UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Robson Pires Borges

Uma abordagem para o Problema da Árvore Geradora Mínima com Restrição de Diâmetro via Metaheurística BRKGA

Mossoró-RN 2024

Robson Pires Borges

Uma abordagem para o Problema da Árvore Geradora Mínima com Restrição de Diâmetro via Metaheurística BRKGA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação - associação ampla entre a Universidade do Estado do Rio Grande do Norte e a Universidade Federal Rural do Semi-Árido, para a obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof$^{o.}$ Dr. Dario José Aloise

Mossoró-RN 2024

UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

ROBSON PIRES BORGES

UMA ABORDAGEM PARA O PROBLEMA DA ÁRVORE GERADORA MÍNIMA COM RESTRIÇÃO DE DIÂMETRO VIA METAHEURÍSTICA BRKGA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA EM: 24 / julho / 2024

Documento assinado digitalmente
gov.br DARIO JOSE ALOISE
Data: 27/07/2024 01:02:34-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Dario José Aloise
Orientador e Presidente

Documento assinado digitalmente
gov.br FABIO FRANCISCO DA COSTA FONTES
Data: 30/07/2024 22:46:26-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Fábio Francisco da Costa Fontes
Membro Interno - Universidade Federal Rural do Semi-Árido - UFERSA

Documento assinado digitalmente
gov.br BRENO BARROS TELLES DO CARMO
Data: 30/07/2024 19:15:51-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Breno Barros Telles do Carmo
Membro Externo - Universidade Federal do Ceará - UFC

Profa. Dra. Andréa Cynthia dos Santos
Membro externo - Université Le Havre Normandie

Agradecimentos

Agradeço a Deus pela vida, pela saúde, força, e ainda por todas as pessoas que se fazem presentes em nossas vidas. Agradeço aos meus pais, por sempre acreditar, e nos mostrar que o caminho da educação vale a pena. Agradeço a minha esposa Karlene Medeiros pelo apoio, compreensão e parceria de sempre, ajudando e segurando as pontas sozinha em casa com nossos filhos durante o período que estive fora de casa e também pelos momentos de ausência. Foram muitas vídeo chamadas, quase que diárias... alguns finais de semana e férias, agradeço por tudo. Agradeço ao meu irmão Ronaldo Pires Borges, pela cumplicidade e parceria nessa jornada. Saímos de nossas casas e viemos juntos para Mossoró, para a realização dessa meta importante em nossas carreiras como docentes. Houveram alguns "perrengues"e muitas semanas longas durante esse percurso. Agradeço ao meu orientador, prof. Dr. Dario José Aloise pelos direcionamentos e conversas sobre o trabalho realizado, pela parceria fora do país com a co-orientadora Drª Andréa Cynthia Santos e suas enormes contribuições. Gratidão ao colega Heitor Nunes Costa, da cidade de Baraúna - RN, o qual me ajudou muito na implementação das buscas locais na etapa final do trabalho. Agradeço a Banca Examinadora pelo tempo, apreço e considerações. Quero agradecer também aos servidores e funcionários da UFERSA que nos receberam muito bem e nos deram todas as condições para o desenvolvimento das pesquisas e estudos no bloco LCC e outros locais.

Agradeço aos meus amigos e a minha família, por superar os dias distantes, pelo apoio e compreensão de todos, que de forma direta ou indireta, contribuíram para a realização desse trabalho e aperfeiçoamento na minha carreira.

Obrigado a todos por tudo!

Resumo

Este trabalho explora abordagens para resolver o problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD), um importante problema de otimização combinatória em teoria dos grafos, que pode ser aplicado a diversos problemas como redes de transporte, abastecimento e energias, além de redes de computadores e internet das coisas (IoT). A proposta do trabalho envolve o desenvolvimento de heurísticas construtivas que, integradas à metaheurística BRKGA (*Biased Random-Key Genetic Algorithm*), visam gerar soluções de alta qualidade de forma eficiente. Para atingir os objetivos, foram desenvolvidas três heurísticas construtivas. A primeira inicia a construção da solução através de um *backbone* central, composto por $D - 1$ vértices. A segunda heurística organiza os vértices em níveis, aproximadamente $D/2$, a terceira a partir de caminhos mínimos que se conectam ao centro da árvore. Em todas as heurísticas a árvore geradora é criada sem a necessidade de verificação de diâmetro durante sua construção. As heurísticas foram integradas ao *framework* BRKGA, que começa com uma população inicial gerada aleatoriamente, onde cada indivíduo representa solução viável para o problema. Além das heurísticas construtivas, o trabalho inclui a aplicação de uma heurística de busca local VND (*variable neighborhood descent*) para refinar as soluções obtidas. Esta busca local é baseada em propriedades e teoremas de grafos, incluindo movimentos como trocas de nós folhas, caracterizando a vizinhança N1 e melhorias entre nós pai e filho no tipo N2. Os algoritmos foram testados em instâncias conhecidas da literatura, com tamanhos variando de 50 a 500 vértices. Os resultados mostraram que as heurísticas propostas, combinadas com o BRKGA, foram capazes de produzir soluções próximas ao ótimo global de maneira consistente.

Palavras-chaves: Árvore Geradora Mínima. Restrição de diâmetro. Heurísticas avança- das. Otimização Combinatória. NP-Difícil.

Abstract

This work explores approaches to solve the Diameter-Constrained Minimum Spanning Tree (DCMST) problem, an important combinatorial optimization problem in graph theory, which can be applied to various problems such as transportation, supply, and energy networks, as well as computer networks and the Internet of Things (IoT). The proposal involves the development of constructive heuristics, integrated with the BRKGA (Biased Random-Key Genetic Algorithm) metaheuristic, aiming to generate high-quality solutions efficiently. To achieve the objectives, three constructive heuristics were developed. The first begins the solution construction through a central backbone, composed of $D - 1$ vertices. The second heuristic organizes the vertices into levels, approximately $D/2$, and the third based on shortest paths that connect to the tree center. In all heuristics, the spanning tree is created without the need for diameter verification during its construction. The heuristics were integrated into the BRKGA framework, which starts with an initial population generated randomly, where each individual represents a feasible solution to the problem. In addition to the constructive heuristics, the work includes the application of a local search heuristic, VND (Variable Neighborhood Descent), to refine the obtained solutions. This local search is based on graph properties and theorems, including moves such as leaf node swaps, characterizing the N1 neighborhood, and improvements between parent and child nodes in the N2 type. The algorithms were tested on well-known instances from the literature, with sizes ranging from 50 to 500 vertices. The results showed that the proposed heuristics, combined with BRKGA, were able to consistently produce solutions close to the global optimum.

Keywords: Minimum Spanning Tree. Diameter Restriction. Advanced Heuristics. Combinatorial Optimization.

Sumário

Lista de ilustrações **8**

Lista de tabelas **9**

1 INTRODUÇÃO **11**
1.1 Motivação **13**
1.2 Objetivos **14**
1.3 Organização do Trabalho **14**

2 REFERENCIAL TEÓRICO **15**
2.1 Árvore Geradora Mínima - AGM **15**
2.1.1 Algoritmo de Kruskal 15
2.1.2 Algoritmo de Prim 16
2.2 Árvore Geradora Mínima com Restrição de Diâmetro - AGMRD **17**
2.3 Abordagens para resolver o AGMRD **18**
2.3.1 Métodos exatos 19
2.3.2 Métodos Heurísticos 19
2.3.3 Métodos Híbridos 20

3 TRABALHOS RELACIONADOS **22**

4 ABORDAGENS HEURÍSTICAS **25**
4.1 Algoritmo OTT **25**
4.2 Metaheurísticas **27**
4.3 Algoritmo Genético **28**
4.4 Algoritmo Genético de Chaves Aleatórias Viciadas **29**
4.4.1 Parâmetros do BRKGA 33

5 ALGORITMOS PROPOSTOS **35**
5.1 Heurística baseada em backbone **35**
5.2 Heurística baseada em organização por Níveis ou camadas **38**
5.3 Heurística construtiva de caminhos mínimos - HCCM **41**
5.4 BRKGA - Decodificador **49**
5.4.1 Função Aptidão 50
5.4.2 Função de cruzamento 50
5.4.3 Função de mutação 51
5.4.4 BRKGA+ 52
5.5 Buscas Locais **53**
5.5.1 Vizinhança N1 54
5.5.2 Vizinhança N2 56
5.5.3 Vizinhança N3 58
5.5.4 Vizinhança N4 59

6 EXPERIMENTOS COMPUTACIONAIS **62**
6.1 Resultados obtidos **62**

6.1.1 Análise da Convergência do BRKGA 63
6.1.2 VND 64

7 CONCLUSÕES E TRABALHOS FUTUROS 67

REFERÊNCIAS 70

APÊNDICES 73

Lista de ilustrações

Figura 1 – Grafo ligando cidades 11
Figura 2 – AGM 12
Figura 3 – AGMRD 17
Figura 4 – AGMRD 18
Figura 5 – AGMRD 18
Figura 6 – Vetor de chaves aleatórias 30
Figura 7 – Recombinação PUC, tendo $\rho_e = 0,70$ 31
Figura 8 – Agrupamento de uma população de p indivíduos em certa geração k 32
Figura 9 – Montagem de uma geração 32
Figura 10 – Fluxograma do Algoritmo BRKGA 33
Figura 11 – Heurística construtiva baseada em *backbone* 36
Figura 12 – Heurística construtiva baseada em níveis. 39
Figura 13 – Grafo completo de 10 vértices. 43
Figura 14 – Construção da Árvore Geradora com Restrição de Diâmetro com HCCM. 44
Figura 15 – Construção da Árvore Geradora com Restrição de Diâmetro com HCCM. 45
Figura 16 – Construção da Árvore Geradora com Restrição de Diâmetro com HCCM. 46
Figura 17 – Árvore Geradora com Diâmetro Limitado a D = 7 construida pelo algoritmo HCCM. 47
Figura 18 – Exemplo de decodificação de chaves aleatórias para uma solução viável 50
Figura 19 – Cruzamento tradicional 51
Figura 20 – Cruzamento Uniforme Parametrizado - PUC 52
Figura 21 – Vizinhança N1 56
Figura 22 – Vizinhança N2 58
Figura 23 – Vizinhança N3 59
Figura 24 – Vizinhança N4 61
Figura 25 – Convergência do BRKGA - instância 50 vértices 63
Figura 26 – Convergência do BRKGA - instância 250 vértices 64
Figura 27 – Gráfico comparativo da evolução do BRKGA e suas variantes 65
Figura 28 – Gráfico com resultados alcançados pelos algoritmos 65

Lista de tabelas

Tabela 1 – Comparativo das metodologias para resolução do problema AGMRD . 21 Tabela 2 – Parâmetros BRKGA com valores recomendados 34
Tabela 3 – Matriz de pesos de um grafo completo de 10 vértices 42
Tabela 4 – Parâmetros para BRKGA 62
Tabela 5 – Resultados alcançados 66

Lista de abreviaturas e siglas

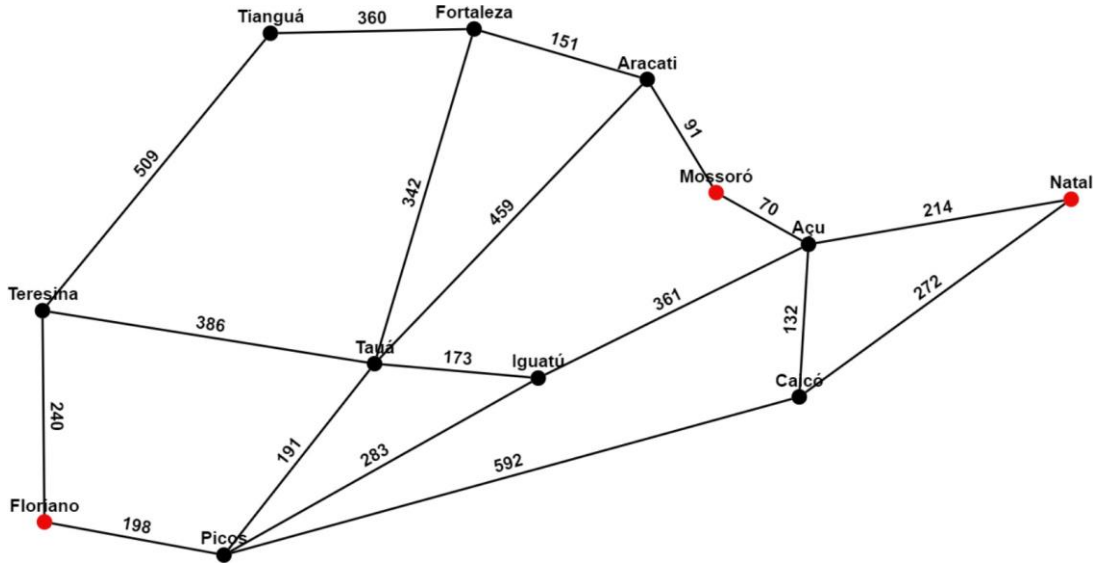
AG	Algorítimo Genético
AGM	Árvore Geradora Mínima
AGMRD	Árvore Geradora Mínima com Restrição de Diâmetro
AGs	Algorítimos Genéticos
BRKGA	Biased Random-Key Genetic Algorithm
GA	Genetic Algorithm
HCCM	Heurística Construtiva de Caminhos Mínimos
IoT	Internet of Things
MSTDC	Minimum spanning Tree with Diameter Constraint
MSTP	Minimum Spanning Tree Problem
OTT	One Time Tree
OTTC	One Time Tree Construction
RGH	Randomized Greed Heuristic
RKGA	Random-Key Genetic Algorithm
TG	Teoria dos Grafos
VND	Variable Neighborhood Descent

1 Introdução

A Teoria dos Grafos é um campo da Matemática e da Ciência da Computação que tem desempenhado um papel fundamental em uma ampla gama de aplicações do mundo real(WEST, 2001). Exemplos de aplicações incluem redes de computadores, internet das coisas, redes neurais dentro do campo da inteligência artificial, gerenciamento de processos em sistemas operacionais, sistemas de trajetórias, mapas e conexões variados.

Explorar a Teoria dos Grafos possibilita uma compreensão mais aprofundada de alguns algoritmos e estrutura de problemas, permitindo sua aplicação em diversos desafios presentes em campos como matemática, combinatória, informática, computação, física, biologia, engenharia, pesquisa operacional e em muitos setores industriais. Quando corretamente modelados com o auxílio de grafos, os problemas nessas áreas frequentemente revelam soluções excelentes (NICOLETTI, 2017).

Figura 1 – Mapa parcial de uma região do nordeste do Brasil



Fonte: criado pelo autor

Formalmente, um grafo G é definido como um par $G = (V, E)$, onde V é um conjunto de vértices e E é um conjunto de arestas, sendo cada aresta um par de vértices (WEST, 2001). Na Figura 1, um grafo ilustra um mapa no qual cada vértice simboliza uma cidade e cada aresta representa uma estrada entre as cidades, incluindo as respectivas distâncias ou pesos. Grafos também podem ser aplicados para representar relações em diversas áreas, como na árvore genealógica, onde os vértices correspondem a pessoas e as arestas indicam suas relações familiares.

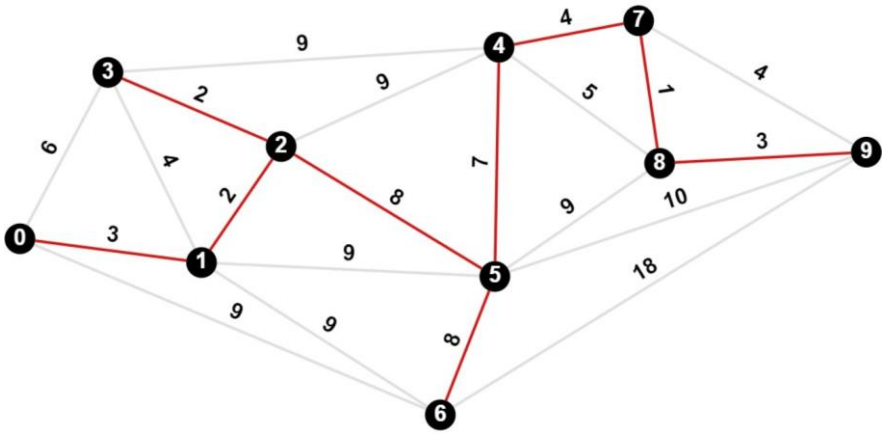
Problemas sobre grafos estão sempre presentes em ciência da computação, e

algoritmos para trabalhar com eles são fundamentais para a área. Centenas de problemas computacionais interessantes são expressos com a utilização de grafos (CORMEN, 2012).

Dentro da Teoria dos Grafos, uma Árvore é um conceito fundamental e importante que descreve uma estrutura de grafo especial. Uma árvore é um tipo específico de grafo acíclico, ou seja, não há caminhos fechados onde possa começar em um vértice e retornar ao mesmo vértice sem repetir arestas. Uma árvore é um grafo conexo (todos os vértices estão interligados por pelo menos um caminho) e acíclico (não possui ciclos) (NETTO, 2017). Já a árvore geradora de um grafo é uma árvore que abrange todos os vértices do grafo original, usando um subconjunto das arestas do grafo. Em outras palavras, uma árvore geradora é uma árvore que é um subgrafo do grafo original e ainda mantém a conectividade entre todos os vértices do grafo.

Ao aprofundar o estudo sobre Árvores Geradoras, destaca-se o problema da Árvore Geradora Mínima (AGM) que é de extrema importância em Teoria dos Grafos e tem uma ampla variedade de aplicações práticas. Uma AGM em um grafo ponderado é um subgrafo que abrange todos os vértices do grafo original, formando uma árvore, e tem um peso total mínimo entre todas as árvores geradoras desse grafo. Em outras palavras, é uma árvore que conecta todos os vértices de maneira eficiente, de modo que a soma dos pesos das arestas na árvore seja a menor possível, como mostrado na Figura 2. A complexidade do problema aumenta quando certas restrições são impostas, como grau, custo, diâmetro entre outros, exigindo a aplicação de métodos sofisticados para sua resolução.

Figura 2 – Exemplo de uma AGM



Fonte: Adaptação de (NICOLETTI, 2017)

Este trabalho aborda o problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD) que é uma variante da AGM ao qual impomos uma restrição específica ao diâmetro da árvore gerada. O diâmetro de uma árvore corresponde ao número de arestas da maior distância entre todos os pares de nós (SANTOS, 2006), ou ainda, é o comprimento do caminho mais longo entre quaisquer dois vértices na árvore. Essa medida ajuda a avaliar a extensão máxima da árvore.

1.1 Motivação

O estudo sobre Árvores Geradoras Mínimas com Restrição de Diâmetro é fundamental devido às diversas aplicações práticas e desafios teóricos que esse problema aborda. Em redes de comunicação, como a internet, é essencial garantir que a latência e o tempo de resposta sejam mantidos dentro de limites aceitáveis. O uso de uma (AGMRD) ajuda a minimizar o atraso máximo entre quaisquer dois nós da rede, garantindo uma comunicação eficiente. Com o avanço e maior presença dos dispositivos de IoT *(Internet of Things)* sistemas de sensores e redes sem fio, a conservação de energia e a minimização da latência são críticas (MAJEED; RAUF, 2020).

Árvores Geradoras Mínimas com Restrição de Diâmetro podem ser usadas em aplicações práticas, como a construção de sistemas de distribuição de água, eletricidade ou gás, garantir que a infraestrutura seja eficiente e mantenha custos mínimos é essencial. O uso dessas soluções pode ajudar a planejar e otimizar a distribuição de recursos. Em logística e planejamento de rotas, a restrição de diâmetro desempenha um papel crítico. Outro exemplo, a aplicabilidade em sistemas de transporte público, como ônibus e metrô, onde é importante minimizar o tempo de viagem entre os pontos de origem e destino, considerando o limite máximo de tempo que os passageiros estão dispostos a esperar.

Segundo Anagnostopoulos (2012), no campo das redes sociais e colaborativas, a eficiência da comunicação é crucial. Encontrar uma árvore geradora com restrição de diâmetro é útil para minimizar o número máximo de saltos ou conexões necessárias para atingir qualquer pessoa na rede, garantindo que a informação seja transmitida de forma rápida e eficaz.

O problema de encontrar uma Árvore Geradora Mínima com Restrição de Diâmetro ainda representa um desafio de grande interesse na teoria dos grafos e na análise de algoritmos. Com o crescimento de aplicações que envolvem big data e IoT, a necessidade de processar grafos de grande escala com restrições específicas como o diâmetro da árvore geradora se torna ainda mais premente. Isso exige novas abordagens e heurísticas que possam lidar com esses desafios de maneira eficaz (SHARMA; TRIVEDI, 2020). Observa-se que novas aplicações para o AGMRD estão surgindo com novas tecnologias e inovações. Segundo Alanis (2018), a crescente complexidade das redes de comunicação modernas, como as redes 5G, e das redes de energia, como as smart grids, exige soluções de otimização que minimizem a latência e garantam a eficiência. Mesmo com muitos algoritmos existentes, as novas restrições e a escala das redes modernas justificam a continuidade da pesquisa em AGMRD.

O estudo das Árvores Geradoras Mínimas com Restrição de Diâmetro (AGMRD) nos dias atuais, mesmo em um campo com uma longa história e muitas soluções propostas, pode ser feito ao considerar a evolução contínua das demandas tecnológicas e o surgimento

de novos desafios em diversas áreas, portanto, esse trabalho visa trazer contribuições para o problema que está ainda mais presente em diversas aplicações do mundo real e necessita de novas abordagens de solução.

1.2 Objetivos

O objetivo principal deste trabalho é desenvolver novas abordagens para resolver o problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD), utilizando heurísticas construtivas combinadas com a meta-heurística BRKGA (*Biased Random-Key Genetic Algorithm*), conhecido ainda como: Algoritmo Genético de Chaves Aleatórias Viciadas.

Os objetivos específicos deste trabalho são:

- Investigar problemas existentes na literatura relacionados à AGMRD, analisando diferentes abordagens e metodologias propostas.
- Propor e implementar heurísticas construtivas, integrá-las ao BRKGA para explorar de forma eficiente o espaço de busca, garantindo a geração de soluções viáveis e de alta qualidade.
- Desenvolver técnicas de busca local para o refinamento das soluções geradas pelo BRKGA, utilizando propriedades dos grafos.
- Avaliar a eficácia das abordagens propostas através de experimentos computacionais em instâncias conhecidas da literatura, comparando os resultados obtidos com métodos tradicionais em termos de qualidade da solução e tempo computacional.
- Contribuir para o avanço do estado da arte na resolução do problema AGMRD.

1.3 Organização do Trabalho

O restante deste trabalho está organizado como segue: O Referencial Teórico necessário para compreender a proposta desta pesquisa e Revisão de Literatura encontra-se no Capítulo 2, que contém a Seção 2.1 sobre o tema AGM e a Seção 2.2 sobre o AGMRD e métodos de resolução. Um conjunto de trabalhos da literatura que propuseram solucionar o problema são apresentados no Capítulo 3. O Capítulo 4 apresenta a proposta desta pesquisa, enquanto o Capítulo 5 descreve a metodologia adotada. Os resultados dos testes realizados nessa proposição de pesquisa estão no Capítulo 7 além da conclusão e indicação de trabalhos futuros.

2 Referencial Teórico

Para melhor entendimento sobre a proposta deste trabalho, serão apresentados alguns conceitos que precedem o problema AGMRD dentro da Teoria dos Grafos.

2.1 Árvore Geradora Mínima - AGM

Uma árvore geradora T de um grafo $G = (V, E)$, onde $|V| = n$ e $|E| = m$, é aquela que contém todos os vértices de G e possui exatamente $n - 1$ arestas. Entre o conjunto de árvores geradoras de um grafo, aquela com menor custo é conhecida como Árvore Geradora Mínima (AGM). Considera-se um Grafo não direcionado, conectado, no qual a cada aresta é atribuído um valor não negativo, como custo ou tempo. O objetivo é encontrar uma estrutura de conexão, uma árvore, na qual todos os nós estejam interligados por um único caminho. Essa estrutura deve ter o menor peso possível, calculado como a soma dos valores associados às arestas escolhidas, com o intuito de minimizá-lo.

Em termos de complexidade computacional, o MSTP, é um problema fácil de resolver e os algoritmos de *Kruskal* e *Prim* demandam, no pior caso, respectivamente, $O(m \log n)$ e $O(n^2)$ operações algébricas elementares e de comparação para resolvê-lo, sendo $n = |V|$ e $m = |E|$(AZEVEDO, 2021).

2.1.1 Algoritmo de Kruskal

O propósito do algoritmo de *Kruskal* consiste em identificar uma árvore geradora mínima em qualquer grafo conexo com pesos atribuídos às arestas. A abordagem subjacente é selecionar as arestas uma a uma, garantindo que não sejam criados ciclos. Isso resultará, ao final, na formação de uma árvore geradora mínima para qualquer tipo de grafo. O algoritmo funciona de forma gulosa, ou seja, as arestas escolhidas para entrar na solução são sempre as de menor peso (SANTOS; LIMA; ALOISE, 2014). Cada vez que um ciclo é identificado, a aresta é excluída e não pode mais ser considerada na solução. Esse procedimento é repetido até que não haja mais arestas disponíveis. Ao término do algoritmo, a árvore estará completamente formada.

Para a implementação, esse algoritmo faz uso de três conjuntos: E, T e V_S. O conjunto T é empregado para armazenar as arestas da árvore expandida. O algoritmo opera transformando uma floresta expandida de G em uma única árvore. O conjunto V_S contém os conjuntos representando as árvores na floresta expandida. Inicialmente, inclui todos os vértices de G, onde cada vértice é um conjunto unitário.

As arestas são escolhidas por ordem crescente de custo. Quando uma aresta une duas subárvores da floresta expandida, estas são unidas em V_S, quando não, ela é descartada, pois forma ciclo. O Pseudocódigo é mostrado com mais detalhes no Algoritmo 1

O procedimento é bastante simples e fácil de implementar. Uma alternativa para implementar o algoritmo de *Kruskal* seria ordenar as arestas de maneira crescente antes de entrar no laço principal. Dessa forma, seria suficiente percorrer o vetor em vez de procurar pela aresta de menor peso. O resultado permanece inalterado, assim como a complexidade.

Algoritmo 1: KRUSKAL
Entrada: Grafo $G = (V, E)$
Saída: S

```
1 início
2     enquanto existirem arestas em E faça
3         Escolher a aresta de menor peso ∈ E e que não esteja na solução S;
4         se não formar ciclo então
5             Aresta removida de E e inserida em S;
6         fim
7         Aresta removida de E;
8     fim
9 fim
```

2.1.2 Algoritmo de Prim

O algoritmo de *Prim* é semelhante ao anterior, também usa uma estratégia gulosa, polinomial e de implementação simples. (SANTOS; LIMA; ALOISE, 2014). Este algoritmo atualmente parte de uma subárvore exclusivamente e continua a conectar os nós a esta subárvore até que a árvore geradora mínima seja concluída. Isso é diferente do método anterior, que se originava a partir de várias árvores isoladas. Em contraste com *Kruskal*, onde a aresta de menor peso é inserida na solução, no algoritmo de *Prim*, o próximo nó a ser adicionado à solução é aquele que está ligado a qualquer nó já presente na solução e tem o peso mínimo. Essa abordagem, além de ser altamente eficiente, também evita a formação de ciclos.

Para implementá-lo, são necessárias essencialmente duas estruturas de dados: uma contendo todos os nós do grafo e outra contendo as arestas que farão parte da solução final. Em cada iteração, um nó é retirado da lista de nós disponíveis e uma de suas arestas é adicionada à solução. Para evitar a formação de ciclos, cada nó que ainda não está na solução só pode ser conectado a um nó que já pertence à solução. O critério para a escolha da aresta é o menor custo. Esse processo se repete até que todos os nós tenham sido conectados à Árvore Geradora de Mínima (AGM), que constitui a solução final. O pseudocódigo é apresentado com mais detalhes no Algoritmo 2.

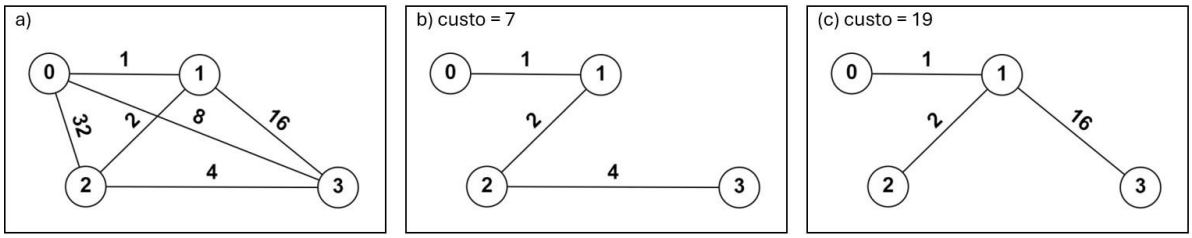
Algoritmo 2: PRIM **Entrada**:
Grafo $G = (V, E)$ **Saída**: S

início
  Escolhe um vértice qualquer e insere o vértice em um conjunto S;
  Retire o vértice de V;
  enquanto *existirem vértices em* V **faça**
    Dentre os vértices em V seleciona aquele com aresta de menor peso e adjacente a qualquer vértice em S;
    O vértice sai de V e a aresta entra em S;
  fim
fim

2.2 Árvore Geradora Mínima com Restrição de Diâmetro - AGMRD

O problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD) é uma especialização do problema da Árvore Geradora Mínima, é definido sobre um grafo $G = (V, E)$, onde V representa o conjunto de vértices e E o conjunto de arestas. Cada aresta $[i, j] \in E$ tem um custo associado $C_{ij} \geq 0$. Em qualquer árvore geradora de G, existe um caminho único P_{ij} entre dois vértices i e j. O diâmetro de uma árvore geradora é dado por $max\{d_{ij} : i, j \in V\}$. O problema AGMRD visa encontrar uma árvore geradora que minimize o custo total, sujeita à condição de que o diâmetro seja pelo menos D, onde D é um número inteiro positivo que satisfaz $2 \leq D \leq |V - 1|$. A Figura 3 ilustra um grafo completo de quatro vértices (a), sua árvore geradora mínima (b), e uma árvore geradora mínima com diâmetro restrito.

Figura 3 – Exemplo para caso de D = 2



Fonte: (SANTOS, 2006)

Em quatro casos particulares o problema é polinomial, segundo (SANTOS, 2006). O primeiro caso ocorre quando $D = |V| - 1$ sua solução é a própria Árvore Geradora Mínima (AGM). Já o segundo, quando $D = 2$, neste caso á Árvore solução é composta por um vértice central conectando arestas aos demais nós do grafo. No terceiro caso é para $D = 3$, sendo que a solução será composta por uma aresta central, ou aresta ponte, conectando dois grafos do tipo estrela. Por fim, o quarto caso, ocorre quando todos os

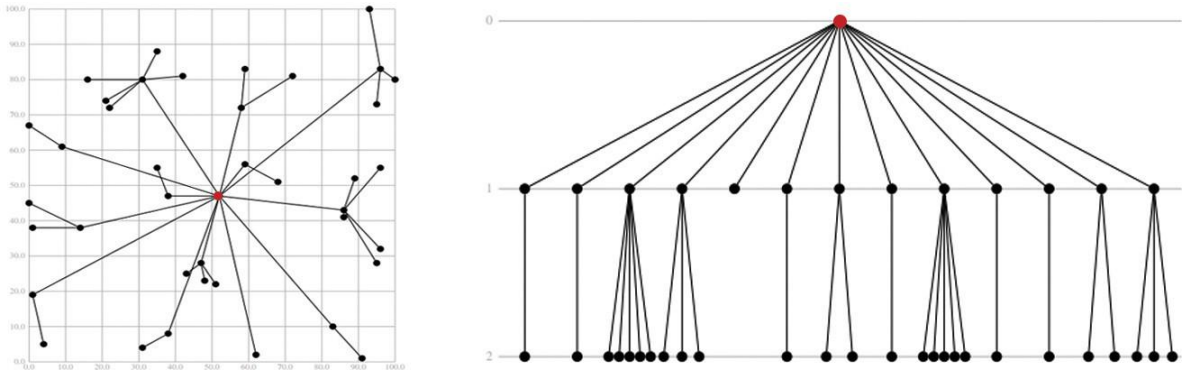


Figura 4 – Árvore de diâmetro $D = 4$, visões no plano à esquerda e hierárquica à direita.
Fonte: (SANTOS, 2006)

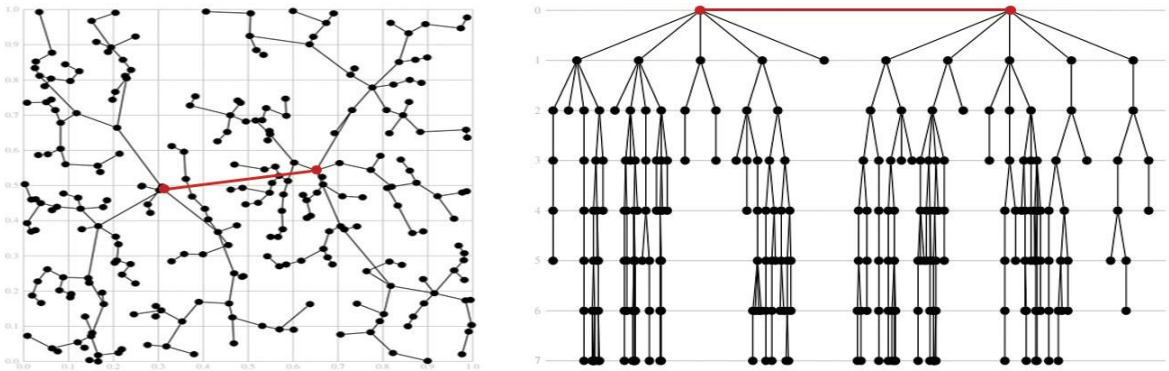


Figura 5 – Árvore de diâmetro $D = 15$, visões no plano à esquerda e hierárquica à direita.
Fonte: (SANTOS, 2006)

pesos das arestas são iguais, não influenciando na construção da árvore, bastando assim conectar todos os nós, obedecendo D, e neste caso temos que qualquer solução é ótima.

Quando o diâmetro D é maior que quatro ($4 \leq D < |n| - 1$), este problema é denominado NP-difícil, indicando que não existem algoritmos conhecidos que possam encontrar a solução exata em tempo polinomial.

De forma geral, quando o diâmetro da árvore é par, o grafo terá um vértice central, como ilustrado na Figura 4, e quando o diâmetro for ímpar, uma aresta central ou aresta ponte, conforme exemplo da Figura 5.

2.3 Abordagens para resolver o AGMRD

Na literatura, existe uma variedade de métodos computacionais para solucionar o problema da Árvore Geradora Mínima com Restrição de Diâmetro, entre eles estão os métodos exatos, as heurísticas, as metaheurísticas, além de abordagens híbridas. As subseções a seguir apresenta de forma sucinta cada um dos métodos.

2.3.1 Métodos exatos

O problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD) tem como objetivo minimizar o custo do somatório das arestas que conectam todos os vértices do grafo, mantendo um diâmetro fixado. Portanto, pode ser formulado como um problema de programação linear inteira:

$$\text{Minimizar} \sum_{e \in E} c(e) \cdot x(e)$$

Sujeito a:

$$\sum_{e \in E} x(e) = |V| - 1$$

$$\delta(u, v) \leq D, \quad \forall u, v \in V, u \neq v$$

$$x(e) \in \{0, 1\}, \quad \forall e \in E$$

$G = (V, E)$ um grafo não direcionado com um conjunto de vértices V e um conjunto de arestas E. $c(e)$: o custo associado a cada aresta e em E. $x(e)$: uma variável binária que indica se a aresta e está presente na árvore ($x(e) = 1$) ou não ($x(e) = 0$). D: o diâmetro menor ou igual ao desejado para a árvore.

Segundo Narsingh Deo e Ayman Abdalla(2000) os métodos exatos para resolver o problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD) podem se tornar ineficientes conforme o tamanho do problema aumenta. Isso ocorre devido à complexidade computacional inerente ao problema, que é classificado como NP-completo. A resolução exata desses problemas exige um tempo exponencial, o que torna impraticável para grafos de grande escala. Estudos mostram que, apesar das técnicas avançadas como *branch-and-bound*, *branch-and-cut* e outros métodos de otimização, a escalabilidade continua sendo um desafio significativo.

No entanto, existem algoritmos heurísticos que podem encontrar soluções próximas à solução ótima (SOUSA; ALOISE; SANTOS,).

2.3.2 Métodos Heurísticos

Os métodos heurísticos são técnicas projetadas para encontrar soluções boas o suficiente para problemas complexos dentro de um tempo computacional aceitável. Esses métodos são especialmente úteis quando os métodos exatos são impraticáveis devido à complexidade exponencial dos problemas. Heurísticas construtivas, heurísticas de busca local e metaheurísticas são algumas das abordagens mais comuns.

Para o AGMRD, várias heurísticas foram propostas e demonstraram eficiência na obtenção de soluções próximas ao ótimo. Neste trabalho dedicamos o Capítulo 4 para este método de solução.

2.3.3 Métodos Híbridos

Os métodos híbridos combinam técnicas exatas e heurísticas ou metaheurísticas para resolver problemas de otimização complexos, particularmente aqueles classificados como NP-difíceis. Esses métodos são altamente eficazes porque aproveitam os pontos fortes de ambos os tipos de abordagem: a precisão e a garantia de otimalidade das técnicas exatas e a eficiência e flexibilidade das heurísticas e metaheurísticas.

Os métodos exatos, como o branch-and-bound, são conhecidos por fornecer soluções ótimas para problemas de otimização. No entanto, sua aplicabilidade é limitada a instâncias de menor escala devido ao seu alto custo computacional. Em contrapartida, as heurísticas e metaheurísticas, como algoritmos genéticos, GRASP (Greedy Randomized Adaptive Search Procedure) e busca tabu, oferecem soluções próximas do ótimo em um tempo computacional significativamente menor, mas sem garantias de otimalidade.

Os métodos de programação matemática podem ser integrados a meta-heurísticas para melhorar o desempenho na resolução de problemas de diversas maneiras. As abor- dagens existentes podem ser amplamente classificadas em duas categorias principais: Combinações colaborativas e Combinações integrativas. As Combinações colaborativas referem-se a métodos em que algoritmos exatos e heurísticos trocam informações, mas não são parte um do outro. Eles podem ser executados sequencialmente, interligados ou em paralelo. Combinações integrativas, por outro lado, envolvem a integração de uma técnica como um componente subordinado de outra. Neste caso, existe um algoritmo mestre distinto, que pode ser exato ou meta-heurístico, e pelo menos um escravo integrado (FAKHRAVAR, 2023).

A literatura oferece diversos exemplos de sucesso na aplicação de métodos híbridos. Em problemas de roteamento de veículos, combinações de programação linear inteira com heurísticas baseadas em colônia de formigas ou algoritmos genéticos têm mostrado resultados promissores. Esses métodos são capazes de resolver instâncias de problemas que seriam intratáveis por abordagens puramente exatas ou heurísticas, proporcionando soluções de alta qualidade em tempo computacional viável.

Após a apresentação das diferentes abordagens para a resolução do problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD), é essencial destacar as principais vantagens e desvantagens de cada metodologia discutida. A seguir, na Tabela 1 temos um quadro comparativo que sintetiza as características fundamentais das formas exatas, dos métodos heurísticos e dos métodos híbridos. Esse quadro oferece uma visão clara e objetiva, facilitando a compreensão das circunstâncias nas quais cada abordagem se mostra mais adequada, bem como os desafios e limitações que podem surgir na sua aplicação. Essa análise comparativa é crucial para fundamentar a escolha da metodologia mais apropriada em diferentes cenários práticos.

Metodologia	Vantagens	Desvantagens
Formas Exatas	• Garante a solução ótima do problema. • Aplicável a proble- mas de pequena escala onde a precisão é cru- cial.	• Elevado custo com- putacional, especial- mente para grafos grandes. • Escalabilidade limi- tada. • Inviável para redes muito grandes devido ao tempo de processa- mento.
Métodos Heurísticos	• Soluções rápidas e apli- cáveis a problemas de grande escala. • Flexibilidade para adaptação a dife- rentes variações do problema.	• Não garantem a solu- ção ótima. • Qualidade da solução depende da qualidade da heurística usada. • Pode exigir ajuste fino ou conhecimento espe- cializado para configu- rar heurísticas efica- zes.
Métodos Híbridos	• Combina a precisão das formas exatas com a eficiência dos méto- dos heurísticos. • Melhor balanço entre qualidade da solução e tempo de execução.	• Complexidade na im- plementação. • Pode exigir mais re- cursos computacionais que os métodos heurís- ticos simples. • Dificuldade em balan- cear a integração dos dois métodos para oti- mização.

Tabela 1 – Comparativo das metodologias para resolução do problema AGMRD

3 Trabalhos relacionados

Este capítulo explora uma série de estudos relacionados ao problema de Árvores Geradoras de Custo Mínimo com Restrição de Diâmetro (AGMRD). Inicialmente, apresen- tamos revisões encontradas na literatura, oferecendo uma compilação de informações sobre as principais técnicas empregadas na resolução do problema. Em seguida, fornecemos uma descrição de trabalhos abordados na literatura no período de 2000 a 2021. Alguns desses estudos mencionados neste capítulo compartilham instâncias semelhantes às que foram utilizadas em nosso próprio trabalho, destacando a relevância e a conexão entre as diferentes abordagens e pesquisas realizadas ao longo dessas duas décadas.

Raildl e Julstrom (2003) abordam o problema da árvore geradora mínima de diâmetro limitado (BDMST), que visa encontrar uma árvore geradora em um grafo conectado, ponderado e não direcionado, onde o caminho entre dois vértices não excede um limite de diâmetro D. Desenvolveram um algoritmo denominado *Randomized Greedy Heuristic* (RGH). Os testes comparativos da RGH com a OTT indicam que a RGH produz resultados substancialmente melhores. A proposta inclui o desenvolvimento de uma nova heurística gulosa aleatória e um algoritmo evolutivo (EA) para resolver o problema de maneira mais eficiente do que as heurísticas existentes. A metodologia inclui duas abordagens principais: uma nova heurística gulosa aleatória e um algoritmo evolutivo (EA). A heurística constrói uma árvore geradora de diâmetro limitado a partir de vértices centrais escolhidos aleatoriamente, anexando cada próximo vértice com a aresta válida de menor peso. Isso difere de heurísticas tradicionais, como a OTTC (One Time Tree Construction), que imitam o algoritmo de Prim e geralmente produzem resultados subótimos. O algoritmo evolutivo proposto codifica árvores geradoras como listas de arestas, complementadas com vértices centrais. Ele aplica operadores de recombinação e mutação que preservam o limite de diâmetro, garantindo que sempre gerem árvores válidas. A recombinação favorece arestas comuns aos pais, enquanto quatro operadores de mutação (exclusão de arestas, movimento central, substituição gulosa de arestas e otimização de subárvore) exploram diferentes aspectos do espaço de busca. O EA começa com uma população inicial gerada pela heurística gulosa aleatória e evolui soluções ao longo de várias iterações.

Um trabalho importante é o de Santos (2006) que traz uma abordagem inovadora para resolver o problema das Árvores Geradoras de Custo Mínimo com Restrição de Diâmetro (AGMRD). O objetivo principal é desenvolver modelos matemáticos e algo- ritmos heurísticos que possam produzir soluções eficientes para problemas da AGMRD. Os modelos propostos são reforçados com desigualdades válidas e utilizam relaxação lagrangeana para melhorar os limites inferiores das soluções. A metodologia utilizada inclui a adaptação de heurísticas construtivas como OTT e RGH, além do desenvolvimento

de quatro estratégias de busca local denominadas 1-opt, 2-opt, adoção de sub-árvore e substituição de caminhos. Além disso, foram implementadas heurísticas avançadas como GRASP (Greedy Randomized Adaptive Search Procedure) e algoritmos híbridos que combinam GRASP com ILS (Iterated Local Search). A abordagem proposta também utiliza técnicas de perturbação DCV e SAR para melhorar a exploração do espaço de soluções e otimizar o desempenho computacional. Os resultados experimentais mostram que os algoritmos desenvolvidos são capazes de gerar soluções muito próximas do ótimo, com desempenho superior aos melhores resultados conhecidos na literatura para várias instâncias de problemas. As heurísticas lagrangeanas, embora consumam mais tempo de processamento devido à complexidade da relaxação lagrangeana, produzem soluções de alta qualidade. Os algoritmos do tipo GRASP e híbridos mostraram-se particularmente eficazes, alcançando soluções ótimas ou muito próximas do ótimo para a maioria das instâncias testadas.

Nghia et al. (2007) propuzeram uma nova heurística gulosa randomizada (NRGH) para inicializar a população em um algoritmo genético (GA) proposto para resolver o problema BDMST. A inovação principal é um novo operador de recombinação chamado operador k-recombinação, que gera um filho a partir de vários indivíduos da geração de pais, em vez dos dois habituais. Esse operador visa aumentar a hereditariedade forte ao incorporar mais potencial de diversidade genética. Realizou-se experimentos comparando três algoritmos: NRGH, um algoritmo genético existente (EARJ) e o novo algoritmo genético utilizando o operador k-recombinação (EA-k). Os testes foram conduzidos em grafos completos euclidianos com tamanhos variando de 100 a 1000 vértices e diferentes limites de diâmetro (10, 15, 20 e 25). Os resultados mostram que o operador k-recombinação superou consistentemente o algoritmo EARJ em termos de custo mínimo, máximo e médio das árvores geradoras. A introdução do operador k-recombinação em algoritmos genéticos para o problema BDMST resultou em soluções de melhor qualidade e maior eficiência computacional. Em testes comparativos, o EA-k não só produziu árvores geradoras de menor custo, mas também o fez de maneira mais rápida do que os métodos anteriores. A abordagem mostrou-se promissora e sugere potencial para aplicação em outras variantes do problema de árvores geradoras e em diferentes tipos de grafos.

Gouveia et al. (2011) abordaram dois problemas de otimização combinatória: o problema da Árvore Geradora Mínima com Restrição de Saltos (HMSTP) e o problema da Árvore Geradora Mínima com Restrição de Diâmetro (DMSTP). Esses problemas são relevantes no design de redes de telecomunicação, onde a qualidade de serviço é crítica. O HMSTP envolve encontrar uma árvore geradora mínima com um número máximo de arestas em qualquer caminho a partir de um nó raiz, enquanto o DMSTP limita o número de arestas em qualquer caminho entre dois nós. Foi proposta uma modelagem inovadora desses problemas como problemas de árvore de Steiner (STP) em grafos em camadas. Demonstrou-se que o HMSTP pode ser transformado em um STP direcionado

em um grafo em camadas, onde cada camada representa um nível de profundidade do grafo original. A transformação permite o uso de um modelo de corte direcionado, que proporciona limites superiores mais fortes e dominantes em comparação com os modelos existentes para o HMSTP. A técnica de corte direcionado envolve a criação de variáveis associadas às arestas e aos cortes no grafo em camadas. Cada aresta no grafo original é representada por duas arestas direcionadas no grafo em camadas. Os cortes direcionados garantem que, para cada sub-conjunto de nós que não inclui o nó raiz, exista pelo menos uma aresta conectando os nós dentro e fora do sub-conjunto. Esta técnica resulta em uma relaxação linear do problema que é mais forte que as abordagens anteriores, proporcionando melhores limites inferiores. Os resultados experimentais mostram que o método proposto é significativamente mais eficiente do que os métodos anteriores. Os testes foram realizados em grafos completos e densos, com até 160 nós. O algoritmo de branch-and-cut desenvolvido pelos autores mostrou-se superior em termos de qualidade das soluções e tempo de computação, resolvendo instâncias que antes eram consideradas difíceis ou impossíveis de serem resolvidas. A modelagem do HMSTP e do DMSTP como problemas de árvore de Steiner em grafos em camadas, utilizando um modelo de corte direcionado, oferece uma solução eficiente e eficaz para esses problemas complexos.

Para o DMSTP, a abordagem é adaptada de maneira semelhante. O DMSTP é inicialmente transformado para um grafo em camadas com duas camadas adicionais representando os possíveis nós centrais do diâmetro da árvore. Esse método permite uma formulação eficiente do problema, tanto para diâmetros pares quanto ímpares.

Após a revisão de estudos relacionados ao problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD), discutidos no Capítulo 3, observa-se que, apesar dos avanços significativos na solução deste problema, ainda há espaço para a exploração de novas heurísticas que possam melhorar a eficiência computacional. As abordagens existentes, embora eficazes, muitas vezes enfrentam desafios relacionados ao custo computacional elevado, especialmente em instâncias maiores e mais complexas.

Neste contexto, o Capítulo 4 introduz, primeiramente, a heurística OTT, uma das primeiras a solucionar o AGMRD com desempenho satisfatório para instâncias até uma quantidade de vértices. Em seguida, o capítulo apresenta o BRKGA, acompanhado de heurísticas construtivas que buscam superar essa limitação. Essa abordagem explora uma lacuna importante, oferecendo uma alternativa que reduz o custo computacional e apresenta soluções mais eficientes para instâncias maiores. A transição para as abordagens heurísticas neste capítulo é, portanto, motivada pela busca por métodos que complementem e aprimorem as soluções já conhecidas na literatura.

4 Abordagens Heurísticas

Heurísticas, ou métodos heurísticos, são abordagens que se destacam por sua eficácia na resolução de questões complexas, particularmente quando as soluções exatas são de difícil obtenção. Essas estratégias envolvem o uso de regras práticas, intuição e experiência para orientar a busca por soluções aceitáveis, embora não necessariamente ótimas. Ao enfatizar eficiência computacional, as heurísticas são frequentemente empregadas em situações em que os métodos exatos seriam computacionalmente inalcançáveis. Elas oferecem uma abordagem pragmática e rápida para explorar espaços de solução, priorizando resultados satisfatórios em detrimento de soluções precisas e completas.

As heurísticas construtivas visam criar uma solução de maneira incremental. Começando com uma solução inicial vazia, o problema é abordado incrementalmente por meio de técnicas de adição, onde novos elementos são gradualmente incorporados à solução a cada iteração. Essas heurísticas comumente empregam métodos gulosos e são mais adequadas para problemas em que as soluções viáveis podem ser obtidas facilmente. Na literatura encontramos uma heurística bastante utilizada para AGMRD: A OTT (One Time Tree) ou OTTC (One Time Tree Construction).

O objetivo deste capítulo é apresentar a heurística OTT e a metaheurística evoluci- onária BRGKA.

4.1 Algoritmo OTT

A heurística OTT (One Time Tree) visa construir uma solução viável utilizando o algoritmo de Prim como base. O algoritmo de Prim é um método guloso que constrói uma Árvore Geradora Mínima (MST) adicionando, a cada etapa, a aresta de menor custo que não forma um ciclo com as arestas já presentes na árvore. Para adaptar esse processo ao problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD), além de evitar ciclos, a heurística OTT também verifica se a inclusão de cada nova aresta não viola a restrição de diâmetro imposta. A heurística OTT está apresentada conforme Algoritmo 3.

A heurística OTT é uma das poucas conhecidas na literatura que resolve o pro- blema da AGMRD com desempenho satisfatório com instâncias de aproximadamente 500 (quinhentos) vértices (SANTOS, 2006).

Embora a heurística OTT tenha sido uma das primeiras a solucionar o problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD) de maneira eficaz em instâncias menores, ela apresenta uma limitação significativa em termos de custo

Algoritmo 3: OTT **Entrada**:
Grafo $G = (V, E)$ **Saída**: T^a

início
  $distância[i][j] \leftarrow 0 \ \forall \ i,j \in V$;
  $max\_distancia[i] \leftarrow 0 \ \forall \ i \in V$;
  Selecionar aleatoriamente um nó $i \in V$;
  $T^v \leftarrow \{i\}$;
  $Q \leftarrow V - \{i\}$;
  enquanto $(Q \neq \emptyset)$ **faça**
    $C_{min} \leftarrow \infty$;
    para $(j \in Q)$ **faça**
      $chave[j] \leftarrow -1$;
      $melhor \leftarrow \infty$;
      para $(i \in T^v)$ **faça**
        se $(C_{ij} < melhor)$ *e* $(max\_distância[i] + 1 \leq D)$ **então**
          $chave[j] \leftarrow i$;
          $melhor \leftarrow C_{ij}$;
        fim
      fim
      $i \leftarrow chave[i]$;
      se $(C_{ij} < C_{min})$ *e* $(i \neq -1)$ **então**
        $C_{min} \leftarrow C_{ij}$;
        $u \leftarrow i$;
        $v \leftarrow j$;
      fim
    fim
    $T^v \leftarrow T^v \cup \{v\}$;
    $Q \leftarrow Q - \{v\}$;
    $T^a \leftarrow T^a \cup \{[v, v]\}$;
    Atualizar *distância* e *max_distância* de v em relação a todos os nós na solução;
  fim
  retorna T^a;
fim

computacional. Essa limitação decorre da necessidade de verificar o diâmetro da árvore a cada passo do algoritmo, o que pode se tornar computacionalmente oneroso, especialmente à medida que o tamanho do grafo aumenta. Essa verificação constante impacta diretamente a escalabilidade da heurística, tornando-a menos adequada para instâncias maiores e mais complexas. Essa limitação, portanto, abre espaço para a exploração de novas abordagens que possam evitar essa verificação contínua e oferecer soluções mais eficientes, como as que serão discutidas nas seções seguintes.

4.2 Metaheurísticas

Metaheurísticas representam abordagens de alto nível projetadas para enfrentar problemas de otimização complexos e de ampla escala. Diferentemente de algoritmos específicos para problemas particulares, as metaheurísticas são estratégias gerais e flexíveis que se aplicam a uma variedade de domínios. Essas técnicas exploram e combinam diferentes heurísticas para guiar a busca por soluções de alta qualidade no espaço de busca do problema. As metaheurísticas buscam um equilíbrio entre exploração ampla e explotação eficaz, adaptando-se dinamicamente para superar desafios computacionais e encontrar soluções aproximadas, muitas vezes em um tempo computacionalmente viável. Sua versatilidade e capacidade de lidar com problemas complexos tornam as metaheurísticas ferramentas valiosas na solução de diversos problemas em diversas áreas.

Segundo (OSMAN I. H.; KELLY, 1996) metaheurísticas são métodos iterativos de geração de soluções que guiam uma heurística subordinada combinando de forma inteligente diferentes conceitos para explorar e explotar o espaço de busca, utilizando estratégias de aprendizagem para estruturar informações a fim de localizar eficientemente soluções próximas do ótimo. À medida que as dimensões da instância aumentam, observa- se um aumento exponencial no tempo de resposta exigido pelos métodos exatos. Nesse contexto, torna-se necessário explorar abordagens não exatas que possam proporcionar soluções próximas ao ótimo em tempo polinomial. Assim, a opção recai sobre o emprego de heurísticas (heurísticas, heurísticas híbridas, metaheurísticas e hiper-heurísticas), as quais possibilitam a descoberta de soluções viáveis, ocasionalmente ótimas.

De acordo com Noronha et al. (2008), diversos métodos têm sido sugeridos para resolver o problema da Árvore Geradora de Custo Mínimo com Restrição de Diâmetro (AGMRD). Esses métodos incluem abordagens baseadas em algoritmos clássicos, como os de Prim e Kruskal, bem como heurísticas e técnicas avançadas de otimização, com o uso combinado de metaheurísticas, como GRASP, colônia de formigas, algoritmos genéticos, entre outros. Entretanto, do nosso conhecimento e até o momento da escrita desse trabalho, a metaheurística BRKGA (*Biased Random-Key Genetic Algorithm*), proposta por Resende 2011, ainda não foi aplicada a esse problema da AGMRD. Assim sendo, por se tratar de uma metaheurística que tem se destacado pela sua eficiência em explorar e otimizar espaços de solução complexos, e aplicada com sucesso em vários problemas de otimização combinatória foi escolhida como foco central neste trabalho. Apresentamos uma versão modificada do BRKGA, que incorpora um avaliador nas sub-rotinas de cruzamento e mutação, resultando em melhorias significativas, especialmente no uso de populações e gerações reduzidas. Também introduzimos duas novas heurísticas construtivas, especificamente desenvolvidas para o BRKGA aplicado ao problema da AGMRD, além de algoritmos de busca local para refinar as soluções obtidas.

Os algoritmos implementados foram submetidos a uma série de testes rigorosos

para avaliar a qualidade dos resultados em relação ao tempo de execução. Além disso, discutimos a escolha dos parâmetros do BRKGA, cujos detalhes serão apresentados nas próximas seções.

Conforme a perspectiva de Blum e Roli (2003), dois elementos cruciais em metaheurísticas são a diversificação e a intensificação. O conceito de diversificação está relacionado à exploração abrangente do espaço de busca, enquanto a intensificação diz respeito à exploração focada na experiência acumulada da busca. Em resumo, a diversificação busca investigar soluções em diferentes regiões do espaço de busca de um problema, enquanto a intensificação utiliza informações de buscas anteriores para aprofundar-se em áreas promissoras desse espaço.

4.3 Algoritmo Genético

O Algoritmo Genético introduzido por Holland (1992) representa uma metaheurís- tica inspirada na Teoria da Evolução e é eficaz para resolver uma variedade de problemas de otimização combinatória. Ele desenvolve uma população de indivíduos, aplicando o princípio de sobrevivência proposto por Charles Darwin. Nesse contexto, os indivíduos evoluem ao longo do tempo por meio da seleção natural, onde os mais adaptados têm uma probabilidade superior de gerar descendentes. A evolução da população ocorre através da aplicação de um conjunto de operadores, incluindo seleção, cruzamento (*crossover*) e mutação.

Conforme (FERNANDES, 2005), o algoritmo genético propõe-se a resolver casos de otimização, visando encontrar a melhor solução em um conjunto de possíveis soluções para um determinado problema. O processo se inicia com um conjunto de indivíduos que serão combinados para gerar novos elementos, os quais serão incorporados à população inicial. Esse mecanismo é repetido ao longo de várias gerações, refinando a população a cada iteração e resultando em soluções aprimoradas para o problema em questão.

De acordo com (RICARDO, 2008), o algoritmo genético têm sido aplicado com êxito em diversos problemas de otimização. Diversos termos apresenta uma correspondência entre as denominações do processo de evolução natural e um problema computacional: População o qual representa um conjunto de soluções, indivíduo que representa a solução de um problema, cromossomo que é a representação de uma solução, gene que é parte da representação de uma solução. Os indivíduos são avaliados por uma função aptidão *(fitness)* através da função objetivo. Temos ainda o cruzamento e mutação que são operadores de busca e evolução da população.

4.4 Algoritmo Genético de Chaves Aleatórias Viciadas

Embora o Algoritmo Genético (AG) clássico demonstre eficácia na resolução de problemas em diversas áreas, ele enfrenta um desafio significativo conhecido como convergência prematura. Esse obstáculo resulta na estabilização das soluções presentes na população, limitando a capacidade do algoritmo de explorar continuamente o espaço de busca, devido à falta de diversidade no material genético.

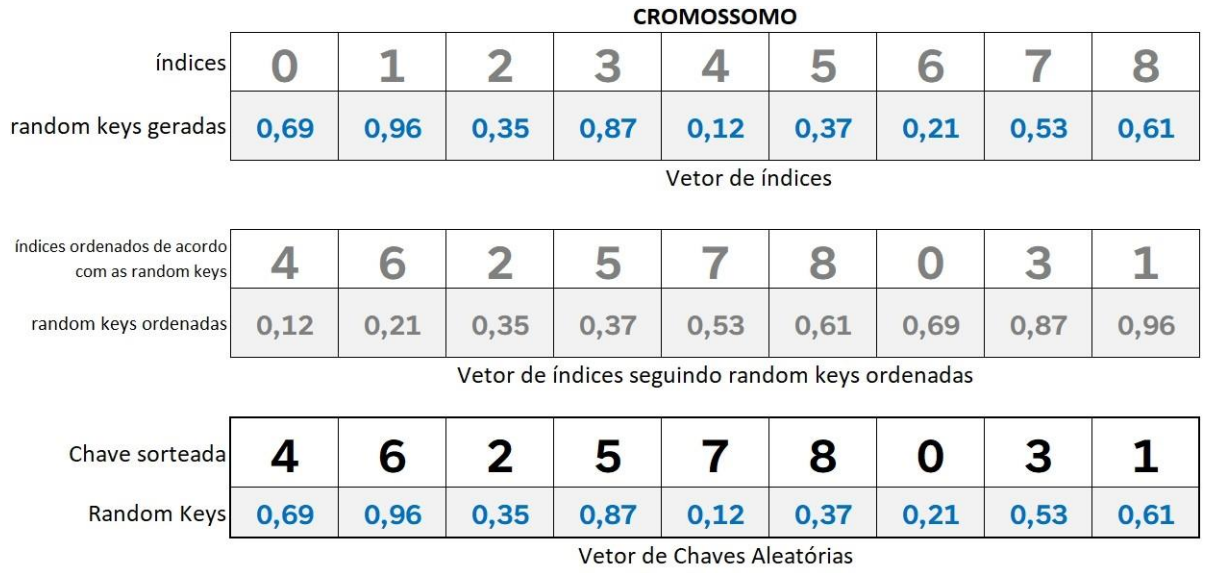
O *Biased Random-Key Genetic Algorithm* (BRKGA), introduzido por (GONÇAL- VES; RESENDE, 2015), representa uma metaheurística evolutiva aplicada a problemas de otimização discreta e global, sendo fundamentado no *Random-Key Genetic Algorithm* (RKGA) de Bean 1994. Embora o BRKGA compartilhe semelhanças operacionais com o Algoritmo Genético convencional, assim como o RKGA, apresenta distinções específicas que serão explicadas a seguir.

O princípio darwinista, fundamentado na teoria da evolução de Charles Darwin, é aplicado de maneira inovadora e eficaz no Biased Random-Key Genetic Algorithm (BRKGA). Este algoritmo evolutivo utiliza conceitos chave da seleção natural, como variação, hereditariedade, competição e seleção diferencial, para resolver problemas de otimização combinatória.

No BRKGA, as soluções potenciais são representadas por vetores de chaves aleatórias, onde cada chave é um número real no intervalo [0, 1]. Em seguida as chaves são ordenadas levando os índices do vetor formando uma sequencia numérica o qual chamamos de chave. A Figura 6 ilustra o processo de geração de uma chave por meio das random keys. Cada chave representa um indivíduo na população que é decodificado em uma solução do problema a ser resolvido, no caso, uma Árvore Geradora com Restrição de Diâmetro.

Este método de representação permite uma variação contínua e uma decodificação eficiente das soluções. A variação, um dos pilares do princípio darwinista, é introduzida no BRKGA através da geração inicial de uma população de soluções aleatórias. Cada indivíduo na população é um vetor de chaves, e as diferenças nas chaves entre os vetores representam a variação genética.

Figura 6 – Vetor de chaves aleatórias



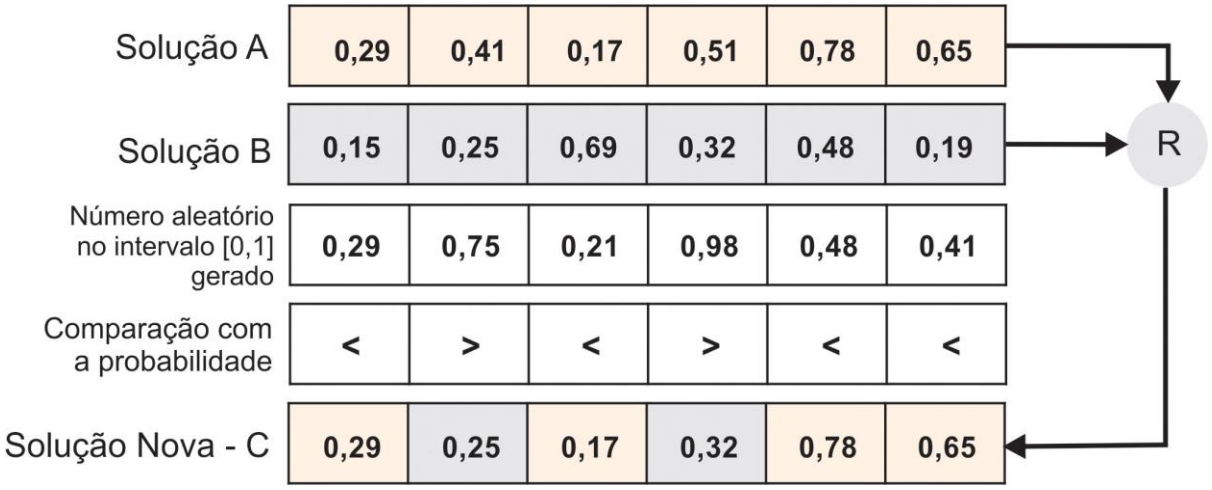
Fonte: Autoria própria.

A hereditariedade no BRKGA é garantida através do processo de cruzamento, onde novos indivíduos (soluções) são gerados combinando características de pais selecionados da população atual. Este cruzamento é enviesado ou "viciado"de forma que as melhores soluções têm uma probabilidade maior de transmitir suas características para a próxima geração. Um indivíduo da População Elite é cruzado com um não-elite, e o valor de probabilidade de herança dos genes do pai elite pode ser definido, dando maior probabilidade de passar suas características aos descendentes.

O BRKGA usa no cruzamento a técnica de cruzamento uniforme parametrizado, também conhecida como PUC (*Parametrized Uniform Crossover*), introduzida por Spears e De Jong (1995) para a geração de descendentes. No contexto do BRKGA, o PUC opera selecionando um indivíduo A, pertencente ao conjunto elite, e um indivíduo B, da população não elite, para o cruzamento. O PUC utiliza um parâmetro ρ_e que determina a probabilidade de um descendente herdar um alelo do indivíduo A. Para criar um descendente C, gera-se aleatoriamente um número $\rho \leq [0, 1]$, que é então comparado com ρ_e. Se $\rho \leq \rho_e$, o alelo do indivíduo A é copiado; caso contrário, o alelo do indivíduo B é selecionado. Geralmente, ρ_e é definido como 0,7, aumentando assim a probabilidade de que o descendente C herde uma quantidade significativa de alelos da solução elite. A Figura 20 detalha o processo de geração de um descendente utilizando a recombinação PUC.

A competição no BRKGA ocorre naturalmente durante a avaliação e seleção das soluções. Cada indivíduo é avaliado com base em uma função de aptidão que mede a qualidade da solução em relação ao problema específico. As melhores soluções, ou seja,

Figura 7 – Recombinação PUC, tendo $\rho_e = 0,70$



Fonte: (GONçALVES; RESENDE, 2011)

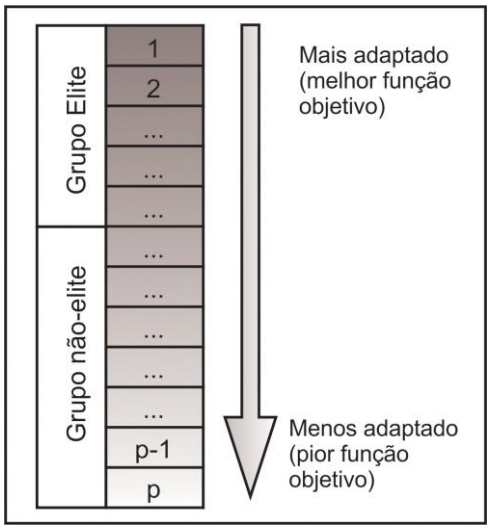
aquelas com maior aptidão, competem com as menos adaptadas pela oportunidade de reproduzir e contribuir com a próxima geração. Este processo de seleção diferencial garante que apenas as soluções mais eficazes continuem a evoluir, semelhante à sobrevivência dos mais aptos na teoria de Darwin.

O Biased Random-Key Genetic Algorithm (BRKGA) incorpora um conceito chamado elitismo, que é crucial para a eficácia do algoritmo. O elitismo no BRKGA assegura que as melhores soluções encontradas ao longo das gerações sejam preservadas e trans- mitidas para a próxima geração, garantindo uma melhoria constante na qualidade das soluções. O elitismo refere-se à estratégia de sempre manter uma determinada proporção, determinado pelo parâmetro Ne que calcula qual percentual da população formará a po- pulação elite, qual dos melhores indivíduos (soluções) da população atual para a próxima geração. Assim, as soluções mais bem adaptadas têm maior probabilidade de sobreviver e propagar suas características vantajosas. A Figura 8 ilustra o agrupamento da população e o elitismo.

Além do elitismo e do cruzamento enviesado, uma pequena taxa de mutação é aplicada para introduzir variações aleatórias nas chaves de alguns indivíduos. Isso ajuda a manter a diversidade genética da população e a evitar a convergência prematura para ótimos locais. Um percentual de genes a ser modificado é definido ao realizar a mutação de indivíduos.

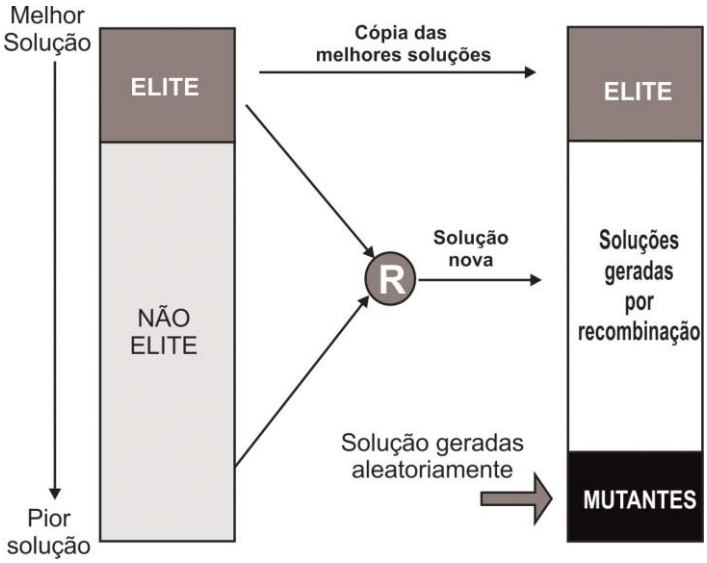
Assim como na evolução natural, onde as espécies se adaptam gradualmente ao ambiente, o BRKGA permite que a população de soluções se adapte ao espaço de solução do problema de otimização. Através de gerações sucessivas de seleção, cruzamento enviesado e mutação, o algoritmo explora eficazmente o espaço de busca e converge para soluções de alta qualidade.

Figura 8 – Agrupamento de uma população de p indivíduos em certa geração k



Fonte: Mainieri (2014)

Figura 9 – Montagem de uma geração



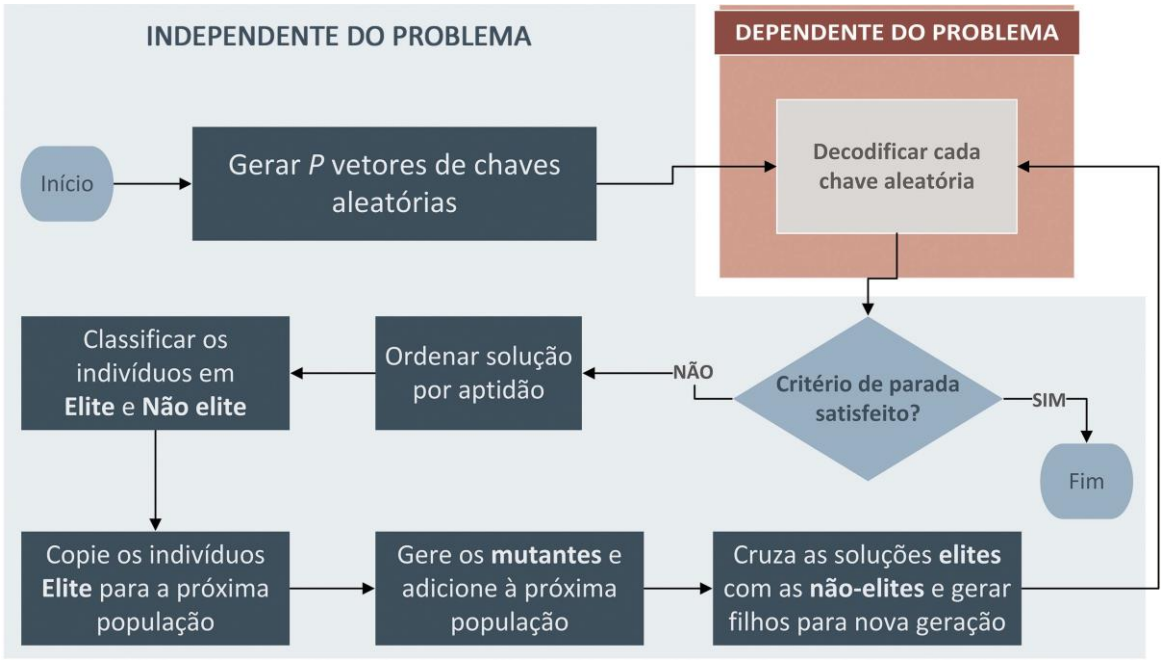
Fonte: Silva et al. (2012)

Embora a diferença entre o Algoritmo Genético e o BRKGA seja pequena em termos de estrutura, há uma diferença significativa no desempenho. O BRKGA é capaz de encontrar soluções de qualidade muito superior quando comparado ao outro algoritmo, especialmente ao se considerar o mesmo tempo de processamento.

A Figura 10, que apresenta o fluxograma do BRKGA, evidencia duas partes bem distintas: uma parte específica ao problema que se pretende resolver e outra independente, aplicável a qualquer tipo de problema. Dessa maneira, a parte independente do BRKGA pode ser implementada uma única vez e reutilizada para resolver diferentes problemas. Quando for necessário aplicar o BRKGA a um novo tipo de problema, basta implementar

o decodificador específico para esse problema.

Figura 10 – Fluxograma do Algoritmo BRKGA



Fonte: (GONçALVES; RESENDE, 2011)

4.4.1 Parâmetros do BRKGA

Alguns parâmetros do BRKGA precisam ser cuidadosamente escolhidos e calibrados. Esses referem-se ao tamanho do vetor de codificação de chaves aleatórias (n), tamanho da população (n_{pop}) O número de soluções elite (n_e), o número das soluções mutantes (n_m) e a probabilidade (ρ_e) de um descendente incorporar a informação do pai elite. A Tabela 2 apresenta os valores recomendados para esses parâmetros conforme descritos por Gonçalves e Resende (2015). Esses valores foram determinados a partir de experimentos realizados em diversos problemas.

Tabela 2 – Parâmetros BRKGA com valores recomendados

Parâmetros	Especificação	Valores recomendados
n_{pop}	Tamanho da população	n_{pop} = a x n onde $1 \leqslant a \in R$ é uma constante que depende de n comprimento de uma proposta de solução codificada pelo vetor de chaves aleatórias
n_e	Tamanho das soluções de elite	$0{,}10\ n_{pop} \leqslant n_e \leqslant 0{,}25\ n_{pop}$
n_m	Tamanho das soluções de mutantes	$0{,}10\ n_{pop} \leqslant n_m \leqslant 0{,}30\ n_{pop}$
ρ_e	Probabilidade de herdara informação da solução do pai elite	$0{,}50\ n_{pop} \leqslant \rho_e \leqslant 0{,}80\ n_{pop}$

Fonte: (GONçALVES; RESENDE, 2011)

Algoritmo 4: Pseudocódigo do *Brkga*

Entrada: P, P_e, P_m, ρ_e, *critério_de_parada*
Saída: Melhor Indivíduo S

```
1  início
2      Inicializa a População
3      Avalia a População
4      Ordena a População
5      para i ← 1 até critério_de_parada faça
6          Particione a População em dois Conjuntos: Elite (Pe) e Não-Elite (P - Pe)
7          Copie os Indivíduos do Conjunto Elite para a Próxima Geração
8          Gere o Conjunto de Mutantes Pm
9          para i ← 1 até P - Pe - Pm faça
10             Selecione um Indivíduo do Conjunto Elite;
11             Selecione um Indivíduo do Conjunto Não-Elite;
12             Realize o Cruzamento Uniforme Parametrizado;
13         fim
14         Atualiza a População
15         Avalia a População
16         Ordena a População
17     fim
18 fim
```

5 Algoritmos Propostos

Uma heurística construtiva é uma técnica de construção de soluções para problemas de otimização, onde a solução é desenvolvida passo a passo, adicionando elementos à solução parcial de acordo com um critério específico. Este tipo de heurística é utilizado para encontrar soluções aproximadas de forma eficiente, especialmente quando os mé- todos clássicos são muito lentos ou incapazes de encontrar uma solução exata devido à complexidade do problema.

Em ciência da computação e otimização matemática, heurísticas construtivas são frequentemente empregadas em problemas NP-difíceis, onde o objetivo é criar uma solução viável que possa ser posteriormente melhorada. Essas heurísticas são projetadas para ter um tempo de execução aceitável, embora possam não garantir uma solução ótima em todos os casos. Exemplos comuns incluem o método guloso e a construção aleatória parcial.

Neste trabalho, abordamos três heurísticas construtivas para a metaheurística BRKGA a fim de solucionar o problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD). A primeira heurística inicia a construção da solução através de um caminho central denominado *backbone*. A segunda heurística organiza os vértices em níveis, na quantidade $D/2$, para formar a árvore geradora, e por fim, a terceira, o qual constrói uma quantidade de caminhos mínimos que se conectam a um nó central ou aresta central, dependendo da paridade do diâmetro.

As três heurísticas desenvolvidas partem do princípio fundamental de que a geração de soluções viáveis com o diâmetro pré-estabelecido pode ser alcançada de maneira direta, sem a necessidade de verificações contínuas durante a construção das árvores. Essa abordagem inovadora permite que as árvores sejam formadas já respeitando as restrições impostas, resultando em uma redução significativa do custo computacional. A eliminação da necessidade de verificação a cada etapa não só simplifica o processo, mas também abre caminho para a aplicação eficiente dessas heurísticas em instâncias de maior complexidade, conforme será detalhado a seguir.

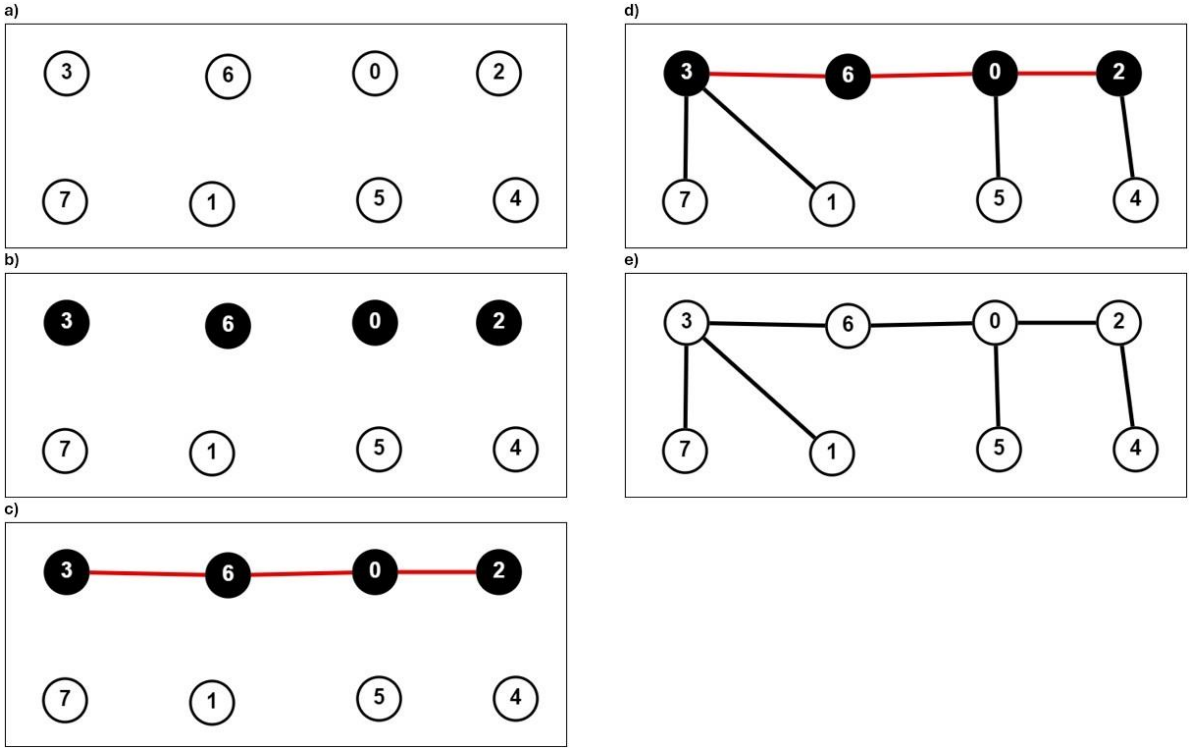
5.1 Heurística baseada em backbone

Nesta heurística construtiva, em vez de partir de uma árvore e repará-la, foi proposta uma construção baseada em caminhos. Na realidade, o diâmetro limita o tamanho dos caminhos em termos de quantidade de arestas que os compõem. Dessa forma, heurísticas baseadas em caminhos foram utilizadas para construir uma Árvore Geradora Mínima com Diâmetro Restrito (AGMRD).

No algoritmo construtivo, o primeiro passo é criar um vetor com o conjunto de vértices ou nós do grafo, distribuídos de forma aleatória. Para a construção de uma solução viável, esse vetor é interpretado da seguinte forma: Selecionamos os $D - 1$ nós para formar o *backbone* da solução, uma espécie de espinha dorsal do grafo. Após a seleção, os nós do *backbone* são conectados entre si. Tendo o *backbone* definido, os nós que ficaram de fora, se conectarão a um dos nós do *backbone* com a aresta de menor peso. Dessa maneira é garantido que o diâmetro da solução inicial não será violado.

Tomando como exemplo a Figura 11 d), considerando $D = 5$, sendo assim serão selecionados quatro nós par formar o *backbone*, ou seja, os 4 primeiros nós do vetor [3, 6, 0, 2]. Através da estrutura de dados utilizada (pode ser uma lista de arestas ou matriz de pesos), verifica-se com qual nó no *backbone*, cada nó fora dele se conecta com a aresta de menor peso.

Figura 11 – Heurística construtiva baseada em *backbone*.



Fonte: Autoria própria

O pseudocódigo da heurística construtiva de *backbone* pode ser visto em Algoritmo 5.

Algoritmo 5: HC-BACKBONE
Entrada: D, cV , $MATRIX$
Saída: *SOL, custo*

início
  $nV \leftarrow \text{len}(MATRIX)$;
  $backbone \leftarrow \emptyset$;
  $backbone\_nos \leftarrow \emptyset$;
  $backbone\_custo \leftarrow 0$;
  $individuo \leftarrow \emptyset$;
  $custo \leftarrow 0$;
  para $i \leftarrow 0$ ***até*** $D - 2$ **faça**
    $j \leftarrow i + 1$;
    $backbone\_nos \leftarrow backbone\_nos \cup \{cV[i]\}$;
    se $j \neq D - 1$ ***e*** $MATRIX[i][j] \neq 0$ **então**
      $backbone\_custo \leftarrow backbone\_custo + MATRIX[cV[i]][cV[j]]$;
      $backbone \leftarrow backbone \cup \{(cV[i], cV[j], MATRIX[cV[i]][cV[j]])\}$;
    fim
  fim
  se D *é PAR* **então**
    $centro \leftarrow \text{int}(\text{len}(backbone)/2) + 1$;
  fim
  senão
    $centro \leftarrow \text{int}((\text{len}(backbone) + 1)/2)$;
  fim
  $R \leftarrow nV - (D - 1)$;
  $nos\_restantes \leftarrow \emptyset$;
  para $restantes \leftarrow D - 1$ ***até*** $nV - 1$ **faça**
    $nos\_restantes \leftarrow nos\_restantes \cup \{cV[restantes]\}$;
  fim
  $infinito \leftarrow \infty$;
  $individuo \leftarrow \text{copy}(backbone)$;
  $u \leftarrow 0$;
  $v \leftarrow 0$;
  para $i \in nos\_restantes$ **faça**
    $menor \leftarrow \infty$;
    para $j \in backbone\_nos$ **faça**
      se $MATRIX[i][j] \neq 0$ ***e*** $MATRIX[i][j] < menor$ **então**
        $menor \leftarrow MATRIX[i][j]$;
        $u \leftarrow i$;
        $v \leftarrow j$;
      fim
    fim
    $individuo \leftarrow individuo \cup \{(u, v, menor)\}$;
  fim
  retorna *SOL, custo*;
fim

5.2 Heurística baseada em organização por Níveis ou camadas.

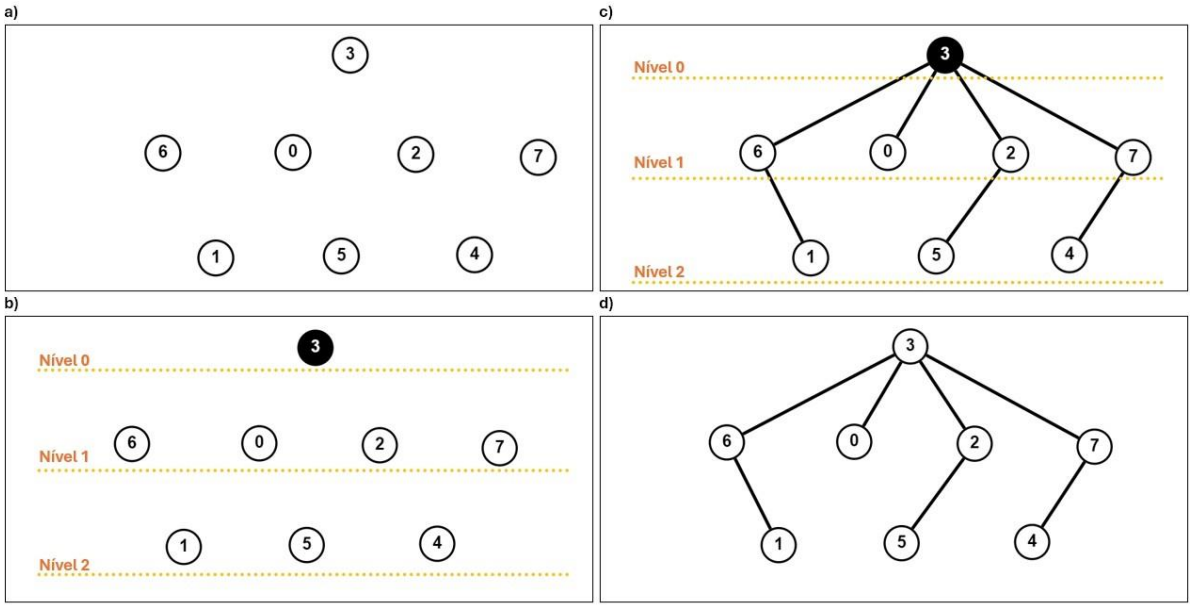
Este formato de organizar os nós do grafo em camadas foi apresentado por Gouveia et al. (2011) e também por Santos (2006) que foi utilizada em seu trabalho como parte da busca local dentro da temática métodos heurísticos e metaheurísticos. Com base nesse modelo, foi observado que seria uma forma inovadora aplicar essa estrutura como heurística construtiva para o problema da AGMRD com o BRKGA. A heurística constrói soluções viáveis organizando os nós, e construindo caminhos que se conectam ao centro da árvore, sendo que este terá um nó para quando o diâmetro for par, e uma aresta central, quando o diâmetro for ímpar. Em uma lista de níveis ou camadas, o nó ou nós do centro são adicionados à camada 0 (zero) da lista. Os demais níveis são criados de acordo com o valor D estabelecido, sendo $D/2$ quantidade de níveis criados. A quantidade de nós para cada nível é distribuída de forma equilibrada para cada nível, até que todos os nós do grafo estejam acomodados em um dos níveis. A alocação dos nós é feita forma aleatória. Com a lista de nós completa, o algoritmo a percorre, agora partindo do nível inferior até o nível 0, conectando cada nó do nível mais baixo a um nó do nível imediatamente superior com a aresta de menor custo. Usando essa heurística dentro do BRKGA, o vetor com os genes é distribuído em um vetor denominado lista de níveis. A quantidade de níveis criada é calculada dividindo-se $D/2$. Se houve resto é porque D é ímpar e assim são alocados dois nós no nível 0, ou nível raiz. Se D for par, então apenas um nó é alocado no nível 0.

Considerando para este exemplo, D = 4, teríamos a quantidade de 2 níveis, e apenas 1 nó no nível 0. A quantidade de nós alocados nos demais níveis é dividida de forma igualitária (quando possível), ou seja, para um grafo de 8 nós, teremos 1 nó no nível 0, e 7 nós distribuídos em 2 níveis, então o nível 1 ficará com 4 nós, e o nível 2 com 3 nós. Construída a lista de níveis, o algoritmo irá conectar cada nó do nível mais baixo a um nó do nível mais alto com uma aresta de menor custo, até chegar na raiz (nível 0).

O decodificador interpreta as chaves geradas e cria uma lista de arestas com a AGDR, assim obtém-se uma solução viável construída e com D em acordo com o determinado. A Figura 12 ilustra como fica alocado a lista de nós e a árvore gerada.

O pseudocódigo da heurística construtiva está apresentado em Algoritmo 6 e 7

Figura 12 – Heurística construtiva baseada em níveis.



Fonte: Autoria própria

Algoritmo 6: HC-NÍVEIS **Entrada**: D, cV, $MATRIZ$ **Saída**: SOL, $custo$

```
1  início
2      SOL ← ∅;
3      custo ← 0;
4      niveis ← dividir_niveis(D, cV)de nós em; se
       quantidade de nós em (niveis[0]) > 1 então
5          SOL ←
           SOL ∪ {(niveis[0][0], niveis[0][1], MATRIZ[niveis[0][0], niveis[0][1]])};
6          custo ← custo + MATRIZ[niveis[0][0], niveis[0][1]];
7      fim
8      para N ← 0 até quantidade de nós em(niveis) − 2 faça
9          para j ∈ niveis[N + 1] faça
10             menor ← ∞;
11             para i ∈ niveis[N] faça
12                 se MATRIZ[i, j] < menor então
13                     menor ← MATRIZ[i, j];
14                     u ← i;
15                     v ← j;
16                 fim
17             fim
18             SOL ← SOL ∪ {(u, v, menor)};
19             custo ← custo + menor;
20         fim
21     fim
22     retorna SOL, custo;
23 fim
```

Algoritmo 7: DIVIDIR NÍVEIS

```
Entrada: N, valores
Saída: niveis
1  início
2      nivel0 ← ∅;
3      se N é PAR então
4          nivel0 ← nivel0 ∪ {valores[0]};
5          valores_restantes ← valores[1:];
6          num_subconjuntos ← min(N/2, len(valores_restantes));
7      fim
8      senão
9          nivel0 ← nivel0 ∪ {valores[:2]};
10         valores_restantes ← valores[2:];
11         num_subconjuntos ← min((N−1)/2, len(valores_restantes));
12     fim
13     subconjuntos ← [∅ para _ em range(num_subconjuntos)];
14     para i, valor em enumerate(valores_restantes) faça
15         subconjuntos[i%num_subconjuntos] ←
           subconjuntos[i%num_subconjuntos] ∪ {valor};
16     fim
17     niveis ← subconjuntos;
18     niveis ← niveis ∪ {nivel0};
19     retorna niveis;
20 fim
```

5.3 Heurística construtiva de caminhos mínimos - HCCM

A HCCM é uma heurística construtiva desenvolvida para resolver o problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD). Ela foi implementada para construir uma árvore geradora que respeita um diâmetro pré-estabelecido, utilizando um conjunto de arestas de entrada do grafo. Esta função busca construir uma solução viável diretamente, sem a necessidade de verificações contínuas do diâmetro durante a construção da árvore.

O algoritmo começa inicializando variáveis essenciais, como o conjunto de nós a serem visitados e os níveis da árvore. A função seleciona um ponto de partida para a construção da árvore com base na paridade do diâmetro estabelecido. Se o diâmetro for ímpar, o algoritmo identifica a melhor aresta inicial para começar a árvore; se for par, escolhe um nó central que atuará como o ponto de origem da árvore. O tamanho dos sub-caminhos é calculado como sendo $(D/2) - 1$ arestas.

Tabela 3 – Matriz de pesos de um grafo completo de 10 vértices

0	4	3	4	6	3	1	11	4	8
4	0	7	5	2	4	3	5	7	1
3	7	0	7	13	13	1	6	6	8
4	5	7	0	4	12	1	12	11	12
6	2	13	4	0	2	8	7	11	10
3	4	13	12	2	0	8	5	8	12
1	3	1	1	8	8	0	12	13	11
11	5	6	12	7	5	12	0	3	1
4	7	6	11	11	8	13	3	0	12
8	1	8	12	10	12	11	1	12	0

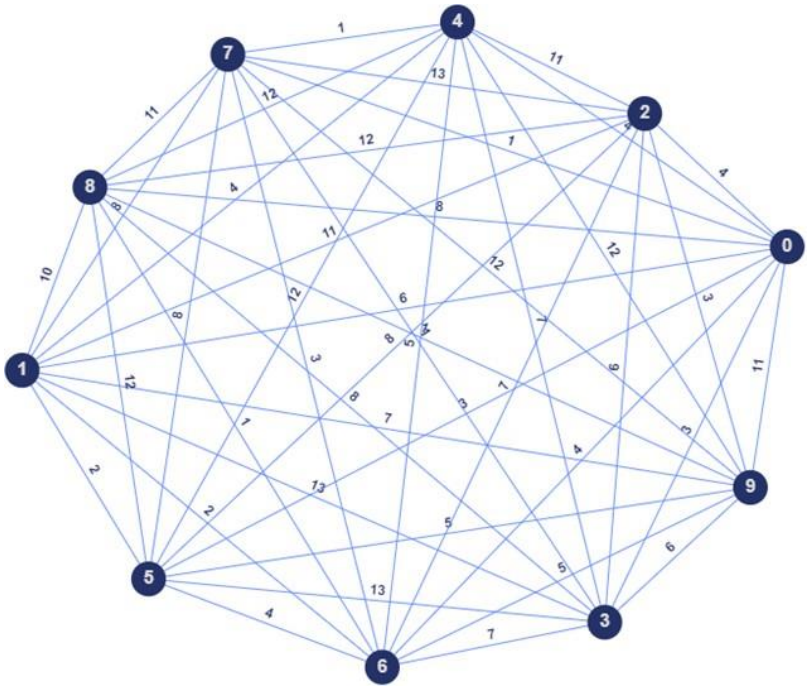
O algoritmo prossegue gerando sub-caminhos a partir dos nós selecionados no vetor de vértices. Esses sub-caminhos são construídos sem a necessidade de verificar o diâmetro a cada passo, pois o algoritmo já é projetado para garantir que o diâmetro definido seja respeitado. Cada sub-caminho é adicionado à solução parcial, com os nós sendo marcados como visitados.

Uma vez que os sub-caminhos são gerados, o algoritmo irá conectar cada um deles ao centro da árvore, com um dos nós de cada sub-caminho, utilizando as arestas de menor custo, assegurando que a árvore resultante seja otimizada em termos de custo.

Ao final, o algoritmo calcula o custo total da árvore gerada e organiza as informações em uma lista de arestas da árvore geradora resultante. O pseudocódigo dessa heurística construtiva é apresentado no Algoritmo 8.

Tomando como exemplo um grafo completo de 10 vértices cuja matriz de pesos está apresentada a seguir na Tabela 3 e uma respresentação visual na Figura 13.

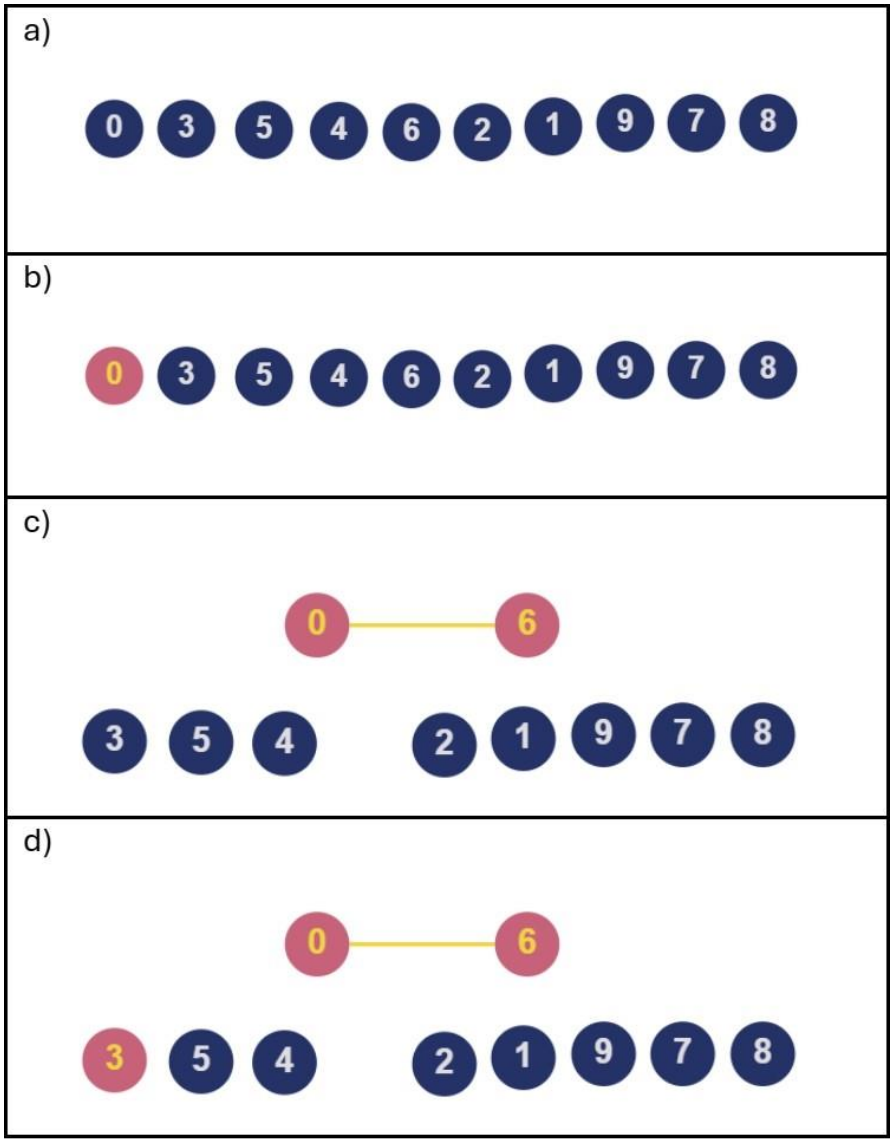
Figura 13 – Grafo completo de 10 vértices.



Fonte: Autoria própria

Inicialmente, como ilustrado na Figura 14(a), um vetor contendo todos os vértices do grafo é gerado de forma aleatória. Essa sequência sorteada será utilizada como base para a construção da solução. A figura 1 ilustra um exemplo de um vetor com os vértices de um grafo de 10 vértices (0 a 9).

Figura 14 – Construção da Árvore Geradora com Restrição de Diâmetro com HCCM.



Fonte: Autoria própria

Neste exemplo, o diâmetro determinado para o grafo é 7, o que indica que ele é ímpar. Sendo assim, o grafo terá uma aresta central. O vértice localizado na primeira posição do vetor de vértices será utilizado para iniciar a construção da ponte central.

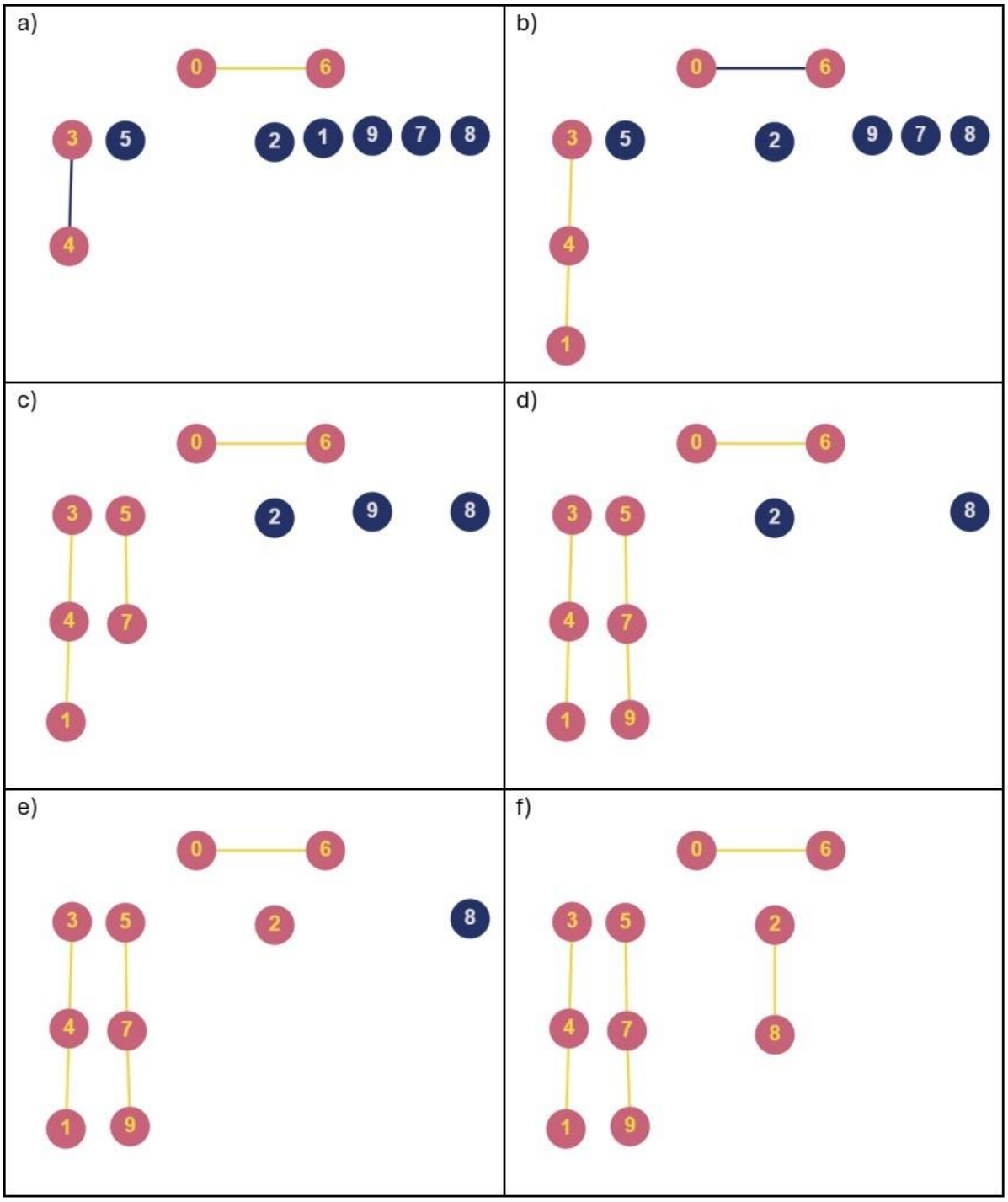
Verifica-se qual é a melhor conexão disponível para o vértice na posição 0. No caso deste exemplo, a melhor conexão é com o vértice 6. Dessa forma, os vértices 0 e 6 passam a fazer parte da lista de nós já visitados. Figura 14(b) e (c).

Após a conexão inicial, o algoritmo continua percorrendo o vetor de vértices. O próximo elemento na sequência é o vértice 3, que iniciará o primeiro subcaminho, Figura 14(d).

O tamanho dos subcaminhos é determinado pela fórmula $(D/2)1$. Com $D = 7$,

o cálculo resulta em 2,5. Este valor é arredondado para o inteiro menor, estabelecendo assim que o tamanho máximo de cada subcaminho será de 2 arestas. Verifica-se, então, qual é a melhor conexão disponível para o vértice 3. A melhor conexão encontrada é com o vértice 4, como mostrado na Figura 15(a).

Figura 15 – Construção da Árvore Geradora com Restrição de Diâmetro com HCCM.



Fonte: Autoria própria

Em seguida, verifica-se entre os vértices disponíveis qual possui a melhor conexão com o vértice 4. A melhor conexão é com o vértice 1. Dessa forma, o primeiro subcaminho é finalizado, respeitando o tamanho máximo de 2 arestas como ilustrado em Figura 15(b).

O algoritmo prossegue para iniciar o próximo subcaminho, percorrendo o vetor de vértices. Caso o vértice já tenha sido incluído na solução, o algoritmo avança para o próximo vértice na sequência. No exemplo, após o vértice 3, o próximo vértice disponível é o 5, que iniciará o novo subcaminho. A melhor conexão disponível para o vértice 5 é com o vértice 7, apresentado em Figura 15(c).

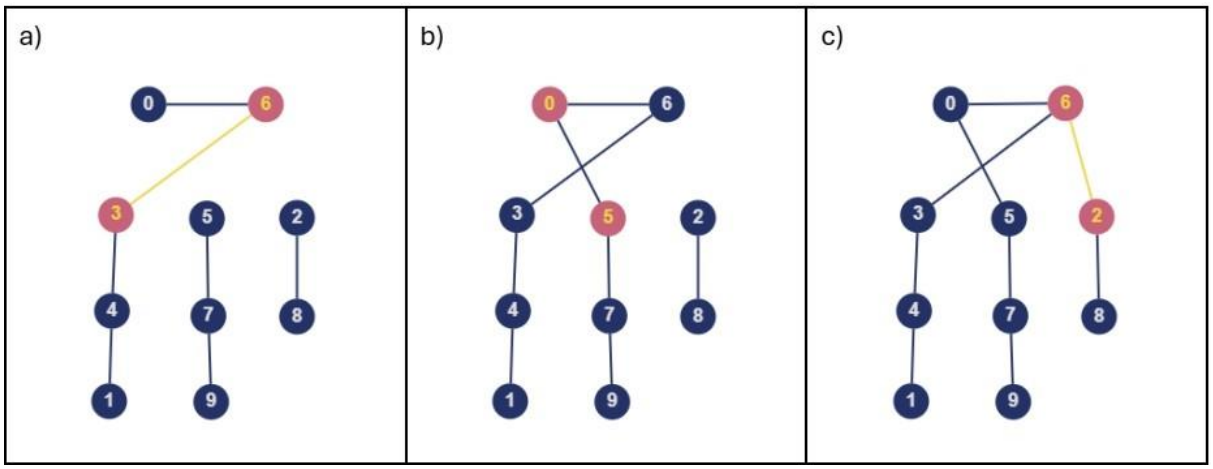
O algoritmo então verifica qual a melhor conexão disponível para o vértice 7. A conexão identificada como mais vantajosa é com o vértice 9, encerrando assim o segundo subcaminho, como mostrado em Figura 15(d).

O próximo elemento no vetor de vértices é o vértice 4, porém, como ele já faz parte da solução, o algoritmo continua percorrendo o vetor até encontrar um vértice disponível. Nesse caso, o vértice 2 é selecionado para iniciar um novo subcaminho.

O vértice 2 só possui uma conexão disponível, que é com o vértice 8, o último vértice ainda não visitado. Assim, o último subcaminho é finalizado, como ilustrado em Figura 15(e) e (f).

Após a construção dos subcaminhos, o algoritmo verifica em cada subcaminho qual é a melhor conexão possível com o centro do grafo. Essas conexões são então estabelecidas para formar a árvore final, como mostra a Figura 16(a), (b) e (c).

Figura 16 – Construção da Árvore Geradora com Restrição de Diâmetro com HCCM.

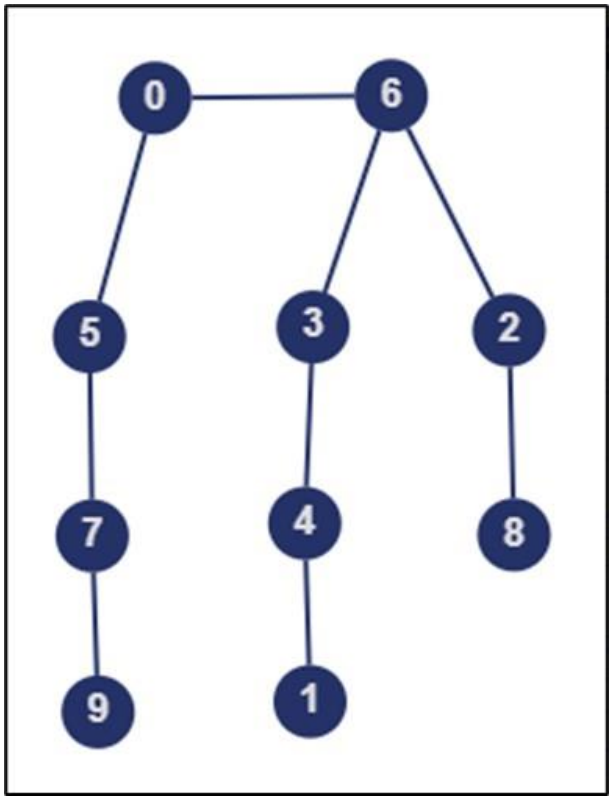


Fonte: Autoria própria

A árvore geradora com diâmetro restrito está, portanto, formada. Embora não haja garantias de que essa é a árvore geradora mínima, uma vez que outras combinações de subcaminhos e conexões poderiam ser exploradas, o diâmetro estabelecido foi mantido

conforme especificado. A seguir, é apresentada a figura da árvore reorganizada, como apresentada na Figura 17.

Figura 17 – Árvore Geradora com Diâmetro Limitado a D = 7 construida pelo algoritmo HCCM.



Fonte: Autoria própria

Algoritmo 8: HCCM: Heurística Construtiva de Caminhos Mínimos

Entrada: Lista de Arestas do Grafo, Diâmetro D
Saída: T

início
  $visitados \leftarrow \emptyset$; (Lista vazia de nós visitados)
  $a\_visitar \leftarrow$ Lista de Nós do Grafo;
  se D *é PAR* **então**
    $centro \leftarrow$ primeiro vértice de $a\_visitar$;
    $a\_visitar$.remover($centro$);
    $visitados$.inserir($centro$);
  fim
  senão
    $aresta\_central \leftarrow$ melhor aresta que conecta o primeiro nó de $a\_visitar$;
    T.incluir($aresta\_central$);
    $a\_visitar$.remover($nó\_1$, $nó\_2$);
    (Remove os dois nós conectados pela aresta central)
    $visitados$.inserir($nó\_1$, $nó\_2$);
  fim
  para $nó \in a\_visitar$ **faça**
    $subcaminho \leftarrow \emptyset$;
    para i *de* 0 *até tamanho de* $subcaminho < \frac{D}{2} - 1$ **faça**
      $melhor\_aresta \leftarrow$
        aresta de menor custo conectando $nó$ a um nó em $visitados$;
      $subcaminho$incluir($melhor\_aresta$);
      $nó \leftarrow$ nó conectado pela $melhor\_aresta$;
      $visitados$inserir($nó$);
      $a_visitar$remover($nó$);
    fim
    T.incluir($subcaminho$);
  fim
  para $subcaminho \in T$ **faça**
    $nó\_final \leftarrow$ último nó em $subcaminho$;
    $aresta\_conexao \leftarrow$
      aresta de menor custo do subcaminho que conecta ao centro;
    T.incluir($aresta\_conexao$);
  fim
  retorna T
fim

5.4 BRKGA - Decodificador

As heurísticas construtivas mencionadas são implementadas como decodificadores na metaheurística BRKGA. Ambas são geradas a partir de um vetor de chaves aleatórias (*random keys*) que, de acordo com o modelo implementado, realizam a decodificação dos dados necessários para uma solução viável do problema.

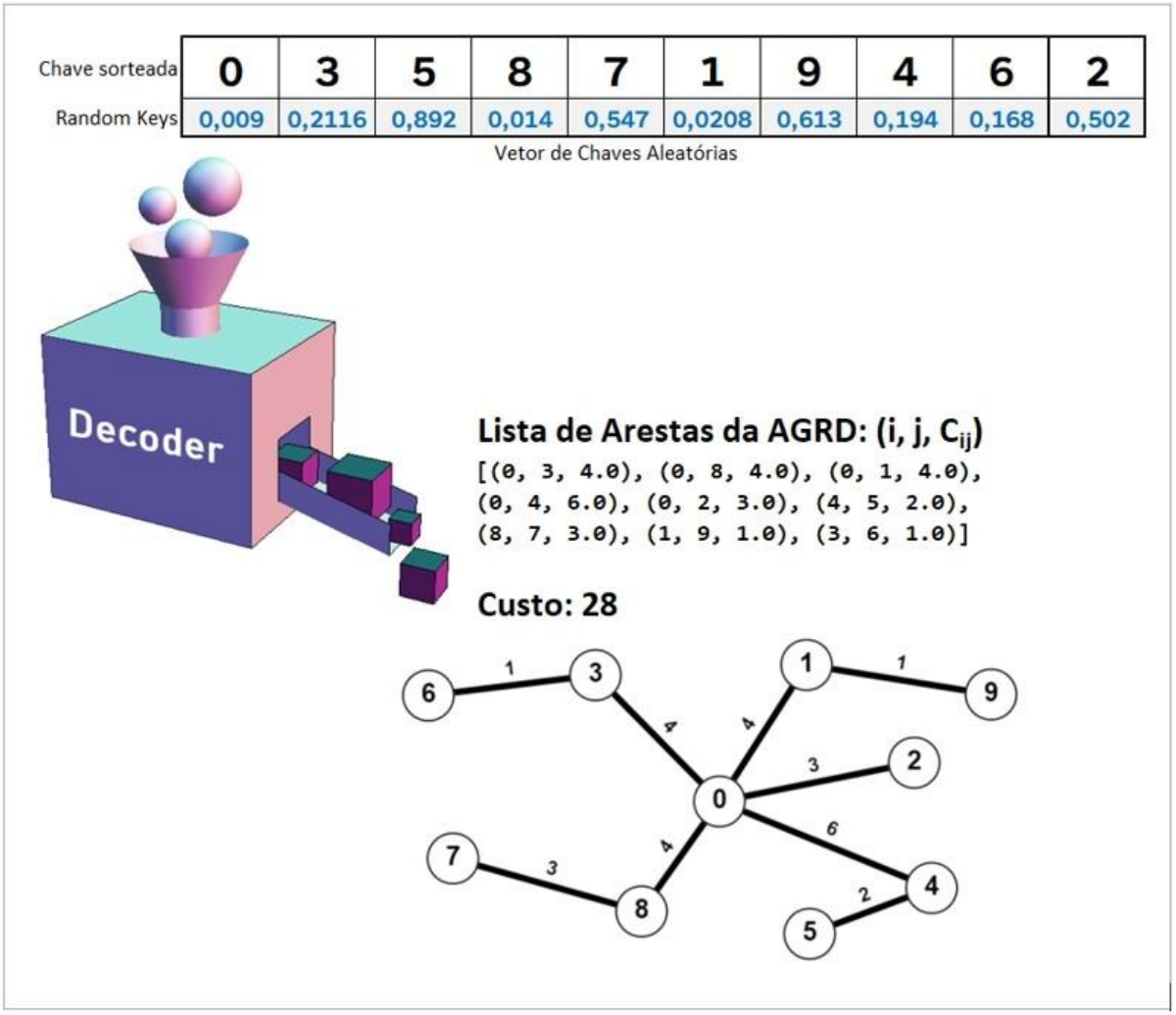
Conforme discutido, o BRKGA gera cromossomos nos quais cada índice do vetor é gerada aleatoriamente uma random key no intervalo [0,1], gerando finalmente um vetor de chaves aleatórias. Para avaliar a qualidade do cromossomo, é necessário realizar a decodificação. Este processo de decodificação consiste em um método heurístico que recebe como entrada um cromossomo composto por números aleatórios e, como saída, fornece uma solução viável do problema (exemplo na Figura 18).

Para o nosso algoritmo, a interpretação das chaves aleatórias foi realizada utilizando um vetor auxiliar associado ao cromossomo. A Figura 6 ilustra um cromossomo em seu estado natural, sem qualquer alteração, e um vetor de índices associado a ele. Nesse vetor,
o primeiro elemento do vetor de índices está ligado ao primeiro elemento do cromossomo,
o segundo elemento do vetor de índices está ligado ao segundo elemento do cromossomo, e assim por diante.

No fluxograma do BRKGA, o decoder é aplicado no momento da avaliação dos indivíduos, somente neste momento é que o BRKGA conhece o quanto vale aquela sequencia de chaves interpretadas pelo decodificador, fazendo uma associação de chave e custo.

Conforme Gonçalves et al.(2015) a principal vantagem do uso do decodificador em RKGAs é que ele garante a viabilidade da solução gerada, mesmo após a aplicação de operadores genéticos como cruzamento e mutação. Isso ocorre porque a ordenação das chaves sempre resultará em uma sequência válida de vértices do grafo.

Figura 18 – Exemplo de decodificação de chaves aleatórias para uma solução viável



5.4.1 Função Aptidão

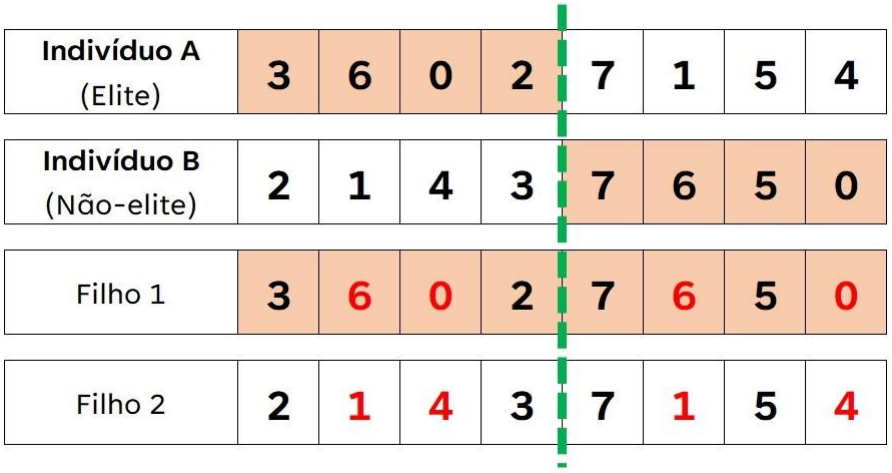
Para se ter uma solução viável para AGMRD, é preciso que todos nós sejam conectados por uma aresta formando uma árvore cujo diâmetro é estabelecido e que tal árvore tenha o menor custo. A avaliação é feita de forma simplificada e função *fitness* baseia-se em obter o menor custo como melhor resultado, penalizando o custo quando a solução não for uma árvore ou contiver ciclos.

5.4.2 Função de cruzamento

No BRKGA, o processo de cruzamento é dito como viciado por favorecer a herança genética dos indivíduos mais aptos, aumentando a probabilidade de gerar soluções de alta qualidade, onde um gene é copiado do pai elite com uma alta probabilidade, garantindo que características vantajosas sejam preservadas na população (GONçALVES; RESENDE, 2011). Nos problemas relacionados a grafos, especialmente árvores geradoras, o cruzamento

tradicional frequentemente resulta em soluções inviáveis devido à duplicação de nós após o cruzamento. A Figura 19 ilustra um cruzamento usando diretamente os valores das chaves com um corte de cruzamento no meio do vetor, ou seja ρ_e = 0,50.

Figura 19 – Cruzamento tradicional



Fonte: Autoria própria. *Crossover* inválido. Gera repetição de nós.

Para sanar esse problema, o BRKGA usa um cruzamento parametrizado (PUC – *Parametric Uniform Crossover*), que utiliza chaves em vez de nós diretamente. Essa abordagem garante que, independentemente da quantidade de genes a serem cruzados, os filhos gerados sejam sempre soluções válidas. A Figura 20 ilustra este processo.

O operador de cruzamento uniforme parametrizado é aplicado, pegando um indiví- duo da Elite e um indivíduo Não-Elite. Sendo ρ_e a probabilidade de herança de um gene de um indivíduo elite, o novo indivíduo herda alelos do primeiro pai com probabilidade ρ_e e com $(1 - \rho_e)$ do segundo. A seleção dos pais é determinística: dado que todos os indivíduos são previamente classificados, um pai (de ambos os tipos) é selecionado sequencialmente. Portanto, é possível que um indivíduo da Elite cruze com vários indivíduos Não-Elite.

A Figura 20 mostra como o operador de cruzamento com ρ_e = 0,7 produz um novo indivíduo que herda mais chaves do pai Elite do que do Não-Elite.

5.4.3 Função de mutação

A mutação no BRKGA é realizada gerando novos cromossomos aleatórios, conheci- dos como mutantes, que são inseridos na população para manter a diversidade genética e evitar a estagnação precoce. Este processo foi detalhado por Gonçalves e Resende [2011], onde uma fração da população é substituída por mutantes gerados aleatoriamente em cada geração, promovendo a exploração de novas áreas do espaço de soluções. No algoritmo o percentual de mutação ρ_m é informado, no qual refletirá na quantidade de genes que serão modificados aleatoriamente. Para o problema de árvores, uma quantidade de genes

Figura 20 – Cruzamento Uniforme Parametrizado - PUC

Fonte: Autoria própria. *Crossover* válido - (PUC). Não gera repetição de nós.

é trocada de posição de forma aleatória de acordo com o percentual ρ_m de mutação do indivíduo, resultando sempre em indivíduos válidos, ou seja, sem repetição de vértices. Para que haja modificações mais sutis, recomenda-se valores entre 0 e 0,10 para ρ_m gerando poucas modificações no indivíduo que sofre a mutação, mantendo características mais próximas do indivíduo original, já para valores de ρ_e mais altos o indivíduo poderá sofrer mudanças mais significativas, o que é bom para fugir de ótimos locais.

5.4.4 BRKGA+

Neste trabalho também foi implementada uma versão modificada do BRKGA, denominada BRKGA+ afim de obter melhores resultados com número menor de itera- ções. Observado que nas operações de mutação e cruzamento, em alguns casos gerava-se indivíduos piorados, implementou-se uma variação onde executando-se um número n de tentativas, a saída das sub-rotinas sempre entrega indivíduos melhores, ou no pior caso, igual aos de entrada, mas nunca piores. O Algoritmo 20 apresenta o pseudocódigo do

BRKGA com as alterações mencionadas nas linhas 7 e 13, com a adição de repetições com uma quantidade de tentativas atribuída à variável.

Algoritmo 9: Pseudocódigo do BRKGa MELHORado ou BRKGA+

Entrada: P, P_e, P_m, ρ_e, *critério_de_parada*, *tentativas*
Saída: Melhor Indivíduo S

```
1  início
2      Inicializa a População;
3      Avalia e Ordena a População;
4      para i ← 1 até critério_de_parada faça
5          Particione a População em dois Conjuntos: Elite (Pe) e Não-Elite (P - Pe);
6          Copie os Indivíduos do Conjunto Elite para a Próxima Geração;
7          para i ← 1 até tentaticas faça
8              Gere o Conjunto de Mutantes "sempre melhores"P;m
9          fim
10         para i ← 1 até P - Pe - Pm faça
11             Selecione um Indivíduo do Conjunto Elite;
12             Selecione um Indivíduo do Conjunto Não-Elite;
13             para i ← 1 até tentaticas faça
14                 Realize o Cruzamento Uniforme Parametrizado "sempre melhores";
15             fim
16         fim
17         Atualiza a População;
18         Avalia e Ordena a População;
19     fim
20 fim
```

5.5 Buscas Locais

A busca local é uma técnica de otimização que explora soluções vizinhas para encontrar uma solução satisfatória para um problema de otimização. A principal ideia da busca local é começar com uma solução inicial e iterativamente mover-se para uma solução vizinha que melhora o valor da função objetivo. O processo continua até que nenhum vizinho melhore a solução atual, indicando que um ótimo local foi encontrado. Métodos comuns de busca local incluem algoritmos como *hill climbing*, *simulated annealing* e *tabu search*, cada um com estratégias diferentes para escapar de ótimos locais e explorar o espaço de soluções (BLUM; ROLI, 2003).

No contexto do problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD), a busca local pode ser aplicada para encontrar uma árvore que minimize o custo total das arestas enquanto satisfaz a restrição de diâmetro. A aplicação da busca local no AGMRD envolve gerar uma árvore inicial, neste trabalho usa-se OTT, HCCM ou BRKGA para tal, e então iterativamente modificar a árvore, como adicionar e remover arestas,

para melhorar a solução. Cada modificação deve garantir que a restrição de diâmetro seja mantida. Essas técnicas permitem explorar eficientemente o espaço de soluções e encontrar árvores que são próximas do ótimo, mesmo para instâncias de problema grandes e complexas.

Após a realização de diversos testes, observou-se que ambas as heurísticas constru- tivas apresentaram um ponto de estagnação, incapazes de evoluir além de ótimos locais. A primeira heurística, baseada no *backbone*, mostrou uma estagnação devido ao espaço de busca na construção da árvore ser bastante limitado, uma vez que fica restrito ao *backbone*. Por outro lado, a segunda heurística, que organiza os vértices em níveis, oferece um espaço de busca significativamente maior ao construir a árvore de solução, permitindo uma melhor evolução com o BRKGA.

O Método de Descida em Vizinhança Variável, VND - *Variable Neighborhood Descent*, é uma técnica de melhoria de solução que pertencente à família dos métodos de busca local, criada por Pierre Hansen e Nenad Mladenović (1997). Sua origem está ligada ao desenvolvimento de metaheurísticas que exploram diferentes estruturas de vizinhança para escapar de ótimos locais e melhorar a qualidade das soluções obtidas. O principal objetivo do VND é sistematicamente explorar múltiplas vizinhanças de uma solução inicial, realizando uma busca local em cada uma delas até que não seja possível encontrar melhorias adicionais. Este processo permite uma exploração mais robusta do espaço de soluções, aumentando a probabilidade de identificar soluções de alta qualidade.

As vantagens do VND incluem sua simplicidade e eficiência em explorar o espaço de busca de maneira diversificada, além de sua capacidade de combinar bem com outras metaheurísticas, melhorando a eficácia e a robustez das abordagens de solução em problemas complexos de otimização (HANSEN; MLADENOVIć, 1997). O Pseudo-código da VND é apresentado no 10.

A escolha de diferentes estruturas de vizinhança permite explorar diferentes regiões do espaço de soluções de forma sistemática. Ao aplicar essas operações de maneira iterativa e controlada, é possível melhorar a solução inicial, encontrando árvores com menor custo que ainda respeitam a restrição de diâmetro. A eficácia da busca local em encontrar boas soluções depende da definição apropriada dessas vizinhanças e da eficiência com que são exploradas.

Neste trabalho utilizou-se de quatro algoritmos de busca em vizinhança para aperfeiçoamento das soluções entregues pelos algoritmos construtivos.

5.5.1 Vizinhança N1

A busca local N1 para o problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD) visa melhorar soluções dentro de uma vizinhança específica de

Algoritmo 10: VND - Busca em Descida por Vizinhança Variável

Entrada: Solução inicial S, Conjunto de Estruturas de Vizinhança N, Número máximo de iterações max_iter
Saída: Solução S^*

```
1  início
2      S* ← S;
3      k ← 1;
4      iter ← 0;
5      k ≤ |N| e iter < max_iter Gerar uma solução S' ∈ Nk(S*);
6      se f(S') < f(S*) então
7          S* ← S';
8          k ← 1;
9      fim
10     senão
11         k ← k + 1;
12     fim
13     iter ← iter + 1;
14     retorna S*;
15 fim
```

uma solução inicial. O objetivo é encontrar uma solução melhor dentro de um "passo"na vizinhança da solução atual, minimizando o custo total das arestas e mantendo o diâmetro dentro do limite definido. Baseada na busca 1-opt, utilizada por SANTOS (2006).

O algoritmo começa obtendo uma árvore como solução inicial a partir da lista de arestas fornecida. Cada aresta é representada como uma tupla de nós i, j. Em seguida, o algoritmo identifica o nó com aresta de maior valor a partir de lista de níveis, que organiza os nós por níveis de profundidade. Utilizando esse valor, um vetor de profundidade p é inicializado com -1 para cada nó, representando inicialmente que nenhum nó possui profundidade definida.

A seguir, o algoritmo atribui profundidades reais aos nós com base na lista de níveis. Cada nó recebe uma profundidade correspondente ao seu nível na árvore, que é armazenada no vetor p.

Com a profundidade dos nós definida, o algoritmo calcula o último nível permitido, que é metade do valor do diâmetro fornecido, arredondado para baixo. A partir daqui, o algoritmo começa a explorar possíveis melhorias na solução inicial. Para cada nó i na árvore, o algoritmo verifica se o nó está no último nível. Se o nó estiver no último nível, ele é considerado uma folha e não possui filhos. O algoritmo então identifica a aresta conectada a este nó, encontrando o valor adjacente e o índice dessa aresta na solução inicial. Em seguida, para cada outro nó j, o algoritmo verifica se a aresta que conecta j a i possui um custo menor do que a aresta atual conectada a i. Se uma aresta com custo menor for encontrada e j estiver em um nível inferior ao último nível, a aresta original é removida

da solução inicial e substituída pela nova aresta. A profundidade de i é atualizada para ser um nível acima da profundidade de j.

Se o nó i não estiver no último nível, o algoritmo realiza um processo semelhante. Ele identifica a aresta conectada a i e verifica para cada nó j se uma aresta de custo menor pode ser encontrada. Se i for uma folha (grau 1) e j estiver em um nível inferior ao último nível, a aresta é trocada e a profundidade de i é atualizada.

Durante essas verificações, se uma melhoria for encontrada (ou seja, se uma aresta com menor custo for adicionada), a solução é atualizada e a função retorna a nova árvore com pesos adicionados e a profundidade dos nós. Se nenhuma melhoria for encontrada durante as iterações, o algoritmo adiciona pesos à árvore final e agrupa os nós por profundidade. A árvore otimizada e a profundidade dos nós são então retornadas. A Figura 21 ilustra o movimento de troca de arestas N1.

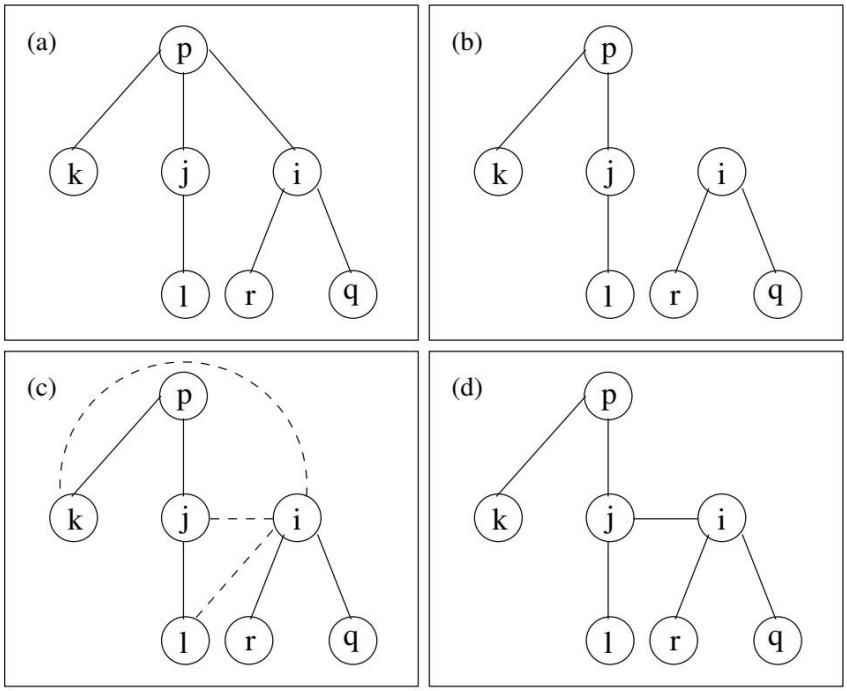


Figura 21 – Vizinhança N1
Fonte: (SANTOS, 2006)

A busca N1 possui ótimo desempenho computacional, porém pode ficar presa em ótimos locais, sendo frequentemente combinada com outras estratégias para escapar desses pontos e encontrar soluções próximas ao ótimo global.

5.5.2 Vizinhança N2

A busca local N2 é baseada na busca denominada "Adoção de sub-árvore"apresentada por SANTOS (2006). O algoritmo é projetado para otimizar a troca de uma subárvore dentro de um grafo, minimizando o custo total da árvore enquanto respeita uma restrição

de diâmetro. Ele começa inicializando a solução inicial com a lista de arestas da árvore inicial. A seguir, o algoritmo configura um vetor de profundidade, p, que armazena a profundidade de cada nó na árvore. A profundidade é determinada com base em outra lista, lista de níveis fornecida, que organiza os nós por seus níveis. Para cada nó na árvore, o algoritmo define sua profundidade no vetor p. Com a profundidade inicializada, o algoritmo calcula o custo total da solução inicial, além disso, o algoritmo calcula o último nível permitido, que é metade do valor do diâmetro fornecido, arredondado para baixo. O próximo passo é a exploração e melhoria da solução inicial. O algoritmo itera sobre todos os nós da árvore. Para cada nó, ele verifica se o nó tem mais de uma aresta conectada (grau maior que 1) e se sua profundidade é maior que zero. Se essas condições forem atendidas, o algoritmo prossegue para criar uma lista de adjacência da solução inicial e obter todos os descendentes do nó atual. Para cada descendente, o algoritmo cria cópias da solução inicial e do vetor de profundidade. Essas cópias são usadas para reverter as alterações, se necessário. Em seguida, o algoritmo tenta melhorar a solução atual trocando e invertendo as arestas entre o nó atual e o descendente. Se a nova solução tiver um custo menor do que a solução atual, o algoritmo atualiza a solução inicial e o vetor de profundidade. A troca das profundidades entre o nó atual e o descendente também é realizada para refletir a nova estrutura da árvore.

Se a nova solução não for melhor, o algoritmo restaura as cópias da solução inicial e do vetor de profundidade, continuando a busca por melhorias. Esse processo de troca e avaliação continua até que todas as possíveis trocas tenham sido exploradas.

Por fim, se nenhuma melhoria adicional puder ser encontrada, o algoritmo adiciona pesos ao grafo com base na solução final e agrupa os nós por profundidade. A árvore otimizada e as profundidades dos nós são então retornadas como resultado final. A Figura 22 ilustra os passos da movimentação N2.

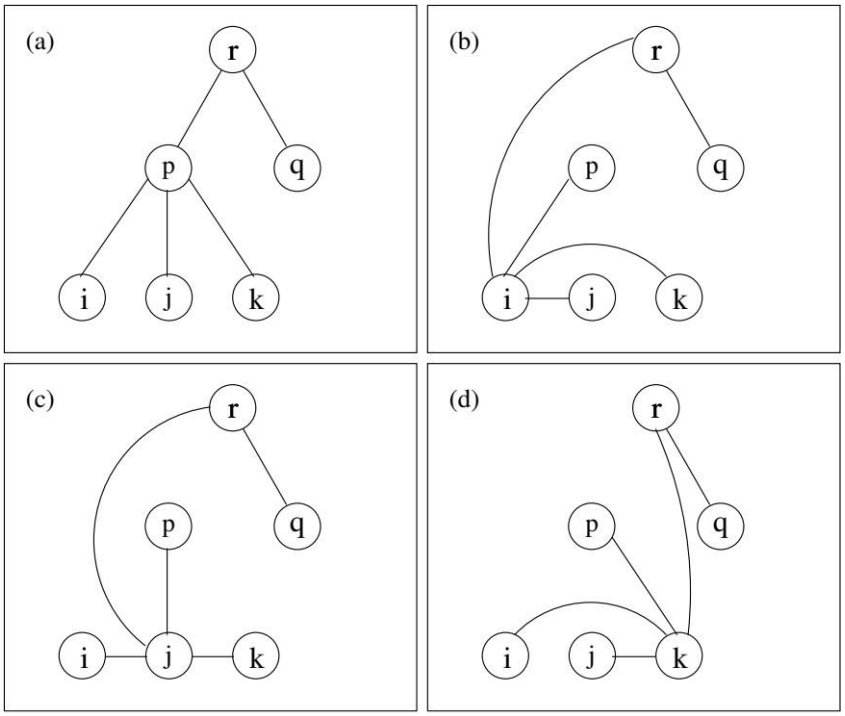


Figura 22 – Vizinhança N2
Fonte: (SANTOS, 2006)

5.5.3 Vizinhança N3

Essa busca local, visa realizar a substituição de caminhos, trocando arestas pai de uma sub-árvore até um dos filhos da raiz dessa sub-árvore.

O algoritmo recebe uma árvore como solução inicial. Em seguida, ele identifica o nó conectado a aresta de maior valor a partir de lista de níveis. que é uma lista aninhada, onde cada sub-lista contém nós em um determinado nível de profundidade na árvore. Com base nisso, um vetor de profundidade p é criado, inicialmente preenchido com -1 para indicar que nenhum nó tem profundidade atribuída. Em seguida, o algoritmo atribui profundidades reais aos nós usando a lista de niveis. Para cada nível de profundidade, ele percorre a sub-lista correspondente e define a profundidade de cada nó no vetor p.

Depois de configurar a profundidade dos nós, o algoritmo calcula o último nível permitido, que é metade do valor do diâmetro fornecido, arredondado para baixo. Ele também cria uma lista de adjacência a partir da solução inicial para facilitar a manipulação das conexões entre os nós.

Se o diâmetro fornecido for par, ele define um nó raiz e uma segunda raiz como o primeiro nó da solução inicial. Se o diâmetro for ímpar, ele define raiz como o primeiro nó da solução inicial e raiz secundária como o segundo nó.

A seguir, o algoritmo de forma iterativa, começa a explorar possíveis melhorias na solução inicial. Para cada nó i na árvore, ele verifica se o nó não é uma das raízes (raiz

ou raiz secundária). Se i não for uma raiz e tiver grau maior que 2, ou seja, tem mais de duas arestas conectadas a ele, o algoritmo verifica se a profundidade de i + 1, mais a profundidade de i é menor ou igual ao último nível.

Se essas condições forem atendidas, o algoritmo identifica os filhos diretos do nó i. Para cada filho direto, ele identifica as arestas conectadas ao nó i e ao filho direto. Em seguida, ele verifica se a troca dessas arestas pode resultar em uma solução de menor custo. Se sim, a troca é realizada, a solução é atualizada e as profundidades dos nós são ajustadas conforme necessário. Durante essas verificações, se uma melhoria for encontrada, a solução é atualizada e a função retorna a nova árvore com pesos adicionados e a profundidade dos nós. Caso contrário, o algoritmo adiciona pesos à árvore final e agrupa os nós por profundidade. A árvore otimizada e a profundidade dos nós são então retornadas.

A Figura 23 ilustra os passos da movimentação N3.

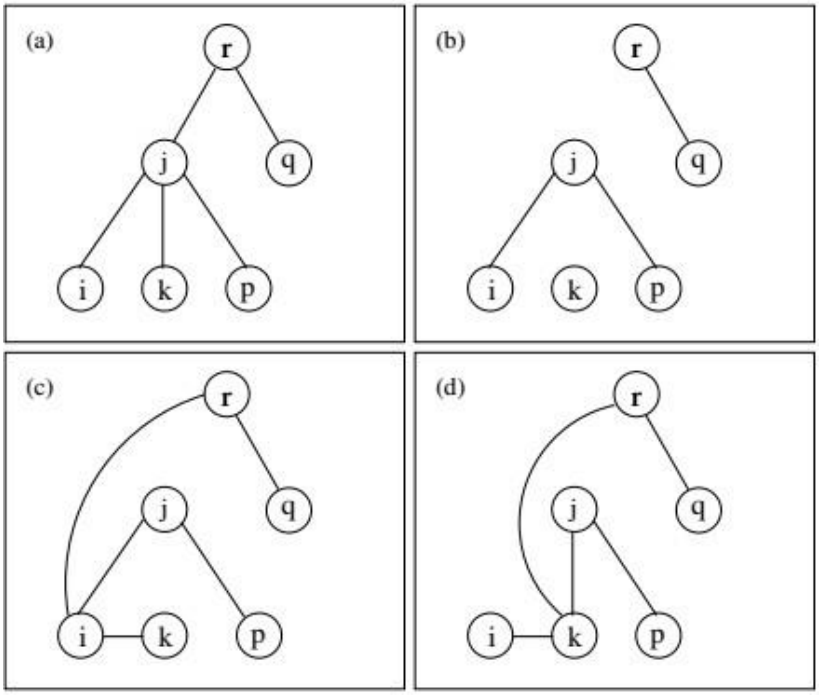


Figura 23 – Vizinhança N3
Fonte: (SANTOS, 2006)

5.5.4 Vizinhança N4

A busca local N4 é baseada no "2-opt"envolve a remoção de duas arestas de uma solução atual e a reconexão dos dois segmentos resultantes de uma maneira diferente, de modo a reduzir o custo total ou melhorar a qualidade da solução. Esse processo é repetido de forma exaustiva até que nenhuma melhoria adicional possa ser encontrada, resultando em uma solução localmente ótima.

O seu funcionamento começa com uma solução inicial, que pode ser gerada de

maneira aleatória ou através de um método heurístico. Após a inicialização o algoritmo faz a exploração da Vizinhança: Para cada par de arestas, as duas arestas são removidas e os segmentos resultantes são reconectados de forma diferente e ao final faz avaliação de melhorias. Se a nova configuração resultar em uma solução de menor custo ou de melhor qualidade, ela substitui a solução atual. Este processo é repetido até que todas as possíveis trocas de arestas tenham sido exploradas e nenhuma melhoria adicional possa ser encontrada.

Essa busca local é uma técnica de intensificação que se concentra em melhorar uma solução local através de pequenas mudanças iterativas. Quando combinada com metaheurísticas, a 2-opt pode intensificar a busca em regiões promissoras identificadas pelas metaheurísticas, refinando as soluções e encontrando ótimos locais de alta qualidade.

A Figura 24 ilustra os passos da movimentação N4.

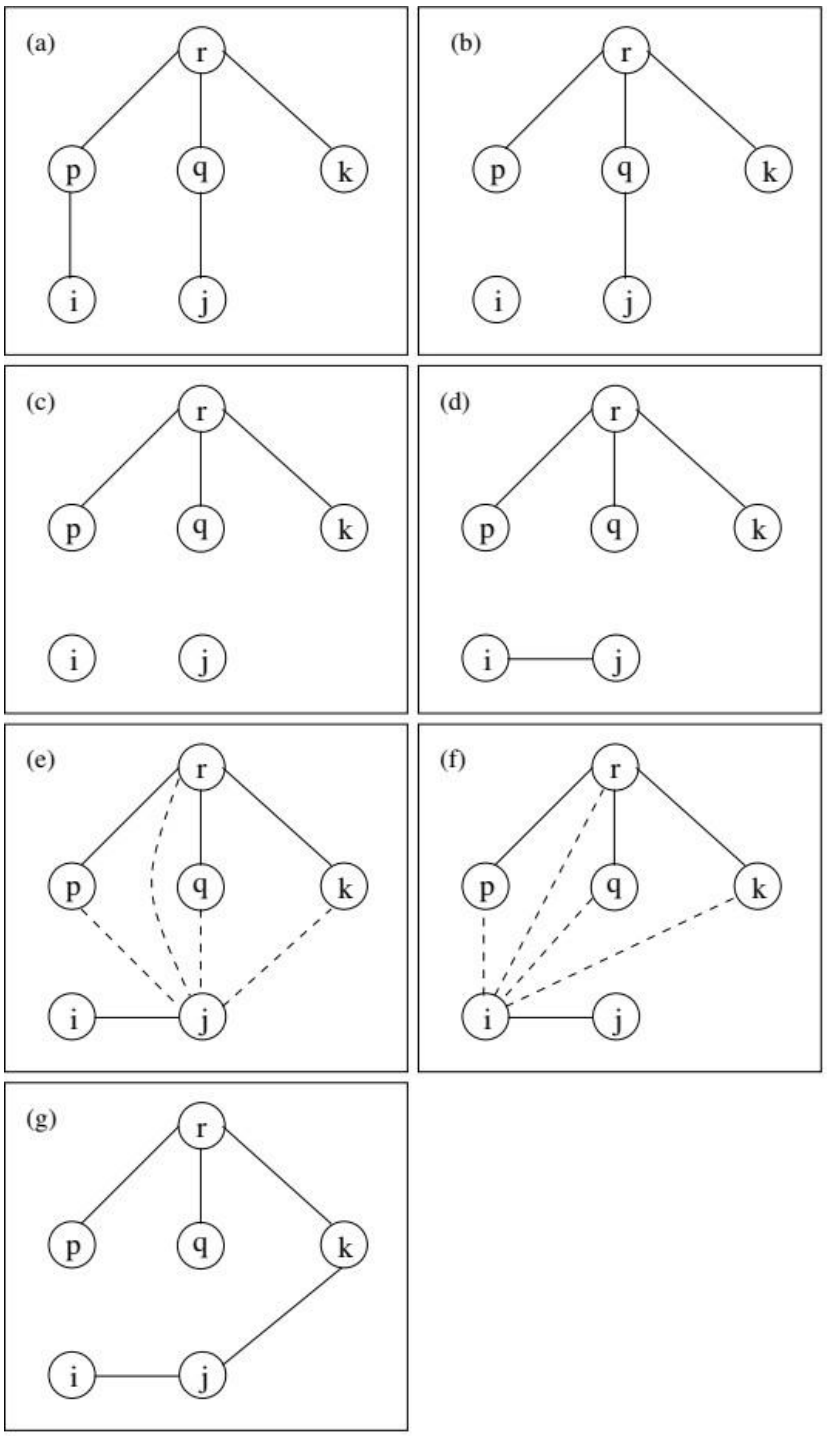


Figura 24 – Vizinhança N4
Fonte: (SANTOS, 2006)

6 Experimentos Computacionais

Os experimentos computacionais foram realizados em um conjunto de instâncias euclidianas de 50 a 500 vértices conhecidos na literatura, como nos trabalhos de Raidl e Julstrom (2003), Santos (2006), Nghia (2007) e Santos et al. (2014). Os algoritmos foram implementados em linguagem Python v3.11 usando a IDE *Microsoft Visual Studio Code* e *Jupyter*, com execução em um computador com processador Core i7, 7ª geração com 20GB de memória RAM, sistema operacional *Windows* 11 64bits.

6.1 Resultados obtidos

Esta seção apresenta os resultados obtidos pelos algoritmos para as instâncias citadas. Embora as fontes não definam, a aplicação do BRKGA tem uma relação direta entre o número de gerações e o tamanho do problema, a experiência demonstrou que problemas maiores podem exigir mais gerações para convergir para uma solução de boa qualidade. Isso ocorre porque, em problemas maiores, o espaço de busca de soluções é mais amplo, demandando mais tempo para que o algoritmo explore as diferentes regiões e encontre soluções melhores (GONçALVES; RESENDE, 2011). Em relação ao tamanho da população, sugere-se que seja proporcional ao tamanho do cromossomo, sendo pelo menos igual a n, onde n é o número de genes no cromossomo. Por esse motivo foi calculado o número de gerações e tamanho da população de forma proporcional conforme apresentados na Tabela 4.

Tabela 4 – Parâmetros para BRKGA

Quant. Vértices n	**População n**	**Gerações n x 3,2**
50	50	160
70	70	224
100	100	320
250	250	800
500	500	1600

Cada algoritmo foi executado 20 vezes para cada instância do conjunto de instâncias com parâmetros calculados proporcionalmente ao número de vértices. O critério de parada adotado foi de 0, 25 vezes a quantidade de gerações para iterações sem evolução na solução. Na Tabela 5 são apresentados os resultados dos algoritmos e também no Gráfico da Figura 28.

6.1.1 Análise da Convergência do BRKGA

A análise da convergência do algoritmo BRKGA revelou questões importantes sobre a minimização dos resultados. Inicialmente, observou-se que a escolha incorreta do conjunto de parâmetros impedia qualquer progressão significativa nos resultados gerados pelo BRKGA. Essa configuração inadequada dos parâmetros resultava em estagnação, onde as soluções não apresentavam melhorias ao longo das iterações.

Após o ajuste e a correta configuração dos parâmetros, uma mudança significativa foi observada. Os resultados começaram a convergir de maneira consistente para soluções de qualidade superior. Essa convergência foi notada em instâncias de diferentes tamanhos, demonstrando a robustez e a eficácia do BRKGA quando corretamente parametrizado. Isso pode ser observado no gráfico da Figura 25, 26 e 27.

Figura 25 – Convergência do BRKGA - instância 50 vértices

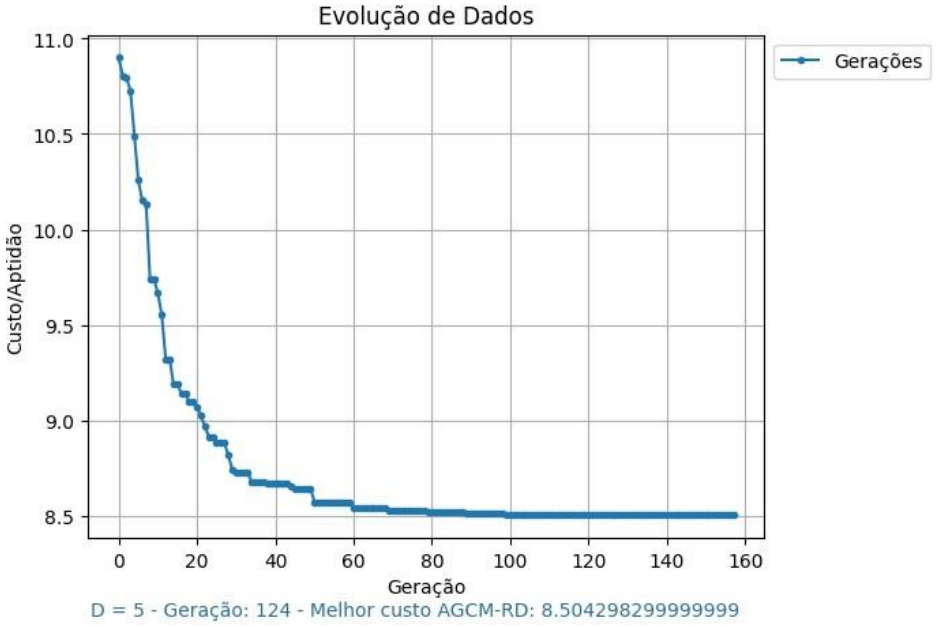
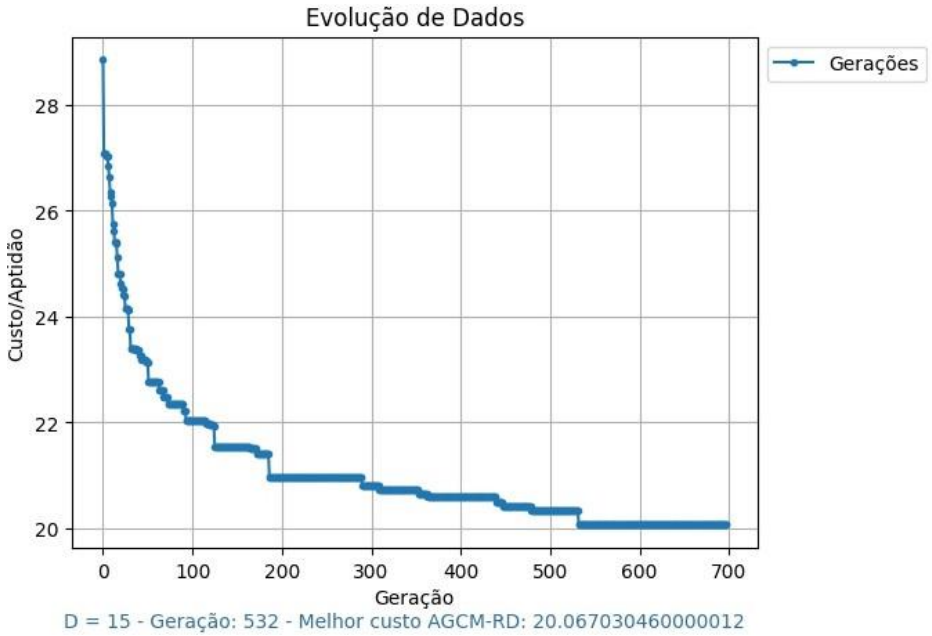


Figura 26 – Convergência do BRKGA - instância 250 vértices



A correta parametrização permitiu ao BRKGA explorar o espaço de soluções de forma mais eficiente, evitando os problemas de estagnação e permitindo uma progressão contínua na qualidade das soluções. A convergência semelhante para instâncias de vários tamanhos indica que o algoritmo, uma vez ajustado, tem um comportamento previsível e confiável, independentemente da escala do problema.

6.1.2 VND

A VND foi aplicada nos melhores resultados obtidos pelos algoritmos que geraram AGRD, no caso BRKGA+ com a heurítica de níveis em seu decodificador. Durante a execução da VND, foi observado que nas quatro vizinhanças, denominadas N1, N2, N3 e N4 dentro da VND. Os resultados obtidos mostraram variações na eficácia das diferentes movimentações em relação ao tamanho das instâncias. As movimentações N1 e N2 demonstraram uma performance consistente ao serem executadas em um maior número de iterações. Essas movimentações foram eficazes em melhorar os resultados para instâncias de todos os tamanhos, proporcionando uma redução significativa nos custos das árvores geradoras mínimas com restrição de diâmetro. Por outro lado, as movimentações N3 e N4 mostraram-se mais eficazes em instâncias de maior porte, especificamente para grafos com 250 e 500 nós. Embora essas movimentações tenham sido executadas em um número menor de iterações, elas ainda conseguiram proporcionar algumas melhorias nas soluções, indicando seu potencial para otimizar casos mais complexos e de maior dimensão.

Figura 27 – Gráfico comparativo da evolução do BRKGA e suas variantes

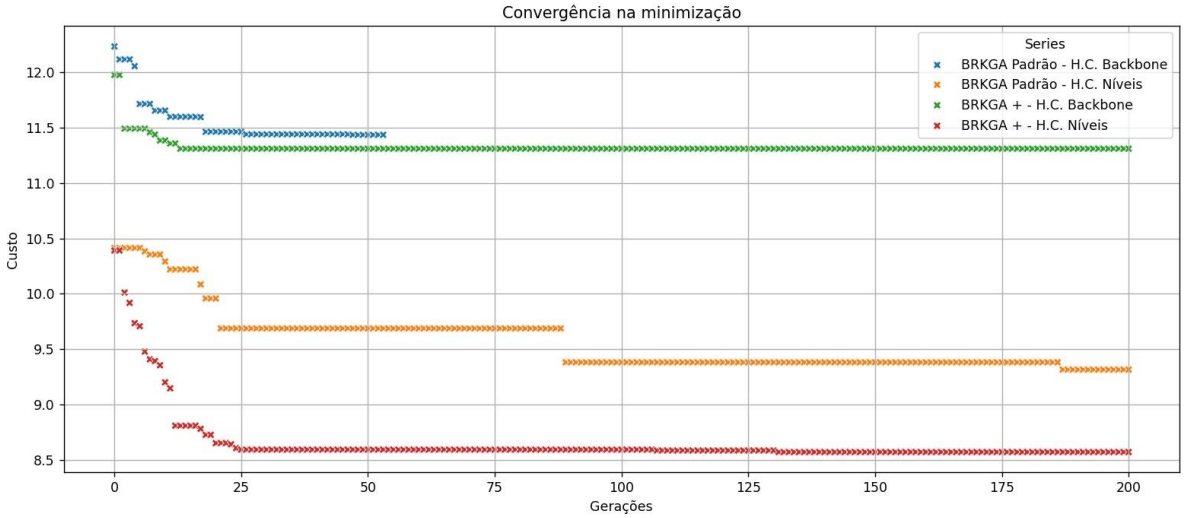


Figura 28 – Gráfico com resultados alcançados pelos algoritmos

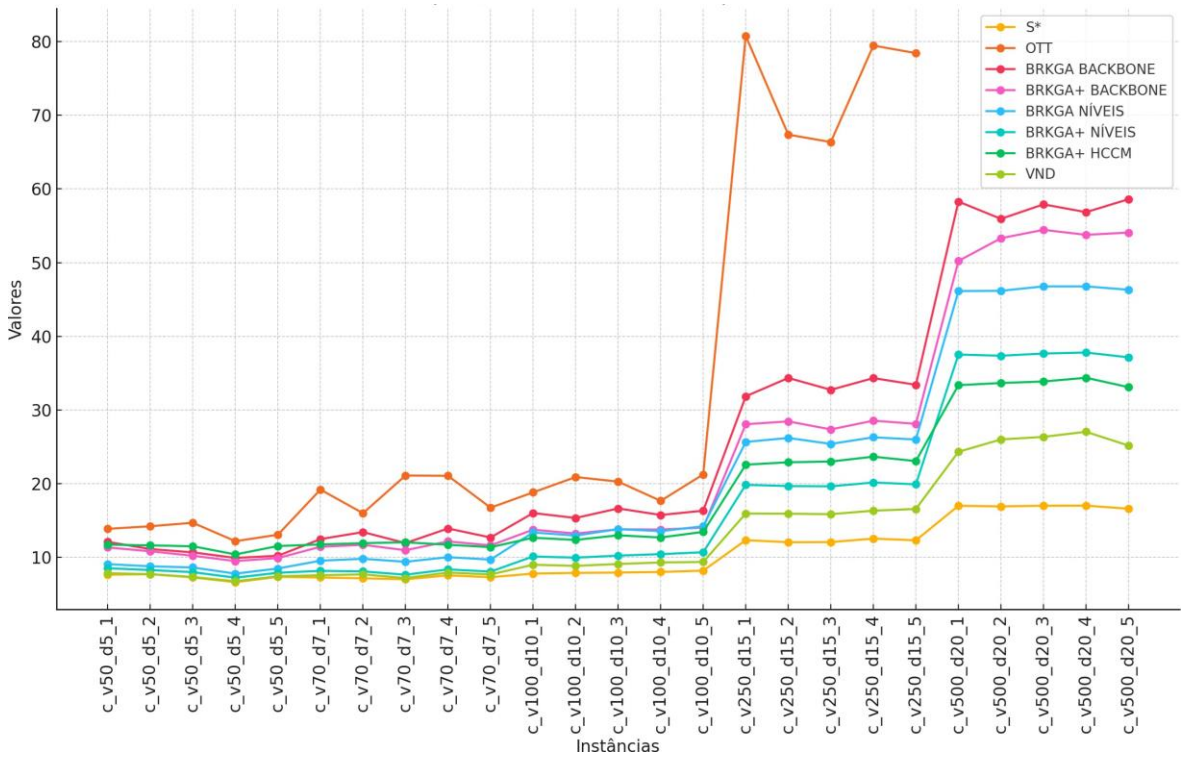


Tabela 5 – Resultados alcançados

Instâncias	**S***	OTT	BRKGA	BRKGA+ BACKBONE	BRKGA NÍVEIS	BRKGA+ NÍVEIS	BRKGA+ HCCM	VND	perc.
$c_v50_d5_1$	**7,60**	13,84	12,07	11,309	9,04	8,490	11,69	**7,81**	0,03
$c_v50_d5_2$	**7,68**	14,19	11,09	10,787	8,75	8,242	11,59	**7,68**	0,00
$c_v50_d5_3$	**7,24**	14,66	10,64	10,185	8,59	7,953	11,47	**7,30**	0,01
$c_v50_d5_4$	**6,59**	12,16	9,89	9,459	7,75	7,204	10,36	**6,71**	0,02
$c_v50_d5_5$	**7,32**	13,04	10,18	9,868	8,45	7,873	11,50	**7,39**	0,01
$c_v70_d7_1$	**7,23**	19,17	12,44	11,398	9,51	8,133	11,72	**7,54**	0,04
$c_v70_d7_2$	**7,12**	15,96	13,40	11,670	9,77	8,056	11,89	**7,65**	0,07
$c_v70_d7_3$	**6,99**	21,08	11,90	10,933	9,36	7,605	12,01	**7,15**	0,02
$c_v70_d7_4$	**7,52**	21,04	13,89	12,129	9,99	8,330	11,68	**7,90**	0,05
$c_v70_d7_5$	**7,27**	16,71	12,66	11,587	9,66	8,028	11,35	**7,64**	0,05
$c_v100_d10_1$	**7,76**	18,79	15,98	13,721	13,35	10,084	12,63	**8,96**	0,15
$c_v100_d10_2$	**7,85**	20,87	15,31	13,198	12,92	9,927	12,33	**8,81**	0,12
$c_v100_d10_3$	**7,90**	20,25	16,61	13,776	13,80	10,202	12,96	**9,06**	0,15
$c_v100_d10_4$	**7,98**	17,64	15,72	13,737	13,52	10,401	12,66	**9,27**	0,16
$c_v100_d10_5$	**8,16**	21,23	16,30	14,007	14,19	10,658	13,43	**9,34**	0,14
$c_v250_d15_1$	**12,30**	80,79	31,84	28,045	25,64	19,815	22,55	**15,92**	0,29
$c_v250_d15_2$	**12,02**	67,38	34,33	28,433	26,18	19,651	22,88	**15,89**	0,32
$c_v250_d15_3$	**12,04**	66,37	32,72	27,342	25,37	19,610	22,98	**15,83**	0,31
$c_v250_d15_4$	**12,51**	79,48	34,31	28,527	26,27	20,135	23,64	**16,30**	0,30
$c_v250_d15_5$	**12,28**	78,45	33,40	28,099	25,97	19,886	23,04	**16,54**	0,35
$c_v500_d20_1$	**16,97**	*	58,29	50,222	46,13	37,512	33,353	**24,30**	0,43
$c_v500_d20_2$	**16,88**	*	55,94	53,310	46,17	37,347	33,645	**25,98**	0,54
$c_v500_d20_3$	**16,98**	*	57,91	54,450	46,77	37,650	33,848	**26,33**	0,55
$c_v500_d20_4$	**16,99**	*	56,85	53,770	46,77	37,784	34,356	**27,02**	0,59
$c_v500_d20_5$	**16,57**	*	58,61	54,080	46,31	37,129	33,079	**25,14**	0,52

7 Conclusões e Trabalhos Futuros

Neste trabalho, foram desenvolvidas três heurísticas construtivas para o decodifica- dor do BRKGA visando resolver o problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD), abordagens estas ainda não combinadas na literatura existente. Heurística construtiva baseada em *backbone*, heurística construtiva baseada em níveis e heurística construtiva baseada em caminhos mínimos. As heurísticas demonstraram bons resultados durante os testes de execução, no entanto, todas apresentaram uma estagnação em um certo ponto.

A heurística construtiva baseada em *backbone* obteve resultados iniciais satisfatórios, mas sua estagnação deveu-se ao seu formato de construção. Esse método assume que existe um caminho mínimo ao qual todos os nós se conectam, o que não reflete a verdadeira diversidade estrutural das árvores. Esse formato rígido limita a capacidade da heurística de explorar soluções alternativas e potencialmente mais eficientes, porém só foi possível detectar esse comportamento após vários testes exaustivos.

Da mesma forma, a heurística construtiva baseada em níveis se mostrou mais eficiente chegando a resultados com custos baixos, porém também mostrou limitações estruturais. Ao dividir a construção em níveis com distribuição igual, a heurística ficou restrita a buscar soluções dentro desse formato específico, o que também impediu uma evolução nos resultados. A construção e procura de soluções limitadas a essa estrutura igualitária impedem a heurística de alcançar resultados mais eficientes.

A Heurística Construtiva de Caminhos Mínimos (HCCM) foi desenvolvida para gerar soluções iniciais eficientes para o problema da Árvore Geradora Mínima com Restrição de Diâmetro (AGMRD). Em sua execução de forma independente, a HCCM demonstrou um desempenho altamente satisfatório, especialmente para grandes instâncias do problema. Sua característica autônoma permite a construção rápida de soluções de qualidade, sem a necessidade de integração com outras meta-heurísticas, como o BRKGA (Biased Random- Key Genetic Algorithm). No entanto, quando combinada com o BRKGA, a HCCM apresentou um desempenho inferior, o que pode ser explicado pela menor flexibilidade no refinamento das soluções ao longo das iterações do algoritmo genético. Apesar disso, os resultados obtidos de forma independente evidenciam o potencial da HCCM, e com ajustes e refinamentos adicionais, em versões futuras, essa heurística poderá alcançar ainda melhores resultados e ser aplicada de maneira mais robusta em diferentes cenários.

Portanto, a principal conclusão deste trabalho é a necessidade de desenvolver heurísticas mais flexíveis, que permitam a construção de grafos em formatos variados. Uma abordagem genérica para a construção de grafos pode superar as limitações observadas

nas heurísticas atuais, proporcionando melhores resultados para o problema das árvores geradoras mínimas com restrição de diâmetro. A flexibilidade no modelo da árvore criada é essencial para explorar plenamente o espaço de soluções e encontrar soluções mais otimizadas.

A construção de soluções sem verificação iterativa de violação do diâmetro, também contribui significativamente em relação ao desempenho dos algoritmos, o que viabiliza construções de soluções viáveis de forma muito rápida, principalmente em se tratando de instâncias de grande porte.

Essas conclusões apontam para futuras direções de pesquisa, onde o foco deve ser na criação de heurísticas que não sejam limitadas por formatos estruturais rígidos, mas que possam adaptar-se dinamicamente à complexidade e diversidade das árvores geradoras mínimas.

Os resultados positivos do BRKGA estão intimamente ligados à qualidade das heurísticas construtivas utilizadas em seu decodificador. O espaço de busca é composto por todas as possíveis combinações de chaves aleatórias, e a exploração eficiente desse espaço é fundamental para encontrar soluções de alta qualidade. Esta diferença ficou evidente ao comparar as duas heurísticas, uma com espaço de busca mais restrito e outra com maior diversificação.

Além das heurísticas construtivas, o trabalho também abordou a utilização de buscas locais para aprimorar as soluções obtidas. Foram implementadas quatro movimentações de busca local denominadas N1, N2, N3 e N4. Essas movimentações são baseadas em propriedades específicas das árvores: N1 utiliza a troca 1-opt, N2 foca na adoção de subárvores, N3 busca e substituição de caminhos e N4 emprega a troca de 2-opt.

A escolha por movimentações de busca local baseadas em propriedades intrínsecas das árvores mostrou-se acertada, pois todas as três estratégias conseguiram reduzir significativamente o custo das árvores geradoras com restrição de diâmetro. Os resultados demonstraram uma melhoria consistente nas soluções iniciais, evidenciando a eficácia das movimentações propostas.

No entanto, apesar dos bons resultados obtidos, ainda não foi possível alcançar o resultado ótimo desejado, principalmente nas instâncias maiores. As limitações observadas indicam que, embora as buscas locais tenham potencial para refinar as soluções, elas ainda enfrentam desafios em otimizar completamente o custo das árvores dentro das restrições de diâmetro impostas.

Em suma, as buscas locais implementadas neste trabalho mostraram-se eficazes na redução de custos, mas evidenciaram a necessidade de estratégias adicionais ou complementares para atingir a otimização completa. Esses achados apontam para a importância de continuar investigando novas movimentações e combinações de técnicas de busca local,

bem como a integração com outras abordagens heurísticas para melhorar ainda mais os resultados obtidos no problema das árvores geradoras mínimas com restrição de diâmetro.

A principal contribuição deste trabalho é a demonstração de que o BRKGA é uma ferramenta extremamente eficaz quando são utilizados algoritmos eficientes em seu decodificador, capazes de explorar um espaço de busca abrangente, resultando em soluções de alta qualidade dentro de um tempo computacional satisfatório. Para trabalhos futuros, sugere-se o uso de outras metaheurísticas em conjunto com as heurísticas construtivas implementadas neste estudo. Além disso, um maior incremento no espaço de busca utilizando o BRKGA pode oferecer melhorias significativas. Outra linha promissora de pesquisa envolve o uso de algoritmos que aproveitem o processamento paralelo, utilizando GPUs ou tecnologias similares, para aumentar a eficiência computacional.

Referências

ALANIS, M.; CARRETERO, J.; XHAFA, F. Optimization of smart grid infrastructures using diameter-constrained minimum spanning tree algorithms. *Journal of Parallel and Distributed Computing*, Elsevier, v. 120, p. 1–15, 2018. Citado na página 13.

ANAGNOSTOPOULOS, A.; BECCHETTI, L.; CASTILLO, C.; GIONIS, A.; LEONARDI, S. Efficient algorithms for team formation with a leader in social networks. *The Journal of Supercomputing*, Springer, v. 62, n. 2, p. 807–823, 2012. Citado na página 13.

AZEVEDO, A. F. de. *Uma abordagem híbrida para o problema de programação de horários de cursos universitários*. 2021. Citado na página 15.

BEAN, J. C. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing, INFORMS, v. 6, n. 2, p. 154–160*, 1994. Citado na página 29.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, ACM, v. 35, n. 3, p. 268–308, 2003. Citado 2 vezes nas páginas 28 e 53.

CORMEN, T. *Algoritmos: Teoria e Prática*. 3ª edição. ed. Grupo GEN, 2012. ISBN 9788595158092. Disponível em: <https://app.minhabiblioteca.com.br/#/books/ 9788595158092>. Citado na página 12.

DEO, N.; ABDALLA, A. Computing a Diameter-Constrained Minimum Spanning Tree in Parallel. In: GOOS, G.; HARTMANIS, J.; LEEUWEN, J. V.; BONGIOVANNI, G.; PETRESCHI, R.; GAMBOSI, G. (Ed.). *Algorithms and Complexity*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000. v. 1767, p. 17–31. ISBN 978-3-540-67159-6 978-3-540-46521-8. Series Title: Lecture Notes in Computer Science. Disponível em: <http://link.springer.com/10.1007/3-540-46521-9_2>. Citado na página 19.

FAKHRAVAR, H. Combining heuristics and Exact Algorithms: A Review. 2023. Citado na página 20.

FERNANDES, A. M. R. *Inteligência Artificial: noções gerais*. [S.l.]: Florianópolis, 2 edição, 2005. Citado na página 28.

GONÇALVES, J. F.; RESENDE, M. G. A biased random-key genetic algorithm for the unequal area facility layout problem. *European Journal of Operational Research*, v. 246, n. 1, p. 86–107, out. 2015. ISSN 03772217. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S0377221715003227>. Citado 3 vezes nas páginas 29, 33 e 49.

GONçALVES, J. F.; RESENDE, M. G. C. Biased random-key genetic algorithms for combinatorial optimization. journal of heuristics. *Springer, v. 17, n. 5, p. 487–525*, v. 17, n. 5, p. 487–525, out. 2011. ISSN 1381-1231, 1572-9397. Disponível em: <http://link.springer.com/10.1007/s10732-010-9143-1>. Citado 6 vezes nas páginas 27, 31, 33, 34, 50 e 62.

GOUVEIA, L.; SIMONETTI, L.; UCHOA, E. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as steiner tree problems over layered graphs. *Mathematical Programming*, Springer, v. 129, p. 123–147, 2011. Citado 2 vezes nas páginas 23 e 38.

HANSEN, P.; MLADENOVIć, N. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, Elsevier, v. 130, n. 3, p. 449–467, 1997. Citado na página 54.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, 1992. ISBN 978-0-262-27555-2. Disponível em: <https://direct.mit.edu/books/book/2574/ adaptation-in-natural-and-artificial-systemsan>. Citado na página 28.

MAINIERI, G. B. *Meta-heurística BRKGA aplicada a um problema de programação de tarefas no ambiente flowshop híbrido. Tese de Doutorado*. Tese (Doutorado) — Universidade de São Paulo, 2014. Citado na página 32.

MAJEED, A.; RAUF, I. Graph theory: A comprehensive survey about graph theory applications in computer science and social networks. *Inventions*, MDPI, v. 5, n. 1, p. 10, 2020. Citado na página 13.

NETTO, S. J. P. O. B. *Grafos: introdução e prática*. [S.l.]: Editora Blucher, 2017. ISBN 9788521211327. Citado na página 12.

NGHIA, N. D.; BINH, H. T. T. New recombination operator in genetic algorithm for solving the bounded diameter minimum spanning tree problem. In: *Proceedings of the IEEE Congress on Evolutionary Computation*. [S.l.]: IEEE, 2007. p. 108–113. Citado 2 vezes nas páginas 23 e 62.

NICOLETTI, M. d. C. *Fundamentos da Teoria dos Grafos para Computação*. 3. ed. Grupo GEN, 2017. ISBN 978-85-216-3477-5. Disponível em: <https://app.minhabiblioteca.com.br/#/books/9788521634775/>. Citado 2 vezes nas páginas 11 e 12.

NORONHA, T. F.; SANTOS, A. C.; RIBEIRO, C. C. Constraint Programming for the Diameter Constrained Minimum Spanning Tree Problem. *Electronic Notes in Discrete Mathematics*, v. 30, p. 93–98, fev. 2008. ISSN 15710653. Disponível em: <https://linkinghub.elsevier.com/retrieve/pii/S1571065308000188>. Citado na página 27.

OSMAN I. H.; KELLY, J. P. Meta-heuristics: an overview. meta-heuristics. *Springer, p.1–21*, 1996. Citado na página 27.

RAIDL, G. R.; JULSTROM, B. A. Greedy heuristics and an evolutionary algorithm for the bounded-diameter minimum spanning tree problem. *Proceedings of the 2003 ACM Symposium on Applied Computing*, p. 1021–1025, 2003. Citado 2 vezes nas páginas 22 e 62.

RICARDO, L. *Algoritmos genéticos : uma importante ferramenta da inteligência computacional*. [S.l.]: 2ª edição Rio de Janeiro: Brasport, 2008. ISBN 9788574523736. Citado na página 28.

SANTOS, A. C. *Modelos e Algoritmos para o Problema da Árvore Geradora de Custo Mínimo com Restrição de Diâmetro*. Tese (Tese de doutorado) — PUC-RIO, Rio de Janeiro, jun. 2006. Citado 12 vezes nas páginas 12, 17, 18, 22, 25, 38, 55, 56, 58, 59, 61 e 62.

SANTOS, A. C.; LIMA, D. R.; ALOISE, D. J. Modeling and solving the bi-objective minimum diameter-cost spanning tree problem. *Journal of Global Optimization*, v. 60, n. 2, p. 195–216, out. 2014. ISSN 0925-5001, 1573-2916. Disponível em: <http://link.springer.com/10.1007/s10898-013-0124-4>. Citado 3 vezes nas páginas 15, 16 e 62.

SHARMA, S.; TRIVEDI, R. Iot-enabled optimized smart home automation: Survey, challenges, and future research directions. *Journal of Network and Computer Applications*, Elsevier, v. 157, p. 102423, 2020. Citado na página 13.

SILVA, R.; RESENDE, M.; PARDALOS, P.; GONÇALVES, J. Biased random-key genetic algorithm for bound-constrained global optimization. In: *GOW 2012: Proceedings of Global Optimization Workshop*. [S.l.: s.n.], 2012. Citado na página 32.

SOUSA, E. G. de; ALOISE, D. J.; SANTOS, A. C. META-HEURÍSTICAS PARA O PROBLEMA BIOBJETIVO DA ÁRVORE GERADORA DE CUSTO E DIÂMETRO MÍNIMOS. p. 11. Citado na página 19.

SPEARS, W. M.; JONG, K. D. D. *On the Virtues of Parameterized Uniform Crossover,:*. Fort Belvoir, VA, 1995. Disponível em: <http://www.dtic.mil/docs/citations/ ADA293985>. Citado na página 30.

WEST, D. B. *Introduction to graph theory*. [S.l.]: Prentice Hall, 2001. Citado na página 11.

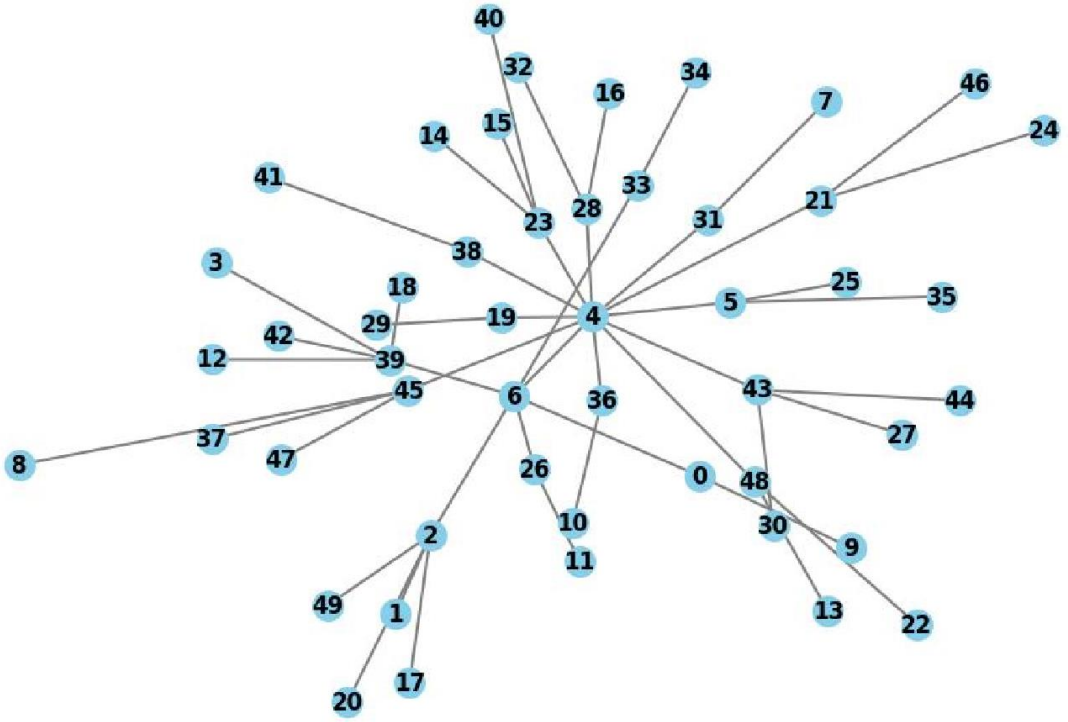
Apêndices

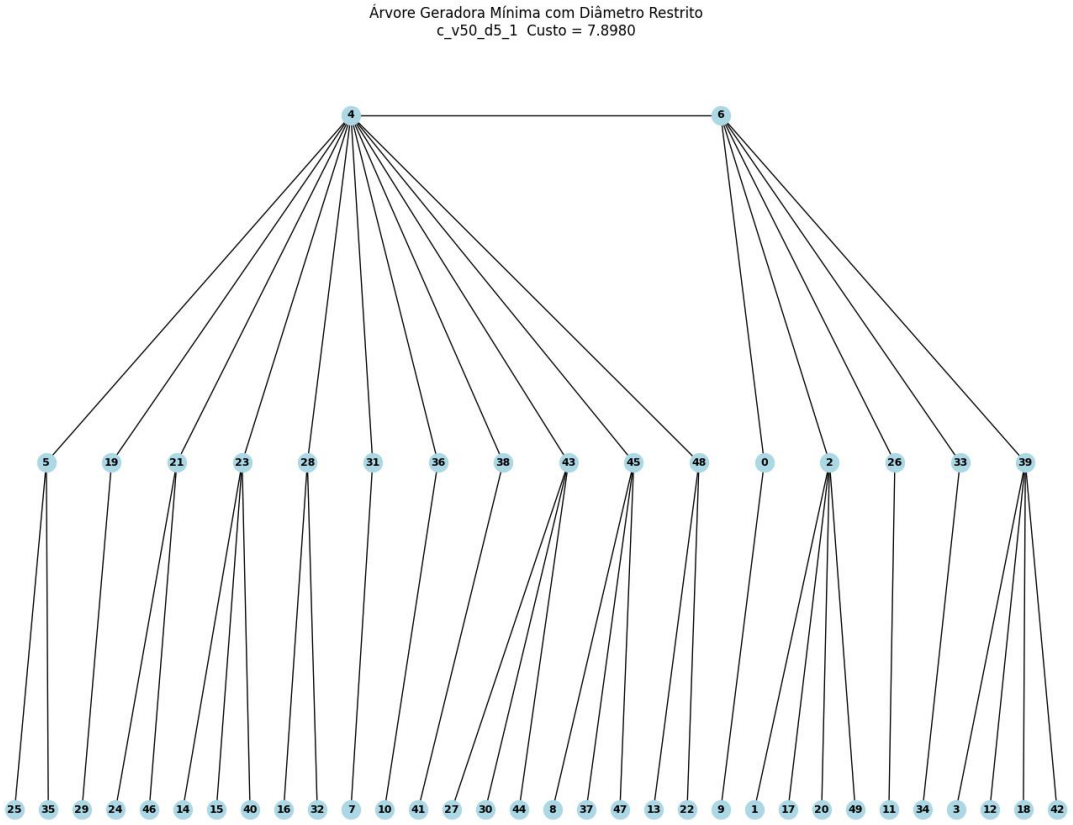
PLOTAGEM DAS ÁRVORES GERADORAS COM DIÂMETRO RESTRITO

c_v50_d5_1 Custo = 7.898 Tempo = 1.349

Árvore Válida (7.897991600000001, 5) Diâmetro da árvore: 5

Custo da árvore: 7.897991600000001

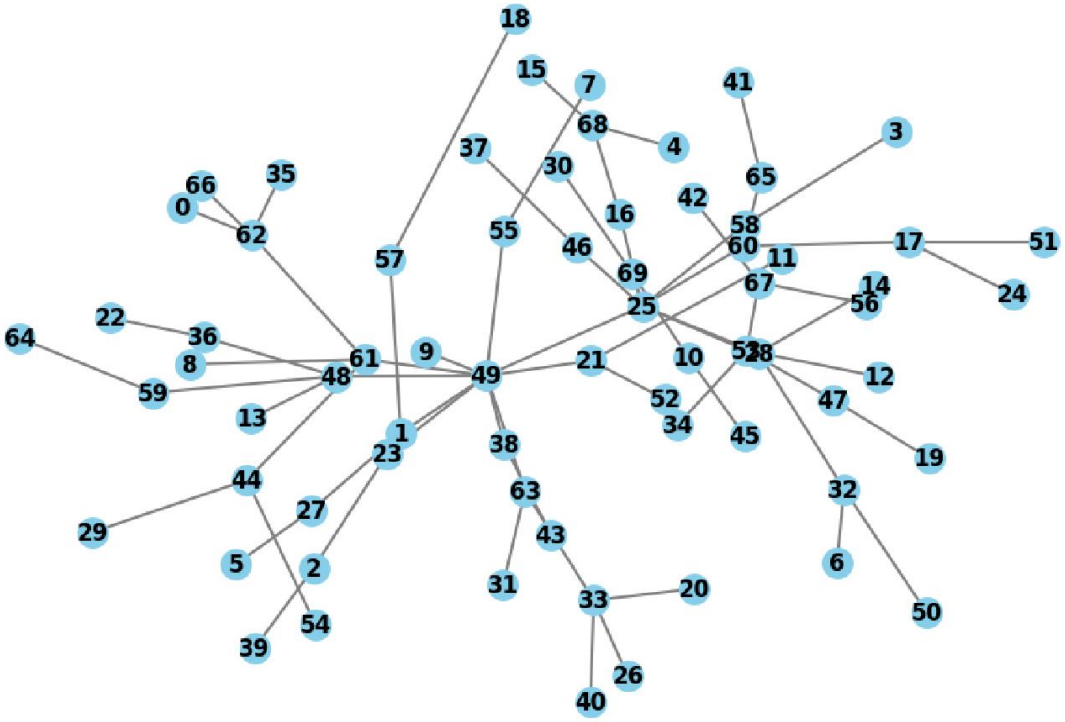




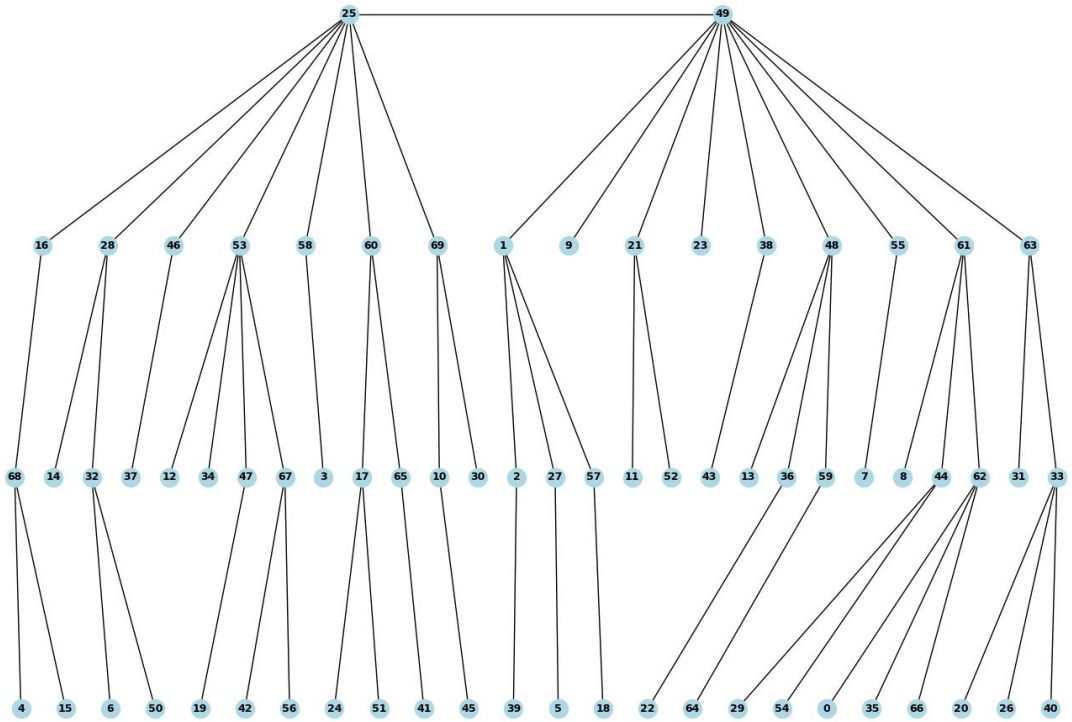
c_v70_d7_1 Custo = 7.594 Tempo = 2.307

Árvore Válida (7.5940758, 7) Diâmetro da árvore: 7

Custo da árvore: 7.5940758



Árvore Geradora Mínima com Diâmetro Restrito
c_v70_d7_1 Custo = 7.5941

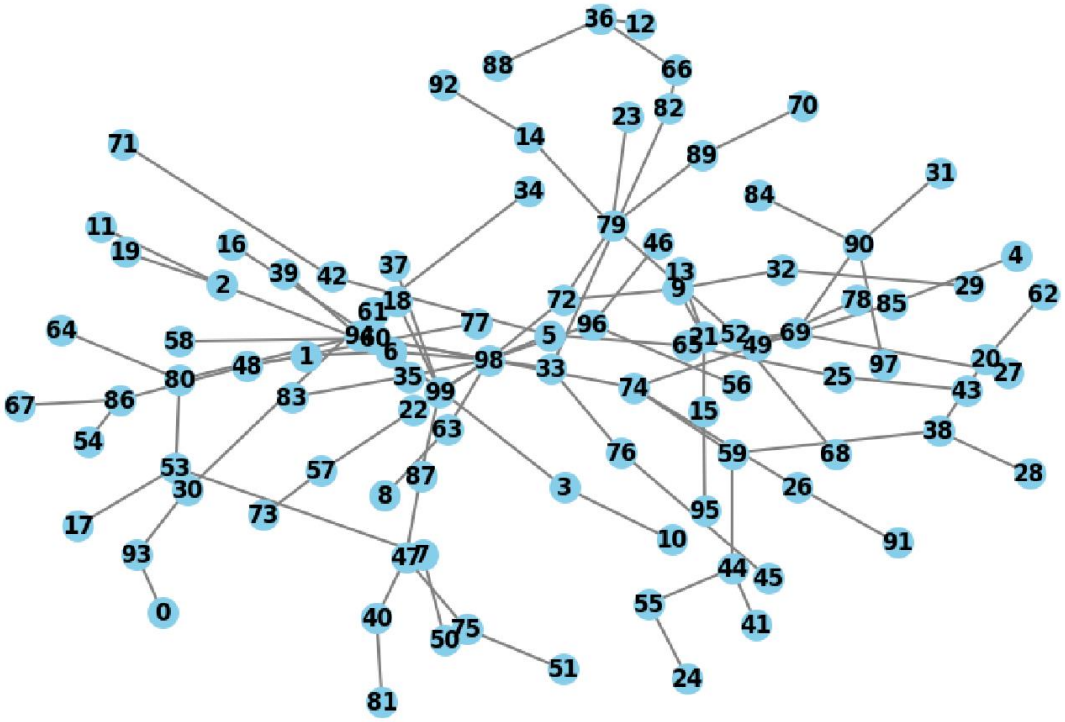


c_v100_d10_1 Custo = 9.364 Tempo = 1.845

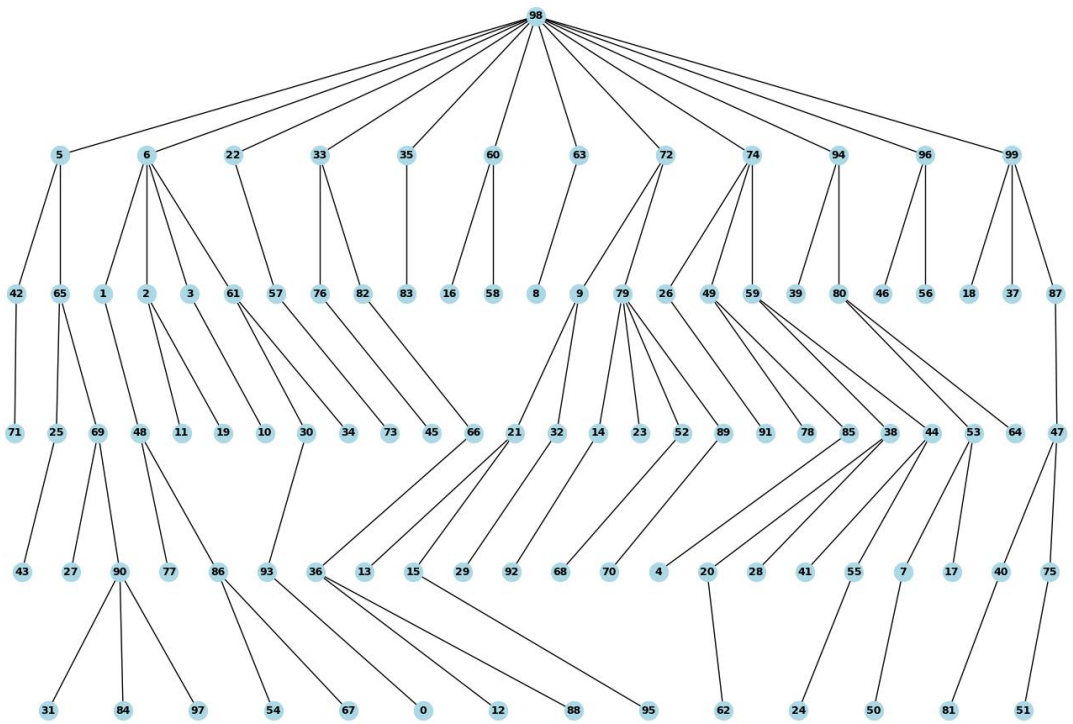
Árvore Válida (9.36423738, 10)

Diâmetro da árvore: 10

Custo da árvore: 9.36423738



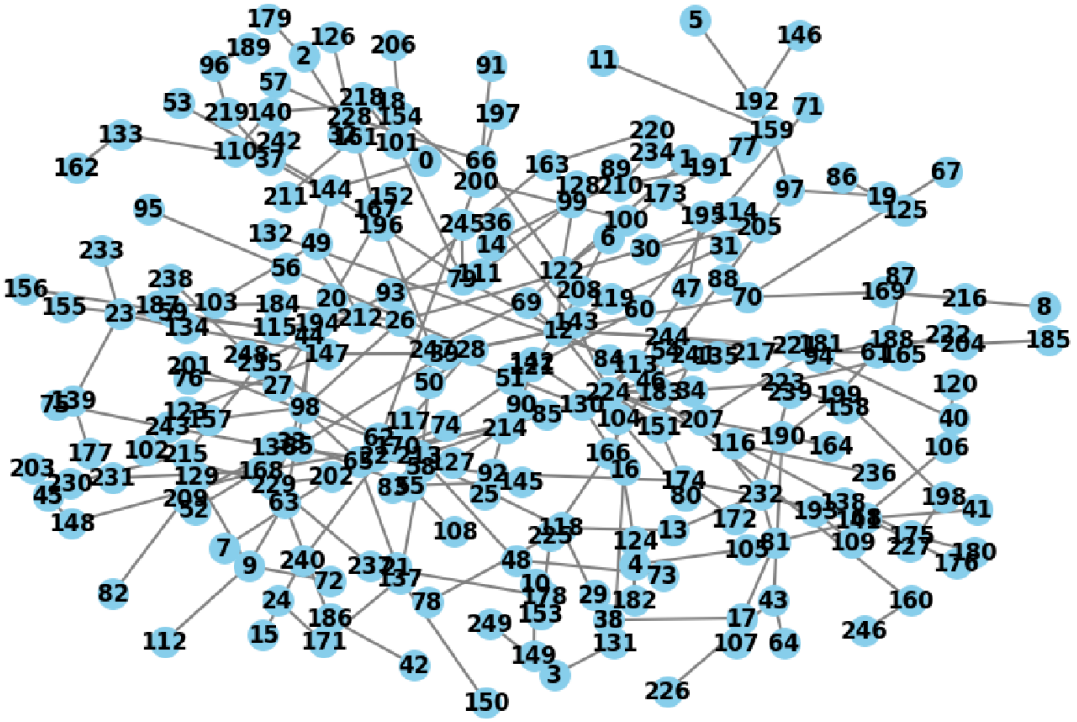
Árvore Geradora Mínima com Diâmetro Restrito
c_v100_d10_1 Custo = 9.3642



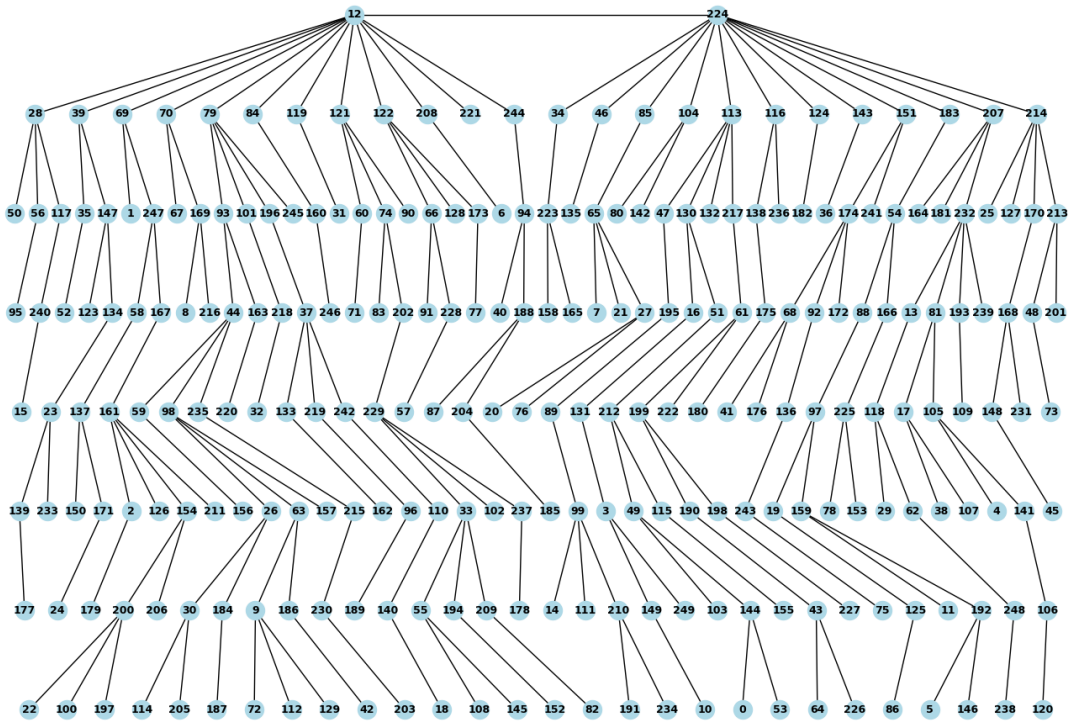
c_v250_d15_2 Custo = 17.239 Tempo = 38.828

Árvore Válida (17.238776249999994, 15)

Diâmetro da árvore: 15



Árvore Geradora Mínima com Diâmetro Restrito
c_v250_d15_2 Custo = 17.2388



Custo da árvore: 17.238776249999994

c_v500_d20_1 Custo = 27.800 Tempo = 652.779

Árvore Válida (27.800245439999973, 20)

Diâmetro da árvore: 20

Custo da árvore: 27.800245439999973