Universidade Federal do Espírito Santo
Programa Institucional de Iniciação Científica
Relatório Final de Pesquisa
Ciências Exatas e da Terra

Um algoritmo *Iterated Local Search* híbrido para solução do problema da Árvore geradora com número mínimo de vértices d-*branch*

Edital:	Edital Piic 2023/2024
Grande Área do Conhecimento (CNPq):	Ciências Exatas e da Terra
Área do Conhecimento (CNPq):	Ciência da Computação
Título do Projeto:	Estudo de centralidades de grafos com aplicações em Otimização e Computação Científica Combinatória.
Título do Subprojeto:	Um algoritmo *Iterated Local Search* híbrido para solução do problema da Árvore geradora com número mínimo de vértices d-*branch*.
Professor(a) Orientador(a):	Maria Claudia Silva Boeres
Estudante:	Marcos Vinícius de Souza Silva

Resumo

O problema da Árvore Geradora com Número Mínimo de Vértices *d-Branch* (d-MBV) busca, em um grafo conexo, não direcionado e não ponderado, identificar uma árvore geradora que minimize o número de vértices com grau maior que d, para $d > 2$. A relevância desse problema se destaca na otimização da alocação de *switches* WDM (Multiplexação por Divisão de Comprimento de Onda) em redes ópticas de *multicast*. Este trabalho investiga um algoritmo *Iterated Local Search* (ILS) adaptado com o uso da centralidade *PageRank* no lugar de grau, explorando variações nos métodos de melhoria da solução, em busca de aprimorar a identificação de padrões em soluções eficientes. A metodologia incluiu um estudo detalhado de cada etapa do ILS (fase construtiva, busca local e perturbação), introduzindo adaptações em cada fase e substituindo o algoritmo construtivo por uma versão randômica encontrada na literatura. Resultados obtidos a partir de experimentos computacionais realizados com um conjunto de instâncias do problema mostram melhorias quando comparadas com os mesmos algoritmos guiados pela centralidade de grau.

Palavras-chave: Otimização Combinatória. Meta-heurística ILS. Centralidade de Grafos. Problema d-MBV.

1 Introdução

Problemas de Otimização Combinatória são frequentemente desafiadores devido à complexidade de encontrar soluções ótimas com algoritmos eficientes. À medida que o tamanho do problema cresce, o número de soluções possíveis aumenta exponencialmente, tornando métodos exatos, da programação linear ou inteira, inviáveis para grandes instâncias. Nesse contexto, as meta-heurísticas têm se destacado como ferramentas eficazes, gerando soluções de alta qualidade em tempos computacionais razoáveis. Embora não garantam soluções ótimas, sua eficiência e adaptabilidade as tornam amplamente utilizadas e constantemente aprimoradas, com foco em robustez, qualidade e flexibilidade. A pesquisa em meta-heurísticas tem buscado combiná-las com métodos exatos, criando

hibridizações que aliam a capacidade exploratória das meta-heurísticas com a precisão dos métodos exatos, como os detalhados no trabalho de PUCHINGER & RAIDL (2005). Isso permite melhorar a eficiência e os resultados, especialmente em problemas de grande escala.

A Teoria dos Grafos desempenha um papel crucial na solução de problemas de otimização ao modelar relações entre entidades por meio de vértices e arestas. A sua versatilidade permite a formulação de problemas de Otimização Combinatória usando a representação por grafos, além do desenvolvimento de algoritmos eficazes para o desafio de solucioná-los. Problemas de otimização, como a busca por uma árvore geradora mínima – uma subestrutura acíclica que conecta todos os vértices de um grafo – são comuns em redes de transporte e telecomunicações, visando minimizar custos. Variações desse problema surgem conforme o foco se desloca para diferentes elementos do grafo, como vértices, arestas ou custos. Conhecer propriedades da estrutura de um grafo muitas vezes auxilia na solução de problemas representados por eles. Centralidade de grafos é uma invariante que ajuda a classificar vértices conforme sua importância no grafo. O grau de um vértice é um exemplo de medida de centralidade. Centralidades são amplamente usadas para analisar redes complexas, como redes sociais ou urbanas. Além disso, distâncias entre vértices também definem formas de centralidade, auxiliando na compreensão da posição relativa dos vértices.

Neste projeto, investigamos o problema da Árvore Geradora com Número Mínimo de Vértices d-Branch, conhecido em inglês como *Minimum d-Branch Vertices Problem* (d-MBV). Este problema foi inicialmente proposto por GARGANO et al. (2002) e mais tarde generalizado por MERABET et al. (2018) e RAMÍREZ et al. (2018). Dado um grafo conexo, não direcionado e sem pesos nas arestas, o objetivo do d-MBV é encontrar uma árvore geradora que minimize o número de vértices cujo grau seja superior a um valor específico d, onde $d > 2$. Estes vértices são denominados de vértices *d-branch* ou ramificações. Em outras palavras, o problema busca uma árvore geradora que reduza a quantidade de vértices altamente ramificados, otimizando assim a estrutura do grafo para ter menos vértices com grau elevado, enquanto mantém a conectividade entre todos os vértices do grafo original.

Com base em estudos anteriores sobre o problema d-MBV, este projeto propõe uma abordagem que combina o uso dos algoritmos desenvolvidos por RAMÍREZ (2018) e OLIVEIRA (2023), com centralidade de grafos diferentes do grau. A proposta visa explorar como essas medidas influenciam a eficiência e a eficácia na convergência para soluções locais ótimas. A integração das centralidades busca aprimorar a escolha e a priorização dos vértices e arestas no processo de otimização, promovendo uma melhor redução do número de vértices *d-branch* e uma abordagem mais robusta para encontrar soluções de alta qualidade. Este estudo avalia o impacto das centralidades na convergência dos algoritmos de solução do problema, investigando se a tornam eficaz.

A seguir, serão apresentados os principais objetivos deste estudo, que incluem o aprimoramento das meta-heurísticas para o problema d-MBV, a experiência adquirida no ambiente de pesquisa científica e a divulgação dos resultados obtidos. A seção de embasamento teórico oferece a fundamentação necessária para a compreensão e aperfeiçoamento dos algoritmos, abordando conceitos essenciais. A metodologia descreve o algoritmo adaptado durante este estudo para resolver o problema d-MBV. Por fim, serão apresentados os resultados e as conclusões das variações realizadas nos algoritmos estudados, destacando o desempenho obtido nas instâncias testadas.

2 Objetivos

O principal objetivo deste projeto é investigar e desenvolver técnicas eficazes para a solução do problema de otimização d-MBV, um tópico de relevância significativa na literatura, abrangendo algoritmos, otimização e teoria

dos grafos. Os objetivos específicos são: (i) Proporcionar a oportunidade de adquirir experiência no campo da pesquisa científica, com foco na aplicação de metodologias avançadas e na resolução de problemas complexos; (ii) Analisar e adaptar algoritmos e meta-heurísticas específicas para a solução do problema d-MBV, investigando abordagens inovadoras e comparando a eficácia de diferentes estratégias. (iii) Explorar e aplicar ferramentas teóricas e práticas relevantes para o estudo, incluindo teoria dos grafos, algoritmos construtivos e de busca local, além de utilizar ferramentas de visualização, como o *Graphviz*, para a representação e análise das soluções. (iv) Disseminar os resultados e as experiências adquiridas durante o projeto, contribuindo para o avanço do conhecimento na área por meio do compartilhamento das descobertas.

3 Embasamento Teórico

Nesta seção é resumido os principais conceitos aprendidos nesta Iniciação Científica, relativos a Teoria dos Grafos e algoritmos heurísticos para a solução de problemas de Otimização Combinatória. A Teoria dos Grafos estuda estruturas formadas por vértices conectados por arestas e é amplamente aplicada em áreas como redes de comunicação, transporte e biologia. Essas estruturas podem ser representadas por listas ou matrizes de adjacência, descrevendo as conexões entre os vértices. Alguns dos conceitos fundamentais incluem adjacência, incidência, conexidade, caminhos, ciclos, subgrafos e vizinhança. Esses conceitos são essenciais para a compreensão de grafos e foram discutidos durante o estudo de NETTO & JURKIEWICZ (2017). Além deles, os conceitos de Teoria dos Grafos que se destacam neste projeto são árvores, particularmente, árvores geradoras. Uma árvore é um grafo conexo e acíclico, onde qualquer par de vértices está conectado por exatamente um caminho, sem formar ciclos. Uma árvore geradora de um grafo G é um subgrafo acíclico que conecta todos os vértices de G com exatamente $|V|-1$ arestas, onde $|V|$ é o número de vértices. Se uma aresta for removida, a árvore se desconecta, dividindo-se em componentes conexos que são árvores.

Outro conceito relevante para este estudo é o de centralidade de grafos. As centralidades são métricas usadas para avaliar a importância dos vértices em um grafo. Uma medida de centralidade é uma função aplicada aos vértices de um grafo com o objetivo de classificá-los, do mais para o menos central, de acordo com um particular critério. Dessa forma, podem ser usadas para classificar os vértices a partir de sua importância na estrutura de ligações do grafo ao qual eles pertencem. Medida de centralidade (ou simplesmente centralidade) é uma invariante do grafo. Uma invariante de um grafo consiste em uma propriedade que é preservada em grafos isomorfos (que representam a mesma estrutura de ligações entre os vértices). Dentre as diversas centralidades, destacam-se: a centralidade de grau, que mede o número de arestas incidentes a um vértice e reflete sua conectividade direta; a centralidade de proximidade, que calcula a distância média de um vértice para todos os outros vértices do grafo; e a centralidade de intermediação, que mede quantas vezes um vértice é intermediário nos caminhos mais curtos entre outros pares de vértices. Este estudo aborda duas centralidades fundamentais: o grau, já discutido, e o *PageRank*.

O *PageRank*, introduzido por BRIN & PAGE (1998), é uma medida de centralidade que avalia a importância de um vértice com base na quantidade e na qualidade de suas conexões. O algoritmo atribui maior relevância a vértices conectados a outros vértices importantes, enfatizando que a importância de um vértice é influenciada pela importância dos vértices aos quais está ligado. Utilizado em sistemas de ranqueamento, como análise de redes sociais e busca na web, o *PageRank* ajuda a identificar conexões influentes dentro de um grafo.

Problemas de Otimização Combinatória são amplamente estudados na literatura, e diversas estratégias foram desenvolvidas para encontrar boas soluções. As meta-heurísticas são uma dessas estratégias de alto nível e flexíveis,

aplicáveis a uma variedade de problemas. Elas combinam técnicas que geralmente envolvem um algoritmo de busca local e um mecanismo de exploração global, proporcionando soluções eficazes e frequentemente de alta qualidade, embora não garantam a solução ótima. Esses conceitos foram abordados e estudados a partir de TALBI (2009).

Algoritmos construtivos incrementam soluções gradualmente, começando com uma solução vazia e adicionando elementos com base em critérios específicos, sem revisar decisões anteriores. Eles são eficientes para gerar soluções iniciais que podem ser aprimoradas posteriormente. Busca local, por sua vez, explora soluções vizinhas a uma solução inicial, fazendo modificações incrementais para melhorar a qualidade. Embora eficaz para refinar soluções, pode estagnar em ótimos locais, limitando a capacidade de encontrar soluções globais. Para superar essa limitação, a técnica de perturbação é usada para introduzir mudanças significativas na solução corrente, ampliando a exploração do espaço de busca e evitando a estagnação. Frequentemente, a perturbação é combinada com a busca local para equilibrar exploração de soluções.

4 Metodologia

Este estudo propõe uma solução para o problema da Árvore Geradora com Número Mínimo de Vértices *d-branch* (brevemente descrito na seção 4.1) fundamentada no algoritmo *Iterated Local Search* (ILS) proposto por (RAMÍREZ, 2018) (descrito na Seção 4.2). A metodologia desenvolvida adapta e expande essa abordagem, considerando uma nova estratégia de algoritmo construtivo desenvolvido em OLIVEIRA (2023), variações na técnica de busca e a inclusão da centralidade *PageRank*, visando aprimorar a qualidade das soluções obtidas.

4.1 O problema d-MBV

Dado um grafo conexo, não direcionado e não ponderado G, composto por um conjunto de vértices V_G e um conjunto de arestas E_G, o problema MBV visa encontrar uma árvore geradora $T = (V_T, E_T)$, onde $V_T = V_G$ e $E_T \subseteq E_G$, que minimize o número de vértices com grau superior a 2 na árvore geradora. Esses vértices são conhecidos como vértices *branch* ou ramificações. O problema MBV foi introduzido por (GARGANO et al., 2002) com a motivação da aplicação, que tinha como objetivo, determinar as melhores posições de alocação de *switches*. A minimização do número de vértices *branch* é crucial devido ao elevado custo desses dispositivos, que pode ser substancialmente reduzido ao limitar a quantidade necessária.

No estudo de MERABET et al. (2018), foi proposta uma generalização do MBV utilizando o conceito de *k-branch*, que define um vértice como *k-branch* se seu grau for estritamente superior a $k+2$. Assim, o problema k-MBV procura encontrar a árvore geradora que minimize o número de vértices *k-branch*. Os autores dessa generalização demonstraram que o problema é *NP-Hard* para qualquer valor de k. E para simplificar a notação, durante o estudo de RAMÍREZ (2018) foi introduzido o parâmetro $d = k+2$, no qual um vértice em um grafo é classificado como *d-branch* se seu grau for estritamente maior que d, para $d > 2$. O problema d-MBV, portanto, busca minimizar o número de vértices *d-branch* nas soluções, herdando todos conceitos e objetivos do problema MBV.

4.2 Iterated Local Search (ILS)

O *Iterated Local Search* (ILS) é uma meta-heurística que busca melhorar as soluções obtidas por métodos de busca local, combinando exploração e intensificação de espaços de soluções. Ele aplica uma busca local em uma solução

inicial e, em seguida, introduz perturbações para escapar de mínimos locais, permitindo explorar novas regiões do espaço de soluções. O processo se repete até atingir um critério de parada, refinando as soluções sucessivamente. O método ILS proposto em RAMÍREZ (2018) foi desenvolvido para receber um grafo conexo G e encontrar uma árvore geradora de G que minimize os vértices *d-branch*. A primeira modificação proposta neste trabalho para o aprimoramento do ILS, foi a adição de variáveis de entrada que permitem selecionar o algoritmo construtivo e diferentes estratégias para a busca local e perturbação.

Algoritmo 1: ILS - Iterated Local Search

Entrada: O grafo $G = (V,E)$, $Construtivo = \{AC_{\text{ILS}}, RBEP\}$, $OrdenacaoBusca = \{Grau, PageRank\}$, $ModoPertubacao = \{Grau, Pagerank\}$ e $SelecaoVizinhos = \{Primeiro, Melhor\}$.

Saída: Uma solução válida T.

1 $O_D \leftarrow Decomposicao\_por\_Pontes(G)$;
2 $T_0 \leftarrow Solucao\_Inicial(G, O_D, Construtivo)$;
3 $T^* \leftarrow Busca\_Local(T_0, O_D, OrdenacaoBusca, SelecaoVizinhos)$;
4 **repita**
5 $\quad T' \leftarrow Perturbacao(T^*, O_D, ModoPertubacao)$;
6 $\quad T'' \leftarrow Busca\_Local(T', O_D, OrdenacaoBusca, SelecaoVizinhos)$;
7 $\quad T^* \leftarrow Criterio\_Aceitacao(T^*, T'', historia)$;
8 **até** *condicao de parada atendida*;
9 **return** T^*;

Inicia-se, então, o Algoritmo ILS (Algoritmo 1) com a construção do conjunto O_D (linha 1), fundamentada na abordagem de decomposição de grafos proposta por MELO et al. (2016), que é baseada na análise de pontes no grafo. Então, seja $\beta(v)$ o número de pontes incidentes no vértice v. Define-se o conjunto como: $O_D = \{u \in V \mid d_G(u) > d \wedge \beta(u) \geq d\}$, $d_G(u)$ sendo o grau do vértice u em G. Um grafo G possui várias árvores geradoras como soluções. Uma característica essencial de uma ponte em um grafo é que sua remoção divide o grafo em duas componentes conexas, o que inviabiliza a conectividade necessária, em qualquer árvore geradora. Portanto, se uma aresta é uma ponte, ela estará presente em todas as possíveis soluções. Assim, vértices com mais de d incidências de pontes serão necessariamente parte do conjunto de vértices *d-branch* em todas as possíveis soluções e, por isso, são incluídos no conjunto O_D.

O próximo passo do Algoritmo 1, é a geração de uma solução inicial T_0 (linha 2). A variável *Construtivo* define qual algoritmo será utilizado nessa etapa, sendo as opções consideradas o Algoritmo Construtivo *ILS* (RAMÍREZ, 2018) e o Algoritmo Construtivo *R-BEP* (OLIVEIRA, 2023), detalhados a seguir:

4.2.1 Algoritmo Construtivo ILS

Recebe como entrada o grafo G e o conjunto de vértices O_D. O processo inicia-se com a criação de uma árvore T que contém todos os vértices de G, $V_T = V_G$, e o conjunto $E_T = \emptyset$. Em seguida, um laço é executado para adicionar arestas a T até que a quantidade total de arestas atinja $|V| - 1$, assegurando assim a formação de uma árvore geradora, totalmente conectada e sem ciclos. Em cada iteração do laço, uma aresta (u, v) do grafo G é selecionada para ser incluída em T. A seleção é baseada em uma lista das arestas de G, ordenadas de forma ascendente por um critério que leva em conta o peso dessas arestas, calculado a partir da soma dos graus dos vértices u e v no grafo G.

A inserção das arestas em T segue um critério de prioridade bem definido. Inicialmente, o algoritmo tenta adicionar arestas que não introduzam novos vértices *d-branch* em T. Caso isso não seja viável, são consideradas arestas que introduzam apenas um novo vértice *d-branch*. Finalmente, para garantir a conectividade da árvore, são adicionadas arestas que possam introduzir até dois novos vértices *d-branch*. Esses critérios visam minimizar o número de vértices *d-branch* na árvore geradora final. Ao término do laço, a árvore T construída é retornada como a solução inicial T_0 da linha 2 do Algoritmo ILS (algoritmo 1). O pseudocódigo detalhado desse algoritmo pode ser encontrado em RAMÍREZ (2018).

4.2.2 Algoritmo Construtivo R-BEP

Recebe como entrada um grafo conexo G e inicia-se com a construção do conjunto B_T. Esse conjunto é composto por vértices cuja remoção de G resultaria na formação de três ou mais componentes conexas. Todo vértice v que satisfaz essa condição é classificado como um vértice *d-branch*, incluído no conjunto B_T e na solução T. Após a construção de B_T, é inicializado a fase de expansão. Nesse momento, o grau dos vértices *d-branch* presentes em B_T são maximizados, uma vez que o problema d-MBV não impõe restrições sobre o grau desses vértices. Então para cada vértice $v \in B_T$, todos os vértices u adjacentes a v no grafo G são incorporados à solução T, juntamente com as arestas (u,v).

Em seguida, é construído o conjunto de vértices chamados de *pontas*, que correspondem às folhas na fase atual da construção. Esses vértices têm grau 1 na árvore T e grau maior que 1 no grafo G, indicando a presença de caminhos a serem explorados para a conclusão do algoritmo construtivo. Originalmente, se não houver vértices na solução atual T que atendam a esse requisito e o conjunto de *pontas* estiver vazio, o vértice v de menor grau em G é adicionado à solução, juntamente com seu vizinho u, também de menor grau, formando a aresta (v,u). Ambos os vértices são então incluídos ao conjunto *pontas* e a aresta (v,u) na solução T. Durante este estudo, foi sugerida uma modificação: em vez de utilizar o menor grau como critério, seleciona a aresta (v,u) com base nos menores valores de *PageRank* dos vértices v e u, buscando uma escolha mais estratégica para a expansão da árvore e na construção do conjunto *pontas*.

Próximo passo do algoritmo é um laço que adiciona arestas à solução T até que a árvore alcance $|V|-1$ arestas, completando uma árvore geradora. Esta fase é o núcleo do algoritmo R-BEP, onde uma roleta é utilizada para a seleção dos vértices que serão inseridos na solução. Cada vértice do conjunto *pontas* é inserido na roleta e recebe um peso proporcional ao seu grau no grafo G. Ao rodar a roleta, um vértice v é sorteado. Em seguida, outra roleta é preenchida com os vizinhos de v, também ponderada pelos graus desses vértices em G. Após o segundo sorteio, o vértice u é selecionado, e a aresta (v,u) é adicionada à árvore T. O conjunto de pontas é então atualizado, e o processo se repete até que T satisfaça o critério de parada, $|V|-1$ arestas.

Se o conjunto de pontas esvaziar durante o processo, o algoritmo relaxa algumas restrições para garantir a completude da solução, o que pode incluir a adição de vértices *d-branch*. Dessa forma, a construção da árvore é garantida até que se forme uma solução viável T_0, que é então retornada para a linha 2 do Algoritmo ILS (algoritmo 1). O pseudocódigo detalhado desse algoritmo pode ser encontrado em OLIVEIRA (2023).

4.2.3 Busca Local ILS

A busca local explora soluções vizinhas a uma solução inicial, realizando modificações incrementais para melhorar gradualmente a qualidade da solução. A cada iteração, pequenas alterações são aplicadas à solução corrente,

e a nova solução é avaliada para verificar se há uma melhoria no critério de qualidade. No método ILS (Algoritmo 1), um procedimento de busca local é executado na linha 3. Esse procedimento é baseado na proposta de RAMÍREZ (2018) e aprimorado por variações introduzidas neste estudo. A função *Busca_Local* (linha 3) recebe como entrada uma árvore inicial T, o conjunto de vértices O_D, e duas variáveis adicionais: *OrdenacaoBusca* e *SelecaoVizinhos*. Essas variáveis permitem controlar as variações do algoritmo, possibilitando uma análise mais detalhada dos resultados. O pseudocódigo da busca local descrita é apresentado abaixo (Algoritmo 2).

Algoritmo 2: Busca Local

Entrada: A solução atual T_{curr}, O_D, $OrdenacaoBusca = \{Grau, PageRank\}$ e $SelecaoVizinhos = \{Primeiro, Melhor\}$.

Saída: A solução T_{best}.

```
1  T_best ← T_curr;
2  repita
3      melhoria ← false;
4      T ← Vizinho_Customizado(G, T_best, O_D, OrdenacaoBusca, SelecaoVizinhos);
5      se T ≠ T_best então
6          T_best ← T;
7          melhoria ← true;
8      fim se
9  até melhoria = false;
10 return T_best
```

Primeiramente, a melhor solução encontrada até o momento é armazenada em T_{best} (linha 1). Em seguida, o Algoritmo 2 entra em um laço, linha 2 até a linha 9, onde a árvore T é modificada de forma iterativa pela função *Vizinho_Customizado* (linha 4). Esse processo continua enquanto forem encontradas melhorias na solução T, representado pelo condicional descrito na linha 5. Caso, em alguma iteração, nenhuma melhoria seja realizada, o laço é encerrado (linha 9) e a melhor solução, T_{best}, é retornada ao Algoritmo ILS (Algoritmo 1).

4.2.4 Vizinho Customizado

Na linha 4 do Algoritmo de Busca Local (Algoritmo 2), a função *Vizinho_Customizado* é invocada com o objetivo de modificar a árvore T para reduzir o número de vértices *d-branch*. Inicia-se criando uma lista *DB* contendo todos os vértices *d-branch*. A lista *DB* é então ordenada em ordem crescente, primeiramente com base no grau de cada vértice v em T. Como alternativa, foi implementada neste trabalho uma nova estratégia de ordenação que prioriza o menor valor do *PageRank* de v em G. A escolha entre essas duas estratégias é controlada pela variável *OrdenacaoBusca*. Com o conjunto *DB* ordenado, a função entra em um laço que percorre os vértices de *DB*. Para cada vértice v selecionado, uma aresta (v,u) é removida de T, onde o vértice u é um vizinho de v escolhido sem um critério específico. Em seguida, a função busca uma nova aresta (v',u') que possa ser adicionada para restaurar a conectividade de T. Após essa adição, verifica-se se a modificação reduziu o número de vértices *d-branch* em T.

Originalmente, o laço era interrompido assim que uma melhoria na redução dos vértices *d-branch* fosse obtida. Entretanto, uma modificação foi introduzida para permitir que o laço continue em busca de mais melhorias, controlada pela variável *SelecaoVizinhos*. Se configurada como *Melhor*, o laço persiste até encontrar a melhor modificação possível; se configurada como *Primeiro*, interrompe-se na primeira melhoria encontrada.

A função *Vizinho_Customizado* (linha 4) busca reduzir o número de vértices *d-branch* em T, seja interrompendo após a primeira melhoria ou continuando até encontrar a melhor solução. Se nenhuma melhoria for identificada, a função garante que não sejam feitas alterações que aumentem o número de vértices *d-branch* em relação à solução inicial. Por isso, durante o laço da Busca Local (Algoritmo 2, linha 2 até a linha 9), se em duas iterações consecutivas a solução T permanecer igual a T_{best}, a Busca Local é encerrada. Isso indica que a função Vizinho Customizado já explorou todas as possíveis modificações nos vértices *d-branch*, sinalizando um ponto de estagnação na solução.

4.2.5 Pertubação

Para superar possíveis estagnações na busca pela solução ótima, o Algoritmo ILS (Algoritmo 1) incorpora uma função de Perturbação (linha 5). O objetivo principal desta etapa é modificar a árvore T, permitindo que a Busca Local aplicada em seguida (linha 6) explore novos caminhos, aumentando as chances de encontrar soluções superiores. A função de perturbação opera de forma semelhante ao algoritmo de Busca Local.

A princípio, todos os vértices *d-branch* são coletados em uma lista *DB*. Em seguida, a função percorre cada vértice v em *DB*, examinando seus vizinhos e identificando a aresta (v,u). O algoritmo tenta remover essa aresta e localizar uma nova aresta (v',u') que mantenha a conectividade da árvore T e tenha pelo menos um dos vértices (v' ou u') classificado como *d-branch*. Se uma nova aresta (v',u') que satisfaz esses critérios for encontrada, a substituição é realizada em T, e a nova solução é retornada para o ILS (Algoritmo 1, linha 5).

Originalmente, o Algoritmo Perturbação era guiado exclusivamente pelo grau dos vértices, tanto na ordenação da lista *DB* (do menor para o maior grau) quanto na seleção da nova aresta (v',u'), que considerava a menor soma dos graus de v' e u'. Neste estudo, foi proposta uma modificação com a introdução da variável $ModoPerturbacao$. Quando $ModoPerturbacao$ é definido como *Grau*, o algoritmo segue a abordagem tradicional, utilizando o grau dos vértices para ordenar *DB* e selecionar (v',u'). Contudo, quando $ModoPerturbacao$ é definido como *PageRank*, a ordenação dos vértices *d-branch* em *DB* é baseada nos valores de *PageRank* (do menor para o maior), e a escolha da nova aresta (v',u') leva em conta a menor soma dos valores de *PageRank*.

Durante o laço principal do ILS (Algoritmo 1, entre as linhas 4 e 8), a função de Perturbação (linha 5) e a Busca Local (linha 6) são chamadas alternadamente até que o critério de parada seja atingido (linha 7). Esse critério pode ser definido por duas condições: quando não há mais substituições de arestas possíveis pela função de Perturbação ou quando o limite de tempo de execução da instância é alcançado. Vale destacar que, se em qualquer momento a função objetiva atingir o valor zero para a quantidade de vértices *d-branch*, a árvore T é imediatamente retornada, e o algoritmo é interrompido, pois não há mais possibilidade de redução no número de vértices *d-branch*.

5 Resultados e Discussão

Os experimentos realizados combinaram os algoritmos descritos nos trabalhos de RAMÍREZ (2018) e OLIVEIRA (2023) com as adaptações propostas neste estudo, e foram aplicadas a 525 instâncias de grafos conexos e esparsos, variando de 20 a 1000 vértices, instâncias desenvolvidas por CARRABS et al. (2013). Os testes foram feitos em uma máquina equipada com um processador *Ryzen* 5 5600GT, 32 GB de memória RAM e rodando o sistema operacional *Linux*.

As tabelas apresentadas nesta seção podem ser divididas em dois tipos: aquelas que são um resumo comparativo dos algoritmos, mostrando para quantas instâncias cada um deles atinge o valor ótimo em alguma execução e

o melhor resultado obtido dentre todos os algoritmos comparados. As Tabelas 1, 4, 5, e 8 são desse tipo. Há também as tabelas onde são apresentadas para cada algoritmo, a média dos resultados obtidos para os conjuntos de instâncias com o mesmo número de vértices, mas números distintos de arestas. Esse é o caso das tabelas 2, 3, 6, 7, 9 e 10. Em todas as tabelas, os melhores resultados são coloridos em azul, enquanto os piores, em vermelho.

Um dos modelos de tabelas deste estudo (tabelas 2, 3, 6, 7, 9 e 10) segue um formato semelhante ao utilizado em outros trabalhos da literatura, para facilitar a comparação. As instâncias foram organizadas conforme as divisões propostas por (CARRABS et al., 2013). No total, 525 instâncias de grafos foram classificadas em dois grupos: instâncias de tamanhos pequenos e médios, com grafos contendo entre 20 e 500 vértices, e instâncias de grande porte, com grafos entre 600 e 1000 vértices. Para cada valor de n (número de vértices) das instâncias pequenas e médias, há cinco conjuntos de cinco grafos que variam quanto ao número de arestas m. Os resultados para essa instâncias são valores correspondentes as médias obtidas a partir de 25 grafos, que possuam o mesmo número de vértices, mas com cinco valores distintos para o número de arestas, enquanto para as instâncias de grande porte, os resultados são baseados em um conjunto de 5 grafos que possuem o mesmo número de vértices e arestas.

5.1 Estudo do Algoritmo Construtivo aplicado ao problema d-MBV

No trabalho de OLIVEIRA (2023), são apresentadas oito variações de algoritmos construtivos para o problema d-MBV, agrupadas em dois principais conjuntos: algoritmos gulosos e suas versões randômicas. Os algoritmos gulosos, e se distinguem pela estratégia de seleção de vértices ou arestas. São chamados de BEP, EEP, CEP e CEEP. Por exemplo, o BEP seleciona vértices com base no menor grau, enquanto o EEP prioriza arestas com a menor soma de graus dos dois vértices. As versões randômicas, nomeadas R-BEP, R-EEP, R-CEP e R-CEEP, são variantes dos algoritmos gulosos que introduzem aleatoriedade por meio de uma roleta probabilística, usada na seleção de vértices ou arestas para compor a solução.

Os testes realizados por OLIVEIRA (2023) revelaram que os algoritmos randômicos apresentaram desempenho significativamente superior em comparação com os outros algoritmos construtivos da literatura e o R-BEP destacou-se como o algoritmo com o melhor desempenho. Os algoritmos gulosos foram executados uma vez para cada instância, enquanto os algoritmos randômicos foram executados cem vezes para cada instância, com o melhor resultado sendo registrado.

A Tabela 1 resume os resultados obtidos neste estudo para a execução de todos os algoritmos construtivos mencionados. A coluna **MORENO** refere-se ao algoritmo desenvolvido por RAMÍREZ (2018), brevemente explicado na seção 4.2.1. Para uma comparação justa, o algoritmo ILS no qual esse algoritmo está inserido, foi interrompido após a fase de construção. Além disso, nas execuções realizadas, o conjunto auxiliar O_D, relacionado à decomposição do grafo original (técnica utilizada pelo autor) foi desativada, não sendo explorada neste primeiro estudo, focando-se apenas na analise dos algoritmos construtivos. As demais colunas da tabela correspondem aos oito algoritmos construtivos discutidos em OLIVEIRA (2023).

Tabela 1: Resultados geral sobre os Algoritmos Construtivos.

Informações	MORENO	BEP	EEP	CEP	CEEP	R-BEP	R-EEP	R-CEP	R-CEEP
Exatos	18	41	51	41	51	**93**	86	91	90
Melhores Resultados	25	122	216	123	215	**374**	327	373	328

Fonte: Produção do próprio autor.

O destaque é o R-BEP, que se sobressaiu em 374 das 525 instâncias testadas. Os resultados mostrados na Tabela 1 confirmam aqueles de OLIVEIRA (2023), reforçando o R-BEP como o algoritmo de melhor desempenho.

Com base no desempenho do R-BEP, este estudo propõe duas variações neste algoritmo, integrando a centralidade *PageRank* para aprimorar a busca por soluções ótimas. A primeira variação, denominada **R-BEP-1**, modifica o critério de seleção dos vértices inseridos no conjunto *pontas*, algoritmo brevemente explicado na seção 4.2.2, ao invés de selecionar arestas a partir da menor soma de grau dos vértices, incorpora os valores de *PageRank* e seleciona as arestas a partir da menor soma do *PageRank* dos vértice. A segunda variação, chamada **R-BEP-2**, amplia essa abordagem e modifica o cálculo dos pesos na roleta probabilística, tornando a probabilidade de seleção inversamente proporcional ao *PageRank* dos vértices, em vez de ao grau. Enquanto no método original atribuía maior peso e favorecia vértices de menor grau, nesta variação, vértices com menor *PageRank* possuem maior chance de serem escolhidos.

Os resultados obtidos são apresentados nas Tabelas 2 e 3. As duas primeiras colunas dessas tabelas apresentam os números de vértices **n'** e as médias **m'** dos números de arestas de cada conjunto de instâncias testadas. A coluna **R-BEP** exibe os resultados mínimos obtidos com o algoritmo original. As colunas **R-BEP-1** e **R-BEP-2** mostram os resultados mínimos das adaptações que incorporam a centralidade *PageRank*.

Tabela 2: Resultados mínimos em instâncias pequenas e médias.

n'	m'	R-BEP	R-BEP-1	R-BEP-2
20	41.8	**0.84**	**0.84**	**0.84**
40	70.8	**3.20**	3.28	3.36
60	95.0	**7.20**	7.24	7.24
80	119.8	**10.20**	10.24	10.24
100	144.0	14.80	**14.72**	14.96
120	168.8	19.08	**19.04**	19.20
140	193.0	22.96	**22.76**	22.96
160	217.8	27.20	27.24	**27.16**
180	242.0	31.20	**31.16**	31.68
200	266.8	35.16	**35.04**	35.40
250	321.0	47.68	**47.44**	47.80
300	380.0	60.40	**60.36**	60.68
350	434.8	**72.16**	72.36	72.40
400	489.0	86.04	**85.80**	86.20
450	548.0	**97.44**	97.64	97.92
500	602.8	**111.12**	111.16	111.52

Fonte: Produção do próprio autor.

Tabela 3: Resultados mínimos em instâncias grandes.

n'	m'	R-BEP	R-BEP-1	R-BEP-2
600	637	**184.6**	**184.6**	**184.6**
600	674	170.0	170.0	**169.8**
600	712	**154.4**	154.6	154.6
600	749	143.2	**143.0**	**143.0**
600	787	131.4	**131.2**	131.6
700	740	**216.2**	**216.2**	**216.2**
700	780	201.2	**201.0**	**201.0**
700	821	**183.8**	184.4	184.0
700	861	169.0	169.0	**168.8**
700	902	**159.0**	159.4	159.4
800	843	**247.8**	**247.8**	**247.8**
800	886	**230.6**	**230.6**	**230.6**
800	930	212.6	**212.4**	212.6
800	973	199.2	**198.8**	199.4
800	1017	**183.0**	**183.0**	183.2
900	944	**282.0**	**282.0**	**282.0**
900	989	**263.4**	263.6	263.6
900	1034	**244.8**	**244.8**	**244.8**
900	1079	**229.8**	**229.8**	230.0
900	1124	212.8	213.0	**212.6**
1000	1047	**314.6**	**314.6**	**314.6**
1000	1095	294.2	**294.0**	294.2
1000	1143	**276.2**	**276.2**	276.4
1000	1191	**257.4**	**257.4**	258.0
1000	1239	241.8	**241.6**	242.0

Fonte: Produção do próprio autor.

A Tabela 4 resume os resultados obtidos em todas as instâncias testadas por **R-BEP**, **R-BEP-1** e **R-BEP-2**. Os resultados indicam que as variações propostas para o algoritmo **R-BEP** não proporcionaram melhorias significativas. A variação **R-BEP-2** apresentou um desempenho inferior ao algoritmo original, com uma piora de $8,6\%$ na linha

Exatos e $12,7\%$ na linha **Melhores Resultados**, evidenciando que a modificação no critério de pesos da roleta não foi eficaz. Em contrapartida, a variação **R-BEP-1** teve um desempenho muito próximo ao algoritmo original, apresentando uma diferença de $2,1\%$ na linha **Exatos** e uma melhoria de $3,3\%$ na linha **Melhores Resultados**, razão pela qual a variação **R-BEP-1** foi selecionada para os próximos estágios do algoritmo durante este projeto.

5.2 Estudo do algoritmo Busca Local aplicado ao problema d-MBV

A Busca Local utilizada como base para este estudo é a desenvolvida como parte integrante do algoritmo *Iterated Local Search* (ILS) proposto por RAMÍREZ (2018), que realiza iterações contínuas na solução atual para buscar melhorias até que não seja mais possível otimizar. Neste estudo, foram introduzidas duas modificações no algoritmo, resultando em três novas variações da busca local.

A primeira modificação refere-se ao conjunto *DB*, utilizado na Busca Local e brevemente descrito na seção 4.2.4. Originalmente, os vértices eram organizados neste conjunto *DB* levando em consideração o grau, do menor para o maior, caracterizando este modo de ordenação como $ModoOrdenacao = Grau$. Nesta pesquisa, foi adotada uma nova abordagem de ordenação, considerando os valores de *PageRank* dos vértices, também do menor para o maior, alterando o critério para $ModoOrdenacao = PageRank$.

A segunda modificação diz respeito à função de escolha das soluções vizinhas. Inicialmente, o algoritmo retornava a primeira solução encontrada que reduzisse a quantidade de vértices *d-branch*, caracterizando a seleção como $SelecaoVizinho = Primeiro$. No entanto, foi implementada uma modificação que avalia todas as possíveis soluções vizinhas e retorna a melhor entre elas, passando a configuração para $SelecaoVizinho = Melhor$.

A Tabela 5 segue o mesmo formato da seção anterior, onde os melhores resultados correspondem aos maiores valores. Todas as quatro soluções iniciais, das versões de Busca Local implementadas, foram geradas pelo algoritmo **R-BEP-1**. As colunas da tabela representam diferentes variações do algoritmo de busca local. A coluna **BL** corresponde ao algoritmo original de (RAMÍREZ, 2018), utilizando $ModoOrdenacao = Grau$ e $SelecaoVizinho = Primeiro$. A coluna **BL-1** refere-se a uma variação onde $ModoOrdenacao = PageRank$, mantendo $SelecaoVizinho = Primeiro$. A coluna **BL-2** descreve a variação com $ModoOrdenacao = Grau$ e $SelecaoVizinho = Melhor$. Por fim, a coluna **BL-3** apresenta a combinação de $ModoOrdenacao = PageRank$ com $SelecaoVizinho = Melhor$.

A análise da Tabela 5 revela que as variações da busca local não mostraram um desempenho significativamente superior àquele do algoritmo original. As variações que utilizam o *PageRank* como critério de ordenação, representadas pelas colunas **BL-1** e **BL-3**, exibiram uma leve melhoria em comparação com o algoritmo de busca local original, com um aumento aproximado de 1% na linha **Exatos** e 2% na linha **Melhores Resultados**. Em vista desses resultados, essas adaptações serão o foco das próximas etapas deste estudo.

Tabela 4: Resultado sobre a variação do R-BEP.

Informações	R-BEP	R-BEP-1	R-BEP-2
Exatos	**93**	91	85
Melhores Resultados	418	**432**	365

Fonte: Produção do próprio autor.

Tabela 5: Resultados sobre a variação da Busca Local.

Informações	BL	BL-1	BL-2	BL-3
Exatos	96	**97**	96	**97**
Melhores Resultados	503	512	504	**513**

Fonte: Produção do próprio autor.

5.3 Estudo sobre o algoritmo Iterated Local Search (ILS) aplicado ao problema d-MBV

O algoritmo ILS, proposto por RAMÍREZ (2018), é estruturado em três fases principais: o algoritmo construtivo, a busca local e a perturbação. Até esse momento, foram implementadas adaptações nas duas primeiras etapas. Agora, o foco passa a ser a fase de perturbação, cujo objetivo é diversificar as soluções e evitar que o algoritmo fique preso em ótimos locais. Como a eficácia da perturbação depende das fases anteriores, os testes futuros serão baseados na execução completa do ILS, utilizando as melhores adaptações propostas: o algoritmo construtivo **R-BEP-1**, e as variações **BL-1** e **BL-3** da busca local.

O objetivo do algoritmo de perturbação é explorar novas soluções ao remover arestas conectadas a vértices classificados como *d-branch* e substituí-las por arestas que mantenham a conectividade da árvore geradora. Na versão original, os vértices *d-branch* são organizados em uma lista chamada *DB*, ordenada pelo grau dos vértices. O algoritmo remove arestas desses vértices e adiciona novas arestas com base na menor soma dos graus dos vértices conectados, caracterizando o caso onde $ModoPerturbacao = Grau$. A adaptação proposta utiliza a centralidade *PageRank*. A lista *DB* é agora ordenada pelos valores de *PageRank*, do menor para o maior, e as novas arestas são selecionadas com base na menor soma dos valores de *PageRank* dos vértices, sendo o caso onde $ModoPerturbacao = PageRank$.

Com o estudo da perturbação concluído e as adaptações finalizadas, as variações propostas neste projeto são apresentadas a seguir. Os resultados do ILS original servem como referência, enquanto as variantes **ILS-1** a **ILS-4** utilizam o algoritmo construtivo **R-BEP-1**, cada uma com diferentes configurações de busca local e perturbação. Assim, o **ILS** é o algoritmo *Iterated Local Search* original descrito em RAMÍREZ (2018), com o conjunto O_D desativado. A **ILS-1** utiliza busca local BL-1 e $ModoPerturbacao = Grau$; a **ILS-2** utiliza busca local BL-1 e $ModoPerturbacao = PageRank$; a **ILS-3** utiliza busca local BL-3 e $ModoPerturbacao = Grau$; e a **ILS-4** utiliza busca local BL-3 e $ModoPerturbacao = PageRank$.

As Tabelas 6 e 7 apresentam os resultados das execuções das variações do algoritmo ILS sobre as 525 instâncias. Essas tabelas indicam que a variação **ILS-4** alcançou os melhores resultados em 28 dos 41 grupos de instâncias. A integração da centralidade *PageRank* nos estágios do algoritmo construtivo R-BEP-1, na ordenação dos vértices do conjunto *DB* durante a busca local e em todas as fases de perturbação contribuiu para essa superioridade.

A Tabela 8 reforça essa melhoria, com a linha **Exatos** mostrando que o **ILS-4** alcançou o valor exato em 154 instâncias e apresentou os melhores resultados em 416 das 525 instâncias, superando as outras variações, que obtiveram desempenhos próximos, mas inferiores.

As Tabelas 9 e 10, mostram a diferença percentual entre cada variação do algoritmo ILS e o resultado ótimo para cada instância. Esta diferença é calculada utilizando a fórmula: $\text{Gap} = \frac{(\text{Algoritmo} - \text{Ótimo})}{\text{Ótimo}} \times 100$. Na tabela, a coluna **ILS** representa o gap do algoritmo original proposto por RAMÍREZ (2018). As colunas **ILS-1** a **ILS-4** correspondem às variações do algoritmo descritas anteriormente. É possível observar que as variações melhoraram os resultados, com os valores de *gaps* se aproximando cada vez mais de zero comparado ao resultado original, principalmente em instâncias maiores.

6 Conclusões

Encontrar soluções para o problema d-MBV é amplamente reconhecido como um desafio significativo na área de otimização. Este estudo teve como objetivo aprimorar e adaptar o algoritmo *Iterated Local Search* (ILS) proposto

Tabela 6: Resultados mínimos em instâncias pequenas e médias.

n'	m'	ILS	ILS-1	ILS-2	ILS-3	ILS-4
20	41.8	**0.76**	0.80	0.80	0.80	0.80
40	70.8	3.12	3.04	3.04	3.00	**2.96**
60	95.0	6.92	**6.72**	**6.72**	**6.72**	**6.72**
80	119.8	9.92	9.84	9.84	**9.80**	9.84
100	144.0	14.36	14.28	14.20	**14.16**	**14.16**
120	168.8	18.60	18.16	18.24	**18.12**	**18.12**
140	193.0	22.48	22.00	21.96	21.92	**21.80**
160	217.8	26.72	26.32	26.20	26.20	**26.04**
180	242.0	30.72	30.16	30.24	**30.04**	30.12
200	266.8	34.60	33.96	33.72	33.80	**33.64**
250	321.0	46.80	46.20	46.08	**45.96**	**45.96**
300	380.0	59.92	58.88	58.84	58.84	**58.76**
350	434.8	70.84	70.48	70.48	70.24	**70.20**
400	489.0	85.12	83.96	83.88	**83.84**	83.92
450	548.0	96.56	95.60	95.64	95.44	**95.36**
500	602.8	110.28	109.20	109.24	**108.88**	109.08

Fonte: Produção do próprio autor.

Tabela 7: Resultados mínimos em instâncias grandes.

n'	m'	ILS	ILS-1	ILS-2	ILS-3	ILS-4
600	637	184.8	**184.2**	**184.2**	**184.2**	**184.2**
600	674	170.4	169.4	**169.0**	169.4	**169.0**
600	712	155.8	**152.8**	153.0	**152.8**	153.0
600	749	143.2	141.8	**141.2**	141.6	**141.2**
600	787	132.6	129.8	129.8	129.8	**129.6**
700	740	216.0	216.0	**215.4**	215.8	**215.4**
700	780	201.4	199.8	199.8	**199.4**	199.6
700	821	185.2	183.2	**182.8**	**182.8**	183.0
700	861	168.8	167.2	166.8	167.2	**166.6**
700	902	161.0	**156.2**	157.0	156.6	156.8
800	843	247.0	247.2	247.0	**246.8**	**246.8**
800	886	230.8	229.6	**229.2**	229.4	**229.2**
800	930	213.6	211.4	210.8	211.0	**210.6**
800	973	200.8	**197.2**	198.0	**197.2**	197.8
800	1017	184.2	180.4	**179.8**	180.0	180.0
900	944	282.0	281.0	**280.6**	281.0	**280.6**
900	989	263.4	262.2	262.2	**261.8**	**261.8**
900	1034	246.0	243.6	243.8	**243.2**	**243.2**
900	1079	230.2	227.0	227.2	**226.6**	227.0
900	1124	214.6	**210.0**	210.2	**210.0**	210.4
1000	1047	**314.0**	**314.0**	**314.0**	**314.0**	**314.0**
1000	1095	294.2	292.8	**292.4**	**292.4**	**292.4**
1000	1143	276.8	274.4	**273.8**	274.4	**273.8**
1000	1191	259.8	255.0	255.4	255.0	**254.6**
1000	1239	244.8	239.4	**239.0**	239.4	**239.0**

Fonte: Produção do próprio autor.

Tabela 8: Resultado geral sobre todas as adaptações no ILS.

Informações	**ILS**	**ILS-1**	**ILS-2**	**ILS-3**	**ILS-4**
Exatos	101	137	142	149	**154**
Melhores Resultados	204	351	374	398	**416**

Fonte: Produção do próprio autor.

por RAMÍREZ (2018). As principais adaptações incluíram a integração de uma nova heurística da literatura, especificamente o algoritmo construtivo desenvolvido por OLIVEIRA (2023), e a incorporação da centralidade *PageRank*.

O algoritmo ILS é composto por três etapas distintas. A inclusão do algoritmo construtivo R-BEP resultou em uma melhoria substancial nos resultados obtidos, demonstrando sua eficácia na abordagem do problema. Por outro lado, a integração da centralidade *PageRank* na fase construtiva não gerou um impacto significativo no desempenho, com a melhor variação apresentando uma média $0,6\%$ de melhoria. Este resultado pode ser atribuído ao fato de que o algoritmo construtivo não depende fortemente da etapa onde o *PageRank* foi integrado. A influência predominante para a solução gerada é atribuída a roleta probabilística e da necessidade de manter a conectividade em uma árvore geradora.

Na etapa da Busca Local, mesmo que pequena foi visível uma melhora nos resultados, os dois algoritmos que

Tabela 9: Gap dos resultados mínimos em instâncias pequenas e médias comparados ao exato.

n'	m'	ILS	ILS-1	ILS-2	ILS-3	ILS-4
20	41.8	**18.75**	25.00	25.00	25.00	25.00
40	70.8	13.04	10.14	10.14	8.70	**7.25**
60	95.0	16.89	**13.51**	**13.51**	**13.51**	**13.51**
80	119.8	7.36	6.49	6.49	**6.06**	6.49
100	144.0	7.81	7.21	6.61	**6.31**	**6.31**
120	168.8	13.69	11.00	11.49	**10.76**	**10.76**
140	193.0	7.46	5.16	4.97	4.78	**4.21**
160	217.8	10.96	9.30	8.80	8.80	**8.14**
180	242.0	5.64	3.71	3.99	**3.30**	3.58
200	266.8	6.00	4.04	3.31	3.55	**3.06**
250	321.0	4.93	3.59	3.32	**3.05**	**3.05**
300	380.0	4.46	2.65	2.58	2.58	**2.44**
350	434.8	3.33	2.80	2.80	2.45	**2.39**
400	489.0	4.01	2.59	2.49	**2.44**	2.54
450	548.0	3.43	2.40	2.44	2.23	**2.14**
500	602.8	3.34	2.32	2.36	**2.02**	2.21

Fonte: Produção do próprio autor.

Tabela 10: Gap dos resultados mínimos em instâncias grandes comparado ao exato.

n'	m'	ILS	ILS-1	ILS-2	ILS-3	ILS-4
600	637	0.5	**0.2**	**0.2**	**0.2**	**0.2**
600	674	1.9	1.3	**1.1**	1.3	**1.1**
600	712	3.5	**1.5**	1.6	**1.5**	1.6
600	749	3.2	2.2	**1.7**	2.0	**1.7**
600	787	5.4	3.2	3.2	3.2	**3.0**
700	740	0.7	0.7	**0.5**	0.7	**0.5**
700	780	1.7	0.9	0.9	**0.7**	0.8
700	821	2.9	1.8	**1.6**	**1.6**	1.7
700	861	2.9	2.0	1.7	2.0	**1.6**
700	902	4.4	**1.3**	1.8	1.6	1.7
800	843	0.6	0.7	0.6	**0.5**	**0.5**
800	886	1.4	0.9	**0.7**	0.8	**0.7**
800	930	2.5	1.4	1.2	1.2	**1.1**
800	973	3.4	**1.5**	2.0	**1.5**	1.9
800	1017	4.5	2.4	**2.0**	2.2	2.2
900	944	0.9	0.5	**0.4**	0.5	**0.4**
900	989	1.6	1.2	1.2	**1.0**	**1.0**
900	1034	2.2	1.2	1.3	**1.1**	**1.1**
900	1079	3.1	1.7	1.8	**1.5**	1.7
900	1124	4.2	**1.9**	2.0	**1.9**	2.1
1000	1047	**0.6**	**0.6**	**0.6**	**0.6**	**0.6**
1000	1095	1.4	1.0	**0.8**	**0.8**	**0.8**
1000	1143	2.1	1.2	**1.0**	1.2	**1.0**
1000	1191	3.4	1.5	1.7	1.5	**1.4**
1000	1239	4.1	1.8	**1.6**	1.8	**1.6**

Fonte: Produção do próprio autor.

foram ordenados pelo *PageRank* tiveram o melhor desempenho e a variação nos métodos de escolha de soluções vizinhas teve impacto mínimo, em geral as melhores adaptações da Busca Local tiveram um melhora de $1,5\%$ comparado ao algoritmo original.

A etapa final do algoritmo ILS envolve a Perturbação e a Busca Local, que são executadas em um laço até que o critério de parada seja atendido. Dessa forma, para avaliar o desempenho da Perturbação, é necessário testar o algoritmo ILS em sua totalidade. Os resultados obtidos mostraram que o algoritmo adaptado com a Perturbação regida pelo *PageRank* alcançou os melhores resultados, apresentando uma melhoria média de $9,7\%$ em comparação com as outras variações, evidenciando a eficácia dessa abordagem integrada com a nova centralidade.

Conclui-se que a principal melhoria nos resultados foi a inclusão do novo algoritmo construtivo. A integração do PageRank mostrou-se benéfica, com ganhos iguais ou levemente superiores tanto no algoritmo construtivo quanto na busca local. No entanto, a maior melhoria foi na etapa de perturbação, onde a distribuição mais ampla dos valores de PageRank possibilitou uma escolha mais precisa das arestas a serem substituídas. A eficácia do PageRank na seleção de vértices d-branch contribuiu significativamente para o aprimoramento nesta etapa do algoritmo.

Para trabalhos futuros, recomenda-se a continuidade do estudo, aprofundando o impacto da centralidade *PageRank*, que pode oferecer *insights* valiosos para pesquisas subsequentes. E durante o próximo ano, quando será dada

continuidade a esse trabalho, com a renovação desse projeto de iniciação científica, planeja-se explorar o algoritmo de decomposição de pontes, originalmente utilizado no ILS, e a integração das adaptações definidas durante este estudo com o algoritmo GRASP, para avaliar como essas combinações afetam a busca de boas soluções para o problema d-MBV.

Agradecimentos

Agradeço à Fundação de Amparo à Pesquisa e Inovação do Espírito Santo (FAPES) pelo apoio financeiro fundamental para o desenvolvimento deste projeto de Iniciação Científica. Este suporte foi essencial para a continuidade e sucesso da pesquisa. Gostaria também de expressar minha sincera gratidão aos meus orientadores, cujo conhecimento, orientação e dedicação foram cruciais para a realização deste trabalho.

Referências Bibliográficas

BRIN, S. & PAGE, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30:107–117.

CARRABS, F., CERULLI, R., GAUDIOSO, M., & GENTILI, M. (2013). Lower and upper bounds for the spanning tree with minimum branch vertices. *Computational Optimization and Applications*, 56:1–34.

GARGANO, L. et al. (2002). Spanning trees with bounded number of branch vertices. In WIDMAYER, P. et al., editors, *Automata, Languages and Programming*, page 355–365, Berlin, Heidelberg. Springer Berlin Heidelberg.

MELO, R. A., SAMER, P., & URRUTIA, S. (2016). An effective decomposition approach and heuristics to generate spanning trees with a small number of branch vertices. *Computational Optimization and Applications*, 65(3):821–844.

MERABET, M., DESAI, J., & MOLNÁR, M. (2018). A generalization of the minimum branch vertices spanning tree problem. In *International Symposium in Combinatorial Optimization*. Springer.

NETTO, P. O. B. & JURKIEWICZ, S. (2017). *Grafos: Introdução e Prática*. Editora Edgard Blucher Ltda., 2 edition.

OLIVEIRA, A. B. (2023). Heurísticas gulosas e randomizadas para o problema da Árvore geradora com número mínimo de ramificações. Mestre em computação, Programa de Pós-Graduação em Computação, Universidade Federal Fluminense, UFF, Brasil, Niterói.

PUCHINGER, J. & RAIDL, G. (2005). Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification. In *Lecture Notes in Computer Science*, volume 3562, pages 41–53.

RAMÍREZ, J. R. M. (2018). *Heuristic and Exact Approaches for Some Combinatorial Optimization Problems on Graphs*. Doutorado em ciência da computação, Programa de Pós-Graduação em Computação, Universidade Federal Fluminense, UFF, Brasil, Niterói.

RAMÍREZ, J. R. M., FROTA, Y. A. D. M., & MARTINS, S. L. (2018). An exact and heuristic approach for the d-minimum branch vertices problem. *Computational Optimization and Applications*, 71(3):829–855.

TALBI, E.-G. (2009). *Metaheuristics: From Design to Implementation*. Wiley Publishing.