SDE e as arestas são os trechos, chaves, reguladores de tensão e transformadores. Quando as chaves normalmente abertas, são retiradas da análise, o grafo forma uma floresta, ou seja, um conjunto de árvores no qual cada uma representando um alimentador.

4.1.1 Estratégia da busca de topologias

A busca de topologias teve como alvo um conjunto das distribuições estatísticas desejadas de características topológicas para a SDE sintética. Assim, o objetivo do processo de busca foi encontrar um conjunto de florestas que minimizassem o erro entre as distribuições desejadas e as distribuições obtidas de cada indivíduo, conforme será detalhado na seção 4.1.2. Na geração de topologias, somente foram consideradas as chaves normalmente fechadas, permitindo que cada alimentador fosse modelado como uma árvore, dado que ao considerar somente as chaves nesse estado, os SDEs possuem estrutura radial.

A figura 4.2 apresenta a estratégia utilizada no processo de busca de topologias. No passo 1, acontece a definição das métricas e as distribuições que caracterizam o SDE que se deseja obter. Para isso, o SDE é carregado para a memória definindo a estrutura de cada alimentador.

No passo 2, as distribuições de frequência que constituem o objetivo do processo de busca são extraídas dos alimentadores obtidos no passo 1. Estas distribuições são fornecidas como entrada para o gerador de SDE e sua saída são vários conjuntos de florestas (passo 3) que minimizem o erro destacado em azul no passo 4.

Tendo como objetivo a geração de topologias de SDEs foram utilizadas três distribuições como objetivo da busca: 1) distribuição de graus (figura 3.10); 2) folhas por alimentador (figura 3.12); 3) barras por alimentador (figura 3.11). Deste modo, o processo

evolutivo adotado foi multiobjetivo. Consequentemente, a saída do gerador de SDE foram os indivíduos da primeira fronteira de Pareto.

É importante salientar que a geração de topologias (passos 3 e 4) é independente da extração das distribuições (passos 1 e 2). Portanto, é possível gerar topologias de qualquer SDE se houver o conhecimento prévio das distribuições desejadas (passo 2), sem que haja necessidade de carregamento e tratamento dos dados. Isso viabiliza a geração de SDEs mesmo quando não se tem acesso aos dados, uma vez que, o detentor destes dados pode informar estas distribuições sem grandes problemas de segurança ou sigilo.

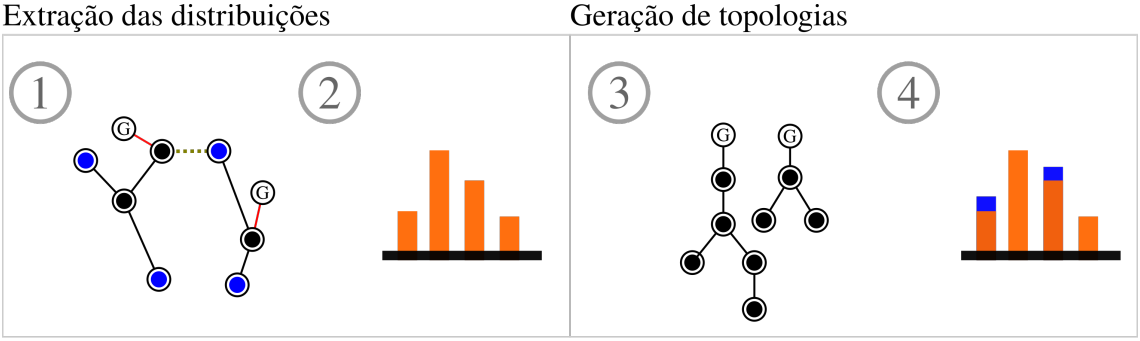


Figura 4.2: *Metodologia de geração de topologia. 1) as estruturas dos alimentadores são carregadas para a memória descartando-se as chaves abertas; 2) as distribuições utilizadas como objetivo da busca são extraídas; 3) com base nas distribuições é obtido um conjunto adequado de florestas durante o processo de busca; 4) obtém-se o erro das distribuições das redes sintéticas obtidas.*

Para o funcionamento do processo evolutivo do passo 3 é necessária uma representação dos SDEs e operadores adequados. E, para isso, foi utilizada a representação NPE e seus operadores SPRN e TBRN. Além disto, este trabalho também contribui com a proposta de novos operadores de inicialização e *crossover* específicos para o problema que serão apresentados nas seções 4.1.3 e 4.1.4.

4.1.2 Busca e função de *fitness*

As distribuições definidas na seção 3.3.1 formam o conjunto M de métricas para a busca, sendo que algumas distribuições possuem interdependência, ou seja, qualquer alteração na característica estudada altera uma ou mais distribuições do objetivo. É importante destacar estas interdependências, pois deste modo a busca global pela solução pode ser feita utilizando a estratégia lexicográfica [68], otimizando a topologia, posicionamento dos consumidores e chaves, nesta ordem.

Entre as distribuições de frequência mencionadas na tabela 3.1, na busca da topologia foram utilizadas somente aquelas classificadas com a categoria “T”: 1) o

grau das barras; 2) o número de barras por alimentador; e 3) o número de folhas por alimentador. Esta busca das soluções dentro de cada categoria apresentada na tabela 3.1 foi tratada como um problema de otimização multiobjetivo. Para isto, foi empregado o NSGA-II [15, 14] como processo evolutivo da solução proposta.

Nestas categorias, os objetivos consistem em um conjunto de métricas destas três distribuições de frequência relativa $M = \{M_1, M_2, M_3\}$ anteriormente mencionadas. Assim, cada M_i $(i = 1,2,3)$ representa um conjunto de pontos de uma distribuição de frequência, ou seja, $M_i = \{m_1^i, m_2^i, m_3^i, \cdots, m_{k^i}^i\}$, onde k^i é o número de classes da i-ésima distribuição de frequência relativa. Por serem frequências relativas sabe-se que $\sum_{j=1}^{k^i} m_j^i = 1$ e é por isso que é possível comparar duas distribuições sem problemas de escala. Neste sentido o *fitness* da busca multiobjetiva é o vetor dos erros de cada distribuição (equação 4-1). Já o erro (equação 4-2) é a diferença absoluta entre o valor desejado e o observado para cada classe de uma distribuição:

$$fitness = \begin{bmatrix} erro(M_1) \\ erro(M_2) \\ erro(M_3) \end{bmatrix} \tag{4-1}$$

$$erro(M_i) = \sum_{classe=1}^{k^i} |m_{classe}^{desejado} - m_{classe}^{observado}| \tag{4-2}$$

A figura 4.3 apresenta um exemplo do cálculo do erro de uma única distribuição de frequência M_i. Neste caso, o número de classes $k^i = 4$, compreendendo as classes de 1 a 10, 10 a 20, 20 a 30 e 30 a 40 barras. Nas distribuições de frequência deste trabalho o intervalo final é sempre aberto, exceto o intervalo para a última classe, neste caso 30 a 40. Tem-se então duas distribuições a desejada e a observada. Assim, o erro é calculado para cada uma das classes da distribuição. O lado direito da figura mostra o erro com a cor vermelha, que neste caso é a diferença absoluta entre o valor desejado e o observado em cada classe. Na primeira e na terceira classe o erro foi 0,2; enquanto que na segunda e na quarta classe o erro foi 0,1; totalizando assim o erro 0,6 para esta distribuição.

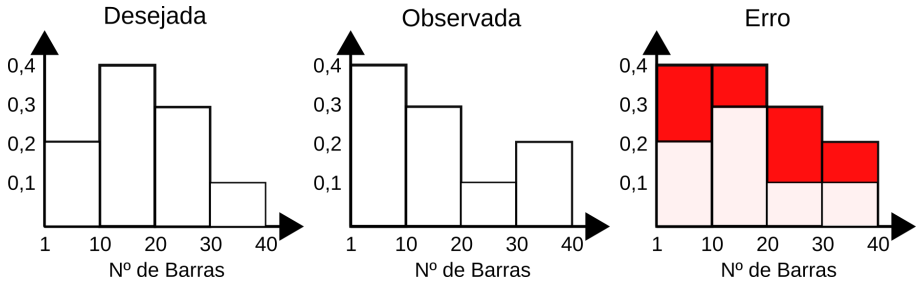


Figura 4.3: *Exemplo de cálculo do fitness da busca topológica.*

Desta forma o *fitness* é calculado com base em todas as distribuições de frequência relativa que servem como objetivo.

4.1.3 Inicialização

Esta seção apresenta o algoritmo de inicialização dos indivíduos e quatro algoritmos propostos de geração de árvores.

Algoritmo de inicialização

A fim de posicionar a população inicial em uma região promissora no espaço de busca foi desenvolvido um método determinístico para criar uma floresta com distribuição de barras por alimentador similar a distribuição obtida no passo 2 da figura 4.2 ou especificada pelo usuário. Este método é apresentado na figura 4.4.

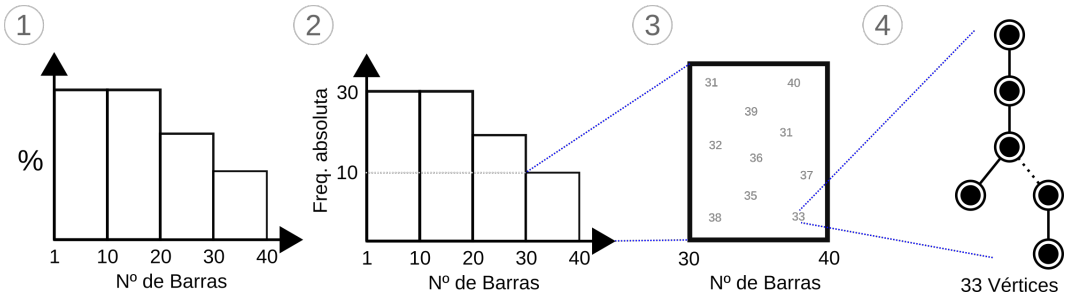


Figura 4.4: *Exemplo do processo de inicialização.*

Em (1), a distribuição da frequência relativa desejada é transformada em uma distribuição de frequência absoluta com base no número de alimentadores desejados para a rede sintética. O algoritmo 4.1 é responsável por este processo e nele deve-se garantir que a frequência absoluta acumulada seja igual ao número desejado de alimentadores. Este total poderia se tornar diferente por meio dos arredondamentos do processo. Ao final deste processo, tem-se o número de alimentadores para cada uma das classes do número de barras.

Deste modo, (2) é a distribuição do número de barras por alimentador em termos absolutos. Na ilustração, pode-se observar que 30 alimentadores possuem de 1 a 10 barras e outros 30 alimentadores possuem de 10 a 20 barras, aproximadamente 20 alimentadores possuem de 20 a 30 barras e 10 alimentadores possuem de 30 a 40 barras.

Assim, em (3) é escolhido um número de barras para cada um destes alimentadores respeitando o início e fim de cada classe (algoritmo 4.2). Como não existem mais informações a respeito destas classes, optou-se por utilizar um sorteio uniforme entre o valor inicial e final de cada classe.

Por se tratar de um processo estocástico, a saída é diferente mesmo quando são utilizados os mesmos argumentos de entrada. Isto acontece porque existem várias combinações de frequência absoluta que possuem a mesma distribuição de frequência relativa. De outro ponto de vista, esta estocasticidade garante uma certa variabilidade genética à população inicial.

Algoritmo 4.1: $frequencia\_absoluta(nA, distribuicao)$

Entrada: número de alimentadores nA, distribuição de frequência relativa $distribuicao$.

Saída: frequência absoluta fa.

```
1  d ← tabela hash (classe => quantidade de alimentadores)
2  acumulada ← 0
3  para (cls, freq) ← distribuicao faça
4      qtde ← arredondar(freq × nA)
5      se acumulada + qtde > nA então
6          qtde ← nA − acumulado
7      fim
8      acumulado ← acumulado + qtde
9      d[cls] ← qtde
10 fim
11 return d
```

Algoritmo 4.2: $distribuir(nA, distribuicao)$

Entrada: número de alimentadores nA, distribuição de frequência relativa $distribuicao$.

Saída: vetor os números de barras q.

```
1  d ← frequencia_absoluta(nA, distribuicao)
2  f ← vetor[nA]              /* Aloca um vetor com nA elementos */
3  i ← 1
4  para (cls, fa) ← d faça
5      para k ← 1 até fa faça
6          f_i ← sortear_numero_inteiro(cls.inicio, cls.fim − 1)
7          i ← i + 1
8      fim
9  fim
10 return f
```

Por fim, em (4) já se sabe quantas barras cada alimentador terá, assim utiliza-se um algoritmo de geração de árvores para criação da árvore de cada alimentador (algoritmo 4.3). O conjunto de árvores de todos alimentadores compõe uma floresta que constitui o indivíduo da busca.

Inicialmente, foi utilizado o método de número de Prüfer para esta finalidade. No entanto, para melhoria de desempenho são propostos novos métodos para esta tarefa,

descritos nas próximas seções. Este método requer duas etapas, a primeira a aleatorização de uma sequência e a segunda a decodificação desta sequência em uma árvore rotulada. Esta árvore, por sua vez, é codificada usando a representação NPE. Este processo de sorteio, decodificação e codificação apresenta um custo computacional relevante que poderá ser objetivo de otimizações em trabalhos futuros.

Algoritmo 4.3: $inicializar(nA, distribuicao)$

Entrada: número de alimentadores nA, distribuição de frequência relativa $distribuicao$.

Saída: floresta f.

```
1 f ← distribuir(nA, distribuicao)
2 flo ← Floresta(nA)      /* Aloca uma floresta com nA árvores */
3 para i ← 1 até nA faça
4 |   flo.arvores[i] ← npe(arvore(gerar_sequencia_prufer(f_i)))
5 fim
6 return flo
```

Quanto a complexidade, o algoritmo 4.1 tem complexidade $\Theta(c)$, onde c representa o número de classes da distribuição uma vez que o tempo de inserção na tabela *hash* na linha 9 é $O(1)$, pois esta não possui colisões entre chaves. O algoritmo 4.2 possui complexidade $\Theta(c+nA)$, onde nA é o número de alimentadores para o problema, porque o somatório de todas as frequências contidas na tabela *hash* d é igual nA. Estes resultados levam a complexidade geral do algoritmo 4.3 que é $O(c+nA+nA\times K)$, onde K é o custo computacional da geração de cada árvore de tamanho fa_i na linha 4.

No caso específico do algoritmo 4.3, tem-se que K é composto pela complexidade de geração da sequência Prüfer de tamanho f_i é $O(f_i)$, e a decodificação desta sequência em uma árvore pela função $arvore()$ é $O(f_i \log f_i)$ uma vez que é possível utilizar uma fila de prioridade mínima para extração do elemento com menor rótulo [57]. Já a codificação da árvore na sequência do NPE pela função $npe()$ tem complexidade $O(f_i)$ já que se trata apenas de uma busca em profundidade na árvore criada e, para essas, o número de arestas é $m = n-1$. Assim, a complexidade de cada execução da linha 4 é $K = O(f_i \log f_i)$. Consequentemente, tem-se que a complexidade do algoritmo 4.3 utilizando o código Prüfer é $O(\sum_{i=1}^{nA} f_i \log f_i)$. Predominantemente, o custo de decodificação dos números Prüfer de cada alimentador.

Algoritmo de construção de árvores

O algoritmo 4.4 apresenta uma proposta de geração de árvores utilizando diretamente a representação do NPE. A figura 4.5 apresenta o funcionamento do algoritmo.

Este adiciona incrementalmente os vértices na árvore, ligando o vértice adicionado a apenas um dos vértices já pertencentes a ela, de forma parecida ao algoritmo de Albert e Barabási [3] exceto pela tendência deste.

Neste algoritmo, S (linha 1) é um vetor que mantém os dados (id e profundidade) dos n vértices. Estes são mantidos na ordem em que são sorteados e não na ordem em que deveriam aparecer na NPE. Fazendo necessário o uso da estrutura $filhos$ (linha 2) para a reordenação proposta no algoritmo 4.5. Esta estrutura $filhos$ mantém os índices dos vértices filhos de cada vértice em um vetor.

Algoritmo 4.4: $npe\_aleatoria(n)$

Entrada: Número de vértices n.
Saída: sequência NPE da árvore gerada.

```
1  S ← vetor com n elementos
2  filhos ← vetor de vetor inicialmente vazio com n elementos
3  S[1] ← Vertice(1,0)
4  para i ← 2 até n faça
5      pai ← randi(1, i−1)
6      prof ← S[pai].profundidade
7      S[i] ← Vertice(i, prof+1)
8      inserir(filhos[pai], i)
9  fim
10 return NPE(S, filhos)
```

Na linha 3, o vértice raiz com id 1 e profundidade 0 é criado. Os demais vértices são criados na estrutura de repetição das linhas 4 a 9. Nesta, o pai do i-ésimo vértice é sorteado entre os vértices já definidos, ou seja, de 1 até $i-1$, na linha 5. Já nas linhas 6 e 7, a profundidade do vértice pai é obtida e usada para criar o novo vértice com um nível a mais. Por fim, o índice do vértice criado é adicionado como filho do vértice com índice pai.

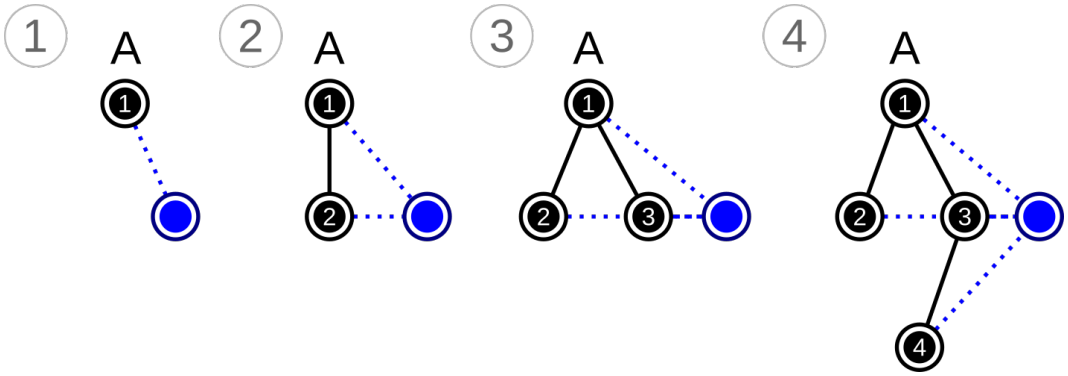


Figura 4.5: *Exemplo de funcionamento da construção de árvores.*

Na linha 10 (algoritmo 4.4), a função NPE é utilizada para ordenar o vetor S em uma sequência NPE válida conforme o algoritmo 4.5. Este algoritmo utiliza uma pilha para colocar os vértices previamente definidos na ordem correta.

Algoritmo 4.5: $NPE(S, filhos)$

Entrada: vértices S e os filhos dos vértices ($filhos$).
Saída: NPE.

```
1  n ← tamanho(S)
2  a ← NPE(n)
3  posicoes ← vetor com n elementos preenchidos com 1
4  indice_pai ← 1
5  a[1] ← S[1]
6  p ← nova pilha
7  i ← 2
8  enquanto i ≤ n faça
9      posicao ← posicoes[indice_pai]
10     se posicao ≤ tamanho(filhos[indice_pai]) então
11         indice_filho ← filhos[indice_pai][posicao]
12         a[i] ← S[indice_filho]
13         posicoes[indice_pai] ← posicao + 1
14         empilhar(p, indice_pai)
15         indice_pai ← indice_filho
16         i ← i + 1
17     senão
18         indice_pai ← desempilhar(p)
19     fim
20 fim
21 return a
```

Para fazer esta ordenação, na linha 1, do algoritmo 4.5, obtém-se o número de vértices no vetor S. Na linha 2, é criada uma estrutura vazia para armazenar a NPE com n vértices. Na linha 3, cria-se um vetor para auxiliar a posição do filho a ser preenchida, portanto inicialmente com todos os valores preenchidos com 1. Na linha 4, define-se o índice do primeiro pai como 1. Na linha 5, o vértice raiz da árvore é preenchido com o mesmo vértice em $S[1]$. Na linha 6, é criada a pilha p que auxiliará na ordenação correta. Na linha 7, é definida a variável i como 2 para armazenar a posição da NPE a a ser preenchida. A estrutura de repetição das linhas 8 a 20 se repetirá enquanto houverem posições da sequência a serem preenchidas.

Na linha 9, é obtida a posição do filho a ser preenchida para o índice pai corrente. Para a primeira iteração, será obtida a posição do primeiro filho do vértice raiz. Na linha 10, é verificado se este vértice possui a quantidade de filhos determinada pela variável *posicao*. Caso ele possua este filho, obtém-se o índice do filho na linha 11. Na linha 12, a i-ésima posição da NPE é preenchida com o vértice filho. Na linha 13, incrementa-se o filho a ser analisado para o índice pai em questão. Na linha 14, o índice pai é inserido na pilha para reanálise futura, pois o próximo vértice do qual os filhos serão analisados é o próprio vértice adicionado na NPE (linha 15). Na linha 16, a posição i é incrementa. Caso todos os filhos, do índice pai corrente, tenham sido inseridos na sequência da NPE, o próximo índice pai é desempilhado (linha 18).

Quanto a complexidade, o algoritmo 4.5 é $O(n)$ uma vez que na linha 3 existe um laço que se repetirá n vezes e o laço das linhas 8 a 20 se repete no mínimo $n-1$ vezes (melhor caso), para um grafo do tipo lista, no máximo $2n-2$ vezes para um grafo estrela (pior caso). Sendo o custo de cada repetição $O(1)$, pois todas as operações inclusive as de empilhar (linha 14) e desempilhar (linha 18) possuem complexidade $O(1)$.

A complexidade do algoritmo 4.4 é determinada pelo laço na linha 2 com n repetições, o laço das linhas 4 a 9 com $n-1$ repetições e a chamada do algoritmo 4.5, na linha 10, que tem complexidade $O(n)$. Assim, complexidade deste algoritmo também é $O(n)$. Portanto, tem-se um gerador de árvores com complexidade linear.

Geração de árvores com grau máximo

Limitar o grau máximo dos vértices é conveniente ao observar que o grau máximo das redes elétricas tende a não passar de 5, assim como apresentado na figura 3.10. Neste sentido, um algoritmo que limite o grau posicionaria a população inicial em uma zona mais promissora do espaço de busca.

O algoritmo 4.6 se assemelha ao algoritmo 4.4 de geração de uma NPE aleatória. Ele inclui, na linha 3, um contador do grau de cada vértice e um vetor de vértices (*candidatos*) que podem ser pai de outro vértice (linha 5). Nas linhas 7 e 8, um dos candidatos a ser o pai do i-ésimo vértice é selecionado, mantendo sua posição no vetor *candidatos* na variável *indice_candidato*. Na linha 12, o vértice recém criado é adicionado como candidato a ser pai, uma vez que seu grau é 1 e o valor mínimo para o grau máximo é 2.

Nas linhas 13 e 14, os graus do i-ésimo vértice e do vértice pai escolhido são computados. E, por fim, nas linhas 15 a 17, se o grau do vértice escolhido como pai chegou ao limite máximo ele é removido do vetor de candidatos (linha 16) através de seu índice no vetor.

Algoritmo 4.6: $npe\_aleatoria\_grau\_maximo(n, grau\_maximo)$

Entrada: Número de vértices n e grau máximo ($grau\_maximo$) permitido para um vértice.

Saída: sequência NPE da árvore gerada.

```
1  S ← vetor com n elementos
2  filhos ← vetor de vetor inicialmente vazios com n elementos
3  graus ← vetor com n elementos iguais a 0
4  S[1] ← Vertice(1,0)
5  candidatos ← [1]
6  para i ← 2 até n faça
7      indice_candidato ← rand_indice(candidatos)
8      pai ← candidatos[indice_candidato]
9      prof ← S[pai].profundidade
10     S[i] ← Vertice(i, prof + 1)
11     inserir(filhos[pai], i)
12     inserir(candidatos, i)
13     graus[i] ← 1
14     graus[pai] ← graus[pai] + 1
15     se graus[pai] = grau_maximo então
16         remover_indice(candidatos, indice_candidato)
17     fim
18 fim
19 return NPE(S, filhos)
```

A figura 4.6 apresenta o processo de construção de uma árvore com grau máximo 3 seguindo o algoritmo 4.6, nela o grau de cada vértice é apresentado ao seu lado. Em todas as etapas, o vértice em azul representa o novo vértice a ser adicionado e a linha pontilhada em azul representa os candidatos a serem vértice pai deste vértice. Na etapa 2 é possível visualizar o vértice 2 sendo adicionado e incrementando os graus dos dois vértices (linhas 13 e 14). Na etapa 5 o vértice 5 foi adicionado como filho do vértice 1, logo o vértice 1 deixou o vetor de candidatos a vértice pai (linhas 15 a 17), pois atingiu o grau máximo 3 para esta construção. É possível visualizar que o vértice 1 deixou o vetor de candidatos pela ausência de linha pontilhada em azul entre ele e o novo vértice a ser adicionado (vértice azul). O mesmo acontece na etapa 6, quando o vértice 6 se torna filho do vértice 3, fazendo com que este último atinja o grau máximo e deixe o vetor de candidatos a vértice pai.

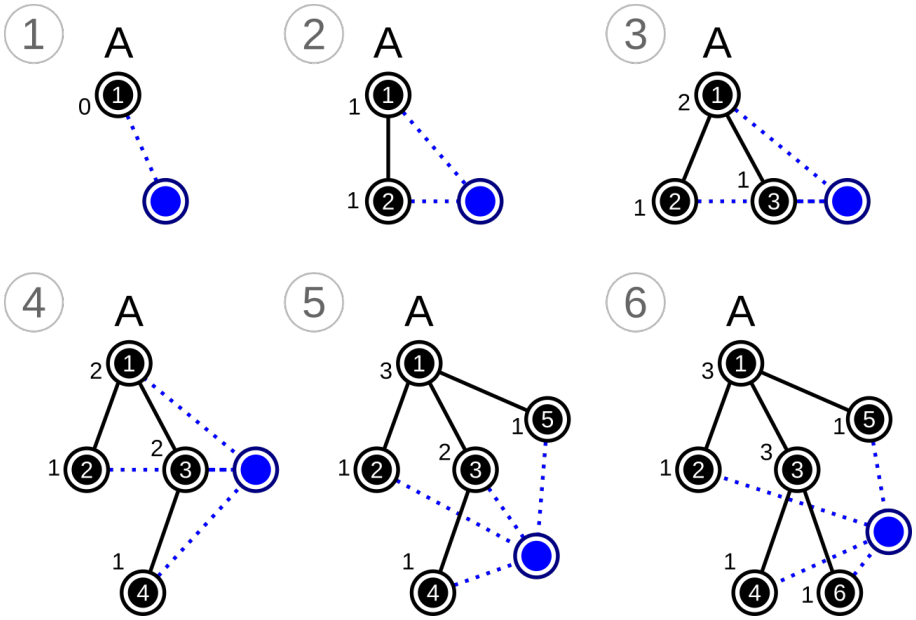


Figura 4.6: *Exemplo de funcionamento da geração de árvores com grau máximo igual a 3.*

O algoritmo 4.6 tem complexidade $O(n^2)$, uma vez que a criação do vetor *graus*, na linha 3, tem complexidade $O(n)$, a inserção de elementos no vetor de candidatos, na linha 12, tem complexidade $O(1)$, assim como a computação dos graus nas linhas 13 e 14, porém a remoção do candidato pelo seu índice também tem complexidade $O(n)$.

Geração de árvores com profundidade máxima

O algoritmo 4.7 apresenta o método proposto para limitar a profundidade máxima de uma árvore. A figura 4.7 o processo de geração de uma árvore com profundidade máxima igual a 2. Para limitar a profundidade o algoritmo mantém uma variável com a profundidade de cada vértice. Esta profundidade é exibida ao lado do vértice na figura 4.7. Apenas os elementos com profundidade inferior à profundidade máxima desejada entram na lista de candidatos a pai de um novo vértice.

A estrutura do algoritmo 4.7 é similar ao dos algoritmos 4.4 e 4.6. No entanto, a criação do vetor de candidatos funciona incluindo os novos vértices cuja profundidade é inferior a máxima desejada, enquanto que no algoritmo 4.6 o processo consiste em excluir os vértices que atingiram o grau máximo. Neste sentido, nas linhas 11 a 13 o algoritmo verifica esta condição e inclui os vértices que podem admitir filhos.

A complexidade do algoritmo 4.7 é $O(n^2)$, assim como o algoritmo 4.6, uma vez que o custo de sorteio e acesso aos candidatos, nas linhas 6 e 7, são $O(1)$, mas o custo de inserção de um novo elemento no vetor, na linha 12, é $O(n)$.

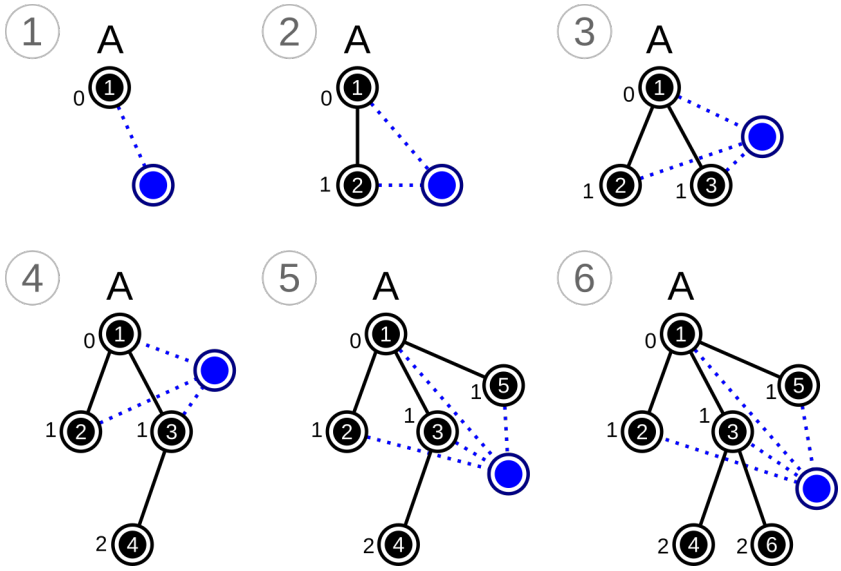


Figura 4.7: *Exemplo de funcionamento da geração de árvores com profundidade máxima igual a 2.*

Algoritmo 4.7: $npe\_aleatoria\_profundidade\_maxima(n, prof\_maxima)$

Entrada: Número de vértices n e profundidade máxima ($prof\_maxima$) permitida para um vértice.

Saída: sequência NPE da árvore gerada.

```
1  S ← vetor com n elementos
2  filhos ← vetor de vetor inicialmente vazios com n elementos
3  S[1] ← Vertice(1,0)
4  candidatos ← [1]
5  para i ← 2 até n faça
6      indice_candidato ← rand_indice(candidatos)
7      pai ← candidatos[indice_candidato]
8      prof ← S[pai].profundidade
9      S[i] ← Vertice(i, prof + 1)
10     inserir(filhos[pai], i)
11     se S[i].profundidade < prof_maxima então
12         inserir(candidatos, i)
13     fim
14 fim
15 return NPE(S, filhos)
```

Geração de árvore por composição de floresta em representação de grafos

Esta seção apresenta um algoritmo capaz de criar uma árvore por composição de uma floresta. Ele foi projetado para diminuir a tendência de geração de grafos próximos ao tipo estrela encontrada no algoritmo 4.4 que será apresentada na seção 5.4.1. A figura 4.8 apresenta seu esquema de funcionamento. Para gerar uma árvore com n vértices, o algoritmo começa criando n árvores com apenas um vértice. Em seguida, ele sorteia duas árvores distintas e conecta a raiz da primeira árvore em um dos vértices da segunda árvore. Este processo é repetido $n-1$ vezes e, no final, é formado apenas uma árvore.

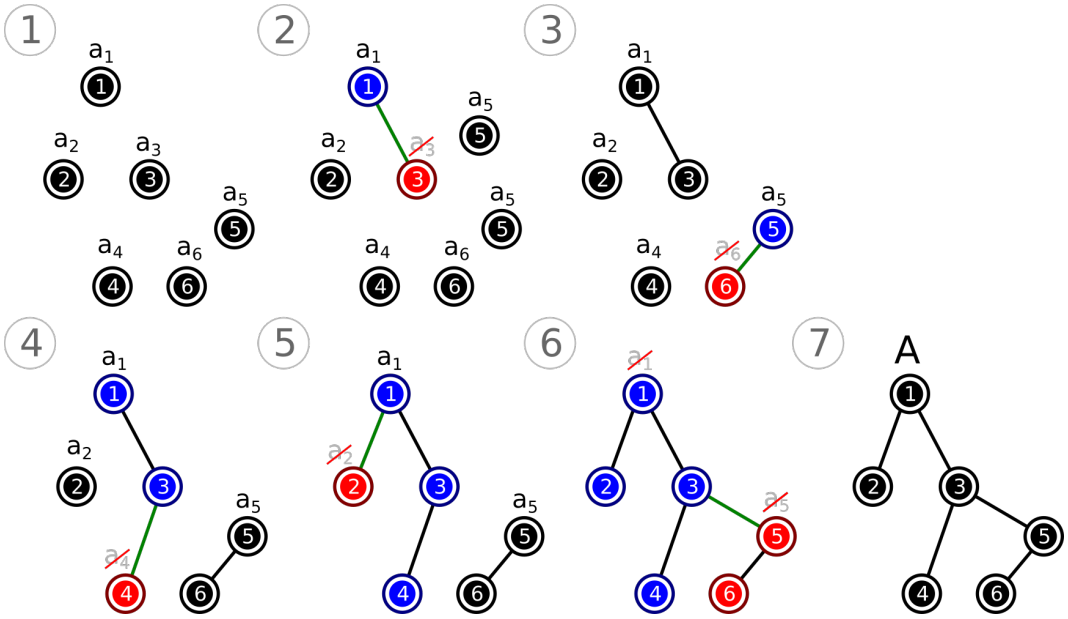


Figura 4.8: *Algoritmo de geração de árvore por composição de floresta. Em azul é destacada a árvore que receberá a árvore com vértices vermelhos.*

O algoritmo 4.8 apresenta os passos computacionais para realização do procedimento apresentado na figura 4.8. Na linha 1, um vetor com n árvores é criado. Nas linhas 2 a 4 este vetor é preenchido com árvores contendo apenas um vértice. Nas linhas 5 a 12 as árvores são conectadas. Na linha 6, dois índices de árvores distintas são sorteados e nas linhas 7 e 8 estes dois objetos são obtidos nas variáveis a_1 e a_2. Na linha 9, é sorteado o vértice que será pai da raiz da árvore a_1. Na linha 10, reside o desempenho do algoritmo, pois as duas árvores a_1 e a_2 são conectadas através do vértice v_{pai}. Na linha 11, o índice i_1 é removido do vetor as de modo que o vetor tenha um elemento a menos. Por fim, na linha 13, a árvore restante é retornada.

Algoritmo 4.8: $arvore\_composicao(n)$

Entrada: Número de vértices n.

Saída: sequência NPE da árvore gerada.

```
1  as ← vetor com n árvores enraizadas
2  para i ← 1 até n faça
3  |   as[i] ← ArvoreEnraizada(1)
4  fim
5  para m ← n até 2 faça
6  |   i1, i2 ← sortear_indices_distintos(1, m)
7  |   a1 ← as[i1]
8  |   a2 ← as[i2]
9  |   v_pai ← sortear_item(a2.vertices)
10 |   inserir(a2, a1, v_pai)
11 |   remover(as, i1)
12 fim
13 return as[1]
```

4.1.4 Operadores de *crossover*

Esta seção apresenta os novos operadores de *crossover* desenvolvidos para o problema de geração de SDEs. Estes operadores foram desenvolvidos para manutenção da qualidade do número de barras por alimentador da inicialização, apresentada na seção 4.1.3.

A operação de *crossover* mais intuitiva consiste em trocar dois alimentadores quaisquer entre dois indivíduos. No entanto, isso prejudica a qualidade da inicialização feita pelo algoritmo 4.3. As variações desse procedimento tentam manter essa propriedade e serão apresentadas a seguir.

Crossover por tamanho da árvore

A solução mais intuitiva para que a distribuição do número de barras por alimentador não se altere exageradamente durante o processo de busca, mesmo quando este critério não seja considerado como objetivo do processo evolutivo foi permitir que somente árvores de tamanhos similares sejam trocadas entre os genitores. Deste modo, uma árvore com 500 vértices não seria trocada por uma árvore com apenas 20 vértices.

Neste sentido, para que duas árvores sejam trocadas entre dois genitores estabeleceu-se uma tolerância. Assim, se a tolerância for 30, somente seria aceitável trocar uma árvore com 500 vértices por árvores contendo entre 470 a 530 vértices.

Crossover por *locus*

Após alguns experimentos, que serão apresentados na seção 5.7, com o *crossover* por tamanho da árvore, descrito na seção anterior, foi possível perceber que ele não mantinha a qualidade da inicialização (seção 4.1.3), pois ao longo das gerações do processo evolutivo ainda era possível que duas árvores de classes diferentes fossem trocadas. Essa troca é intensificada com o aumento da tolerância ou com a redução da amplitude entre as classes das distribuições de frequência.

Assim, para garantir que não fossem trocadas árvores pertencentes a classes diferentes da distribuição do número de barras por alimentador, criou-se um novo conceito de *crossover*. Neste, cada árvore recebeu uma marcação indicando a qual classe da distribuição ela pertence durante a inicialização dos indivíduos. Deste modo, na operação de *crossover*, só foi permitida a troca de árvores cujos *locus* fossem exatamente iguais.

A figura 4.9 apresenta um exemplo da aplicação deste operador. Nota-se que cada árvore criada durante a inicialização recebe uma marcação de *locus* no passo 4. Isto exige uma pequena alteração no algoritmo de inicialização apresentado na seção 4.1.3. Neste exemplo, durante a operação de *crossover*, a árvore RNP 3a foi selecionada no indivíduo SDE A aleatoriamente, como seu *locus* é a tupla $(10,\ 20)$, então as únicas árvores que podem ser trocadas no indivíduo SDE B são RNP 3b e 4b, destacadas em amarelo. Uma delas é escolhida ao acaso e a troca é realizada.

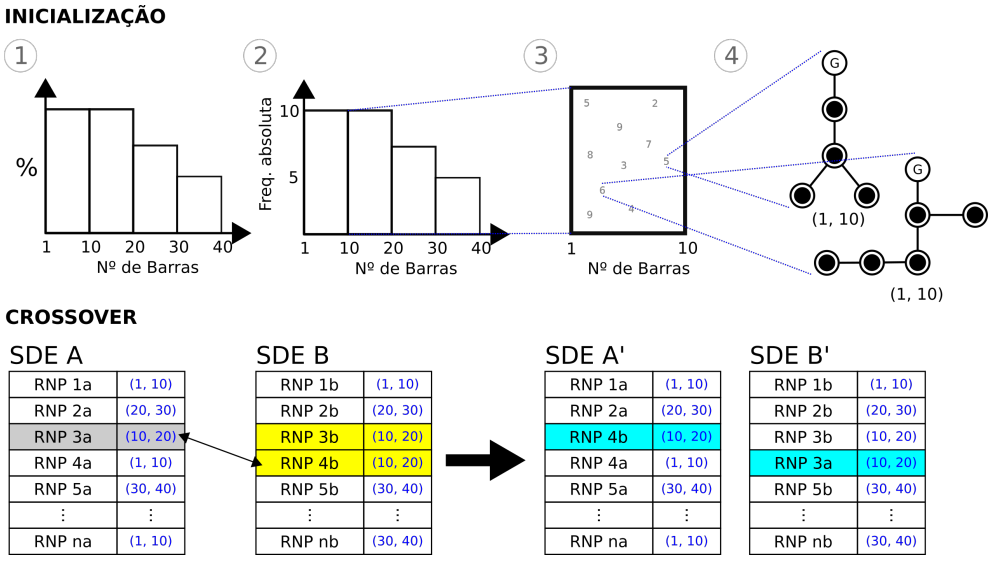


Figura 4.9: *Exemplo de aplicação do crossover por locus.*

É importante notar que em ambos genitores a quantidade de árvores do *locus* $(10,\ 20)$ não foi alterada. Isto garante que a distribuição do número de barras por alimentador não se altere. Permitindo remover essa distribuição da função objetivo do processo de busca, se os demais operadores estiverem alinhados com esta mesma característica.

4.1.5 Mutação

Esta seção apresenta os operadores de mutação propostos neste trabalho. Entre eles uma variação do SPR e do TBR que visa manter o grau do vértice raiz igual a 1 e dois operadores probabilísticos de poda inspirados nesses operadores chamados de Probabilistic Subtree Pruning and Regrafting (PSPR) e Probabilistic Tree Bisection and Reconnection (PTBR).

SPR e TBR mantendo a raiz do alimentador com grau 1

Para a mutação das topologias utilizou-se uma variação do SPRN e do TBRN que mantinham a raiz com grau igual a 1. Isto porque através do estudo descritivo (seção 3.3.1), observou-se que a raiz dos alimentadores sempre possuía grau 1. Para isso foi necessário restringir os limites de sorteio dos vértices de poda, inserção e novo vértice raiz destes operadores conforme apresentado nos algoritmos 4.9 e 4.10.

Algoritmo 4.9: $sprn\_raiz\_grau\_1(A)$

Entrada: NPE A.
Saída: sequência NPE da árvore alterada mantendo o vértice raiz com grau 1.

1 $n \leftarrow tamanho(A)$
2 $i_p \leftarrow aleatorizar\_entre(3, n)$
3 $i_a \leftarrow aletorizar\_entre([2, i_p - 1], [indice\_fim(A, i_p) + 1, n])$
4 **return** $sprn(A, i_p, i_a)$

Na linha 1 do algoritmo 4.9, obteve-se o tamanho da NPE A. Na linha 2, é realizada a escolha do índice de poda. Esta só pode acontecer a partir do terceiro vértice da sequência, uma vez que, ao podar o segundo vértice, só haverá uma opção que seria reconectá-lo a raiz. Já na linha 3, é escolhida a posição que receberá a subárvore iniciada na posição i_p. Nela exclui-se a possibilidade da raiz receber a subárvore podada através da modificação do primeiro intervalo $[2, i_p - 1]$, iniciando-o na posição 2. Na linha 4, o SPRN é executado com os índices escolhidos.

O algoritmo 4.10 funciona de modo similar ao algoritmo 4.9 restringindo os índices escolhidos. No entanto, diferentemente do algoritmo 4.9, na linha 2 permite-se a poda do segundo vértice, uma vez que, a conexão com o vértice raiz pode acontecer com outro vértice da subárvore iniciada no segundo vértice. Na linha 3, obtém-se a posição final da subárvore iniciada em i_p. Caso o segundo vértice seja escolhido para poda, então o vértice que receberá a subárvore será o vértice raiz (linha 5), caso contrário aplica-se o mesmo procedimento usado no algoritmo 4.9 (linha 7). Na linha 9, o novo vértice raiz é

escolhido na subárvore podada. Por fim, na linha 10, realiza-se o TBRN usando os índices escolhidos.

Algoritmo 4.10: $tbrn\_raiz\_grau\_1(A)$

Entrada: NPE A.
Saída: sequência NPE da árvore alterada mantendo o vértice raiz com grau 1.

```
1  n ← tamanho(A)
2  i_p ← aleatorizar_entre(2, n)
3  i_fim ← indice_fim(A, i_p)
4  se i_p = 2 então
5  |   i_a ← 1
6  senão
7  |   i_a ← aletorizar_entre([2, i_p − 1], [i_fim + 1, n])
8  fim
9  i_r ← aleatorizar_entre(i_p, i_fim)
10 return tbrn(A, ip, i_r, ia)
```

Assim, no procedimento de mutação, cada floresta possuiu de chance de ser modificada. Caso ela fosse eleita para alteração, cada uma de suas árvores foi sujeita aos operadores apresentados nos algoritmos 4.9 e 4.10. Ambos operadores se expressaram com probabilidades iguais.

Operadores probabilísticos de poda

Para inserção de tendência nos operadores SPR e TBR é proposto a utilização de funções que definem pesos para cada um dos vértices de uma árvore a fim de alterar as probabilidades de escolha dos vértices nestes operadores, possibilitando direcionar a busca de topologias. Estas variantes foram chamadas de PSPR e PTBR.

Apesar de aumentar a probabilidade de escolher um vértice em detrimento de outros, ainda é possível gerar todos os tipos de árvores com estes operadores desde que nenhum vértice receba peso zero. Mantendo assim as mesmas propriedades de exploração do espaço de busca do SPR e TBR.

Tanto o PSPR quanto o PTBR utilizam uma aleatorização por roleta viciada. Nesta, os vértices com maior peso possuem maior probabilidade de serem selecionados. O funcionamento do PSPR é apresentado na figura 4.10, nela uma subárvore com raiz no vértice p é podada entre os vértices, excluindo a raiz da árvore, ponderados pela função f_p e, em seguida, um vértice a da árvore remanescente é selecionado para ser o novo pai da árvore podada ponderados pela função f_a.

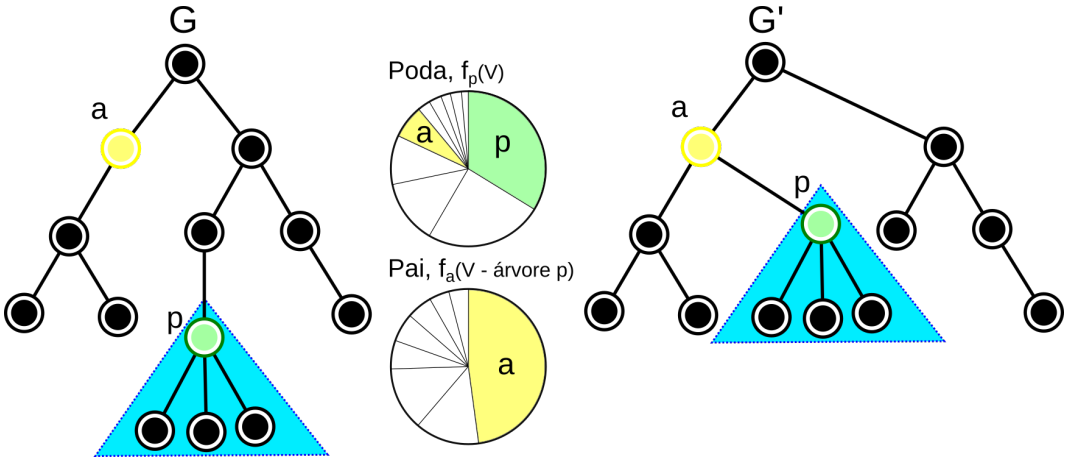


Figura 4.10: *Exemplo de funcionamento do PSPR.*

O Probabilistic Subtree Pruning and Regrafting on NPE (PSPRN) é apresentado no algoritmo 4.11. Nota-se que neste algoritmo é utilizada a sintaxe $[i:f]$ para representar um vetor com os elementos de i até f de 1 em 1. Além disso, a função $seletor\_escala()$ é a responsável pela atribuição dos pesos a cada vértice através da função que ela recebe por parâmetro e pelo sorteio do vértice com base nestes pesos.

Algoritmo 4.11: $pspr\_npe(A, f_p, f_a)$

Entrada: árvore em representação por profundidade A, funções f_p e f_a de pesos.

Saída: sequência A'.

```
1  n ← numero_vertices(A)
2  i_p ← seletor_escala([2 : n], f_p)
3  i_pf ← indice_fim(A, i_p)
4  candidatos_pai ← insercao([2 : i_p − 1], [i_pf + 1 : n])
5  se candidatos_pai ≠ ∅ então
6  |   i_a ← seletor_escala(candidatos_pai, f_a)
7  |   return sprn(A, i_p, i_a, i_pf)
8  senão
9  |   return A
10 fim
```

Primeiramente, na linha 1, o algoritmo obtém o número de vértices da árvore A usando NPE. Na linha 2, o algoritmo seleciona um dos vértices, que não seja a raiz (vértice 1), para sofrer poda de acordo com a escala criada pela função f_p. Em seguida, na linha 3, é calculado ou obtido o fim da árvore iniciada no vértice escolhido. Na linha 4, é construído o vetor de candidatos a vértice pai da subárvore iniciada em i_p, estes são todos os vértices que não fazem parte desta subárvore.

Se houverem candidatos, na linha 6, é selecionado o vértice pai seguindo a escala criada pela função f_a. Por fim, uma vez escolhidos o índice da subárvore a ser podada e o novo vértice pai, ela é podada usando uma definição expandida do SPRN que aceita estes índices i_p e i_a. Nota-se que o último argumento, evita que o SPRN precise calcular novamente o fim da subárvore iniciada em i_p.

O PTBR é muito similar ao PSPR, porém recebe uma terceira função f_r para ponderar a seleção do novo vértice raiz da subárvore que foi removida, conforme apresentado na figura 4.11.

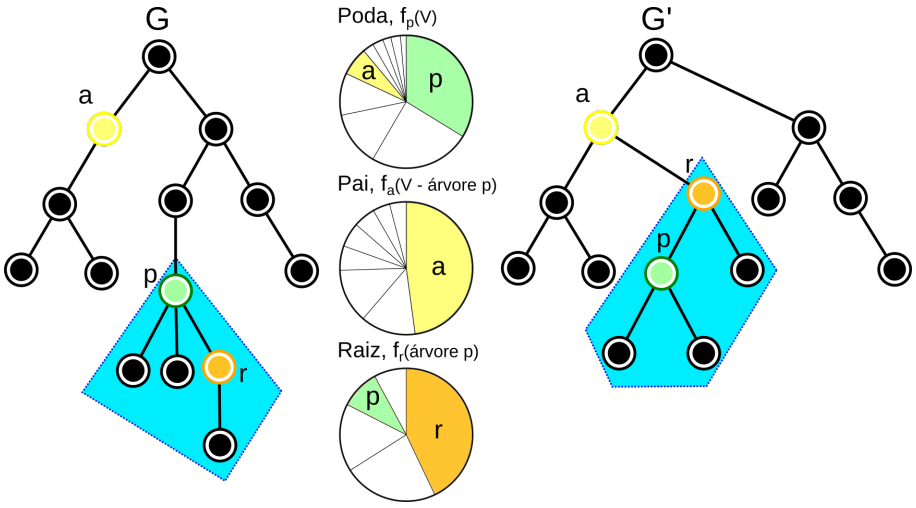


Figura 4.11: *Exemplo de funcionamento do PTBR.*

O Probabilistic Tree Bisection and Reconnection on NPE (PTBRN) é apresentado no algoritmo 4.12. Nota-se que ele se assemelha ao PSPRN, acrescentando a linha 7 e alterando a função chamada na linha 8.

Na linha 7, ocorre o sorteio do índice do novo vértice raiz (i_r) da subárvore podada em i_p. Os vértices candidatos para serem a nova raiz são apenas os vértices contidos no intervalo de i_p a i_{pf} no vetor. Nota-se também que, na linha 8, a função $tbr()$ é chamada através de uma extensão que aceita os índices das subárvores e o índice final da subárvore iniciada em i_p.

A complexidade do algoritmo 4.12 é $O(n) + +O(f_p(v_{restantes})) + O(f_a(vs_a)) + O(f_r(vs_r))$, onde $O(f_r(vs_r))$ é a complexidade de calcular f_r para todos os vértices da árvore podada (vs_r). Assim, tanto a complexidade do PTBR quanto a do PSPR dependem intrinsecamente da complexidade das funções que eles recebem como argumento: f_p, f_a e f_r para o PTBR.

Algoritmo 4.12: $ptbr\_npe(A, f_p, f_a, f_r)$

Entrada: árvore em representação por profundidade A, funções f_p e f_a de pesos.

Saída: sequência A'.

```
1  n ← numero_vertices(A)
2  i_p ← seletor_escala([2 : n], f_p)
3  i_pf ← indice_fim(A, i_p)
4  candidatos_pai ← insercao([2 : i_p − 1], [i_pf + 1 : n])
5  se candidatos_pai ≠ ∅ então
6  |   i_a ← seletor_escala(candidatos_pai, f_a)
7  |   i_r ← seletor_escala([i_p : i_pf], f_r)
8  |   return tbr(A, i_p, i_r, i_a, i_pf)
9  senão
10 |   return A
11 fim
```

4.1.6 Visão geral do gerador de topologias

Embora, a análise descritiva tenha sido feita em SDEs com centenas de alimentadores, o emprego da frequência relativa permite encontrar as mesmas características em SDEs sintéticas de tamanhos variados, podendo ser menores ou maiores que as utilizadas na análise inicial. Desta forma, além das distribuições esperadas da topologia da SDE sintética o usuário também informa o número de alimentadores que ele deseja que sejam criados. Estas conversões acontecem durante a inicialização da população (seção 4.1.3) e na avaliação dos objetivos, sempre ao manipular frequências relativas.

O algoritmo 4.13 apresenta uma visão geral do gerador de topologias. Neste foi utilizada a notação ($\rightarrow$) para indicar a criação de funções anônimas. Nas linhas 1, 2 e 3, as funções de adaptação são criadas. Elas recebem uma floresta (*npes*), neste caso, um indivíduo do processo evolutivo, e avaliam sua distribuição em relação a distribuição esperada para os graus, número de folhas e número de barras por alimentador. Na linha 4, é formada a tupla com as três funções-objetivo que serão informadas como argumento do NSGA-II.

Na linha 5, o processo evolutivo começa especificando a forma como o i-ésimo indivíduo é inicializado, seguindo o algoritmo 4.3, restringindo a raiz com grau 1, conforme a propriedade observada no estudo descritivo (seção 3.3.1). Para isto, pode-se utilizar qualquer algoritmo de geração de árvores apresentado na seção 4.1.3 ou números Prüfer gerando uma árvore com $n-1$ vértices e, em seguida, adicionando-se uma nova raiz. Os seguintes argumentos do NSGA-II são: a tupla de funções de adaptação (fit), o

tamanho da população (P), a probabilidade de *crossover* (P_{cross}), a função de *crossover* (f_{cross}) utilizada foi o *crossover* por *locus* (seção 4.1.4) dadas as características que serão apresentadas na seção 5.7, a probabilidade de mutação (P_{mut}), a função de mutação (f_{mut}) escolhida foi o SPR e TBR que mantém a raiz dos alimentadores com grau 1 (seção 4.1.5), a seleção foi feita por torneio (f_{sel}), o número de iterações desejadas (n_{iter}) do NSGA-II.

Embora os operadores de mutação PSPR e PTBR serem capazes de inserir tendência na busca, conforme será apresentado na seção 5.5. Eles precisam de variantes para serem utilizados quando o objetivo são distribuições.

Algoritmo 4.13: $gerar\_topologias(nA, ct, P, p_{cross}, f_{cross}, p_{mut}, f_{mut}, f_{sel}, n_{iter})$

Entrada: Número de alimentadores nA, distribuições dos critérios topológicos ct, tamanho da população P, percentual de *crossover* p_{cross}, função de *crossover* f_{cross}, percentual de mutação p_{mut}, função de mutação f_{mut}, função de seleção f_{sel}, número de iterações n_{iter}

Saída: Melhores topologias encontradas.

1 $f_1 \leftarrow npes \rightarrow fit\_grau(npes, ct.grau)$
2 $f_2 \leftarrow npes \rightarrow fit\_folhas(npes, ct.n\_folhas\_alimentador)$
3 $f_3 \leftarrow npes \rightarrow fit\_barras\_alimentador(npes, ct.n\_barras\_alimentador)$
4 $fit \leftarrow (f_1, f_2, f_3)$
5 **return** $NSGA\_II(\ i \rightarrow inicializar(nA, ct.n\_barras\_alimentador),\ fit,\ P,\ p_{cross},\ f_{cross},\ p_{mut},\ f_{mut},\ f_{sel},\ n_{iter}\ )$

O resultado do algoritmo 4.13 são os indivíduos da primeira fronteira de Pareto (F1). Nota-se que o número de indivíduos retornados pode ser no máximo igual ao tamanho da população.

4.2 Posicionamento dos consumidores

Para o posicionamento dos consumidores, procedeu-se uma análise de regressão a fim encontrar relações entre o número de barras do alimentador e o número de consumidores que este possuía (seção 3.3.2). Para verificar o ajuste de cada regressão calculou-se o coeficiente de correlação de Pearson (r) e o coeficiente de determinação (R^2).

Após o estudo de regressão, foi decidido que seriam considerados apenas alimentadores urbanos, uma vez que, a qualidade da correlação ($r = 0,21$) para todos os alimentadores foi classificada como fraca ($0,20 \leq r < 0,40$). Desta forma, os alimentadores foram segmentados nos tipos: urbano, rural e misto, permitindo um melhor entendimento

da relação destas duas variáveis, pois o estudo descritivo, que foi apresentado na seção 3.3.1, já havia mostrado que estes tipos de alimentadores possuem características diferentes. Portanto, acreditou-se que esta segmentação aumentaria ainda mais a qualidade da correlação da regressão permitindo melhores inferências.

Neste sentido, os consumidores foram estimados com base no número de barras dos alimentadores urbanos, uma vez que, a qualidade da predição para estes alimentadores foi considerada forte ($0,70 \leq r < 0,90$) e a quantidade de amostras era grande (817 alimentadores). Embora os trabalhos futuros a este tenham que relacionar a carga dos consumidores residenciais, comerciais e industriais separadamente, visto que as demandas de carga destas classes serem bem diferentes (seção 3.3.2), a correlação entre o número de consumidores comerciais e o tamanho do alimentador foi fraca ($0,20 \leq r < 0,40$), conforme apresentado na seção 3.3.2, assim neste trabalho optou-se por determinar o número de consumidores total de cada alimentador em função de seu tamanho.

Para o posicionamento dos consumidores nas barras dos alimentadores, procedeu-se a análise descritiva da proporção de folhas com e sem consumidores. A porção de folhas sem consumidores, foi dividida em dois grupos: folhas que possuíam chave aberta e folhas que não possuíam chave aberta, conforme apresentado na figura 3.18.

Com base nos resultados das regressões dos números de consumidores (seção 3.3.2), descrito anteriormente, determinou-se o número de consumidores total em função do tamanho do alimentador gerado. Em seguida, com base na análise descritiva (figura 3.18), calculou-se o número de folhas e quais folhas de cada alimentador que possuiriam consumidores. Assim, estes consumidores foram distribuídos igualmente nas folhas escolhidas aceitando uma variação equivalente a dois desvios-padrão do número de consumidores por folha, uma vez que, no estudo do problema de distribuição dos consumidores não foram encontradas relações entre a quantidade de consumidores ou carga em função da profundidade das barras.

4.3 Posicionamento das chaves

Em relação ao posicionamento das chaves abertas e fechadas apresentado no terceiro passo da metodologia (figura 4.1), procedeu-se uma análise de regressão para determinar o número de chaves normalmente fechadas nos alimentadores e o número de chaves normalmente abertas no SDE apresentado na seção 3.3.2.

Após o posicionamento dos consumidores na topologia (seção 4.2) foram posicionadas as chaves. Para isso, utilizou-se um AG mono-objetivo com a finalidade de minimizar o número de consumidores afetados por uma falta após o reestabelecimento da energia nos setores à jusante da falta em um SDE sintético. Neste sentido, foram geradas faltas em todas as barras da topologia e, foi considerado que, era possível recompor todos

os setores à jusante de uma falta se esses possuíssem pelo menos uma chave aberta. Este relaxamento do problema foi realizado porque a transferência de consumidores entre alimentadores necessita de um estudo aprofundado sobre a carga e suas flutuações durante o dia que não fizeram parte do escopo deste trabalho. Deste modo, não houveram restrições elétricas avaliadas pelo AG.

4.3.1 Cálculo do *fitness*

Apesar do relaxamento do problema, calcular o *fitness* considerando faltas em todas as barras é um desafio, pois esse cálculo para uma única falta demanda uma busca em profundidade em todas as barras à jusante da barra em falta, consequentemente, realizá-lo em todas as barras requer que várias buscas sejam realizadas. Ao associar este problema com o tamanho da rede elétrica, aplicar um AG com uma função de *fitness* tão complexa, tornaria o tempo demandado para execução muito alto ou até mesmo impraticável.

Assim para que o cálculo do *fitness* fosse otimizado, foi desenvolvido o algoritmo 4.14 que utilizou programação dinâmica durante a busca em profundidade nos setores dos alimentadores (algoritmo 4.16). Este algoritmo foi capaz de calcular o impacto de todas das faltas em uma única busca em profundidade nos setores.

Na linha 1, do algoritmo 4.14, é obtido o setor inicial do alimentador a. Este é o mesmo setor atribuído a barra raiz do alimentador. Na linha 2, é criada uma tabela *hash* que conterá os contadores de falha para cada setor do alimentador. Este contador de falha envolve o número de consumidores afetados caso ocorra uma falha no setor ($qcfs$), o número de consumidores afetados caso ocorra uma falha no setor à montante do setor ($qcfm$) e se há uma chave aberta no setor ou à jusante do setor. Na linha 3, é realizada a busca em profundidade por setores, detalhada no algoritmo 4.16. Esta função recebe o *callback* cb_setor que é o algoritmo 4.15.

Algoritmo 4.14: $qtde\_consumidores\_afetados\_apos\_recomposicao(a)$

Entrada: alimentador a

Saída: contagem de falhas por setor do alimentador mp.

```
1 si ← setor_inicial(a)
2 mp ←tabela hash (Setor => Contadores de falhas)
3 busca_profundidade_por_setor(a, si, cb_setor, mp)
4 return mp
```

O algoritmo 4.15, é chamado quando ocorre uma descoberta e uma finalização de setor durante a busca em profundidade realizada no algoritmo 4.16. Na linha 1, restringe-se o uso desse *callback* para a finalização de cada setor. Na linha 2, é calculado o

número de consumidores no setor s. É importante observar que este número não inclui os consumidores à jusante do setor. Na linha 3, assume-se que não há chave aberta à jusante até que seja encontrada uma.

Algoritmo 4.15: $cb\_setor(a, s, mp, status)$

Entrada: alimentador a, setor s, parâmetros mp, status da descoberta do setor $status$ (d - descoberta, f - finalização).

```
1  se status = 'f' então
2      qcfs ← qtde_consumidores(a,s) /* Quantidade de
           consumidores afetados caso ocorra uma falha no setor
           */
3      tem_chave_aberta_jusante ← false
4      para sa ← setores_adjacente_jusante(a,s) faça
5          qcfs ← qcfs + mp[sa].qcfm
6          se mp[sa].tem_chave_aberta então
7              tem_chave_aberta_jusante ← true
8          fim
9      fim
10     tca ← tem_chave_aberta(s) ou tem_chave_aberta_jusante
         /* Calcula a quantidade de consumidores afetados caso
        ocorra uma falha à montante do setor s */
11     se tca então
12         mp[s].qcfm ← 0
13     senão
14         mp[s].qcfm ← qcfs
15     fim
16     mp[s].qcfs ← qcfs
17     mp[s].tem_chave_aberta ← tca
18 fim
```

Nas linhas 4 a 9, são analisados todos os setores adjacentes à jusante ao setor s. Como estes setores estão à jusante, então seus contadores já foram calculados, uma vez que o *callback* só analisa o setor s após a finalização da análise dos setores à jusante dele. Na linha 5, a quantidade de consumidores afetados caso ocorra uma falha no setor ($qcfs$) é acumulada usando a quantidade de consumidores afetados caso ocorra uma falha à montante ($qcfm$) do setor sa. Deste modo, caso não haja como recompor a subárvore iniciada no setor sa, o total de consumidores afetados é acumulado para o setor s. Nas linhas 6 a 8 é avaliado se há uma chave aberta no setor sa ou à jusante dele. Caso haja, na linha 7, é marcado que há uma chave aberta a jusante do setor s.

Na linha 10, é avaliado também se há uma chave aberta diretamente ligada ao setor s. Nas linhas 11 a 15, é calculado a quantidade de consumidores afetados por uma falha a montante. Caso haja uma chave aberta em s ou a sua jusante, este número é 0, caso contrário, é igual a quantidade de consumidores afetados caso ocorra uma falha no setor ($qcfs$). Na linha 16 e 17 os dados já calculados são atribuídos na tabela *hash* para o setor s.

O algoritmo 4.16 apresenta a busca em profundidade por setor. Nela, uma função de *callback* cb é chamada nas linhas 1 e 5, durante a descoberta e finalização do setor. E, nas linhas 2 a 4, são analisados os setores adjacentes à jusante do setor s, usando recursividade na linha 3.

Algoritmo 4.16: $busca\_profundidade\_setor(a,s,cb,mp)$

Entrada: alimentador a, setor s, callback cb, parâmetros mp.

```
1 cb(a,s,mp,'d') /* d - Descoberta                    */
2 para sa ← setores_adjacente_jusante(a,s) faça
3 |   busca_profundidade_setor(a,sa,cb,mp)
4 fim
5 cb(a,s,mp,'f') /* f - Finalização                   */
```

4.3.2 Representação

Uma vez definida a função de *fitness* de alto desempenho na seção anterior, foi escolhida a representação a ser utilizada para o AG. Esta envolveu a codificação de $k+1$ cromossomos, conforme apresentado na figura 4.12, onde k é o número de alimentadores na topologia gerada, neste exemplo $k = 3$. Para cada um dos k alimentadores foi determinado o número de chaves fechadas com base em seu número de barras de acordo com a regressão apresentada na seção 3.3.2. Deste modo, o cromossomo de cada alimentador foi representado pelos números dos trechos que seriam substituídos por chaves fechadas. Sendo os números dos trechos sequenciais e individuais de cada alimentador. Esta contagem não inclui a ligação da barra inicial à segunda barra do alimentador, pois esta sempre é uma chave fechada conforme o estudo estatístico (seção 3.3.2).

O último cromossomo da representação ficou responsável por determinar as chaves abertas da topologia gerada. Assim como a quantidade de chaves fechadas em cada alimentador, o número de chaves abertas foi determinado pelo número de barras na rede elétrica. Cada chave aberta foi representada por uma tupla de dois elementos contendo o número sequencial de uma folha sem carga.

Por exemplo, na figura 4.12, os trechos 2, 7 e 11, no alimentador A_1, devem ser substituídos por chaves fechadas, conforme representado no lado direito da figura. É

importante observar, que o estudo de regressão estaria inferindo, com base no tamanho, que esse alimentador teria 4 chaves normalmente fechadas. No entanto, na representação são posicionadas apenas três delas, visto que a primeira chave normalmente fechada está posicionada entre a barra raiz do alimentador e a segunda barra.

Para que esse processo fosse realizado com eficiência sem ter que replicar toda a estrutura do grafo, o *framework* de grafos teve que ser adaptado para que fosse possível gerar subgrafos a partir do grafo da topologia sintética. Deste modo, G_1 é um subgrafo da topologia $G = (V, E)$ de modo que $G_1 = (V, E_1)$ e E_1 é o mesmo conjunto de trechos E menos os trechos a serem substituídos, unido às chaves fechadas e abertas codificadas na representação. Tal estrutura implicou diretamente na redução do uso de memória durante o processo e requereu que os dados das arestas do grafo não estivessem armazenados nos vértices. O único dado mantido no vértice foi o número de consumidores, uma vez que, ele é igual em todos os indivíduos.

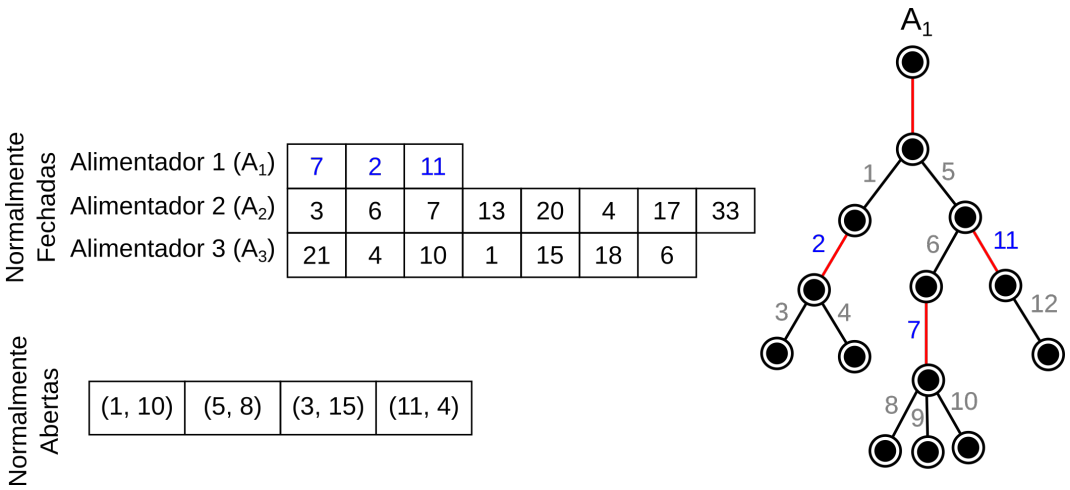


Figura 4.12: *Exemplo de representação das chaves fechadas e abertas para $k = 3$ alimentadores.*

4.3.3 Inicialização, *crossover* e mutação

Como operador de *crossover* foram utilizadas duas funções com meio-a-meio de probabilidade de execução de uma ou da outra. O operador Partially Mapped Crossover (PMX) foi aplicado para trocar informações sobre as chaves fechadas, desta maneira garante-se que não haja repetições de trechos no cromossomo de um alimentador, consequentemente a quantidade de chaves fechadas desejada sempre será posicionada. Já para as chaves abertas utilizou-se o operador *crossover* de dois pontos.

Como mutação foram feitas duas funções uma para as chaves fechadas e outra para as abertas com 50% de probabilidade de execução de uma ou outra. O operador para as chaves fechadas escolheu outro trecho a ser substituído por uma chave normalmente fechada, desde que este não estivesse no cromossomo. Já o operador de mutação das

chaves abertas altera a segunda barra folha da tupla de modo que ela não esteja no mesmo alimentador da primeira barra folha e não possua consumidores.

Para a inicialização foram utilizados dois procedimentos, o primeiro para aleatorizar as chaves fechadas e o segundo para as chaves abertas. No primeiro, foram sorteados k trechos, conforme a regressão do número de chaves fechadas pelo número de barras do alimentador, de modo que não houvessem repetições de trechos. Se houvessem repetições neste ponto o PMX entraria em *loop* durante o *crossover*. No segundo, as duas folhas da tupla de cada chave normalmente aberta foram sorteadas entre as folhas sem consumidores de modo que as folhas não estivessem no mesmo alimentador.

4.4 Experimentos

Esta seção apresenta o plano dos experimentos apresentados no capítulo 5. Estes experimentos tem como objetivo:

- avaliar a qualidade dos operadores desenvolvidos e compará-los com os já existentes na literatura (apresentados no capítulo 2);
- avaliar as melhores configurações dos operadores;
- avaliar o desempenho do processo de busca como um todo.

A condução destes experimentos foi realizada utilizando linguagem Julia, versão 1.5.3, com uma implementação própria do NSGA-II.

4.4.1 Amostrador não tendencioso no espaço de busca

Para as análises de tendência, que serão apresentadas nas seções a seguir, foi preciso o desenvolvimento de um algoritmo não tendencioso que serviu de referência. Este algoritmo será apresentado na seção 5.2. Para observar que ele não era tendencioso, executou-se um Delineamento Inteiramente Casualizado (DIC). Neste, cada topologia gerada tinha exatamente 4 árvores com até 5 vértices, o que resulta em um espaço de busca de 84 mil florestas. No experimento, foram realizadas 20 repetições observando a frequência de ocorrência de cada floresta na topologia gerada. Em cada repetição foram realizados 2.505.630 sorteios, equivalentes a 30 vezes o tamanho do espaço de busca. Ao todo foram feitos 75.168.900 sorteios.

Ao final da coleta de dados, procedeu-se com a análise de variância (ANOVA) e a média da frequência de ocorrência das árvores foi avaliada a 1% de significância com o teste de Tukey [56].

4.4.2 Direcionamento correto da tendência da inicialização

O objetivo deste experimento foi demonstrar que a tendência do algoritmo 4.3 está correta. Para isso, a tendência da inicialização (seção 4.1.3) foi avaliada em comparação ao amostrador uniforme, apresentado e demonstrado na seção 5.2. Neste sentido, ambos constituíram os dois tratamentos de um DIC com 2 mil repetições. Neste experimento o amostrador uniforme serviu de testemunha. Os objetivos de cada unidade experimental eram dinâmicos sendo gerados a partir de uma floresta com 4 árvores na qual cada uma podia possuir de 1 a 15 vértices.

Como o algoritmo é direcionado para criar árvores levando em consideração a distribuição de barras por alimentador, somente este critério foi estudado. Deste modo, para medir se o direcionamento estava correto, foi utilizado o módulo da diferença entre a frequência relativa desejada e a frequência relativa obtida que neste contexto também pode ser chamado de erro. Os resultados deste experimento são apresentados na seção 5.3.

4.4.3 Tendência dos operadores de geração de árvores

A fim de avaliar a tendência dos algoritmos 4.4, 4.6, 4.7 e 4.8 de geração de árvores, foram realizados DICs, com árvores de 18 vértices e 1.500 repetições, a cada repetição gerou-se uma árvore, observando a distância de Hamming mínima para um grafo estrela e a distância mínima para um grafo do tipo lista em relação a um amostrador uniforme e a geração de árvore com os números Prüfer. Para os algoritmos 4.6 e 4.7, que limitam o grau máximo e a profundidade máxima, foi avaliada a tendência observando os limites de 2 a 17. Os resultados desse estudo são apresentados nas seções 5.4.

4.4.4 Observação do tempo de geração de árvores com o NPE

Para corroborar com a dedução de complexidade linear para o algoritmo 4.4, apresentada na seção 6, elaborou-se um experimento que visa gerar árvores aleatórias utilizando o algoritmo proposto com 100 mil até 1 milhão de vértices de 100 em 100 mil vértices de intervalo e com 30 repetições. Procedeu-se então a regressão linear do tempo demandado para a geração destas árvores em função de seu tamanho. Esta regressão é apresentada na figura 5.6 da seção 5.4.1.

4.4.5 Inserção de tendência nos operadores PSPR e PTBR

Para avaliar a tendência dos novos operadores PSPR e PTBR foi estabelecido um DIC cujo objetivo foi obter um grafo estrela. Nesta busca uma árvore com 100 vértices foi gerada aleatoriamente através de um código Prüfer em cada uma das 30 repetições

do experimento. Após a casualização, cada árvore foi submetida aos operadores SPR, TBR, PSPR e PTBR. Estes foram aplicados 1.000 vezes a estas árvores a fim de obter o grafo estrela desejado. Para avaliar a tendência, observou-se o maior grau de vértice na árvore final. Além disso, para permitir uma comparação melhor, foi avaliada a testemunha que representa o grau máximo das árvores geradas no processo sem a aplicação dos operadores. Os resultados desse estudo são apresentados na seção 5.5.

4.4.6 Busca topológica

Para as avaliações do processo de busca de topologias, os experimentos foram conduzidos com 20 alimentadores. Ao final destes experimentos avaliou-se o histórico de cada uma das funções-objetivo e os indivíduos retornados na busca. Destes, observou-se o melhor indivíduo de cada objetivo e o indivíduo com menor erro geral, considerando cada objetivo igualmente importante. Os resultados desse experimento serão apresentados na seção 5.6.

O processo evolutivo escolhido foi o NSGA-II porque cada subproblema continha entre dois ou três (seções 3.3 e 5.7) e, por isso, ainda é adequada sua aplicação [14, 67]. Os parâmetros do gerador de topologias (seção 4.1.6) foram: 80% de percentual de *crossover*, 20% de mutação, seleção por torneio, 300 indivíduos na população e 5.000 iterações do processo evolutivo.

4.4.7 Comparação do posicionamento das chaves na rede elétrica real

Para observar a similaridade entre o posicionamento das chaves da rede elétrica real com a metodologia proposta na seção 4.3, separou-se 21 alimentadores urbanos isolando sua topologia, substituindo suas chaves normalmente fechadas por trechos e retirando as chaves normalmente abertas. Nota-se que o AG de posicionamento das chaves foi restrito a posicionar exatamente o mesmo quantitativo de chaves abertas e fechadas já presente na rede elétrica. Ao final deste processo, foi solicitado ao algoritmo que realizasse o posicionamento das chaves. Em seguida, comparou-se o número de posicionamentos iguais aos observados na SDE real tanto para as chaves abertas quanto para as fechadas. Os resultados deste experimento são apresentados na seção 5.10.

CAPÍTULO 5

Resultados e Discussão

Este capítulo tem como objetivo apresentar os resultados das buscas topológicas com o emprego dos operadores propostos, apresentados no capítulo 4, e dos experimentos apresentados na seção 4.4. Ele está estruturado da seguinte forma. As seis primeiras seções deste capítulo avaliam a tendência dos operadores criados. Para isso, a seção 5.1 apresenta o estudo do espaço de busca que é utilizado para criação de um amostrador não tendencioso (seção 5.2).

A seção 5.3 é feito um estudo da tendência do algoritmo de inicialização, proposto na seção 4.1.3. A seção 5.4 apresenta a tendência dos algoritmos que constroem árvores, apresentados na seção 4.1.3. E, na seção 5.5, são apresentados os resultados da inserção de tendência no PSPR e PTBR.

A seção 5.6 apresenta os resultados do experimento do processo de busca topológica, definidos na seção 4.4.6, apresentando exemplos de alimentadores encontrados pelo gerador topológico. Na seção 5.7, são apresentados os resultados da busca usando somente dois objetivos. Finalmente, na seção 5.9 são apresentados os resultados obtidos no posicionamento dos consumidores e das chaves.

5.1 Análise do espaço de busca

Para verificar a tendência da inicialização (assunto da seção 5.2 e 5.3), é preciso compreender primeiro o espaço de busca do problema. Para isso, é preciso observar que neste contexto o espaço de busca é infinito, logo é preciso limitá-lo para realizar esta avaliação.

Mesmo quando se tem no máximo 8 vértices por árvore e o número fixo de 4 árvores, o espaço de busca é de 1,6 bilhões de florestas distintas. Este valor é parcialmente calculado com a função que determina o número de árvores enraizadas com n vértices

adaptada de Finch [26, p. 296]:

$$T_n = \begin{cases} 1, & \text{se } n = 1 \\ \frac{1}{n-1} \sum_{k=1}^{n-1} \left(\sum_{d|k} T_d \cdot d \right) \cdot T_{n-k}, & \text{se } n > 1 \end{cases} \tag{5-1}$$

Esta sequência, apresentada na tabela 5.1, tem crescimento exponencial conforme mostrado na figura 5.1. No entanto, este é apenas um fator deste espaço de busca, visto que, uma amostra nele é um arranjo com repetição de k árvores.

Tabela 5.1: *Sequência de crescimento do número de árvores enraizadas em função do número de vértices.*

Número de vértices	1	2	3	4	5	6	7	8	9	10
Número de árvores enraizadas	1	1	2	4	9	20	48	115	286	719
Número acumulado de árvores	1	2	4	8	17	37	85	200	486	1.205

Usando o limite de até 8 vértices por árvore, tem-se 200 árvores diferentes como opção para cada alimentador. Uma vez que, dois ou mais alimentadores podem possuir árvores idênticas, tem-se então o espaço de busca, citado anteriormente, de tamanho 200^4 florestas, com 4 alimentadores. Assim, o tamanho do espaço de busca com no máximo t vértices e k alimentadores pode ser definido como:

$$F = \left(\sum_{i=1}^{t} T_i \right)^k \tag{5-2}$$

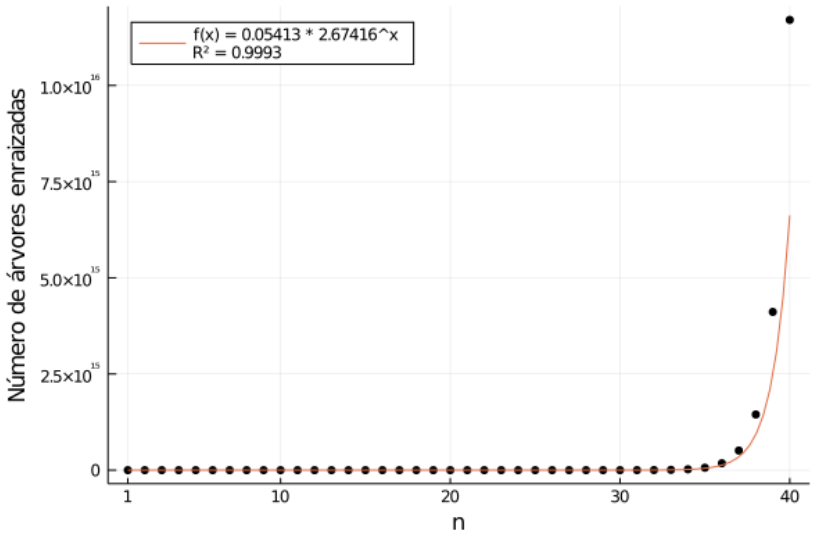


Figura 5.1: *Curva do número de árvores enraizadas.*

Deste modo pode-se afirmar que o espaço de busca por topologias radiais de SDEs possui crescimento exponencial e, por isso, justifica a utilização de métodos heurísticos no processo de busca.

5.2 Amostrador não tendencioso

Para fins de avaliação de tendência, é preciso criar um amostrador do espaço de busca descrito, que não seja tendencioso. O amostrador basicamente escolhe k árvores do total das árvores acumuladas. Para a demonstração de ausência de tendência deste amostrador foi fixado $k = 4$ e o valor escolhido do número máximo de vértices foi 5, que possui um espaço de busca de quase 84 mil florestas, de acordo com a tabela 5.2:

Tabela 5.2: *Tamanho do espaço de busca em função do número máximo de vértices por alimentador.*

Número de vértices	Tamanho do espaço com 4 alimentadores
1	1
2	16
3	256
4	4096
5	83.521
6	1.874.161
7	52.200.625
8	1.600.000.000
9	55.788.550.416
10	> 2,1 trilhões

Teoricamente, se o sorteio for repetido um número r de vezes muito superior as 84 mil florestas, cada indivíduo deve aparecer com uma frequência muito próxima. Então, foi realizado o sorteio 16.704.200 vezes, equivalente a 200 vezes o tamanho do espaço de busca.

Deste modo, foi observada a frequência de cada indivíduo na figura 5.2, que apresentou como esperado média e mediana de 200 ocorrências (linha preta) por indivíduo com desvio padrão de $\pm 14,18$, mínimo de 144 ocorrências ($\approx 3,9$ desvios) e máximo de 269 ($\approx 8,8$ desvios), com o primeiro quartil em 190 e o terceiro em 209 ocorrências.

Ainda assim, é possível notar uma diferença em termos absolutos da frequência de algumas árvores. Para responder se essa diferença se deve ao acaso foi executado o DIC descrito na seção 4.4.1. O teste indicou que não existe diferença entre as médias de ocorrência de cada floresta no espaço de busca a 1% de significância com o teste de Tukey. Provando que o amostrador não é tendencioso assim como desejado, ao menos

para árvores com no máximo 5 vértices com 4 alimentadores. Para generalização desta afirmação procedeu-se com a prova matemática.

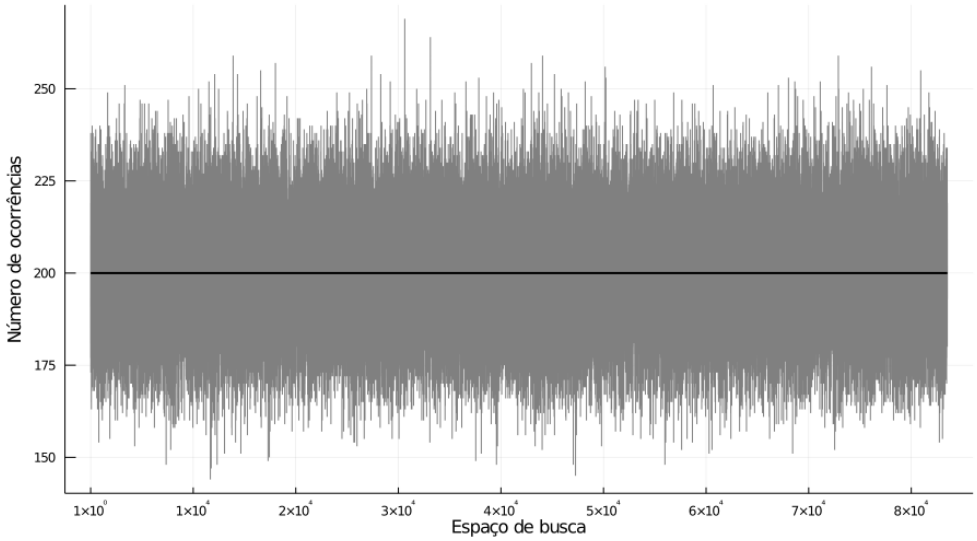


Figura 5.2: *Frequência de ocorrência em um amostrador não tendencioso.*

Teorema 5.1 *Ao selecionar a árvore de cada alimentador uniformemente entre todas as possíveis árvores, tem-se um amostrador uniforme para o espaço de busca com k alimentadores e árvores com no máximo t vértices.*

Prova. A quantidade de árvores enraizadas, com no máximo t vértices, pode ser calculada com o auxílio da equação 5-1, como:

$$A = \sum_{i=1}^{t} T_i \tag{5-3}$$

Já o número de arranjos com repetição de A árvores em k alimentadores é dado por:

$$F = A^k \tag{5-4}$$

Neste sentido, sabe-se que um amostrador uniforme neste espaço de busca de F diferentes florestas, tem a mesma probabilidade de escolha para cada uma delas, ou seja, a probabilidade de escolha de um indivíduo I é:

$$P(I) = \frac{1}{F} \tag{5-5}$$

Agora, é preciso demonstrar que a seleção uniforme da árvore para cada alimentador, também produz uma escolha uniforme para o espaço de busca. Neste sentido,

sabe-se que a probabilidade da formação de um indivíduo é:

$$P(I) = \left(\frac{1}{A}\right)^k = \frac{1}{A^k} \tag{5-6}$$

Uma vez que é escolhida uma árvore entre todas as A árvores enraizadas com no máximo t vértices com a mesma probabilidade para cada um dos k alimentadores. Por fim, ao substituir 5-4 em 5-6, temos:

$$P(I) = \frac{1}{F} \tag{5-7}$$

□

Deste modo, a escolha de k árvores entre todas as árvores enraizadas com no máximo t vértices não possui tendência, uma vez que, a probabilidade de escolha de um indivíduo (equação 5-7) tem a mesma probabilidade de escolha de um amostrador uniforme (equação 5-5).

5.3 A tendência da inicialização

Esta seção tem como objetivo mostrar que a inicialização utilizando o algoritmo 4.3 provê indivíduos com a distribuição do número de barras por alimentador mais próxima a desejada, conforme descrito no experimento 4.4.2. Neste experimento o algoritmo de inicialização 4.3 foi comparado ao amostrador uniforme (seção 5.2), que serviu de testemunha, através de um DIC com 2 mil repetições. Os objetivos de cada unidade experimental foram dinâmicos, gerados a partir de uma floresta com 4 árvores na qual cada uma podia possuir entre 1 e 15 vértices.

O experimento mostrou que todas as florestas geradas pelo inicializador possuíram erro 0,0 enquanto que o amostrador uniforme (testemunha) o erro foi de 0,58 (figura 5.3). E, o teste Tukey permitiu a conclusão que a 1% de significância, as médias dos erros entre os tratamentos foram diferentes. Assim, tem-se um inicializador tendencioso com direcionamento adequado ao objetivo proposto.

Neste experimento é importante observar que o algoritmo 4.3 pôde criar 4 de árvores, ou seja, a mesma quantidade de árvores da qual foi gerada a distribuição que é fornecida como argumento para o algoritmo. Deste modo, o erro apresentado na figura 5.3 foi exatamente igual a zero. Caso haja diferença entre a quantidade de árvores, da qual foi gerada a distribuição de graus, e da quantidade de árvores a serem inicializadas é possível que o erro seja superior a zero, pois pode não ser possível obter a mesma distribuição.

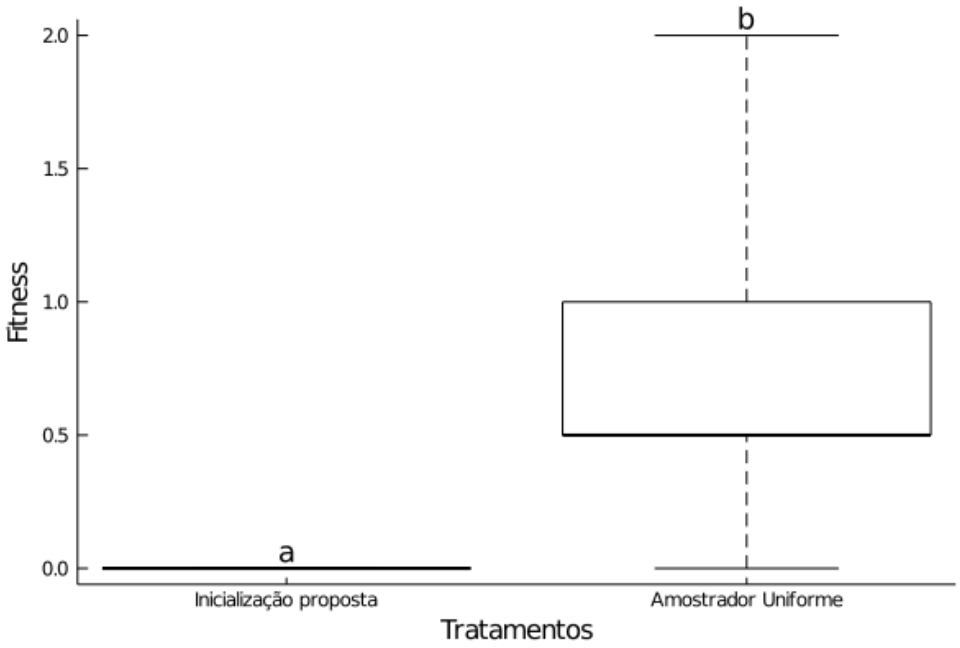


Figura 5.3: *Resultado experimental sobre a tendência da inicialização proposta. Os tratamentos com letras diferentes são estatisticamente diferentes a 1% de significância usando o teste de Tukey.*

5.4 Análise de tendência dos operadores de geração de árvores

Esta seção apresenta as análises dos algoritmos de criação de árvores usando NPE apresentados na seção 4.1.3 conforme o experimento, detalhado na seção 4.4.3, que analisa cada método de criação de árvore em relação ao amostrador uniforme (seção 5.2) e ao método Prüfer.

5.4.1 Algoritmo de construção de árvores

A principal vantagem do algoritmo 4.4 é gerar uma sequência diretamente na representação da NPE. Para verificar se o método proposto possuía tendência foi executado o experimento descrito na seção 4.4.3. Este experimento consiste em comparar o algoritmo 4.4 com o amostrador uniforme e o método Prüfer através de um DIC com árvores de 18 vértices e 1.500 repetições. Para cada repetição gerou-se uma árvore, observando a distância de Hamming mínima para um grafo estrela e para um grafo do tipo lista.

Através deste teste estatístico observou-se que a média da distância de Hamming para árvores com topologia estrela e lista diferiram estatisticamente segundo o teste de

Tukey a 1% de significância comparando o algoritmo proposto, o amostrador do espaço de busca e o método de Prüfer, conforme apresentado nas figuras 5.4 e 5.5.

Isto implica que, tanto o método proposto no algoritmo 4.4 quanto o método de Prüfer possuem tendências em relação ao espaço de busca estudado na seção 5.1. Na figura 5.4, a distância média do amostrador uniforme para um grafo estrela foi de 13,70; já a distância média do método proposto foi de 12,67; indicando que ele tende a gerar árvores mais próximas do tipo estrela, e a distância média do método Prüfer foi de 14,11; indicando que ele tende a gerar árvores mais distantes do tipo estrela do que o amostrador uniforme.

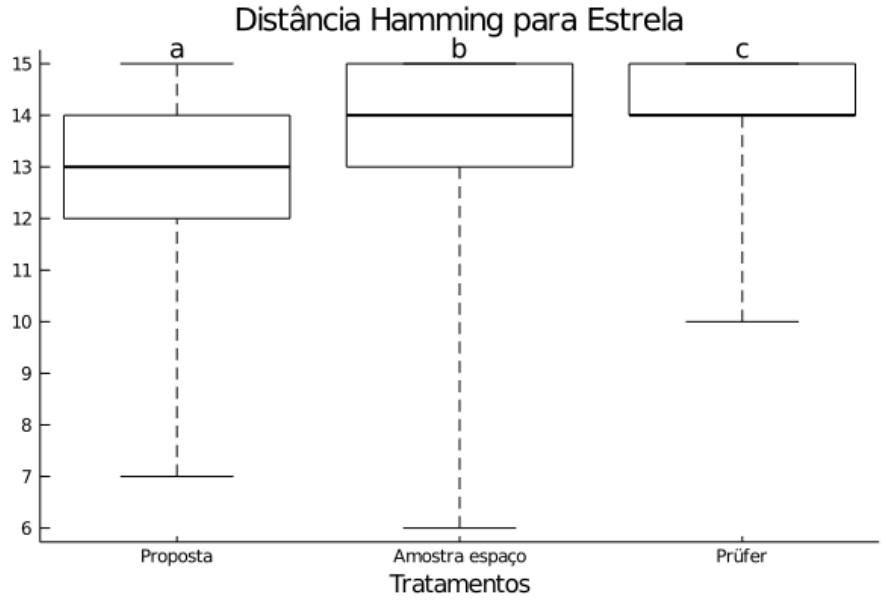


Figura 5.4: *Distância de Hamming para grafo estrela com 18 vértices. Os tratamentos com letras diferentes são estatisticamente diferentes segundo o teste de Tukey a 1% de significância.*

Na figura 5.5, a distância média do amostrador uniforme para um grafo do tipo lista foi de 6,41; já a distância média do método proposto foi de 7,11; indicando que ele tende a gerar árvores mais distantes do tipo lista, já a distância média do método de Prüfer foi de 5,23; indicando que ele tende a gerar árvores mais próximas do tipo lista do que o amostrador uniforme.

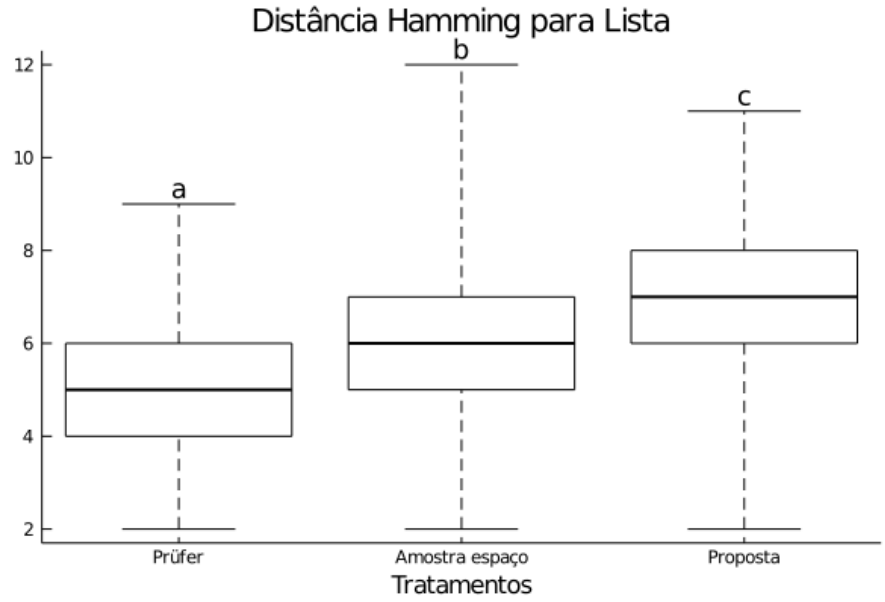


Figura 5.5: *Distância de Hamming para grafo do tipo lista com 18 vértices. Os tratamentos com letras diferentes são estatisticamente diferentes segundo o teste de Tukey a 1% de significância.*

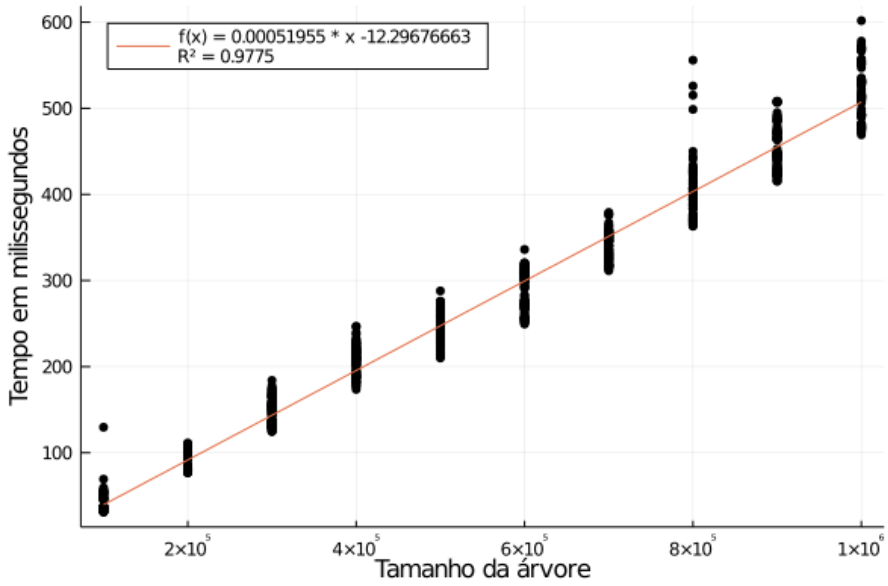


Figura 5.6: *Tempo de geração de árvores de 100 mil a 1 milhão de vértices com intervalo de 100 mil vértices.*

A figura 5.6 apresenta a observação do tempo de geração de árvores do algoritmo 4.4, conforme o experimento descrito na seção 4.4.4. Através do gráfico e do coeficiente de determinação (R^2) com valor 0,9775, é possível observar a complexidade linear prevista nos cálculos (seção 4.1.3), uma vez que, o coeficiente de correlação de Pearson (r) igual a 0,99 indicou uma correlação muito forte ($r \geq 0,90$).

5.4.2 Algoritmo de construção de árvores com grau máximo

Para avaliar a tendência do algoritmo 4.6 variou-se o limite de grau de 2 até 17, uma vez que, o comportamento do algoritmo depende desse parâmetro. Neste sentido, para cada limite de grau gerou-se 1.500 árvores com 18 vértices. Estes resultados foram comparados com uma amostra de mesmo tamanho do amostrador uniforme (seção 5.2) e o método de Prüfer, conforme descrito no experimento 4.4.3. Para cada tratamento, foi observado o valor mínimo, médio e máximo.

O resultado do experimento de análise de tendência do algoritmo 4.6 é mostrado na figura 5.7. A linha preta, representa a distância média do amostrador uniforme para um grafo do tipo estrela, cuja média foi 13,69; sendo a distância máxima encontrada igual a 15 para todos os métodos, a distância média para o método Prüfer foi de 14,11 e mínima de 10.

Já como esperado, a tendência do método proposto muda conforme o limite de grau máximo, cuja distância mínima encontrada foi 8 e média em 12,97. É importante notar que o método proposto apresenta distância máxima quando o grau máximo é 2, gerando apenas árvores do tipo lista e a estabilização da média ocorre por volta do grau máximo 5.

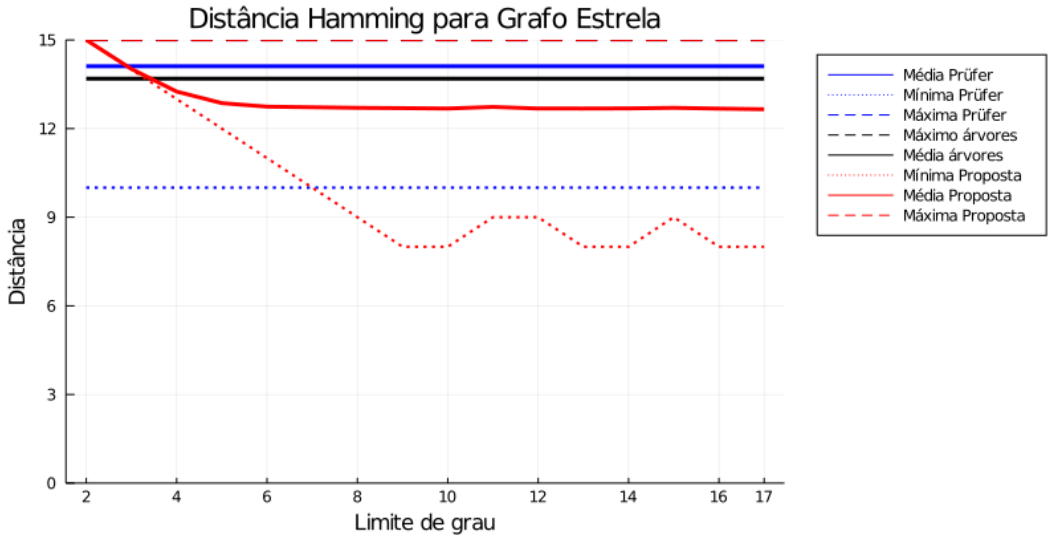


Figura 5.7: *Distância de Hamming mínima para grafo estrela com 18 vértices.*

A mesma análise foi realizada em relação a distância mínima para um grafo do tipo lista, conforme apresentado na figura 5.8. A distância média do amostrador uniforme foi de 6,51 e máxima de 15; já a distância mínima, média e máxima do método Prüfer foram de 2; 5,23; e 9. O algoritmo proposto teve distância máxima de 12 e média 6,49. Observou-se que, assim como supracitado, quando o limite de grau é 2, somente são gerados grafos do tipo lista, ou seja, distância mínima, média e máxima iguais a 0. Além disto, conforme o grau máximo permitido aumenta, também aumenta a distância média

do grafo tipo lista. É importante notar que, a média começa a estabilizar quando o grau máximo é 5 mantendo-se próxima da média de um amostrador uniforme.

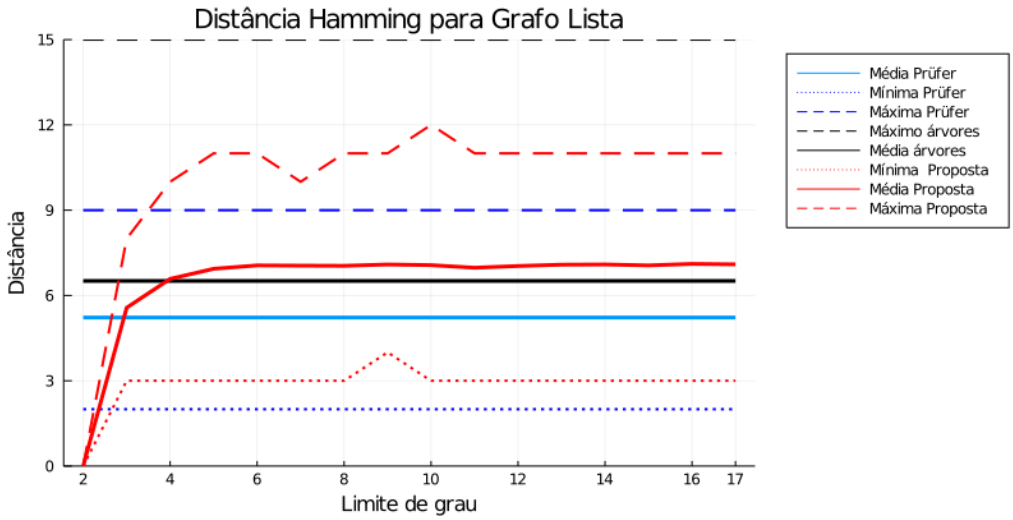


Figura 5.8: *Distância de Hamming mínima para grafo to tipo lista com 18 vértices.*

5.4.3 Algoritmo de construção de árvores com profundidade máxima

De modo similar ao experimento apresentado na seção 5.4.2, para avaliação da tendência do algoritmo 4.7 variou-se o limite de profundidade de 2 até 17, para cada limite de profundidade gerou-se 1.500 árvores com 18 vértices. Estes resultados foram comparados com uma amostra de mesmo tamanho do amostrador uniforme (seção 5.2) e o método de Prüfer, conforme descrito no experimento 4.4.3. Para cada tratamento, foi observado o valor mínimo, médio e máximo.

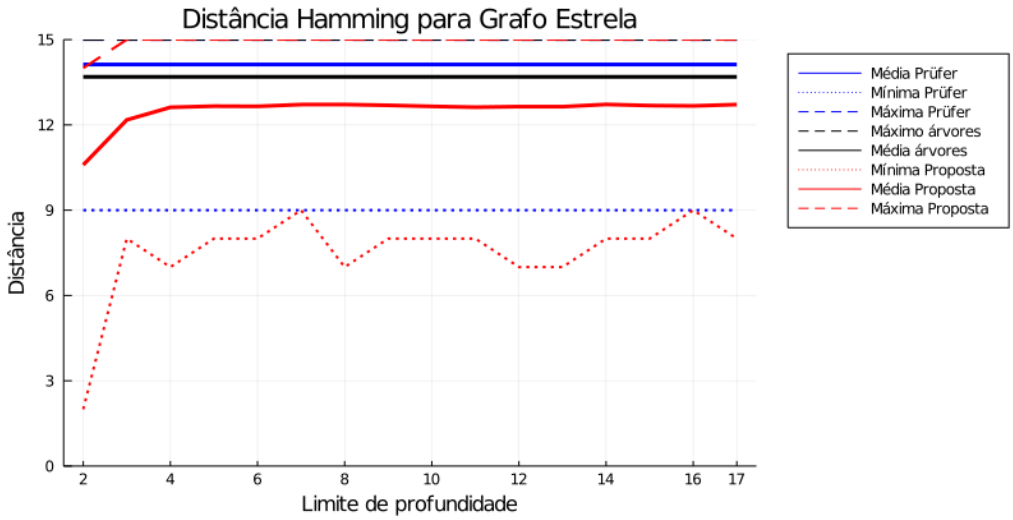


Figura 5.9: *Distância de Hamming mínima para grafo to tipo estrela com 18 vértices.*

Nas figuras 5.9 e 5.10 o amostrador uniforme e o método de Prüfer tiveram resultados muito parecidos aos do resultado do experimento apresentado na seção 5.4.2. A figura 5.9 apresenta os resultados obtidos do experimento de análise de tendência do algoritmo 4.7 em relação a grafos do tipo estrela. Em relação ao método proposto, a distância média estabilizou quando o limite de profundidade foi 4, tendo média geral de 12,51, mínima de 2 e máxima 15.

A mesma análise foi realizada para o grafo do tipo lista. Os resultados obtidos são apresentados na figura 5.10. A distância mínima, média e máxima para o método proposto foram de 2; 7,45 e 14. Observou-se também que a distância média começa a estabilizar quando o limite de profundidade se aproxima de 5.

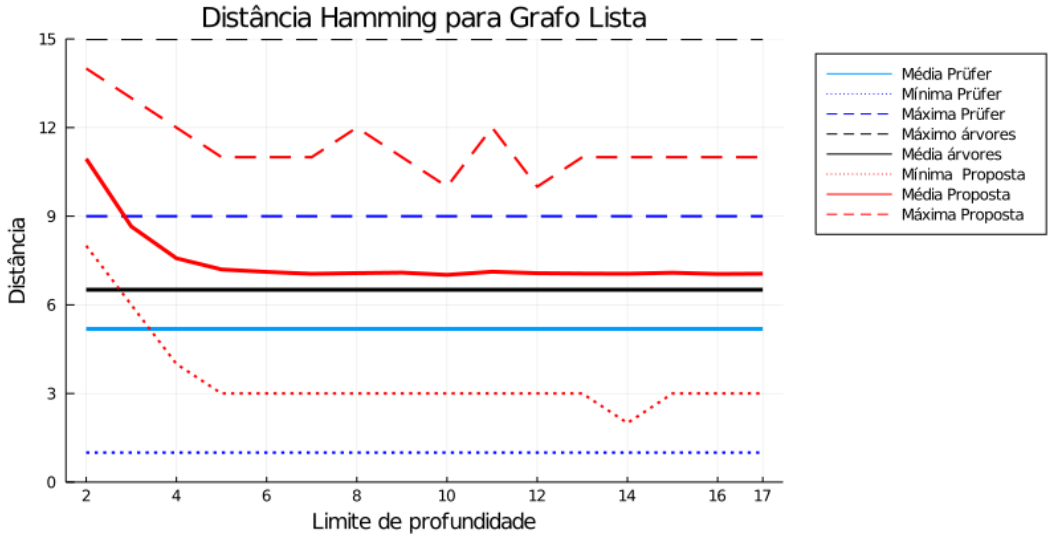


Figura 5.10: *Distância de Hamming mínima para grafo to tipo lista com 18 vértices.*

5.4.4 Tendência na geração de árvores por composição de floresta

Assim como a análise de tendência apresentada na seção 5.4.1. Esta seção apresenta o experimento que visa a análise da tendência do algoritmo 4.8 através da comparação com o amostrador uniforme e o método Prüfer utilizando um DIC com árvores de 18 vértices e 1.500 repetições. Para cada repetição gerou-se uma árvore, observando a distância de Hamming mínima para um grafo estrela e para um grafo do tipo lista, conforme descrito na seção 4.4.3.

Após o experimento foi possível observar que a 1% de significância tanto o método de Prüfer quanto o algoritmo de criação de árvore por composição possuíram tendência em relação ao gerador de amostras não tendencioso.

Na figura 5.11 a média do amostrador uniforme foi de 13,65; já a média do método Prüfer foi de 14,08 e a média do método proposto foi de 14,42. Indicando que o método proposto tende a gerar grafos mais distantes de um grafo estrela. Já na figura 5.12

a média do amostrador uniforme foi de 6,49; a média do método Prüfer foi de 5,26 e a média do método proposto foi de 4,17. Indicando que o algoritmo proposto tende a gerar grafos mais próximos de grafos tipo lista.

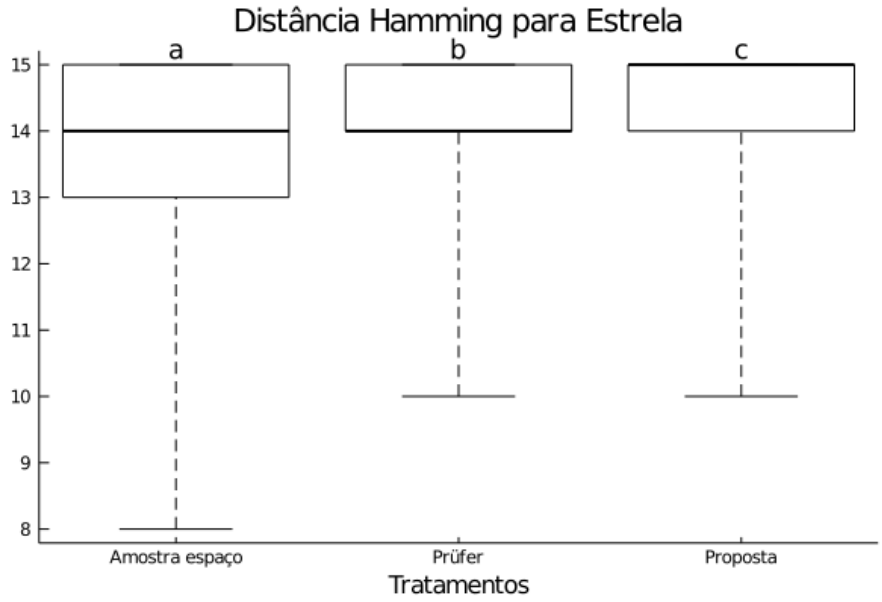


Figura 5.11: *Distância de Hamming para grafo estrela com 18 vértices. Os tratamentos com letras diferentes são estatisticamente diferentes segundo o teste de Tukey a 1% de significância.*

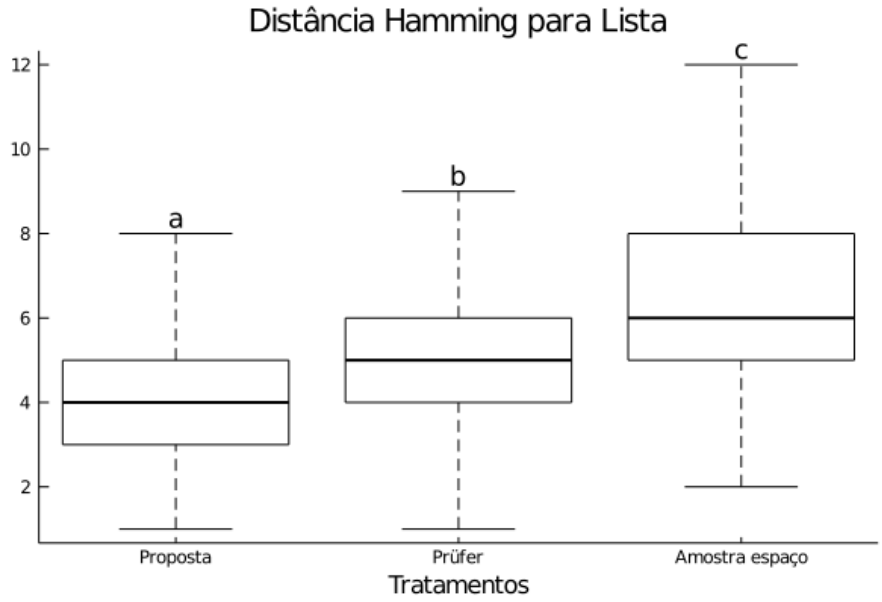


Figura 5.12: *Distância de Hamming para grafo do tipo lista com 18 vértices. Os tratamentos com letras diferentes são estatisticamente diferentes segundo o teste de Tukey a 1% de significância.*

5.5 Inserção de tendência com o PSPR e PTBR

Nesta seção, as versões convencionais do SPR e TBR serão comparadas com as versões probabilísticas PSPR e PTBR a fim de avaliar os impactos do vetor de probabilidade e a melhoria na convergência. Este experimento está descrito na seção 4.4.5 e seu objetivo é obter uma árvore estrela.

As funções utilizadas no PSPR e PTBR para direcionar a tendência foram $f_p(v) = grau\_maximo(G) - grau(v.pai)$ e $f_a(v) = f_r(v) = grau(v)$. Onde $grau\_maximo(G)$ é o maior grau da árvore G e $grau(v)$ é o grau do vértice v. De forma simplificada, para f_p uma subárvore tem maiores chances de ser podada se o vértice pai dela possui baixo grau, já para f_a quanto maior o grau do vértice, maior a probabilidade de ele receber a subárvore podada e consequentemente aumentar o seu grau, e para f_r quanto maior o grau, maior a chance de ele ser a nova raiz da subárvore.

Após a coleta de dados, a análise de variância identificou que ao nível de 1% de significância o grau máximo das árvores finais era estatisticamente diferente. A fim de identificar cada classe realizou-se o teste de Tukey. As médias e as classes de cada tratamento são apresentados na figura 5.13.

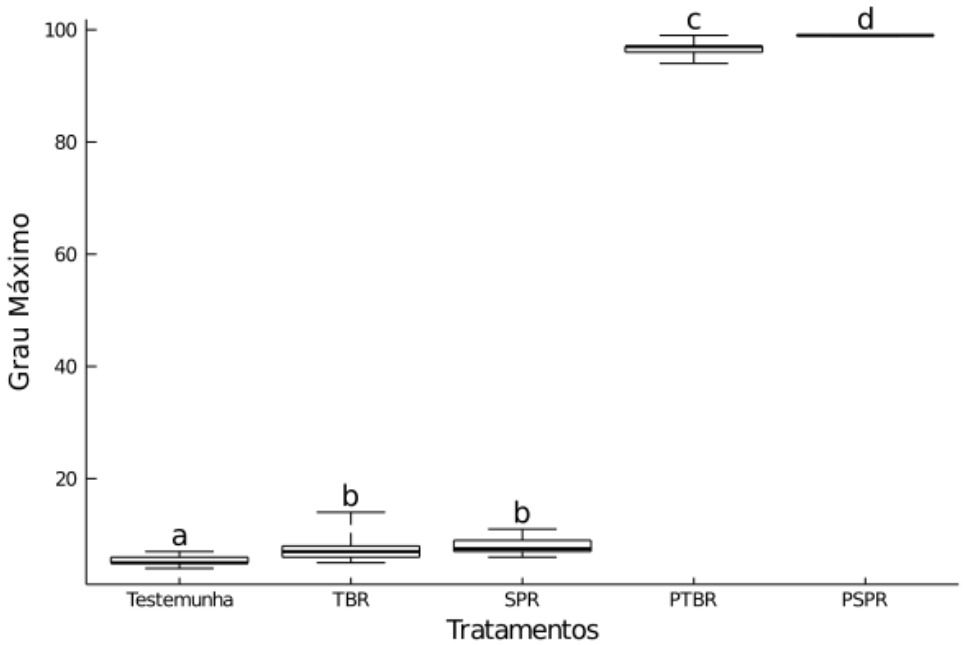


Figura 5.13: *Avaliação de tendência nos operadores. Médias seguidas pela mesma letra não diferem significativamente entre si pelo teste de Tukey ao nível de 1% de significância.*

O experimento mostrou que através de funções de pesos apropriadas é possível direcionar os operadores PSPR e PTBR, pois o grau máximo médio destes operadores

foram respectivamente 99,00 e 96,63. Sendo muito próximo do objetivo desejado que era 99.

No entanto, o grau médio máximo obtido pelo SPR e TBR foi respectivamente 7,83 e 7,47 que em relação à testemunha com 5,23 não indica uma tendência forte, em termos absolutos, na geração de grafos estrela corroborando com os resultados encontrados por Lima et al. [13] a respeito da ausência de tendência destes operadores.

É importante mencionar que nesta configuração quando os operadores PSPR e PTBR produzem o grafo estrela desejado todos os vértices não-raiz tem a mesma probabilidade $f_p(v) = 0$. Uma vez que tanto no SPR quanto no TBR a raiz da árvore não é candidata para poda por não possuir vértice pai, todos os possíveis vértices da árvore têm a mesma probabilidade de serem escolhidos fazendo com que o PSPR e PTBR voltem a agir como o SPR e TBR, selecionando um vértice qualquer da árvore e possivelmente reduzindo o grau máximo da árvore. Isto explica parcialmente porque a média destes operadores não chegou ainda mais próxima do valor 99 desejado.

5.6 Avaliação do processo de busca topológica do gSDE

Esta seção apresenta os resultados dos experimentos da busca topológica apresentando a evolução do *fitness* de cada objetivo, a análise de tempo demandado no processo e a qualidade dos resultados obtidos, conforme o planejamento realizado na seção 4.4.6. Na configuração de referência da busca foram utilizados os parâmetros apresentados na tabela 5.3.

Parâmetro	**Valor**
Tamanho da população	300
Método de inicialização	variante do algoritmo 4.3 de raiz com grau 1 (seção 4.1.6) com o método de Prüfer
Método de seleção	seleção por torneio
Taxa de *crossover*	80%
Crossover	troca de um alimentador entre os dois genitores
Taxa de mutação	20%
Mutação	variantes SPRN e TBRN que mantém a raiz com grau 1 (algoritmos 4.9 e 4.10)
Número de objetivos	3
Funções de *fitness*	distribuição de graus (figura 3.10), folhas (figura 3.12) e barras por alimentador (figura 3.11)
Número de iterações	5.000

Tabela 5.3: *Parâmetros de referência utilizados na busca topológica.*

As figuras 5.14 e 5.15 apresentam dados do processo de busca seguindo a parametrização apresentada na tabela 5.3. Na figura 5.14, é apresentada a evolução de cada objetivo ao longo do processo de busca. Nesta figura, é importante perceber que a qualidade inicial do número de barras por alimentador se manteve ao longo do processo de busca. Indicando que a qualidade do algoritmo de inicialização (seção 4.1.3) foi adequada dentro de seus limites. Além disso, é relevante observar que os objetivos sobre a distribuição de graus e a quantidade de folhas por alimentador conseguiram melhorar ao longo do processo de busca. De forma qualitativa, as grandes melhorias ocorreram até 1.000 gerações.

Figura 5.14: *Evolução da adaptação do melhor indivíduo de cada objetivo ao longo do processo de busca para os três objetivos.*

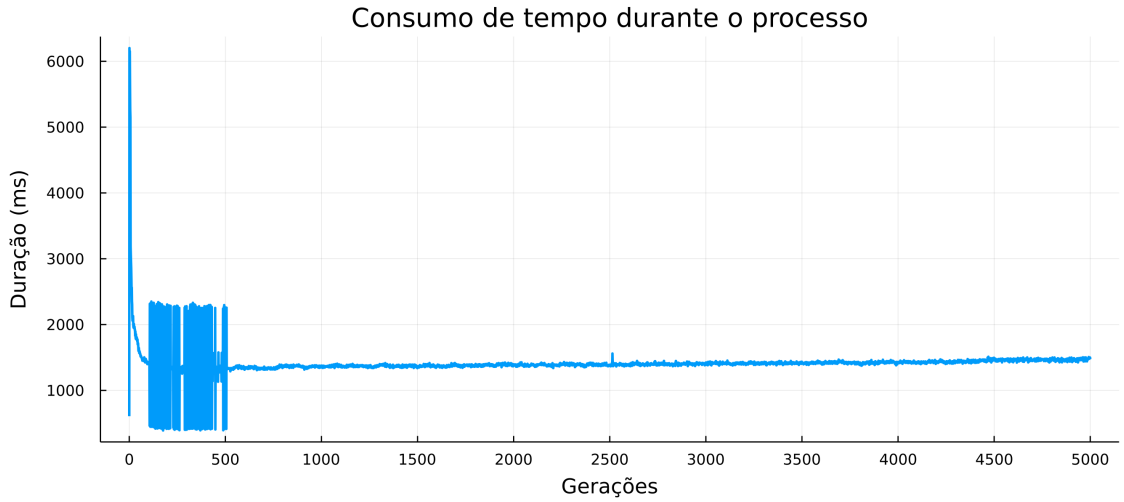


Figura 5.15: *Consumo de tempo em cada geração durante o processo de busca.*

A figura 5.15 apresenta o consumo de tempo durante as gerações do processo de busca. Nela foi importante observar que o tempo da inicialização foi superior a média de

tempo demandado. Isto acontece por dois fatores: 1) tempo de pré-compilação de código da linguagem Julia; 2) demanda de conversões das representações em números Prüfer para NPE. Além disso, o tempo médio de cada geração foi de 1.407 milissegundos. Uma das principais influências deste tempo é a distribuição de barras por alimentador (figura 3.11). Quando a última classe desta distribuição possui um limite superior de barras muito alto e sua frequência é alta, espera-se que as árvores geradas tenham muitos vértices influenciando no tempo de avaliação do *fitness*, *crossover* e mutação.

As figuras 5.16, 5.17 e 5.18 apresentam as comparações das distribuições do indivíduo de menor erro geral (IMEG) com as distribuições desejadas, este indivíduo é selecionado na primeira fronteira de Pareto considerando o menor erro absoluto entre as funções objetivo.

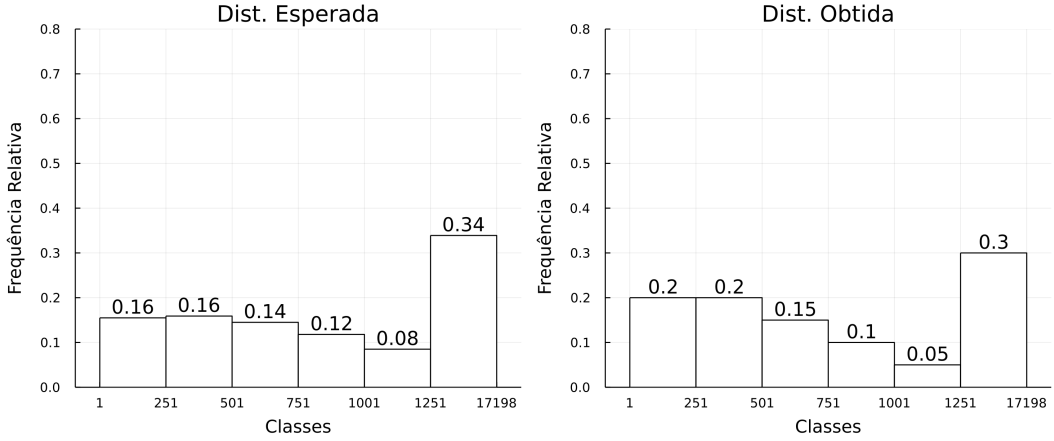


Figura 5.16: *Distribuição do número de barras por alimentador do IMEG comparada a distribuição esperada. Erro: 0,18.*

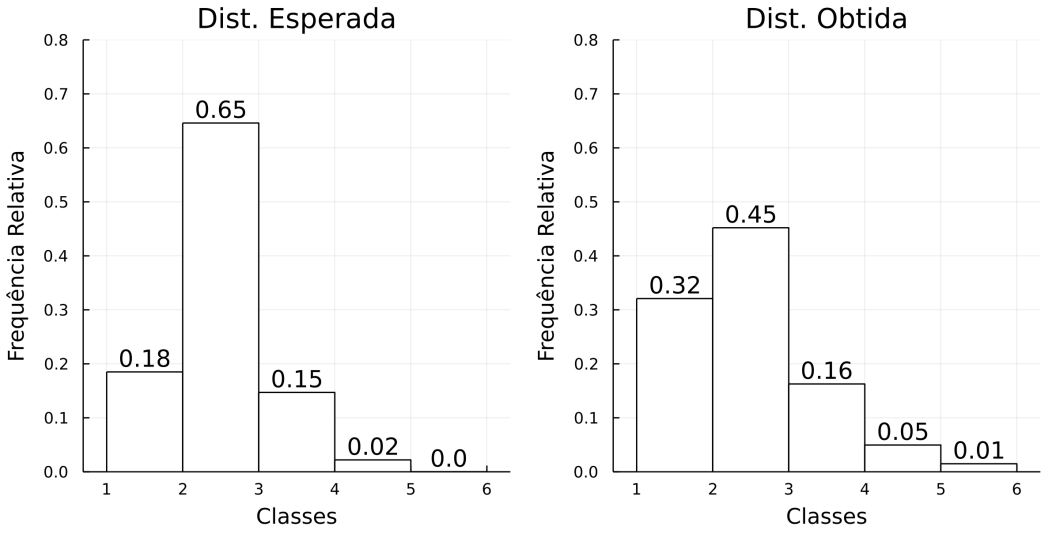


Figura 5.17: *Distribuição dos graus das barras do IMEG comparada com a distribuição esperada. Erro: 0,39.*

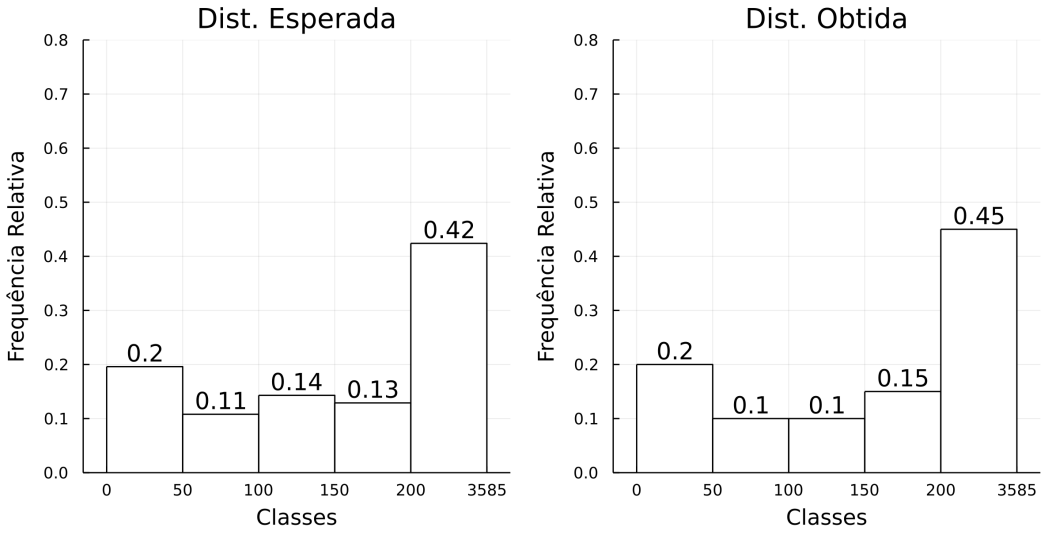


Figura 5.18: *Distribuição do número de folhas por alimentador IMEG comparada com a distribuição esperada. Erro: 0,10.*

É importante notar que, o IMEG não é o melhor indivíduo da busca para cada uma das distribuições de forma separada. Deste modo, acreditou-se que esta seria a forma mais justa de avaliar qualitativamente a solução. Os melhores indivíduos para cada busca não serão mostrados aqui porque geralmente foi possível observar um indivíduo muito próximo do ótimo para cada uma das distribuições desejadas. De modo gráfico, isto representa duas distribuições praticamente idênticas.

Mesmo observando apenas a distribuições do IMEG foi possível notar que elas estiveram muito próximas das distribuições desejadas. Juntamente as observações realizadas a respeito da evolução do *fitness* na figura 5.14, isto mostra que a escolha do processo de busca, representação e operadores foram adequadas.

As figuras 5.19 e 5.20 apresentam dois exemplos de topologias de alimentadores retornados pelo processo de busca. Nestes exemplos são observáveis segmentos longos de grau 2 que também são observáveis em SDEs reais. Isto, mostra que além dos dados quantitativos, questões qualitativas também foram atingidas no final da busca.

A figura 5.21 apresenta a fronteira de Pareto (F1) de uma busca topológica. Nela é possível perceber que os indivíduos encontrados estavam bem distribuídos.

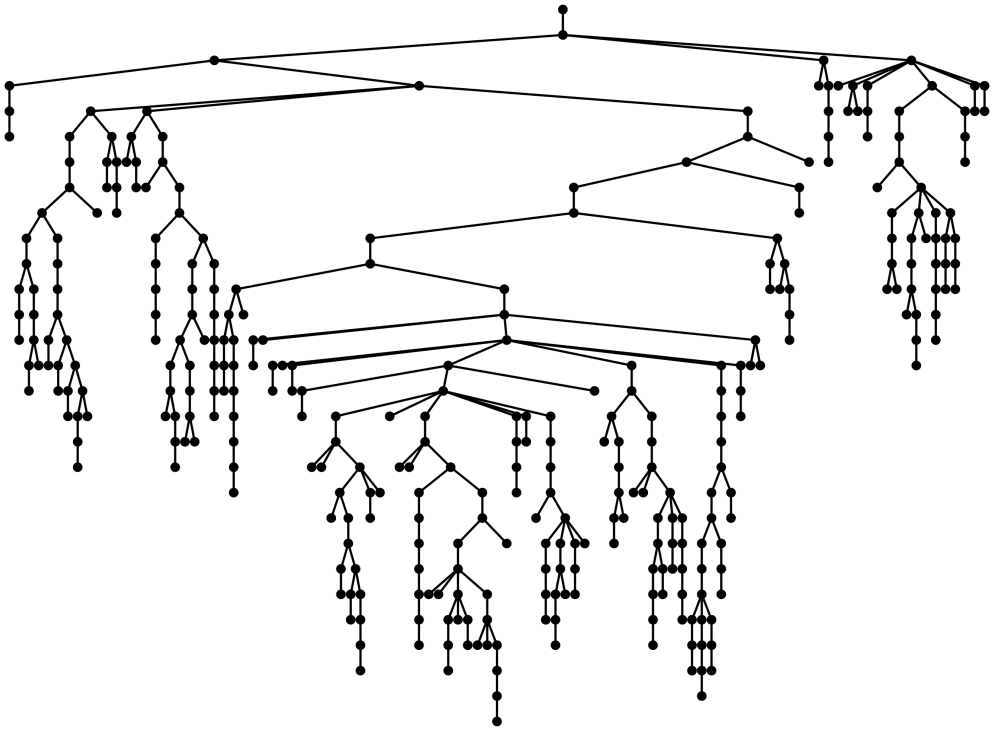


Figura 5.19: *Exemplo 1 de topologia retornada pelo processo de busca.*

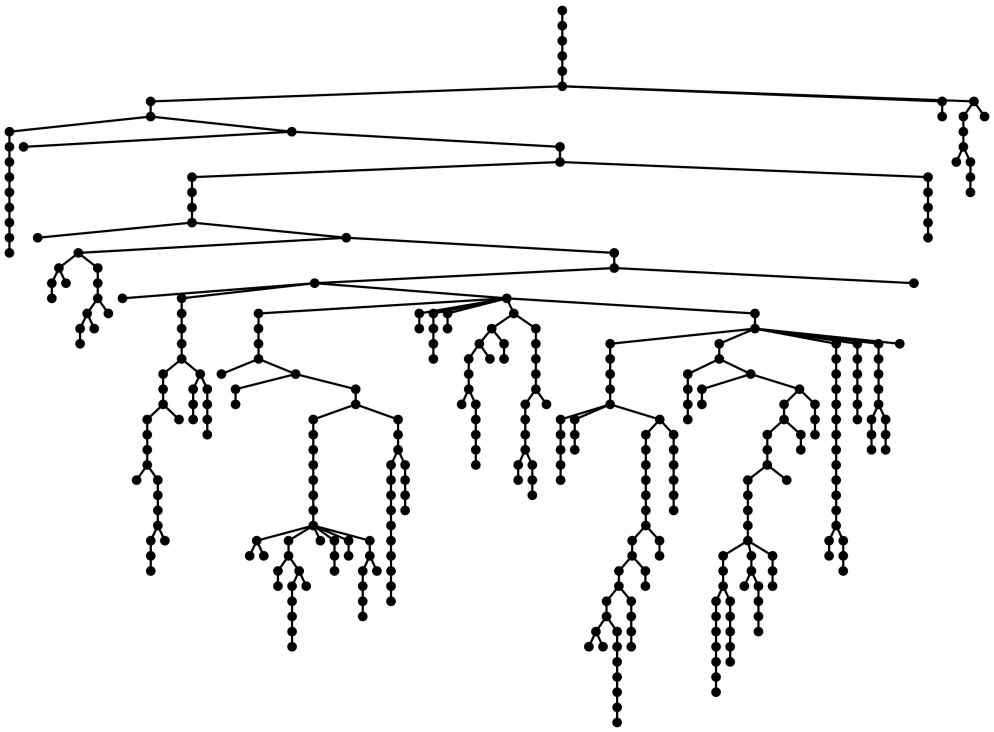


Figura 5.20: *Exemplo 2 de topologia retornada pelo processo de busca.*

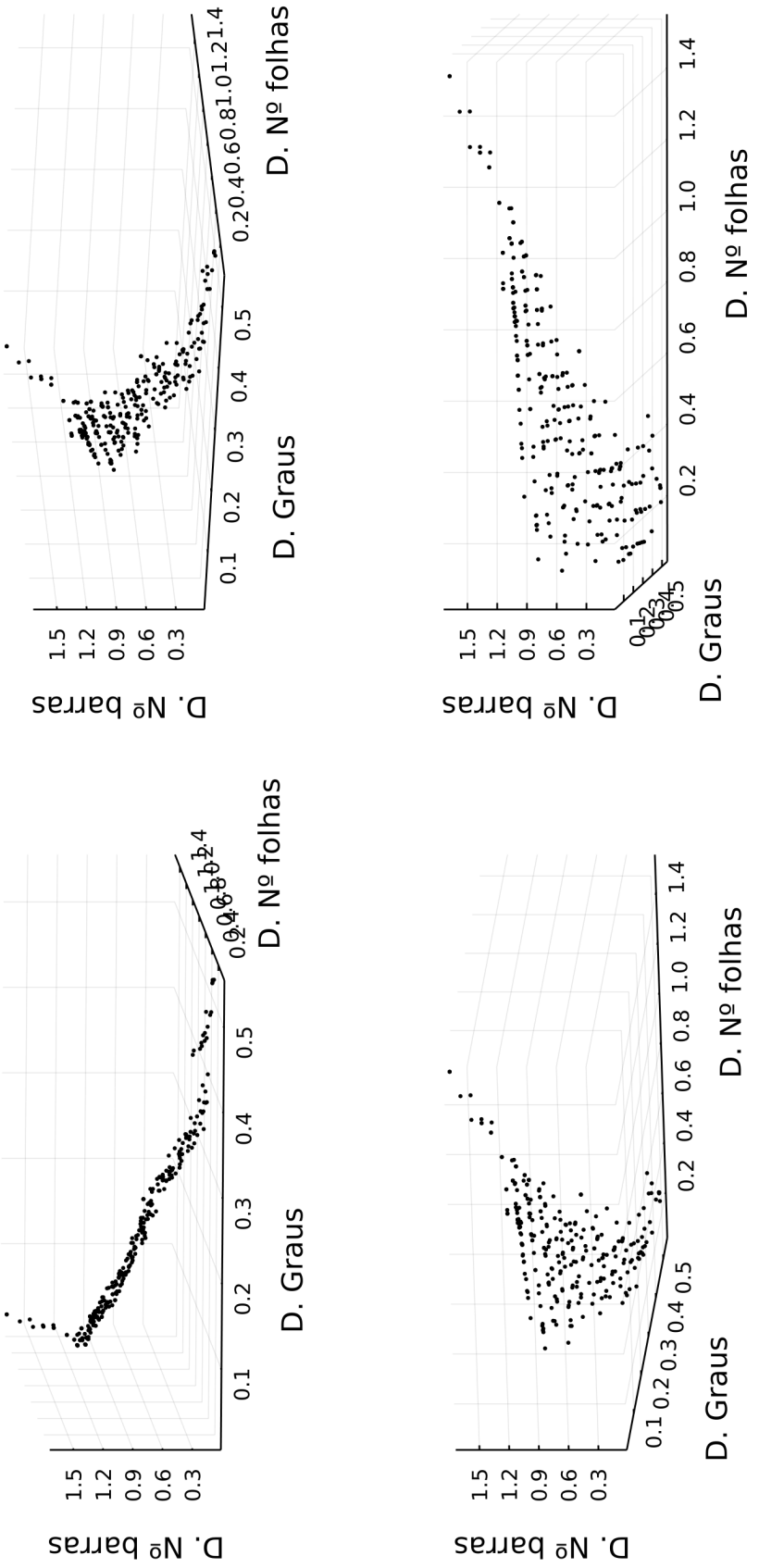


Figura 5.21: *Fronteira de Pareto da busca topológica de referência seguindo a parametrização da tabela 5.3.*

5.7 Busca sem o número de barras por alimentador como objetivo

Uma vez que a inicialização do algoritmo 4.3 provê um bom posicionamento da população inicial conforme demonstrado na seção 5.3. É racional pensar que tal objetivo possa ser excluído da busca, pois economizaria processamento computacional durante as gerações.

No entanto, ao retirar este objetivo da busca este critério piora ao longo do processo, conforme apresentado na figura 5.22, no qual todos os quartis aumentam durante as gerações. Isto acontece porque durante a troca de material genético (*crossover*) entre os indivíduos, duas árvores de tamanhos diferentes podem ser trocadas mudando a relação do número de barras por alimentador.

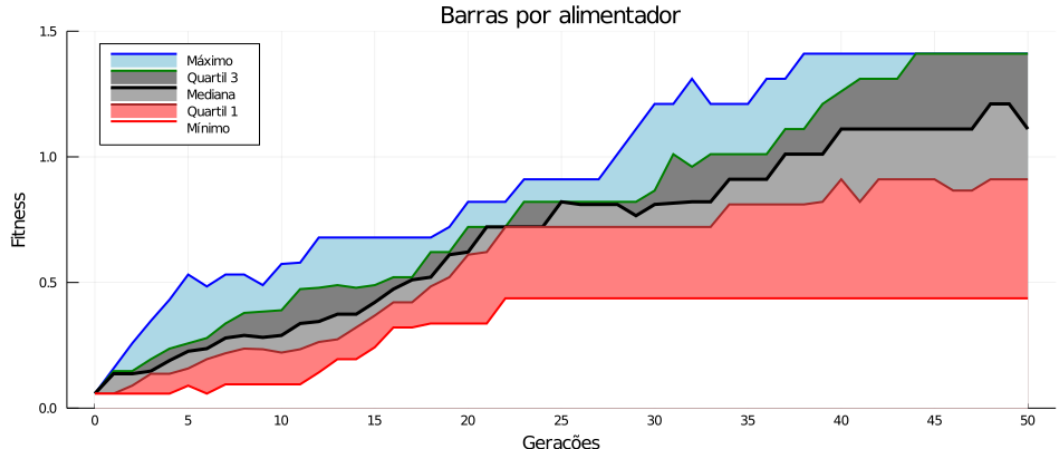


Figura 5.22: *Evolução da quantidade de barras por alimentador quando este critério é removido dos objetivos.*

Ao retirar as operações de *crossover* do mesmo experimento, o número de barras por alimentador permanece imutável, preservando a qualidade dos indivíduos da inicialização. Assim, concluiu-se que ou este objetivo permanece na busca, ou seria utilizado um novo operador de *crossover* que considere o número de vértices em cada árvore trocada durante a operação.

A figura 5.23 apresenta os resultados experimentais em relação a este objetivo quando empregamos o *crossover* por tamanho de árvore descrito na seção 4.1.4 com 100 vértices de tolerância. Nota-se a perda de qualidade da inicialização foi menos acentuada do que a troca indiscriminada de árvores entre os genitores. Mesmo assim, ainda houve perda da qualidade da inicialização.

A figura 5.24 apresenta os resultados experimentais em relação ao objetivo quando empregou-se o *crossover* por *locus* descrito na seção 4.1.4. Nota-se que a distribuição do número de barras por alimentador foi mantida do início ao fim da busca.

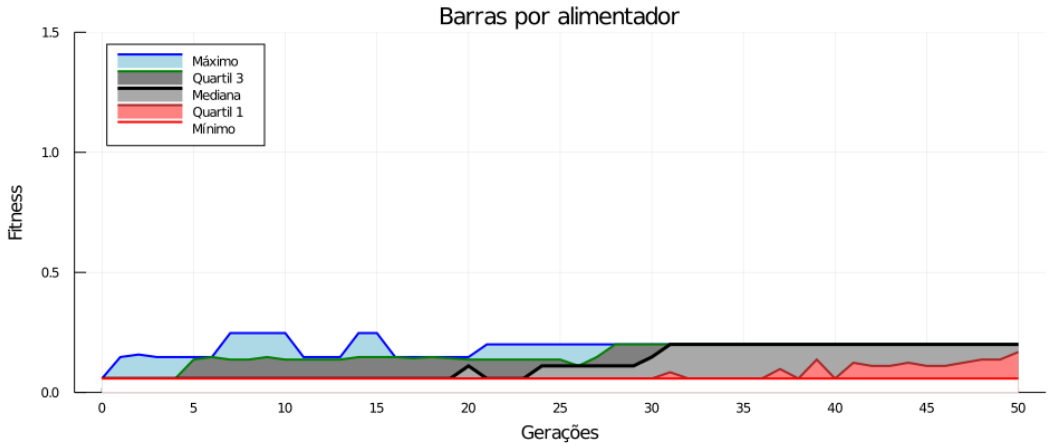


Figura 5.23: *Evolução da quantidade de barras por alimentador quando este critério é removido dos objetivos utilizando o crossover por tamanho de árvore.*

Figura 5.24: *Evolução da quantidade de barras por alimentador quando este critério é removido dos objetivos utilizando o crossover por locus.*

Dada a manutenção da qualidade da distribuição de barras por alimentador fornecida pelo *crossover* por *locus*, elaborou-se um novo experimento com este operador, conforme a parametrização apresentada na tabela 5.4. Neste experimento além da nova escolha de operador de *crossover*, também foi utilizada a inicialização por composição de árvores do algoritmo 4.8. Este algoritmo de criação de árvores foi escolhido porque ele tende a gerar árvores mais próximas de um grafo lista, conforme apresentado na figura 5.12, em relação ao método de Prüfer. Como o SDE em estudo apresenta características próximas de grafos tipo lista, considerou-se esta a escolha mais adequada.

A figura 5.25 apresenta a evolução dos objetivos considerando somente as duas distribuições. Nota-se que a busca rapidamente conseguiu encontrar o melhor indivíduo para a distribuição de folhas por alimentador. O algoritmo 4.8 de composição de árvores posicionou a população inicial em zonas mais promissoras quando comparado ao método

Prüfer (figura 5.14). No entanto, a convergência da distribuição de graus aconteceu de modo mais lento quando comparada com a evolução com três objetivos (figura 5.14). Acredita-se que isto tenha relação à redução da diversidade genética embutida pelo *crossover* por *locus*. Tal característica poderia ser compensada aumentando-se a taxa de mutação do processo de busca.

Parâmetro	**Valor**
Tamanho da população	300
Método de inicialização	variante do algoritmo 4.3 de raiz com grau 1 (seção 4.1.6) com o algoritmo 4.8 de composição de árvores
Método de seleção	seleção por torneio
Taxa de *crossover*	80%
Crossover	*crossover* por *locus*
Taxa de mutação	20%
Mutação	variantes SPRN e TBRN que mantém a raiz com grau 1 (algoritmos 4.9 e 4.10)
Número de objetivos	2
Funções de *fitness*	distribuição de graus (figura 3.10) e distribuição de folhas por alimentador (figura 3.12)
Número de iterações	5.000

Tabela 5.4: *Parâmetros utilizados na busca topológica com 2 objetivos.*

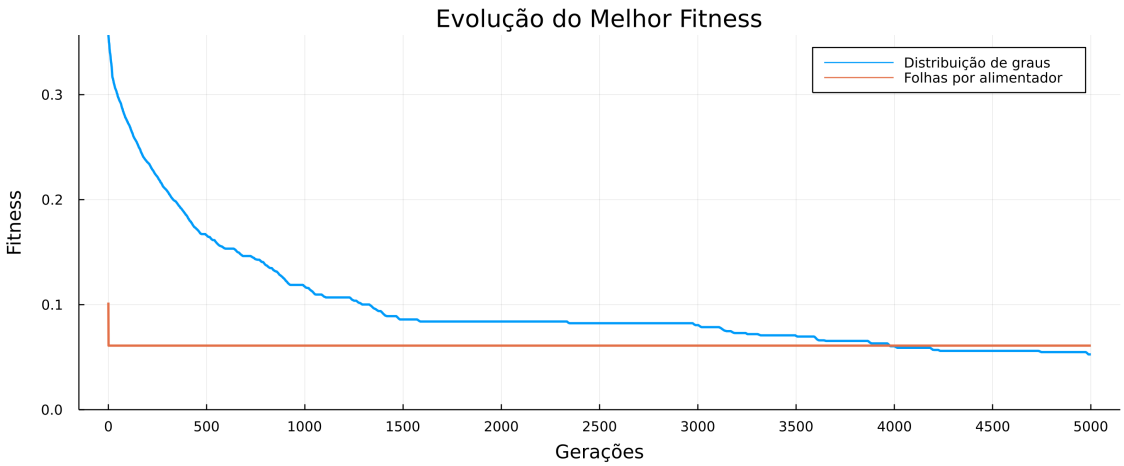


Figura 5.25: *Evolução da adaptação do melhor indivíduo de cada objetivo ao longo do processo de busca para os dois objetivos.*

A figura 5.26 apresenta o consumo de tempo ao longo das gerações. Observou-se que o tempo médio demandado por geração foi de 2.214 milissegundos. Acredita-se que

estes 807 milissegundos a mais do que o método de referência se deve ao comportamento da população. Após o *crossover* a primeira fronteira de Pareto (F1) frequentemente superou o tamanho máximo da população, fazendo com que a escolha seja dos indivíduos seja feita somente na F1 através da distância populacional do NSGA-II.

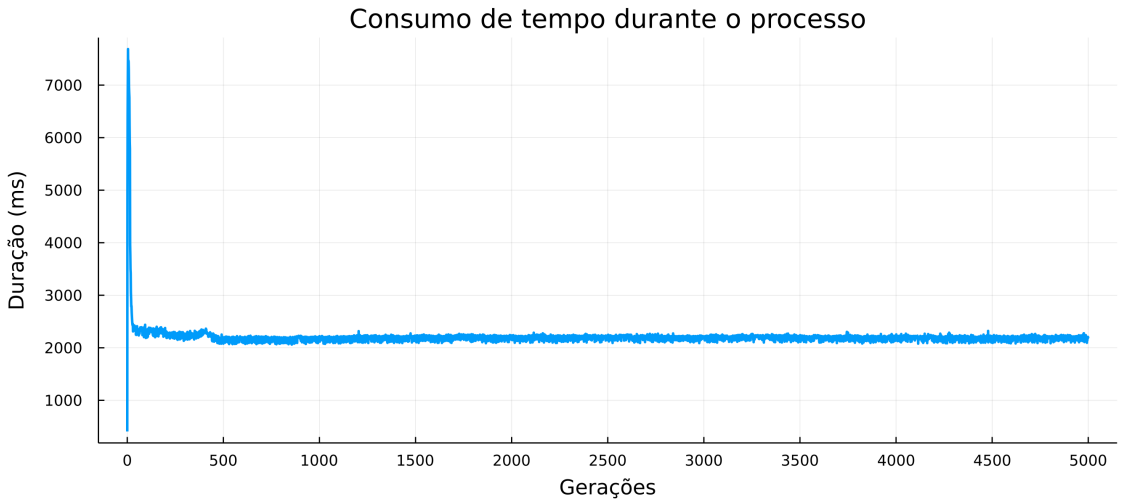


Figura 5.26: *Consumo de tempo em cada geração durante o processo de busca com dois objetivos.*

As figuras 5.29, 5.27 e 5.28 apresentam a comparação entre a distribuição obtida do indivíduo de menor erro geral (IMEG) e a distribuição esperada na busca para as três distribuições: graus das barras, folhas por alimentador e barras por alimentador.

As figuras 5.30 e 5.31 apresentam dois exemplos de árvores do IMEG. Nelas, também é possível observar segmentos formando linhas assim como são observados nos alimentadores reais.

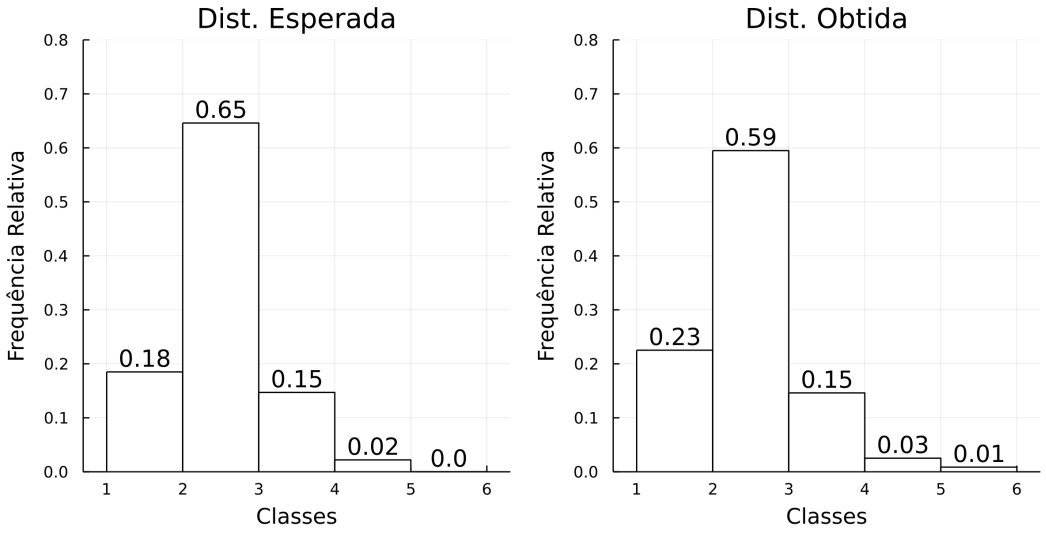


Figura 5.27: *Distribuição dos graus das barras do IMEG comparada com a distribuição esperada. Erro: 0,13.*

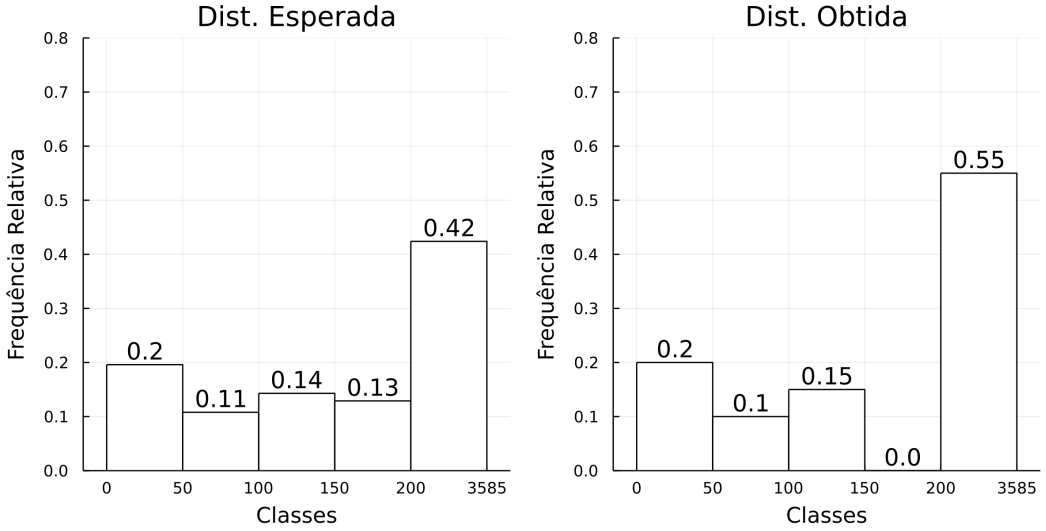


Figura 5.28: *Distribuição do número de folhas por alimentador IMEG comparada com a distribuição esperada. Erro: 0,28.*

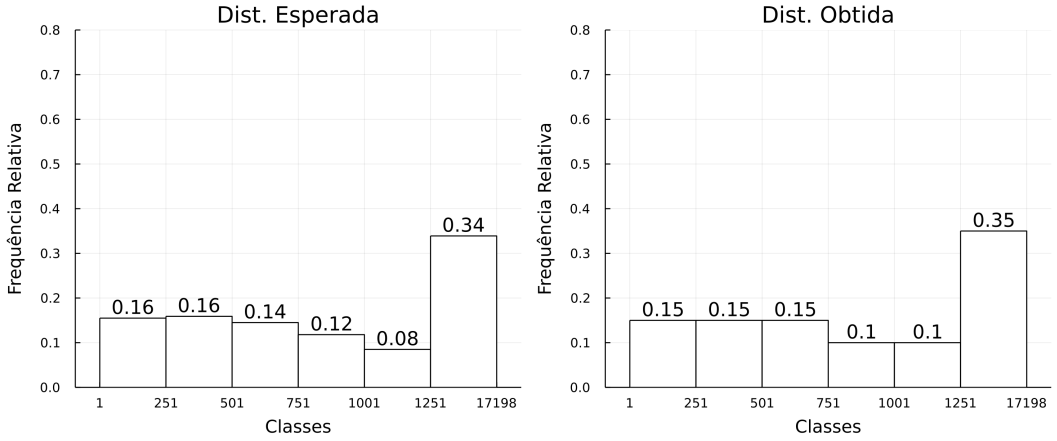


Figura 5.29: *Distribuição do número de barras por alimentador do IMEG comparada a distribuição esperada. Erro: 0,08.*

A fim de validar se as distribuições esperadas e obtidas eram similares foi utilizado o teste de Kolmogorov-Smirnov biamostral [29]. Neste teste o número de amostras da distribuição esperada foi 2.265 alimentadores enquanto que a amostra obtida na busca topológica foi de 20 alimentadores gerados. O teste indicou que todas as distribuições apresentadas nesta seção e na seção 5.6 são estatisticamente semelhantes ao nível de significância de 1%. Desta forma pode concluir que a busca topológica atingiu seu objetivo de modo quantitativo e qualitativo assim como desejado.

Por fim, a figura 5.32 apresenta a primeira fronteira de Pareto retornada pelo processo de busca. Nela é possível visualizar que foram encontrados indivíduos muito próximos de cada distribuição desejada e indivíduos entre esses dois extremos.

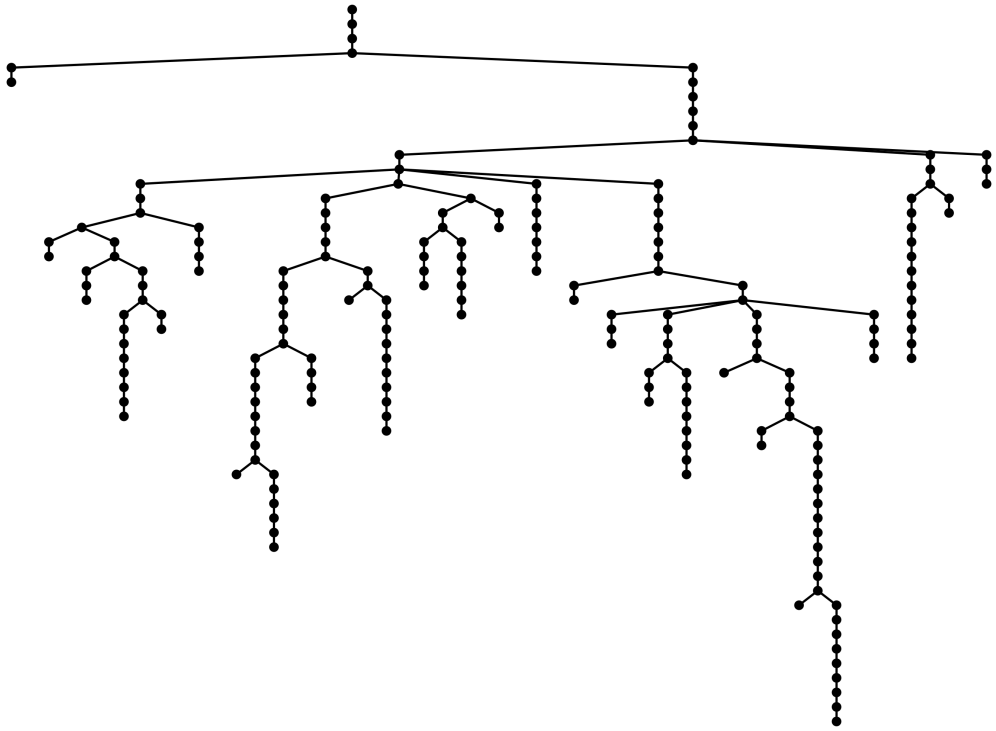


Figura 5.30: *Exemplo 1 de topologia retornada pelo processo de busca com 2 objetivos.*

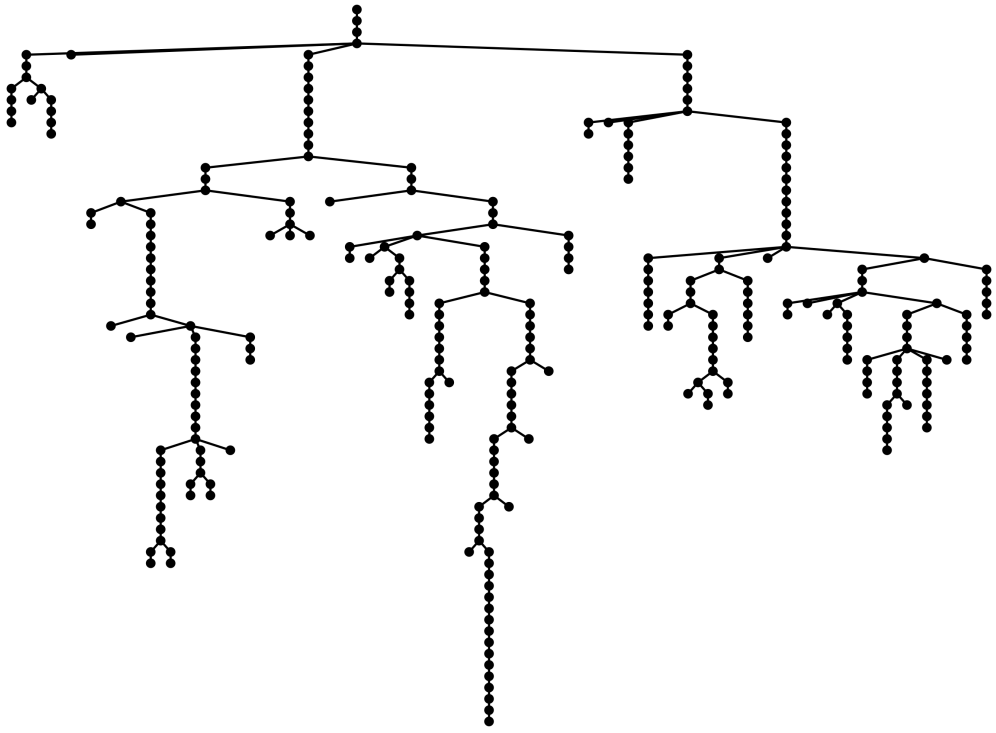


Figura 5.31: *Exemplo 2 de topologia retornada pelo processo de busca com 2 objetivos.*

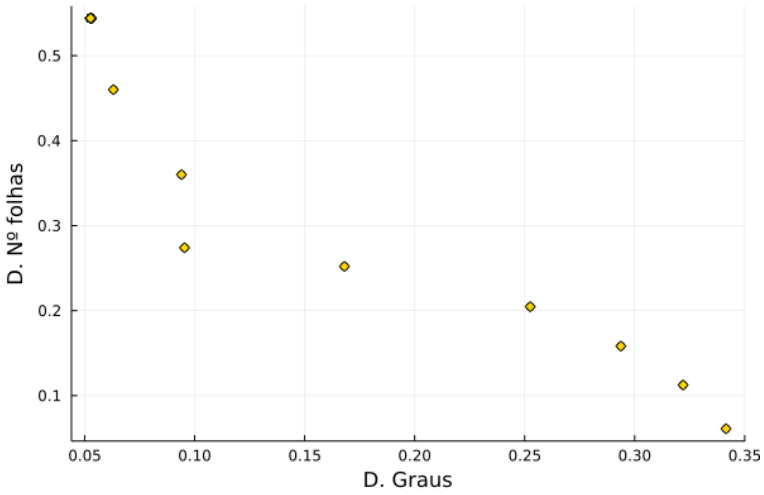


Figura 5.32: *Fronteira de Pareto da busca topológica com dois objetivos seguindo a parametrização da tabela 5.4.*

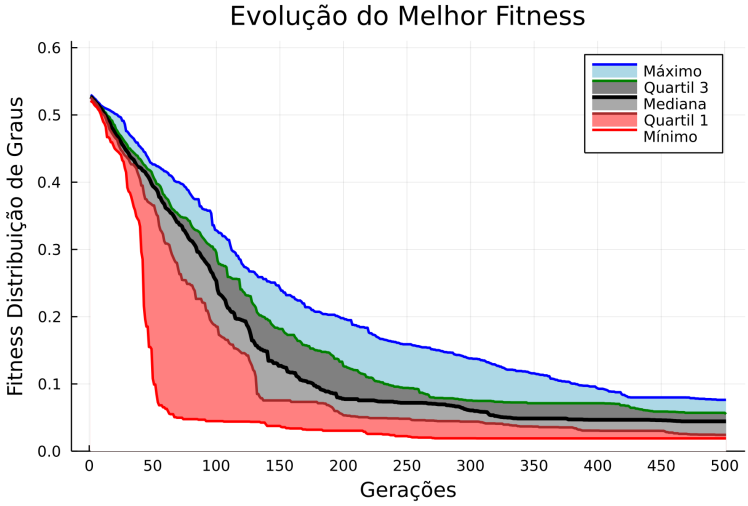
5.8 Comparação das configurações com 2 e 3 objetivos

Esta seção apresenta a comparação entre o processo de busca topológica com 2 e 3 objetivos. Para a busca com três objetivos analisou-se também a inicialização com o uso criação de árvores com os números Prüfer e com o algoritmo 4.8 de composição de árvores. Cada busca da comparação a seguir foi repetida 30 vezes observando a evolução dos objetivos de cada busca. Ao final foram calculados o mínimo, 1º quartil, mediana, 3º quartil e máximo de cada objetivo em cada geração da configuração da busca.

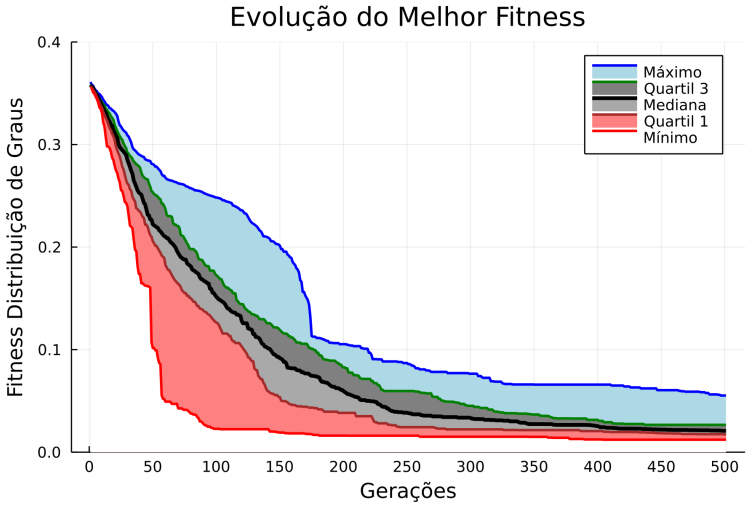
A figura 5.33 apresenta a evolução do *fitness* da distribuição de graus. Ao comparar as duas primeiras configurações, figuras 5.33(a) e 5.33(b), é possível perceber que embora o algoritmo 4.8 de composição de árvores tenha posicionado a população inicial em um espaço de busca mais promissor, ambas configurações convergiram quanto a esse critério com aproximadamente 250 gerações. Já a configuração com 2 objetivos na figura 5.33(c) apresentou uma convergência mais lenta.

Já a figura 5.34 apresenta a evolução do *fitness* da distribuição do número de folhas por alimentador. Nestas distribuições, também é possível perceber o mesmo padrão no qual as configurações com três objetivos, figuras 5.34(a) e 5.34(b), rapidamente convergem e a configuração com dois objetivos, figura 5.34(c), converge de modo mais lento. Além disso, também é possível perceber que a inicialização por composição posiciona os indivíduos em uma posição melhor no espaço de busca para estes critérios quando comparado com os números Prüfer.

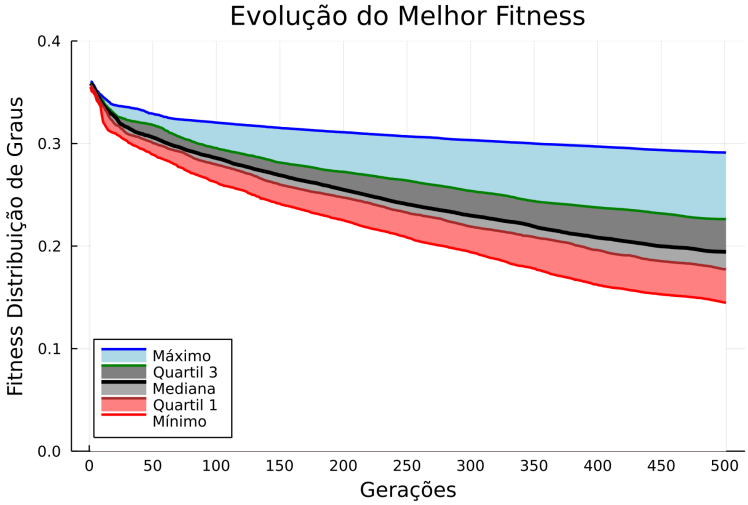
Para as configurações com três objetivos a distribuição do número de barras por alimentador manteve-se constante assim como o observado nos experimentos anteriores como na figura 5.14.



(a) *3 objetivos e criação de árvores na inicialização com números Prüfer.*



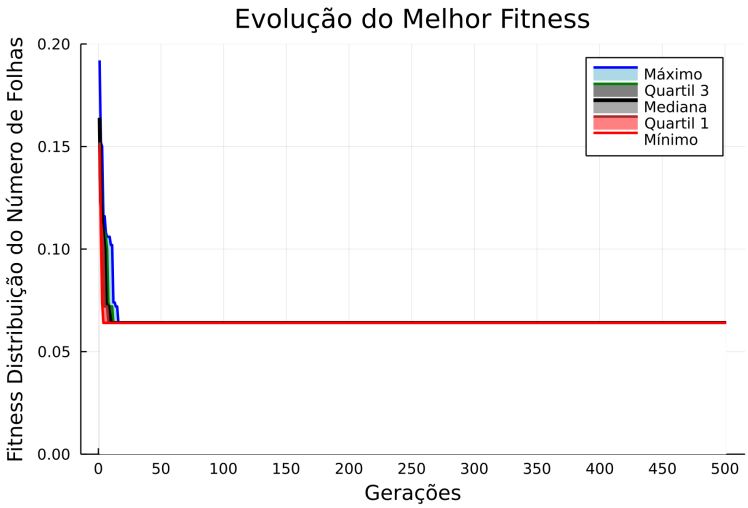
(b) *3 objetivos e criação de árvores na inicialização com o algoritmo 4.8 de composição de árvores.*



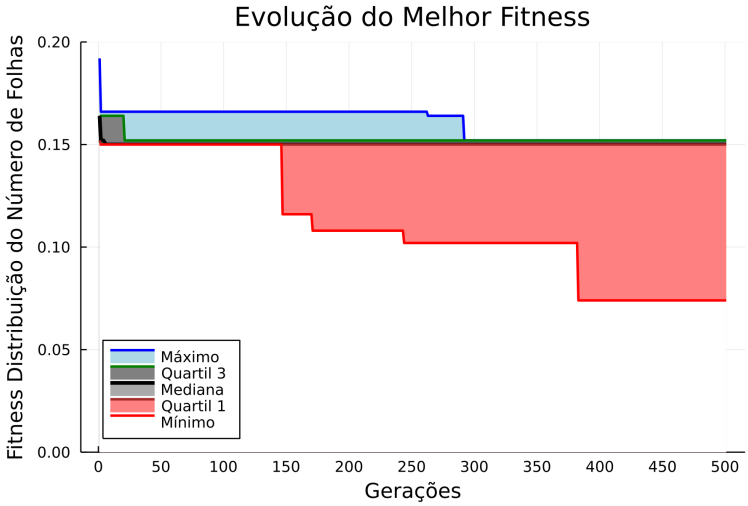
(c) *2 objetivos e criação de árvores na inicialização com o algoritmo 4.8 de composição de árvores.*

Figura 5.33: *Evolução do fitness da distribuição de graus com 30 repetições para cada configuração.*

(a) *3 objetivos e criação de árvores na inicialização com números Prüfer.*



(b) *3 objetivos e criação de árvores na inicialização com o algoritmo 4.8 de composição de árvores.*



(c) *2 objetivos e criação de árvores na inicialização com o algoritmo 4.8 de composição de árvores.*

Figura 5.34: *Evolução do fitness da distribuição do número de folhas com 30 repetições para cada configuração.*

5.9 Posicionamento de chaves

Após a determinação da topologia (seção 5.7), foi determinado o posicionamento dos consumidores conforme o procedimento descrito seção 4.2. Estes foram posicionados em barras consideradas folha seguindo a proporção das folhas com carga (figura 3.18). Em seguida, procedeu-se com a determinação e posicionamento das chaves abertas e fechadas da rede elétrica sintética, conforme descrito na seção 4.3.

A figura 5.35 apresenta a evolução do *fitness* do AG para o posicionamento das chaves. É importante observar que ao longo das gerações a quantidade de consumidores afetados reduziu e estabilizou por volta de 3.500 gerações. Nos experimentos, geração demandou em média 4,5 segundos mostrando que os algoritmos 4.14, 4.15 e 4.16 foram eficientes.

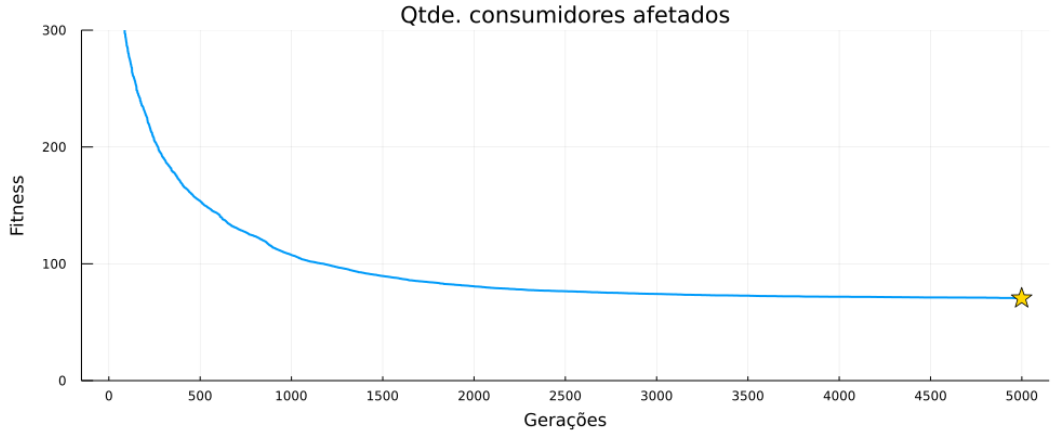


Figura 5.35: *Evolução do fitness do AG.*

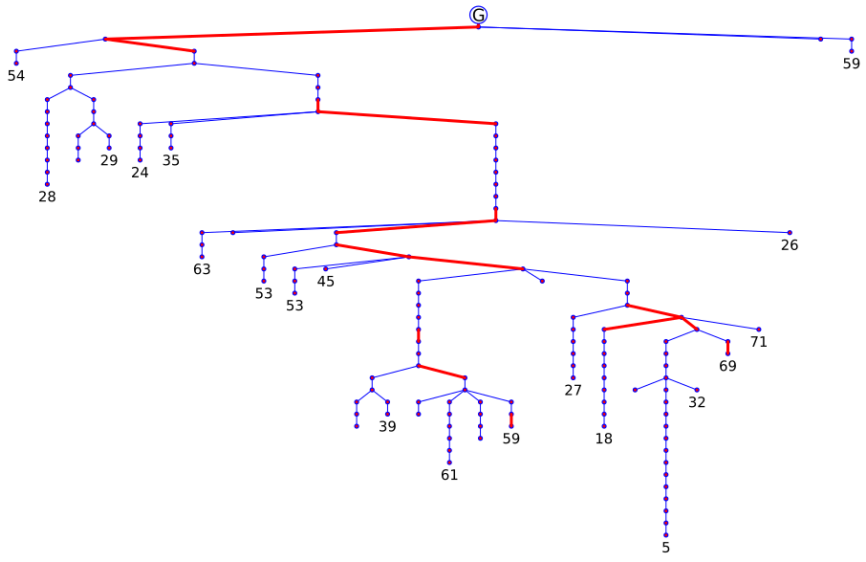


Figura 5.36: *Exemplo de alimentador com posicionamento de chaves encontrado. Os números nos vértices indicam o número de consumidores nele.*

A figura 5.36 apresenta as chaves fechadas de um dos alimentadores do melhor indivíduo da busca. Nela, os números nas barras indicam a quantidade de consumidores que foram alocados a cada uma, conforme o procedimento descrito na seção 4.2. A inspeção visual mostra que o AG alocou o posicionamento das chaves nos trechos imediatamente à montante de uma subárvore e nos trechos de ramificações das subárvores priorizando as maiores subárvores.

5.10 Comparação do posicionamento das chaves

Esta seção apresenta o resultado do experimento descrito na seção 4.4.7 que visa comparar o posicionamento de chaves normalmente abertas e fechadas, descrito na seção 4.3, com o posicionamento real das chaves do SDE.

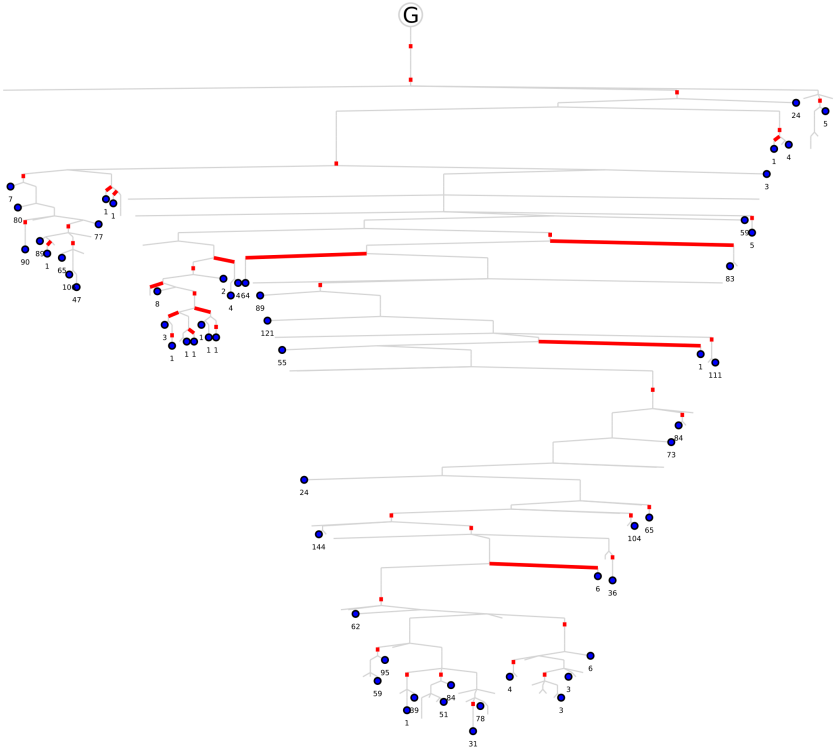
A comparação mostrou que o posicionamento foi igual para 96 chaves normalmente fechadas entre as 941 possíveis do experimento, um total de 10,2% de similaridade. No entanto, nenhuma chave normalmente aberta coincidiu com as existentes no SDE real. A figura 5.37 apresenta a comparação do posicionamento das chaves em um alimentador real e no mesmo alimentador com o posicionamento do algoritmo. Nela é possível notar que o AG tem preferência por posicionar chaves normalmente fechadas em regiões onde há mais consumidores visando diminui a quantidade de consumidores afetados por uma falha.

Embora em termos de similaridade o posicionamento das chaves não tenha sido o mesmo, a quantidade média de consumidores afetados por uma falha foi drasticamente melhorada. Enquanto na rede elétrica real a média e mediana de consumidores afetados são 300,6 e 161,5; no AG estes números são 77,5 e 60, ou seja, dada uma falha qualquer na rede, menos consumidores ficaram desenergizados. Deste modo, a busca foi capaz de encontrar um posicionamento ainda melhor que o encontrado na SDE real.

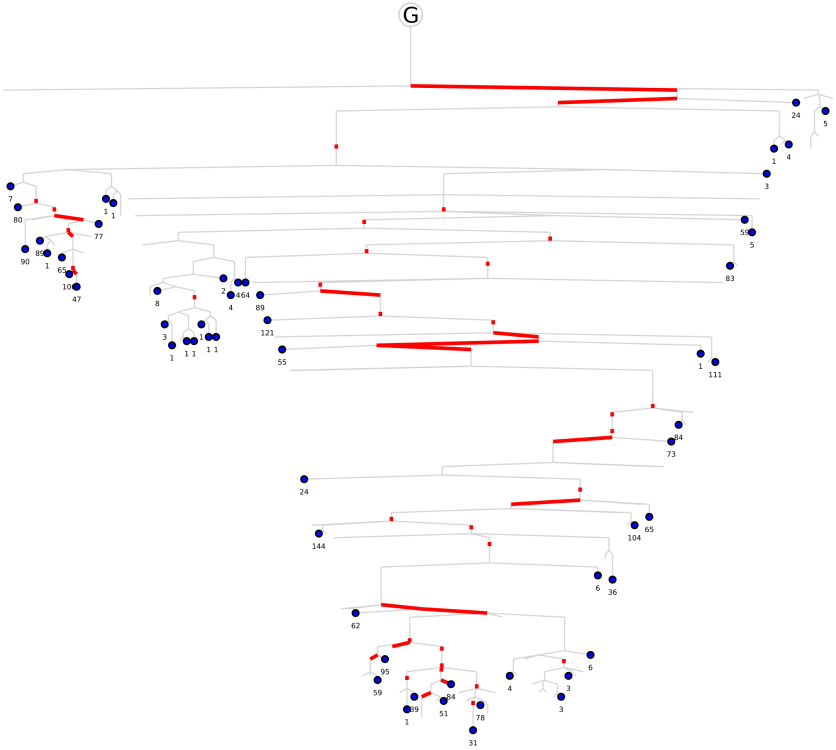
Neste momento cabe ressaltar que, na SDE real o alimentador é criado a partir de uma previsão de carga, mas sem conhecimento da quantidade real de carga ou consumidores que existirão no local. Já as chaves são posicionadas a partir do conhecimento das equipes de planejamento das companhias.

Além disso, há três pontos relevantes quanto ao posicionamento das chaves feito pelo gerador de SDE (gSDE). Em primeiro lugar, o posicionamento das chaves leva em consideração um relaxamento do problema, descrito na seção 4.3, no qual é possível redirecionar qualquer quantidade de carga de um alimentador para outro desde que haja uma chave normalmente aberta para manobra. Esta foi uma decisão tomada para que o posicionamento das chaves pudesse ser feito, visto os entraves da determinação da potência demandada, descritos na seção 3.3.2. Em segundo lugar, é esperado que um bom algoritmo reduza a quantidade de consumidores afetados, visto que, muitas vezes

no SDE o posicionamento das chaves é feito com uma estimativa de consumidores, já o gSDE proposto teve acesso ao número real de consumidores. Por fim, em terceiro lugar, o posicionamento das chaves poderia também levar em consideração a potência demandada pelos consumidores além da quantidade absoluta deles.



(a) *Posicionamento real das chaves.*



(b) *Posicionamento das chaves feito pelo algoritmo.*

Figura 5.37: *Exemplo de alimentador da comparação do posicionamento das chaves normalmente fechadas (arestas vermelhas).*

CAPÍTULO 6

Conclusões e trabalhos futuros

Neste trabalho, foi possível desenvolver um método para geração de topologias, posicionamento de consumidores e de chaves normalmente abertas e normalmente fechadas em um SDEs. Para isso foram criados e avaliados os novos operadores desenvolvidos para inicialização (seções 4.1.3 e 5.3), *crossover* (seções 4.1.4 e 4.3.3) e mutação (seções 4.1.5, 5.5 e 4.3.3). O uso do NSGA-II, como processo de busca, e da NPE, como representação, mostraram-se adequados ao problema, tanto pelo desempenho em relação ao tempo quanto pela qualidade das soluções geradas, conforme apresentado na seção 5.6.

Identificou-se que a metodologia utilizada pelo gerador de SDE (gSDE) para a geração de topologias de SDEs (seção 4.1) é adequada, pois diminui o erro de cada um dos objetivos ao longo do processo de busca (seção 5.6). As distribuições do indivíduo de menor erro geral também se mostraram adequadas do ponto de vista quantitativo e, qualitativamente. Os alimentadores gerados pelo gSDE mostraram características observáveis nos alimentadores reais. Neste processo, observou-se a eficiência do NPE como representação juntamente com seus operadores SPR e TBR.

Foram extraídas várias características e relacionamentos de um SDE conforme o estudo estatístico do capítulo 3. Através destas análises, foi possível definir um método de posicionamento de consumidores na rede elétrica (seção 4.2).

Também foi desenvolvido um AG capaz de alocar as chaves abertas e fechadas visando minimizar o número de consumidores afetados em um SDE (seção 4.3). Para isso foi desenvolvido um método especial de cálculo de *fitness* altamente eficiente que permitiu simular falhas em todas as barras do SDE sintético (seção 4.3.1). Foi possível observar que o posicionamento das chaves reduziu o número de consumidores afetados no SDE, conforme apresentado na seção 5.10.

Além disso, a metodologia do gSDE também pode ser aplicada em diversas áreas que envolvam grafos. Dado que ela não restringe seu uso em SDEs, pois sabendo-se a distribuição do grafo desejado é possível buscar topologias similares. Adicionalmente, também é possível utilizar o algoritmo de posicionamento de chaves para predição de *links* ou para redução de vulnerabilidade em redes de computadores.

As seções a seguir apresentam as contribuições realizadas por este trabalho (seção 6.1) e algumas propostas de trabalhos futuros (seção 6.2).

6.1 Contribuições realizadas

Entre as contribuições deste trabalho, destacam-se:

- desenvolvimento de uma metodologia completa abrangendo desde a geração da topologia, posicionamento dos consumidores, chaves normalmente abertas e normalmente fechadas (capítulo 4);
- utilização do NSGA-II no processo de geração de topologias (seção 4.1), permitindo a busca de diversas características topológicas;
- utilização de três distribuições (seção 4.1.2) como objetivo da busca, ao invés de utilizar somente uma média ou mediana da característica desejada;
- definição de um método de inicialização de topologias (seção 4.1.3) com um posicionamento ótimo em relação ao espaço de busca, permitindo que a busca foque em outras características;
- criação de novos operadores de mutação Probabilistic Subtree Pruning and Regrafting (PSPR) e Probabilistic Tree Bisection and Reconnection (PTBR) (seção 10) assim como o estudo de inserção de tendências neles (seção 5.5);
- desenvolvimento de quatro algoritmos de construção de árvores, incluindo limitações de grau e profundidade, utilizados na inicialização de topologias descritos na seção 4.1.3, assim como o estudo da tendência deles (seção 5.4);
- estudo e entendimento do espaço de busca do problema (seção 5.1);
- definição de uma metodologia de posicionamento de consumidores (seção 4.2) levando em consideração características reais dos SDEs (seção 3.3.2);
- desenvolvimento de um AG de posicionamento de chaves (seção 4.3) com uma função de *fitness* extremamente eficiente utilizando programação dinâmica (seção 4.3.1);

6.2 Trabalhos futuros

Como trabalhos futuros, observa-se principalmente a melhoria da predição de potência descrita na seção 3.3.2. Nesta faz-se necessário o estudo do relacionamento de cada tipo de alimentador com as demandas dos consumidores de maneira abrangente e como estas características podem ser adicionadas ao gerador de SDE. Este estudo também se faz necessário para relacionar o número de consumidores para os tipos de alimentadores rurais e mistos que não foram profundamente estudados nesta tese.

Além disso, entende-se que os trabalhos posteriores devem adicionar todas as demais características elétricas para que seja possível realizar cálculos como o fluxo de carga nos alimentadores sintéticos. Em uma visão ainda mais ampla, é possível visualizar o uso do gerador de SDEs para construção de cidades virtuais na qual possam ser realizadas comparações entre diferentes tipos de algoritmos para resolução dos problemas da área.

Referências Bibliográficas

[1] AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA. **12.007, De 29 De Julho De 2009, N**. *Resolução Normativa nº 414*, p. 205, 2010.

[2] AGÊNCIA NACIONAL DE ENERGIA ELÉTRICA. **Distribuição de energia elétrica**. https://www2.aneel.gov.br/area.cfm?idArea=77&idPerfil=2, mar 2021.

[3] ALBERT, R.; BARABÁSI, A.-L. **Statistical mechanics of complex networks**. *Reviews of Modern Physics*, 74(1):47–97, jan 2002.

[4] BACH, B.; SPRITZER, A.; LUTTON, E.; FEKETE, J. D. **Interactive random graph generation with evolutionary algorithms**. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7704 LNCS:541–552, 2013.

[5] CAMILLO, M. H. M. **Avaliação de uma metodologia para restabelecimento de energia baseada em algoritmos evolutivos multi-objetivos no sistema de distribuição de energia da COPEL na cidade de Londrina**. PhD thesis, Universidade de São Paulo, São Carlos, sep 2013.

[6] CAMILLO, M. H.; FANUCCHI, R. Z.; ROMERO, M. E.; DE LIMA, T. W.; DA SILVA SOARES, A.; BOTAZZO DELBEM, A. C.; MARQUES, L. T.; MACIEL, C. D.; AUGUSTO LONDON, J. B. **Combining exhaustive search and multi-objective evolutionary algorithm for service restoration in large-scale distribution systems**. *Electric Power Systems Research*, 134:1–8, 2016.

[7] CAMILLO, M. H.; ROMERO, M. E.; FANUCCHI, R. Z.; DE LIMA, T. W.; MARQUES, L. T.; DELBEM, A. B.; LONDON, J. B. **Validation of a methodology for service restoration on a real Brazilian distribution system**. *2014 IEEE PES Transmission and Distribution Conference and Exposition, PES T and D-LA 2014 - Conference Proceedings*, 2014-Octob:0–5, 2014.

[8] CAMILLO, M. H.; ROMERO, M. E.; FANUCCHI, R. Z.; DE LIMA, T. W.; DA SILVA SOARES, A.; LONDON, J. B. A.; DELBEM, A. C. B.; MARQUES, L. T.; MACIEL,

C. D. **A multi-objective evolutionary algorithm with efficient data structure and heuristic initialization for fault service restoration**. *Procedia Computer Science*, 80:2367–2371, 2016.

[9] CAYLEY, A. **The Collected Mathematical Papers of Arthur Cayley**. Forgotten Books, 2012.

[10] CHENG, C. S.; SHIRMOHAINMADI, D. **A three-phase power flow method for real-time distribution system analysis**. *IEEE Transactions on Power Systems*, 10(2):671–679, 1995.

[11] DAS, H.; MISHRA, S. K.; ROY, D. S. **The Topological Structure of the Odisha Power Grid : A Complex Network Analysis**. *International Journal of Mechanical Engineering and Computer Applications (IJMCA)*, 1(1):12–16, 2013.

[12] DE ALMEIDA RIBEIRO, L.; DA SILVA SOARES, A.; DE LIMA, T. W.; JORGE, C. A. C.; DA COSTA, R. M.; SALVINI, R. L.; COELHO, C. J.; FEDERSON, F. M.; GABRIEL, P. H. R. **Multi-objective genetic algorithm for variable selection in multivariate classification problems: A case study in verification of biodiesel adulteration**. *Procedia Computer Science*, 51(1):346–355, 2015.

[13] DE LIMA, T. W.; DELBEM, A. C. B.; SOARES, A. D. S.; FEDERSON, F. M.; JUNIOR, J. B. A. L.; BAALEN, J. V. **Node-depth phylogenetic-based encoding, a spanning-tree representation for evolutionary algorithms. part I: Proposal and properties analysis**. *Swarm and Evolutionary Computation*, 31:1–10, 2016.

[14] DEB, K.; JAIN, H. **An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints**. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, 2014.

[15] DEB, K.; PRATAP, A.; AGARWAL, S.; MEYARIVAN, T. **A fast and elitist multiobjective genetic algorithm: NSGA-II**. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[16] DEKA, D.; VISHWANATH, S.; BALDICK, R. **Topological vulnerability of power grids to disasters: Bounds, adversarial attacks and reinforcement**. *PLOS ONE*, 13(10):e0204815, oct 2018.

[17] DELBEM, A. C. B.; DE LIMA, T. W.; TELLES, G. P. **Efficient Forest Data Structure for Evolutionary Algorithms Applied to Network Design**. *IEEE Transactions on Evolutionary Computation*, 16(6):829–846, 2012.

[18] DELBEM, A. C. B.; DE CARVALHO, A.; POLICASTRO, C. A.; PINTO, A. K. O.; HONDA, K.; GARCIA, A. C. **Node-Depth Encoding for Evolutionary Algorithms Applied to Network Design**. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3102:678–687, 2004.

[19] DOS SANTOS, A. C. **Algoritmo evolutivo computacionalmente eficiente para reconfiguração de sistemas de distribuição**. PhD thesis, Universidade de São Paulo, 2009.

[20] DOS SANTOS, A. C.; DELBEM, A. C. B.; BRETAS, N. G. **Algoritmo Evolutivo Eficiente para Restabelecimento de Energia em Sistemas de Distribuição de Energia Elétrica**. *XVIII Seminário Nacional de Distribuição de Energia Elétrica*, p. 11, 2008.

[21] ELYAS, S. H.; WANG, Z. **A multi-objective optimization algorithm for bus type assignments in random topology power grid model**. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2016-March(January):2446–2455, 2016.

[22] ELYAS, S. H.; WANG, Z. **Improved Synthetic Power Grid Modeling with Correlated Bus Type Assignments**. *IEEE Transactions on Power Systems*, 32(5):3391–3402, 2017.

[23] ELYAS, S. H.; WANG, Z.; THOMAS, R. J. **On the Statistical Settings of Generation and Load in a Synthetic Grid Modeling**. In: *The 10th Bulk Power Systems Dynamics and Control Symposium (IREP'2017)*, 2017.

[24] ERDÖS, P.; RENYI, A. **On random graphs I.** *Publicationes Mathematicae*, 6:290–297, 1959.

[25] FELSENSTEIN, J. **Inferring Phylogenies**. Sinauer Associates, Inc., aug 2004.

[26] FINCH, S. R. **Mathematical Constants**. Cambridge University Press, 2003.

[27] GASPAR CUNHA, A.; TAKAHASHI, R.; HENGGELER ANTUNES, C. **Manual de computação evolutiva e metaheurística**. Editora UFMG, Coimbra, 1 edition, 2012.

[28] HARTMANN, T.; FOUQUET, F.; KLEIN, J.; LE TRAON, Y.; PELOV, A.; TOUTAIN, L.; ROPITAULT, T. **Generating realistic Smart Grid communication topologies based on real-data**. In: *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, p. 428–433. IEEE, nov 2014.

[29] HOLLANDER, M.; A. WOLFE, D.; CHICKEN, E. **Nonparametric Statistical Methods**. Wiley Series in Probability and Statistics. Wiley, jul 2015.

[30] KRUSKAL, J. B. **On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem**. *Proceedings of the American Mathematical Society*, 7(1):48, 1956.

[31] MA, S.; YU, Y.; ZHAO, L. **Dual-stage constructed random graph algorithm to generate random graphs featuring the same topological characteristics with power grids**. *Journal of Modern Power Systems and Clean Energy*, 5(5):683–695, 2017.

[32] MAHANTY, R.; GUPTA, P. **Voltage stability analysis in unbalanced power systems by optimal power flow**. *IEE Proceedings-Generation, Transmission and ...*, 151(3):201–212, 2004.

[33] MANSOUR, M. R.; DOS SANTOS, A. C.; LONDON, J. B.; DELBEM, A. C.; BRETAS, N. G. **Energy restoration in distribution systems using multi-objective evolutionary algorithm and an efficient data structure**. *2009 IEEE Bucharest PowerTech: Innovative Ideas Toward the Electrical Grid of the Future*, p. 1–7, 2009.

[34] MANSOUR, M. R.; DOS SANTOS, A. C.; LONDON, J. B. A.; DELBEM, A. C. B.; BRETAS, N. G. **Node-depth encoding and evolutionary algorithms applied to service restoration in distribution systems**. *IEEE PES General Meeting, PES 2010*, p. 1–8, 2010.

[35] MARQUES, L. T.; DELBEM, A. C. B.; LONDON, J. B. A. **Service Restoration With Prioritization of Customers and Switches and Determination of Switching Sequence**. *IEEE Transactions on Smart Grid*, 9(3):2359–2370, 2018.

[36] MARQUEZ, R. A. C. **Restabelecimento de energia em sistemas de distribuição de energia elétrica com priorização de chaves automáticas**. PhD thesis, Universidade de São Paulo, São Carlos, mar 2014.

[37] MASSIGNAN, J. A. D.; LONDON, J. B. A.; BESSANI, M.; MACIEL, C. D.; DELBEM, A. C. B.; CAMILLO, M. H. M.; DE LIMA SOARES, T. W. **In-field validation of a real-time monitoring tool for distribution feeders**. *IEEE Transactions on Power Delivery*, 33(4):1798–1808, 2018.

[38] NGUYEN, V. N.; JENSSEN, R.; ROVERSO, D. **Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning**. *International Journal of Electrical Power and Energy Systems*, 99(September 2017):107–120, 2018.

[39] NICOLETTI, M. C.; HRUSCHKA JR., E. R. **Fundamentos da Teoria dos Grafos para Computação**. LTC, Rio de Janeiro, 3 edition, 2018.

[40] PALMER, C. C.; KERSHENBAUM, A. **Representing trees in genetic algorithms**. *IEEE Conference on Evolutionary Computation - Proceedings*, 1:379–384, 1994.

[41] PALMER, C. C.; KERSHENBAUM, A. **An approach to a problem in network design using genetic algorithms**. *Networks*, 26(3):151–163, 1995.

[42] PRADO, R. S.; PEDROSA SILVA, R. C.; NETO, O. M.; GUIMARÃES, F. G.; SANCHES, D. S.; LONDON, J. B. A.; DELBEM, A. C. **Differential evolution using ancestor tree for service restoration in power distribution systems**. *Applied Soft Computing Journal*, 23:498–508, 2014.

[43] RAIDL, G. R.; JULSTROM, B. A. **Edge sets: An effective evolutionary coding of spanning trees**. *IEEE Transactions on Evolutionary Computation*, 7(3):225–239, 2003.

[44] REYNOLDS, D. **Gaussian Mixture Models**. In: Li, S. Z.; Jain, A., editors, *Encyclopedia of Biometrics*, p. 659–663. Springer US, Boston, MA, 2009.

[45] ROTHLAUF, F. **Representations for Genetic and Evolutionary Algorithms**. Springer, Berlin, Heidelberg, 2 edition, 2006.

[46] ROTHLAUF, F. **Design of Modern Heuristics**. Springer, Berlin, 2011.

[47] ROTHLAUF, F.; GOLDBERG, D. E.; HEINZL, A. **Network random keys - A tree representation scheme for genetic and evolutionary algorithms**. *Evolutionary Computation*, 10(1):75–97, 2002.

[48] SANCHES, D. S.; CASTOLDI, M. F.; LONDON, J. B. A.; DELBEM, A. C. B. **Analysis of Solution Quality of Multiobjective Evolutionary Algorithms for Service Restoration in Distribution Systems**. *Journal of Control, Automation and Electrical Systems*, 28(6):796–805, 2017.

[49] SANCHES, D. S.; LONDON, J. B. A.; DELBEM, A. C. B.; PRADO, R. S.; GUIMARÃES, F. G.; NETO, O. M.; DE LIMA, T. W. **Multiobjective evolutionary algorithm with a discrete differential mutation operator developed for service restoration in distribution systems**. *International Journal of Electrical Power and Energy Systems*, 62:700–711, 2014.

[50] SANCHES, D. S.; LONDON JUNIOR, J. B. A.; DELBEM, A. C. B. **Multi-Objective Evolutionary Algorithm for single and multiple fault service restoration in large-scale distribution systems**. *Electric Power Systems Research*, 110:144–153, 2014.

[51] SCHNEIDER, K. P.; MATHER, B. A.; PAL, B. C.; TEN, C. W.; SHIREK, G. J.; ZHU, H.; FULLER, J. C.; PEREIRA, J. L.; OCHOA, L. F.; DE ARAUJO, L. R.; DUGAN, R. C.; MATTHIAS, S.; PAUDYAL, S.; MCDERMOTT, T. E.; KERSTING, W. **Analytic Considerations and Design Basis for the IEEE Distribution Test Feeders**. *IEEE Transactions on Power Systems*, 33(3):3181–3188, 2018.

[52] SCHULZ, J. E.; ROCHA, C. R. M. **Orientação a Objetos Aplicada na Solução do Fluxo de Potência para Sistemas de Distribuição de Energie Elétrica**. *Anais do XX Congresso Brasileiro de Automática*, p. 1911–1918, 2014.

[53] SOLÉ, R. V.; ROSAS-CASALS, M.; COROMINAS-MURTRA, B.; VALVERDE, S. **Robustness of the European power grids under intentional attack**. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 77(2):1–7, 2008.

[54] SOLTAN, S.; ZUSSMAN, G. **Generation of synthetic spatially embedded power grid networks**. *IEEE Power and Energy Society General Meeting*, 2016-Novem, 2016.

[55] TRAVERS, J.; MILGRAM, S. **An experimental study of the small world problem**. *The Structure and Dynamics of Networks*, 9781400841356(4):130–148, 2011.

[56] TUKEY, J. W. **Comparing Individual Means in the Analysis of Variance**. *Biometrics*, 5(2):99, jun 1949.

[57] WANG, X.; WANG, L.; WU, Y. **An Optimal Algorithm for Prufer Codes**. *Journal of Software Engineering and Applications*, 02(02):111–115, 2009.

[58] WANG, Z.; ELYAS, S. H. **On the Scaling Property of Power Grids**. *Proceedings of the 50th Hawaii International Conference on System Sciences (2017)*, p. 3148–3155, 2017.

[59] WANG, Z.; ELYAS, S. H.; THOMAS, R. J. **A novel measure to characterize bus type assignments of realistic power grids**. *2015 IEEE Eindhoven PowerTech, PowerTech 2015*, 2015.

[60] WANG, Z.; ELYAS, S. H.; THOMAS, R. J. **Generating synthetic electric power system data with accurate electric topology and parameters**. *Proceedings - 2016 51st International Universities Power Engineering Conference, UPEC 2016*, 2017-Janua:1–6, 2016.

[61] WANG, Z.; SCAGLIONE, A.; THOMAS, R. J. **On Modeling Random Topology Power Grids for Testing Decentralized Network Control Strategies**. *IFAC Proceedings Volumes*, 42(20):114–119, sep 2009.

[62] WANG, Z.; SCAGLIONE, A.; THOMAS, R. J. **Generating statistically correct random topologies for testing smart grid communication and control networks**. *IEEE Transactions on Smart Grid*, 1(1):28–39, 2010.

[63] WANG, Z.; THOMAS, R. J. **On bus type assignments in random topology power grid models**. *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2015-March:2671–2679, 2015.

[64] WANG, Z.; THOMAS, R. J.; SCAGLIONE, A. **Generating random topology power grids**. *Proceedings of the Annual Hawaii International Conference on System Sciences*, p. 1–9, 2008.

[65] WATTS, D. J.; STROGATZ, S. H. **Collective dynamics of 'small-world' networks**. *Nature*, 393(6684):440–442, jun 1998.

[66] ZAN, J. C. **Metodologia para restabelecimento de energia em sistemas de distribuição considerando reguladores de tensão, bancos de capacitores e as características operacionais de vários tipos de chaves seccionadoras**. PhD thesis, Universidade de São Paulo, São Carlos, feb 2015.

[67] ZHANG, Q.; LI, H. **MOEA/D: A multiobjective evolutionary algorithm based on decomposition**. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[68] ZYKINA, A. V. **A lexicographic optimization algorithm**. *Automation and Remote Control*, 65(3):363–368, 2004.