Universidade Federal Fluminense
Instituto de Computação
Departamento de Ciência da Computação

RELATÓRIO FINAL DE ATIVIDADES DE INICIAÇÃO CIENTÍFICA

Victor Pereira de Lima
Bolsista Iniciação Científica/FAPERJ
Bacharelado em Ciência da Computação
Data de Ingresso no Programa: 11/2022
Orientadora: Simone de Lima Martins

Área do Conhecimento: 1.03.03.00-6 — Área do CNPq

CAMPUS NITERÓI, 2024

VICTOR PEREIRA DE LIMA
SIMONE DE LIMA MARTINS

UM ALGORITMO ITERATED LOCAL SEARCH HÍBRIDO PARA SOLUÇÃO DO PROBLEMA DA ÁRVORE GERADORA COM NÚMERO MÍNIMO DE VÉRTICES D-BRANCH

Relatório de Pesquisa do Programa de Iniciação Científica da FAPERJ.

NITERÓI, 2024

SUMÁRIO

RESUMO

Dado um grafo G conexo, não direcionado e não valorado, o problema da Árvore Geradora com Número Mínimo de Vértices d-branch (d-MBV) consiste em encontrar uma árvore geradora que possua a menor quantidade de vértices com grau estritamente maior que d, para $d \geq 2$. A aplicação direta desse problema é, por exemplo, na alocação de switches em projetos de redes óticas. Aplicações de sucesso de estratégias híbridas em problemas de Otimização trazem o uso de técnicas de mineração de dados combinadas com meta-heurísticas, com o objetivo de viabilizar a extração e utilização de padrões que representem soluções de boa qualidade. Centralidades de grafos são famosas na literatura pelo seu uso como ferramenta para análise de redes. Estão relacionadas com informações estruturais do grafo, permitindo determinar a importância de um vértice em um grafo, de acordo com diferentes critérios. O objetivo geral deste trabalho de pesquisa consiste em investigar uma solução para o problema d-MBV por meio de um algoritmo híbrido que considera a meta-heurística ILS combinada com centralidades de grafos e técnicas de mineração de dados para extração de padrões de boas soluções.
Palavras-chave: problema d-MBV; Otimização Combinatória; Meta-heurísticas; Grafos
.

INTRODUÇÃO

Problemas de otimização combinatória considerados difíceis consistem em verdadeiros desafios para a obtenção de soluções ótimas por algoritmos eficientes. As meta-heurísticas têm se mostrado uma ferramenta muito útil na solução de problemas de Otimização Combinatória. Não garantem a obtenção do ótimo, mas são algoritmos eficientes que geram soluções consideradas de muito boa qualidade, para as mais diversas aplicações. A pesquisa em meta-heurísticas trata características como robustez, qualidade de solução e eficiência como objeto de investigação constante na busca por algoritmos cada vez mais poderosos. Neste sentido, estratégias como hibridizações entre meta-heurísticas e destas com métodos exatos são muito encontradas na literatura, buscando principalmente eficiência e maior flexibilidade no tratamento de problemas muito grandes. Além disso, o conhecimento de características do problema a ser resolvido e de seu espaço de soluções são muito relevantes para a escolha e adaptação da meta-heurística para a sua solução.

No início dos anos 90, pesquisas em Banco de Dados identificaram que, por meio de estratégias específicas, informações importantes, a princípio desconhecidas, poderiam ser extraídas de grandes bases de dados. Esses estudos pioneiros levaram à consolidação da Mineração de Dados. Em busca da identificação de informações importantes nas soluções de um problema de otimização, surge a ideia de se integrar técnicas

de mineração de dados com meta-heurísticas com o objetivo de viabilizar a extração de padrões que representem boas soluções para problemas de otimização e sua posterior utilização na busca por melhores soluções em um menor tempo computacional.

Em Teoria dos Grafos, distâncias entre vértices de um grafo são usadas para definir medidas de centralidade. Uma medida de centralidade é uma função definida a partir das distâncias relativas entre os vértices, com o objetivo de classificar um conjunto de elementos, do mais para o menos central, de acordo com um particular critério. Dessa forma, podem ser usadas para classificar os vértices de acordo com a 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). Graus dos vértices são exemplos de invariantes de um grafo. As medidas de centralidade têm sido usadas como ferramentas importantes na análise de redes complexas. Permitem, por exemplo, estimar o quanto uma pessoa é influente em uma rede social ou como uma estrada é bem utilizada em uma rede urbana.

Muitos dos problemas de Otimização Combinatória podem ser formulados usando representação por grafos, e entre eles estão os que buscam uma árvore geradora mínima. Esse tipo de problema aparece em diversas aplicações como redes de transporte, de telecomunicações, de distribuição, entre outros, normalmente para o planejamento de redução de custos. Dado um grafo conexo e não orientado G, uma árvore geradora T de G é um subgrafo conexo e acíclico que liga todos os vértices de G. Com motivação das mais diferentes aplicações, variações do problema da árvore geradora mínima foram surgindo. A maior ou menor dificuldade do problema, assim como quais elementos do grafo estão no foco, depende do critério de otimização a ser aplicado para obtenção da árvore e pode estar associado com vértices, arestas, e custos.

O Problema da Árvore Geradora com Número Mínimo de Vértices Branch (do inglês, Minimum Branch Vertices Problem ou simplesmente MBV) consiste em encontrar uma árvore geradora de um dado grafo G conexo, não direcionado e não valorado, que possua a menor quantidade de vértices com grau maior que 2, dentre todas as árvores geradoras de G. Tais vértices são denominados vértices branch. Esse problema foi proposto por Gargano et al. [1] a fim de determinar as melhores posições de alocação de switches. Uma generalização do MBV foi proposta em Merabet et al. [2] usando o conceito de k-branch, que é um vértice com grau estritamente maior que k+2. O problema k-MBV consiste em procurar uma árvore geradora com o número mínimo de vértices k-branch. Os autores provaram que esse problema é NP-hard para qualquer valor de k. Moreno et al. [3], a fim de simplificar a notação, introduziram o parâmetro d = k + 2, onde um vértice no grafo com grau estritamente maior que d, para $d \geq 2$, é um vértice d-branch. O problema d-MBV consiste em achar, em um grafo G não

direcionado com n vértices, uma árvore geradora com número mínimo de vértices d-branch. O problema que estamos interessados em investigar neste projeto é o d-MBV. No desenvolvimento desse projeto, o interesse da pesquisa é estudar e adaptar o algoritmo Iterated Local Search (ILS) [4] proposto em [3], para resolver o problema d-MBV, desenvolvendo uma versão híbrida que combine novas estratégias de diversificação, uso de centralidades e de técnicas de extração de padrões em soluções, com o objetivo de ajudar na convergência para ótimos locais de melhor qualidade. Para auxílio no processo de adaptação do algoritmo ILS, pretendemos lançar mão de ferramentas para análise de redes disponíveis na literatura, como o software de visualização Gephi [5].

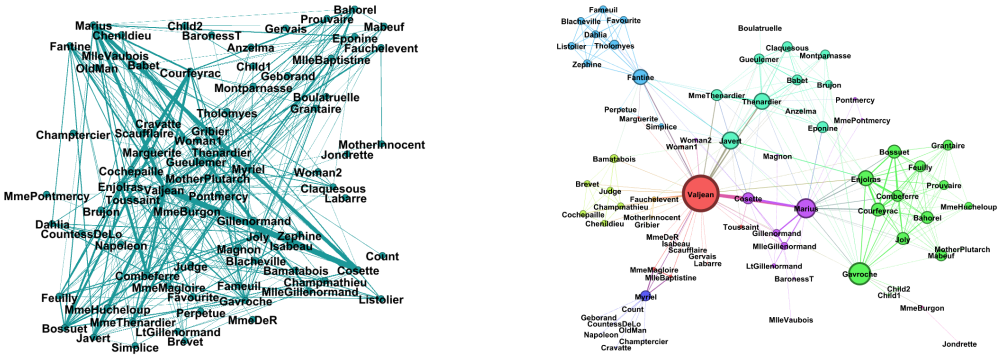
OBJETIVOS

Como principal objetivo deste projeto, destacamos o estudo de técnicas de solução de um problema de otimização relevante na literatura, envolvendo temas como algoritmos, otimização, teoria dos grafos e mineração de dados. Como objetivos específicos, temos: (i) estudar técnicas de busca local e meta-heurísticas para o problema de otimização d- MBV; (ii) estudar ferramentas para análise de dados em grafos; (iii) estudar centralidades de grafos e técnicas de Mineração de Dados, aplicando-as aos algoritmos de solução estudados para o problema d-MBV. São também objetivos deste projeto formar pessoal especializado, gerar propriedade intelectual e publicar trabalhos científicos relevantes. As motivações para a realização deste projeto de pesquisa são o acúmulo de conhecimento, tecnologias e inovações desenvolvidas na academia, no sentido de ampliar o escopo das pesquisas realizadas com vistas ao contínuo desenvolvimento científico e tecnológico para o benefício da sociedade.

MATERIAL E MÉTODOS

O estudo se desenvolveu a partir do uso de um conjunto de dados gerado em Moreno et al. [3] por meio de um gerador de grafos fornecido por Merabet et al. [2]. Chamaremos de SPD este conjunto, que contém 25 instâncias de grafos geradas randomicamente para cada número de vértices n, onde $n \in$ {20, 40, 60, 80, 100, 120, 140, 160, 180, 200, 250, 300, 350, 400, 450, 500, 600, 700, 800, 900, 1000}. A meta-heurística ILS originalmente desenvolvida para resolver o problema d-MBV está codificada na linguagem C, e o modelo matemático para obter soluções exatas para o problema foi resolvido utilizando-se o IBM ILOG CPLEX Optimization Studio 22.1.1. Para trabalhar com os grafos, gerar as imagens para visualização das soluções, e calcular os valores das centralidades, foi utilizada a biblioteca C versão 0.9.9 do pacote

open source de análise de redes igraph [6]. A etapa de análise teve como ferramentas o software livre de visualização e exploração de grafos Gephi, que serviu para buscar ver possíveis relações entre os vértices d-branch e as medidas de centralidade, e o Google Colab, onde utilizando a linguagem Python e as bibliotecas Pandas, Numpy, Plotly, e Matplotlib, foi possível executar a análise de dados e produzir os gráficos, que explicitaram as conclusões. Um exemplo do potencial elucidativo do Gephi é visto abaixo, onde temos exposto o grafo da rede de relacionamentos dos personagens da obra Os Miseráveis, do romancista francês Victor Hugo.



(a) Sem layout. (b) Com layout.

Figura 1. Grafo da rede de relacionamentos dos personagens de Os Miseráveis, com e sem layout do Gephi. Percebe-se que o programa explicita comunidades bem demarcadas, assim como os principais personagens do romance.

Definição das Centralidades. O estudo realizado neste trabalho visa pesquisar se medidas de centralidade dos nós do grafo podem ser utilizadas de alguma maneira para detectar nós do grafo que devem ou não ser d-branches na árvore geradora.

Considerando um grafo não-direcionado e não ponderado $G = (V,E)$, onde $n = |V|$ e $m = |E|$, definimos, conforme [7], as centralidades:

- **Degree (Grau):** É o número de vértices conectados a um vértice v, denotado por $d(v)$. Também define-se que é o número de arestas incidentes a v.

- **Closeness (Proximidade):** Esta centralidade metrifica o quão próximo um vértice está dos outros; num contexto em que os nós representam imóveis, por exemplo, esta medida seria muito útil para definir os mais bem centralizados. Dado um $v \in V$, seu valor de closeness corresponde a $c(v) = (\sum_{a \in V, a \neq v} dist(v,a))^{-1}$, onde $dist(\cdot,\cdot)$ é a distância entre dois nós de G.

- **Betweenness (Intermediação):** Indica o quanto um nó v controla a comunicação numa rede, neste caso, a soma, para todos os outros vértices, da

proporção de vezes em que v aparece no caminho quando se busca o caminho mais curto entre dois vértices. Logo, o valor de betweenness é calculado por $b(v) = \sum_{a,b \neq v}(g_{avb}/g_{ab})$, onde g_{ab} é o número de menores caminhos entre a e b, e g_{avb} é o total destes que contêm v.

- **Eigenvector (Autovetor):** Nesta centralidade parte-se do princípio de que a conexão de um nó com vizinhos influentes contribui para a influência do mesmo. Para o cálculo define-se um vetor $p \in \mathbb{R}^n$, onde cada entrada representa os valores desta centralidade associados a cada $v \in V$, e a matriz de adjacência $A_{n \times n}$ associada ao grafo G. Inicialmente, $p = p_0 = (1,1,\ldots,1)^T$, em seguida ocorre o cálculo iterativo $p_k = Ap_{k-1}$, para $k > 0$, onde p_k é normalizado pelo valor de sua maior entrada antes de ser passado para a iteração seguinte. Para um valor k significativo de passos, temos que $p_k = Ap_{k-1} \approx \lambda p_{k-1}$, e por meio do Método da Potência encontramos o autovetor dominante, que está associado a λ, que é o maior dos autovalores. As entradas deste autovetor final que serão os valores finais desta centralidade associados a cada vértice v.

- **PageRank:** Utilizado pelo mecanismo de busca Web do Google para ranquear resultados conforme relevância, o PageRank, assim como o Eigenvector, considera a influência da vizinhança; como nasceu do contexto da web, modela os vértices de uma rede como páginas, e as arestas como hyperlinks, resultando na probabilidade de se chegar a uma página (vértice) ¨surfando¨ aleatoriamente pelos hyperlinks (arestas). Inicialmente, todos os vértices são inicializados com um mesmo valor de PageRank, que pode ser $1/n$. Se um vértice $v \in V$ com PageRank $p(v)$ aponta para $k > 0$ outros vértices, estes passarão a ter um PageRank igual a $\frac{p(v)}{k}$, e assim segue-se para cada nova ligação, iterativamente até o algoritmo convergir. No entanto, surgem alguns problemas nesta formulação, como no caso de ciclos, onde o algoritmo entra em loop ao distribuir os valores da centralidade circularmente. A solução desenvolvida foi a criação de um fator de amortecimento, que ajuda a simular a probabilidade de uma pessoa se entediar e sair da navegação. Finalmente, para grafos não-direcionados, formula-se que $p(v) = \frac{1-damping}{N} + damping * \sum_{\{v,w\} \in E} \frac{p(w)}{d(w)}$, onde $damping$ é a probabilidade de navegação aleatória (normalmente 0.85), e $d(w)$ é o grau do vértice w.

A Meta-heurística Iterated Local Search (ILS). A formulação matemática desenvolvida para o problema d-MBV em [3] é capaz de fornecer soluções ótimas para instâncias até um determinado tamanho. Para tentar se achar soluções de boa qualidade em

tempo computacional viável para instâncias de maior tamanho foi desenvolvida a meta-heurística Iterated Local Search (ILS) [4], apresentada na Figura 2, capturada de [3].

Algorithm 2: ITERATED LOCAL SEARCH

Input: The graph G.
Output: A valid solution T.

```
1 x_0 ← Initial_Solution(G)
2 x* ← Local_Search(G)
3 repeat
4 |   x' ← Perturbation(x*)
5 |   x*' ← Local_Search(x')
6 |   x* ← AcceptanceCriterion(x*, x*', history)
7 until termination condition met
8 return x*
```

Figura 2. Algoritmo da Meta-heurística Iterated Local Search (ILS).

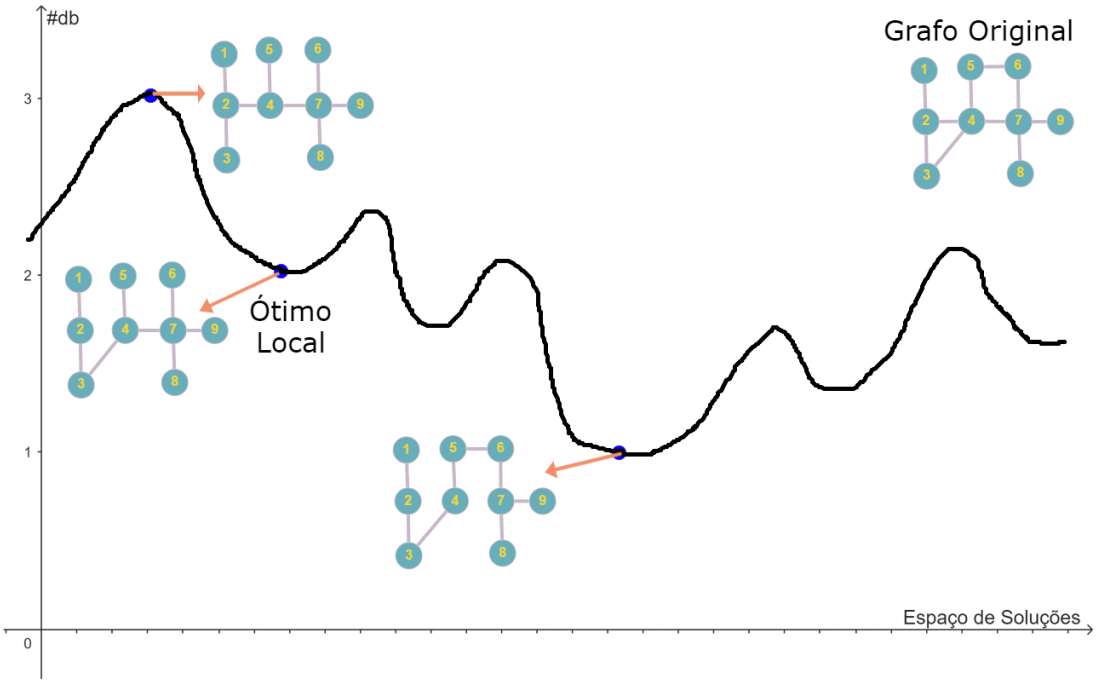


Figura 3. Imagem criada para auxílio ao entendimento e visualização da meta-heurística ILS. Grafos criados utilizando a plataforma web Graph Online.

Fonte: autoria própria.

Para facilitar o entendimento da heurística, vamos simular que nosso grafo original é o apresentado no canto superior direito da Figura 3, onde $\#db$ é a quantidade de vértices d-branch do grafo contido no espaço de soluções. Na linha 1 é gerada uma solução inicial x_0, que podemos assumir como o grafo com $\#db = 3$ da Figura; na linha 2,

efetua-se uma busca na vizinhança da solução inicial x_0, visando encontrar uma outra solução que tenha $\#db$ menor. Na linha 3 entra-se num loop onde são executadas várias iterações onde, buscando se evitar ficar preso a um ótimo local, como a solução em que $\#db = 2$, ocorre uma perturbação para produzir um novo resultado visando levar a busca para outra área do espaço de soluções, como a do "vale"contendo o provável Ótimo Global, $\#db = 1$. É feita outra busca local nesta nova solução e avalia-se o resultado pelo critério de aceitação: a nova solução deve ter uma quantidade de nós d-branch menor do que a melhor obtida até o momento. Estas iterações acabam quando o procedimento de perturbação não consegue mais gerar uma solução nova.

O algoritmo da Figura 2 apresenta o princípio geral da meta-heurística ILS. Os algoritmos que compõem a heurística ILS desenvolvida para resolver o problema d-MBV utilizam medidas da centralidade Grau para avaliar os vértices do grafo.

Como será mostrado na subseção ¨Análise das Centralidades¨ da seção ¨Resultados e Discussões¨, avaliações experimentais mostraram que valores da centralidade Pagerank de um vértice podem indicar que ele é um vértice d-branch. Então, modificamos os algoritmos da heurística ILS que usa a centralidade Grau para usar Pagerank para verificar o impacto de utilizar uma outra medida de centralidade. A seguir, explicaremos estes algoritmos e quais foram e onde ocorreram as mudanças efetuadas para adaptá-los para o uso da centralidade PageRank.

O algoritmo para gerar a Solução Inicial (Initial Solution) mostrado na Figura 4 visa produzir uma árvore geradora T com uma quantidade de vértices d-branch reduzida. Sua entrada é o grafo $G = (V,E)$ sem pontes, onde cada vértice $v \in V$ possui sua quantidade de pontes associada representada por $l(v)$, e temos dois conjuntos: O_D, conjunto dos vértices que serão obrigatoriamente d-branch na solução ótima, e A, conjunto de arcos de G construído gerando-se dois arcos para cada aresta de E.

Algorithm 3: INITIAL SOLUTION

Input: A graph $G = (V, E)$ without bridges, the value $l(v)$ associated to each vertex, the set O_D and the set of arcs A such that for all $\{i, j\} \in E$, then $(i, j), (j, i) \in A$.
Output: A spanning tree T of the graph

1 $T \leftarrow (V, \emptyset)$
2 $m \leftarrow 0$
3 **repeat**
4 Find the arc $(u, v) \in A$ such that $d_T(u) = 0$ and $d_T(v) + l(v) \neq d$ and whose associated edge $\{u, v\}$ has minimum w_{sOD} value; then, add the edge $\{u, v\}$ in T.
5 $m \leftarrow m + 1$
6 **until** *There is no arc that satisfies this condition*
7 **repeat**
8 Consider criteria (a)-(f), whose priorities are in descending order ((a) has the highest priority). Find an arc (u, v) in A with associated edge $\{u, v\}$, such that $T \cup \{u, v\}$ is a tree and no other arc satisfies a higher priority criterion.

(a) arc (u, v) has maximum value $d_G(v) + l(v) + n \cdot f_D(v)$ such that $d_T(u) + l(u) < d$, $d_T(v) + l(v) > d$.
(b) edge $\{u, v\}$ has maximum w_{aOD} in T, and arc (u, v) with $d_T(u) + l(u) > d$, $d_T(v) + l(v) > d$.
(c) edge $\{u, v\}$ has minimum w_{sOD} in T and arc (u, v) with $d_T(u) + l(u) < d$, $d_T(v) + l(v) < d$.
(d) edge $\{u, v\}$ has maximum w_{aOD} in T and arc (u, v) with $d_T(u) + l(u) > d$, $d_T(v) + l(v) = d$.
(e) edge $\{u, v\}$ has maximum w_{aOD} in T and arc (u, v) with $d_T(u) + l(u) < d$, $d_T(v) + l(v) = d$.
(f) edge $\{u, v\}$ has maximum w_{aOD} in T and arc (u, v) with $d_T(u) + l(u) = d$, $d_T(v) + l(v) = d$.

Add $\{u, v\}$ in T
$m \leftarrow m + 1$
9 **until** $m < |V| - 1$
10 **return** T

Figura 4. Algoritmo para gerar a solução inicial (Iniital Solution).

O algoritmo é construtivo, pois, inicializamos T como um grafo sem arestas, e construímos a árvore geradora final inserindo arestas seguindo condições que evitem a criação de vértices d-branch. Há a definição de novas informações para orientar a escolha, como os valores das funções $d_G(\cdot)$ e $d_T(\cdot)$ (grau de vértice em G e T, respectivamente), e os pesos w_{sOD} e w_{aOD} para arestas $\{u,v\}$, definidos abaixo nas Equações 1 e 2.

$$w_{sOD} = d_G(u) + l(u) + d_G(v) + l(v) - f_D(u) \cdot n - f_D(v) \cdot n \tag{1}$$

$$w_{aOD} = d_G(u) + l(u) + d_G(v) + l(v) + f_D(u) \cdot n + f_D(v) \cdot n \tag{2}$$

onde

$$f_D(v) = \begin{cases} 1, & v \in O_D \\ 0, & v \notin O_D \end{cases} \tag{3}$$

As linhas 3 a 6 do algoritmo selecionam os arcos (u,v) de A seguindo condições que garantam que as arestas de E associadas sejam inseridas sem gerar ciclos e novos d-branches, assim como, ao selecionar arcos com valor mínimo de w_{sOD}, priorizam-se

Algorithm 4: LOCAL SEARCH

Input: The current solution T_{curr}.
Output: The best solution T_{best}.

```
1 T_best ← T_curr
2 repeat
3 |   improvement ← 0
4 |   T ←FirstBest_Neighbor(G,T_best, O_D)
5 |   if T ≠ T_best then
6 |   |   T_best ← T
7 |   |   improvement ← 1
8 until improvement = 0
9 return T_best
```

Figura 5. Algoritmo de Busca Local (Local Search).

aqueles que ligam vértices d-branch com grau em G baixo, com isso os vértices d-branch obrigatórios terão mais arestas adicionadas. O loop finaliza quando não houver mais arcos que satisfaçam as condições.

Para finalizar a construção da árvore, no loop iniciado na linha 7, inserem-se novos arcos conforme os critérios (a) a (f), ordenados decrescentemente conforme a prioridade. Estes critérios visam inserir arestas de forma a evitar a geração de nós d-branch, o que ocorre utilizando (a) a (c), porém, (d) e (e) geram um novo, enquanto (f) produz dois. O critério (a) usa as informações de grau para escolher um arco (u,v) tal que u não seja d-branch, v seja e possua alto grau; (a) sendo insatisfeito, segue-se para (b), onde a aresta associada ao arco com valor máximo de w_{aOD} em T e que liga dois vértices d-branch é escolhida; em (c) a aresta é escolhida se tiver valor mínimo de w_{sOD} e ligar nós não-d-branch, priorizando vértices de baixo grau e deixando os de alto para análise posterior; (d) a (f) produzem novos vértices d-branch, mas compensam por escolher arcos com vértices de grau alto, o que aumenta a chance do próximo a ser escolhido já ser incidente a um dos nós do que foi escolhido anteriormente. Após todo esse processo de decisão, a aresta associada ao arco selecionado é inserida em T. O loop executa até que sejam escolhidas as $|V| - 1$ arestas da árvore geradora inicial, retornada na linha 10.

Adaptando o algoritmo para o uso da centralidade PageRank definimos uma nova dinâmica de passos que utilizarão as medidas de Grau e PageRank. Foram realizadas duas modificações. A primeira foi substituir em todo o algoritmo os pesos w_{sOD} por $w(\{u,v\}) = PageRank(u) + PageRank(v)$. E a segunda foi substituir no critério (a) $d_G(v) + l(v) + n.F_D(v)$ por $PageRank(v)$.

O algoritmo de Busca Local (Local Search), mostrado na Figura 5, é o mesmo, seja para Grau ou PageRank. Ele recebe a solução inicial (T_{curr}) e, no loop das linhas 2 a 8 executa melhorias visando encontrar soluções vizinhas com menor quantidade de vértices d-branch, até não conseguir produzir novas. Este procedimento de busca por uma solução melhor ocorre no procedimento ¨FirstBest_Neighbor¨.

Algorithm 5: FIRSTBEST_NEIGHBOR

Input: A graph $G = (V, E)$ without bridges, the current solution T_{curr} and the set O_D of obligatory d-branch vertices.
Output: The solution T.

```
1  T ← T_curr
2  L_D ← O_D
3  DB ← {v ∈ V \ O_D | d_T(v) + l(v) > d}
4  Sort DB in ascending order by value (d_T(v) + l(v))
5  foreach v ∈ DB do
6      foreach u ∈ N_T(v) do
7          Remove the edge {u, v} from T
8          e ←Find_Edge(u, v, T, L_D)
9          if e ≠ ∅ then
10             Insert the edge e in T
11         else
12             Insert the edge {u, v} in T
13
14         /* v or u are d−branch vertices no more */
15         if d_T(v) + l(v) ≤ d or ((d_T(u) + l(u) ≤ d) ∧ (u ∈ DB)) then
16             return T
17     L_D ← L_D ∪ {v}
18 return T
```

Figura 6. Algoritmo de FirstBest_Neighbor().

Em FirstBest_Neighbor (Figura 6) o objetivo é executar trocas de arestas de forma a reduzir o grau dos vértices d-branch a um valor menor ou igual a d. O algoritmo recebe como entrada o grafo $G = (V,E)$ sem pontes, a melhor solução atual encontrada (T_{curr}), e o conjunto de nós que serão obrigatoriamente d-branch (O_D). Nas linhas 1 e 2 as variáveis T e L_D recebem T_{curr} e O_D, respectivamente. Como mudanças nos elementos de O_D não afetarão a quantidade de d-branches, estas são executadas nos vértices d-branch não obrigatórios, contidos no conjunto DB, definido na linha 3. Em seguida ordenam-se os seus nós crescentemente conforme o valor de grau em T de cada um, pois nós de grau baixo vão requerer a retirada de menos ligações para que deixem de ser d-branch. Em seguida, executa-se um loop em que, para cada $v \in DB$, cada vértice u pertencente a sua vizinhança $N_T(v)$ é analisado de forma a remover $\{u,v\}$ e encontrar uma aresta $e \neq \{u,v\}$ que possa ser inserida em T sem aumentar a quantidade de vértices d-branch. Caso esta seja reduzida, a nova árvore-solução é retornada, se não,

adiciona-se v a L_D e continua-se o loop.

A mudança que ocorre para adaptar o algoritmo para fazer uso do PageRank corresponde a inicializar L_D como vazio ($L_D \leftarrow \emptyset$) na linha 2, definir DB como o conjunto de todos os nós d-branch (obrigatórios ou não), e ordená-los também crescentemente, mas conforme seus valores de PageRank. O resto do processo se mantém o mesmo executado entre as linhas 5 a 20, mas como a quantidade de pontes $l(v), v \in V$, não é utilizada na heurística com PageRank, a condição da linha 15 passa a ser $(d_T(v) \leq d) \vee ((d_T(u) \leq d) \wedge (u \in DB))$.

Algorithm 6: FIND EDGE

Input: Vertices u and v, a forest T, where $T_u = (V_u, E_u)$ and $T_v = (V_v, E_v)$ are the connected subtrees containing u and v respectively, and the set L_D of d-branch vertices.
Output: An edge $e \neq \{u, v\}$ between T_u and T_v or $\emptyset$ if not exists.

1 $U_1 \leftarrow L_D \cap V_u$
2 $U_2 \leftarrow \{w \in V_u \mid d_T(w) + l(w) < d\}$
3 $U_3 \leftarrow \{w \in V_u \setminus L_D \mid d_T(w) + l(w) > d\}$
4
5 $V_1 \leftarrow L_D \cap V_v$
6 $V_2 \leftarrow \{w \in V_v \mid d_T(w) + l(w) < d\}$
7 $V_3 \leftarrow \{w \in V_v \setminus L_D \mid d_T(w) + l(w) > d\}$
8
9 $e \leftarrow \{u', v'\}$ such that $\{u', v'\} \neq \{u, v\}$, $u' \in U_i$, $v' \in V_j$ and (i, j) is lexicographically smaller than any other valid pair $(i, j \in \{1, 2, 3\})$.
10 **return** e

Figura 7. Algoritmo de Find_Edge().

O algoritmo ¨Find Edge¨ utilizado na linha 8 da Figura 6 é mostrado na Figura 7, onde temos como entrada o vértice d-branch não obrigatório v, seu vizinho u, a floresta T composta das subárvores $T_u = (V_u,E_u)$ e $T_v = (V_v,E_v)$, subárvores obtidas quando $\{u,v\}$ é retirada de T , e o conjunto L_D contendo os nós d-branch de T.

A linha 1 inicializa o conjunto U_1 com os vértices de T_u que são d-branch conforme L_D, U_2 com os vértices de T_u que não são d-branch e possuem grau menor que d, ou seja, que podem receber uma nova ligação (aresta) e não se tornar d-branch, e U_3 com os vértices de T_u que são d-branch e não estão no conjunto L_D. O análogo acontece para V_1, V_2, e V_3, nesta ordem. O objetivo do algoritmo é se utilizar destes conjuntos para encontrar uma aresta diferente de $\{u,v\}$ que liga T_u a T_v produzindo uma nova árvore T com menos d-branches. Caso tal ligação não seja encontrada retorna-se $\emptyset$ (vazio).

A linha 9 apresenta a busca por uma aresta $\{u',v'\} \neq \{u,v\}$, onde $u' \in U_i, v' \in V_j$, e (i,j) é lexicograficamente menor que qualquer outro par $(i,j \in \{1,2,3\})$. A última condição será alcançada seguindo escolhas para as combinações de (i,j) conforme a ordem abaixo:

1. Dados $u' \in U_1, v' \in V_1$, escolhe-se $\{u',v'\}$ que tenha o menor valor de $d_G(u') + d_G(v') - n \cdot f_D(u') - n \cdot f_D(v')$, assim priorizando arestas com vértices obrigatoriamente d-branch e com grau pequeno.
2. Dados $u' \in U_1, v' \in V_2$, escolhe-se $\{u',v'\}$ que tenha o menor valor de $d_G(u') - n \cdot f_D(u') + d_G(v')$, com isso escolhendo uma aresta que liga um vértice d-branch obrigatório a um não-d-branch de baixo grau, para que este seja parte da solução.
3. Dados $u' \in U_1, v' \in V_3$, escolhe-se $\{u',v'\}$ que tenha o menor valor de $d_G(u') - n \cdot f_D(u') - d_T(v')$, objetivando ligar um vértice d-branch obrigatório a um não obrigatório de alto grau, que provavelmente será d-branch na solução ótima.
4. Dados $u' \in U_2, v' \in V_2$, escolhe-se $\{u',v'\}$ que tenha o menor valor de $d_G(u') + d_G(v')$, pois, como ambos não são d-branch e a sua ligação não produzirá novos, buscam-se escolher nós de grau baixo.
5. Dados $u' \in U_2, v' \in V_3$, escolhe-se $\{u',v'\}$ que tenha o menor valor de $d_G(u') - d_T(v')$, de forma a ligar um vértice não-d-branch de baixo grau em G a um d-branch não obrigatório de alto grau em T.
6. Dados $u' \in U_3, v' \in V_3$, escolhe-se $\{u',v'\}$ que tenha o maior valor de $d_T(u') + d_T(v')$, visando conectar vértices d-branch não obrigatórios que possuam alto grau em T, pois estes provavelmente estarão na solução ótima.

Neste caso, a adaptação para utilizar a medida PageRank foi bastante simples: nas linhas 2, 3, 6, e 7, apenas troca-se $d_T(w) + l(w)$ por $d_T(w)$.

Algorithm 7: PERTURBATION

Input: A graph $G = (V, E)$ without bridges, the current solution T_{curr} and the set O_D of obligatory d-branch vertices.
Output: The solution T.

1 $T \leftarrow T_{curr}$
2 $DB \leftarrow \{v \in V \setminus O_D \mid d_T(v) + l(v) > d\}$
3 Sort DB in ascending order by value $(d_G(v) + l(v))$
4 **foreach** $v \in DB$ **do**
5 **foreach** $u \in N_T(v)$ **do**
6 Remove the edge $\{u, v\}$ from T
7 Let $\{u', v'\} = argmin_{\{i,j\}}\{d_G(i) + d_G(j) \mid i \in T_u,$
 $j \in T_v$ such $d_T(i) = d$ or $d_T(j) = d$, but not both $\}$ and $\{u', v'\}$ is not forbidden
8 **if** $\{u', v'\} \neq \emptyset$ **then**
9 Insert the edge $\{u', v'\}$ in T and mark this edge as forbidden.
10 **return** T
11 **else**
12 Insert the edge $\{u, v\}$ in T
13 **return** T

Figura 8. Algoritmo de Perturbação (Perturbation).

O algoritmo de Perturbação (Figura 8) recebe o grafo $G = (V,E)$ sem pontes, a solução atual, T_{curr}, e o conjunto de nós d-branch obrigatórios, O_D. Ele foi criado visando permitir fugir dos ótimos locais e entregar diversidade à meta-heurística ILS. Isto é alcançado buscando substituir uma aresta $\{u,v\}$ em T que tenha pelo menos um d-branch não obrigatório entre os seus vértices, por uma aresta $\{u',v'\} \neq \{u,v\}$, onde $u' \in T_u$ e $v' \in T_v$, de forma a criar outro nó d-branch (u' ou v').

A execução começa na linha 1 ao se inicializar a variável T com a solução atual (T_{curr}). Em seguida o conjunto DB recebe como elementos os vértices d-branch não obrigatórios, que são ordenados crescentemente pelo valor grau na linha 3. Entre as linhas 4 a 12 acontece o loop em que, para cada nó $v \in DB$, cada vértice u de sua vizinhança $N_T(v)$ é analisado visando encontrar a aresta $\{u',v'\}$ que minimiza $d_G(i) + d_G(j)$, onde $i \in T_u$, $j \in T_v$, $d_T(i) = d$ ou $d_T(j) = d$ (mas não ambos), e ainda não foi inserida em T e marcada como proibida. A inserção de arestas ligando vértices com graus baixos em G dá chance para que, posteriormente, os de grau alto ¨roubem¨ as arestas dos nós d-branch durante a busca local.

O mecanismo de marcação de arestas como proibidas é importante para evitar a criação de ciclos quando se retira e insere a mesma aresta sem obter melhorias. Ele também dita a parada do método ILS quando não é possível se encontrar nós não marcados para continuar a perturbação.

Para a aplicação do PageRank neste algoritmo, na linha 2, DB recebe o conjunto contendo todos os vértices d-branch de T atual ($DB \leftarrow \{v \in V | d_T(v) > d\}$), que são ordenados de forma crescente segundo o PageRank de seus elementos na linha 3. Na

linha 7 a aresta $\{u',v'\}$ escolhida é aquela que minimiza $PageRank(i) + PageRank(j)$.

RESULTADOS E DISCUSSÃO

Análise das Centralidades. O primeiro estudo realizado foi avaliar quais medidas de centralidade se mostram melhores na identificação de nós d-branch. Para isso utilizaram-se as métricas de análise de dados F-Score e quatro medidas de ranqueamento (Ranking Measure). Tais métricas, adaptadas de Zhao et al. [8] e Pereira et al. [9] por Boeres et al. [7] para o problema d-MBV, são explicadas abaixo:

- **F-Score:** Métrica usada para determinar quão bem uma característica discrimina um conjunto de classes, sendo computada a partir das variâncias entre (between-group) e dentro (within-group) destas. Para o problema d-MBV consideramos duas classes de nós: d-branch e não-d-branch, e a característica corresponde à medida de centralidade. A fórmula para esta métrica segue abaixo:

$$\text{F-Score} = \frac{\text{Variância entre classes}}{\text{Variância dentro das classes}} \tag{4}$$

$$\text{Variância entre classes} = \sum_{i=1}^{c} \frac{N_i(\bar{Y}_i - \bar{Y})^2}{c-1} \tag{5}$$

$$\text{Variância dentro das classes} = \sum_{i=1}^{c} \sum_{j=1}^{N_i} \frac{(Y_{ij} - \bar{Y}_i)^2}{N-c} \tag{6}$$

  Onde c é o número de classes (2: d-branch e não-d-branch), N_i é o número de nós da classe i, N é o total de nós da instância analisada, e Y é o valor da característica (centralidade) a ser analisada, onde $\bar{Y}_i$ é a média desta na classe i, $\bar{Y}$ é a sua média geral, e Y_{ij} é o seu valor para o nó j da classe i. Quanto maior for o valor do F-Score, melhor a característica discrimina os elementos entre as classes.

- **Ranking Measure 1 (k_ERROR):** Para calcular as Ranking Measures, utilizamos as soluções ótimas obtidas pela resolução do modelo matemático. Nestas soluções, os vértices de uma instância estão definidos como d-branch ou não. Os vértices da instância são ordenados de forma não-crescente em função dos seus valores para a centralidade escolhida na análise; imaginando os nós ordenados pela centralidade de grau podemos organizar os dados como no conjunto de tuplas do exemplo abaixo:

$$\{(a,1,9),(b,0,5),(c,1,5),\ldots(v_i,\lambda_i,C_i),\ldots,(v_n,\lambda_n,C_n)\} \tag{7}$$

Onde n é o número de vértices, v_i é a identificação do vértice de rank (posição) i, λ_i é igual a 1 se este for d-branch, 0 caso contrário, e C_i é o valor da centralidade escolhida para a ordenação. A medida k-ERROR avalia a frequência com que nós não-d-branch aparecem entre as k primeiras posições do ranking, e consiste na seguinte fórmula:

$$\text{k\_ERROR} = \frac{1}{k}\sum_{i=1}^{k}\delta(\lambda_i) \tag{8}$$

onde

$$\delta(\lambda) = \begin{cases} 1, & \lambda = 0 \quad \text{(não-d-branch)} \\ 0, & \lambda = 1 \quad \text{(d-branch)} \end{cases}$$

Tomando como exemplo uma instância de 20 vértices com 5 vértices d-branch, se ordenarmos por uma centralidade e separarmos os $k = 5$ primeiros para o cálculo da Ranking Measure 1, imaginando que destes os 3 primeiros são d-branch e o resto não, temos que Ranking Measure 1 (RM1) $= k\_ERROR = (0+0+0+1+1)/5 = 0.4$.

- **Ranking Measure 2 (Spread Ranking ou Coverage):** Esta medida avalia o quanto é necessário descer na lista dos vértices do grafo de uma instância ordenada por uma centralidade para se encontrar todos os vértices d-branch. Portanto quanto mais no topo eles estiverem, menor será o Spread Ranking. Seguindo o padrão apresentado em (7), e definindo D como o conjunto dos vértices d-branch encontrados, obtemos a seguinte fórmula:

$$\text{Spread Ranking} = \frac{1}{|D|}\sum_{i=1}^{n}\lambda_i * i \tag{9}$$

Exemplificando, imaginemos um grafo de 20 vértices, destes, 4 são vértices d-branch, e suas respectivas posições após a ordenação por uma determinada centralidade são 1, 2, 10, 20. Consequentemente, Ranking Measure 2 = Spread Ranking $= (1*1, 1*2, 1*10, 1*20)/4 = 8.25$.

- **Ranking Measure 3 (Ranking Loss):** Indica a quantidade que vezes que nós que não são d-branch são ranqueados antes dos que são. Denotando D como o conjunto dos nós d-branch encontrados para uma instância, e $\bar{D}$ o dos outros nós, temos a seguinte fórmula:

$$\text{Ranking Loss} = \frac{|\{(v,w) : r(v) > r(w), (v,w) \in D \times \bar{D}\}|}{|D||\bar{D}|} \tag{10}$$

Supondo uma instância com 11 vértices, em que os vértices d-branch ordenados ocupam as posições 1, 3, e 6, temos para cada, respectivamente, 0, 1, e 3 nós não-d-branch em posições anteriores, portanto Ranking Measure 3 = Ranking Loss $= (0 + 1 + 3)/(3 \times 8) = 0.1\overline{6}$. Como consequência, se todos os d-branches ocuparem as primeiras posições, Ranking Loss será igual a zero.

- **Ranking Measure 4 (Average precision):** Calcula para cada vértice d-branch a proporção de d-branches no conjunto dos nós ranqueados do topo até este, e após efetua uma média dessas proporções; se todos os d-branch estiverem nas primeiras posições, a medida será igual a 1. Definindo D como o conjunto dos vértices d-branch de uma instância, e a função r, onde $r(v)$ = rank (posição, começando por 1) de v na lista de vértices ordenados por uma centralidade, obtemos que:

$$\text{Average Precision} = \frac{1}{|D|} \sum_{v \in D} \frac{|\{w : w \in D, r(w) \leq r(v)\}|}{r(v)} \tag{11}$$

Em uma instância em que os nós de posições 1, 2, 6, e 8 fossem d-branch, Ranking Measure 4 = Average precision $= (\frac{1}{1} + \frac{2}{2} + \frac{3}{6} + \frac{4}{8})/4 = 0.75$.

Foi realizado o cálculo destas medidas para as instâncias SPD visando analisar as seguintes centralidades: **D**egree (Grau), **C**loseness, **B**etweenness, **P**ageRank, **E**igenvector.

Os gráficos nas Figuras 9,10,11,12,13 mostram os resultados obtidos.

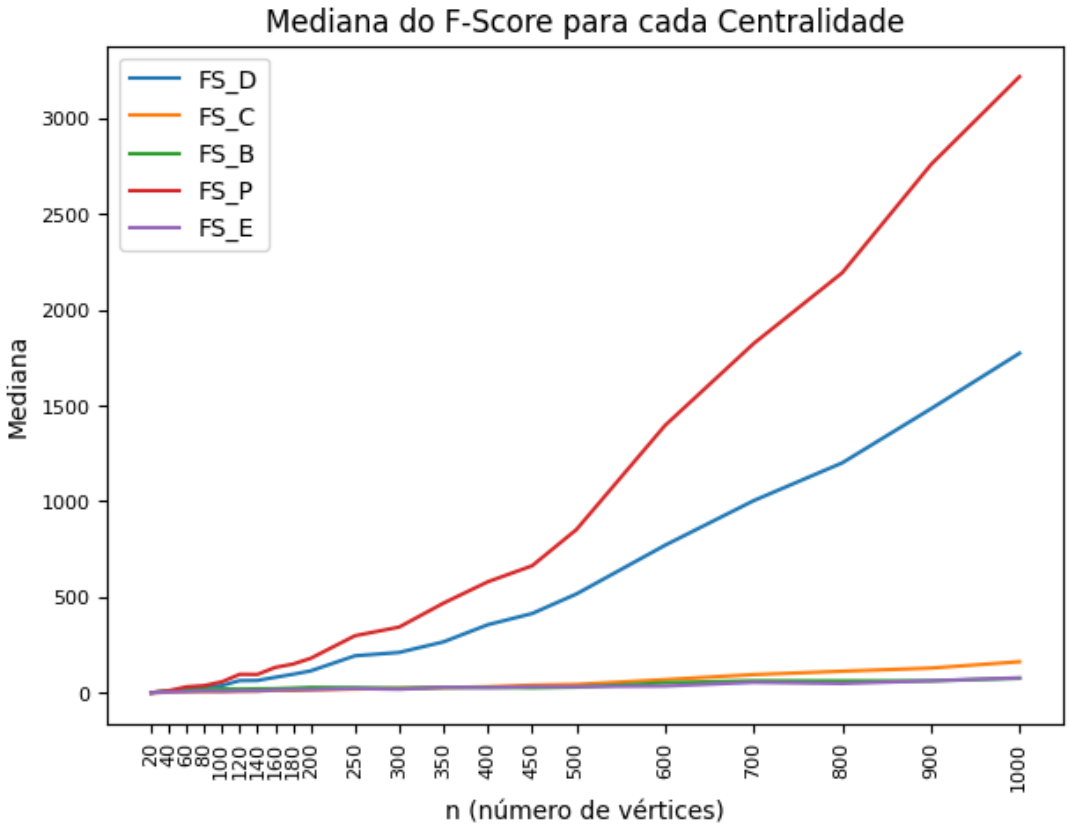


Figura 9. Mediana do F-Score dentro de cada conjunto de instâncias com $|V| = n$, para cada centralidade.

A Figura 9, assim como as Figuras 10 a 13, utilizam para cada centralidade a mediana da distribuição da sua métrica de análise calculada para cada conjunto de instâncias com $|V| = n$. A escolha da mediana se deu pelo fato desta ser menos afetada por valores discrepantes (outliers) do que a média, tornando-a mais adequada para representar as distribuições.

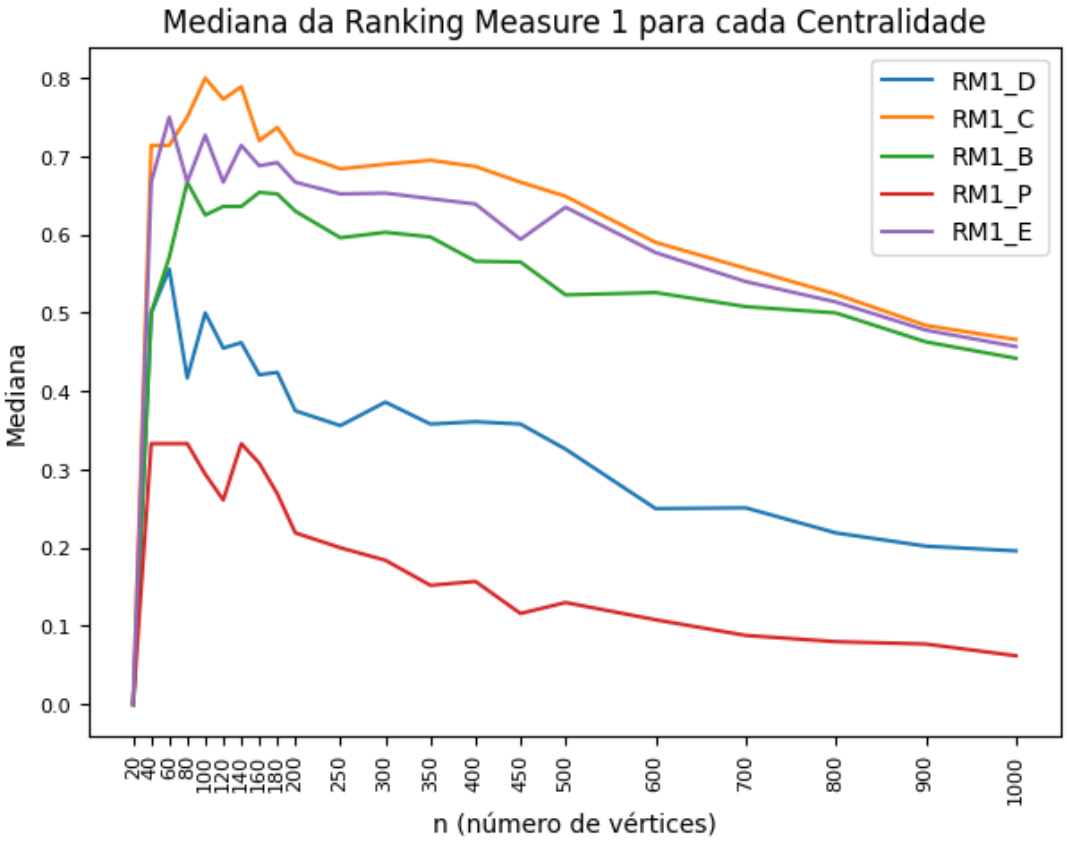


Figura 10. Mediana da Ranking Measure 1 dentro de cada conjunto de instâncias com $|V| = n$, para cada centralidade.

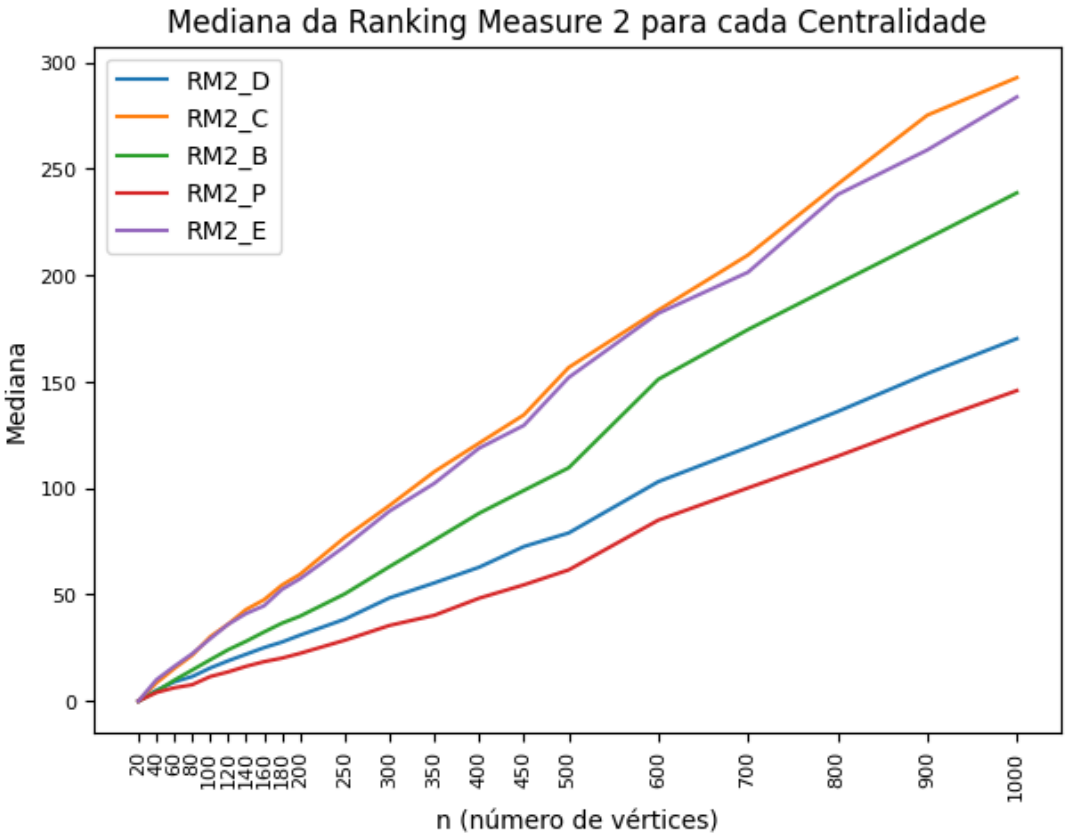


Figura 11. Mediana da Ranking Measure 2 dentro de cada conjunto de instâncias com $|V| = n$, para cada centralidade.

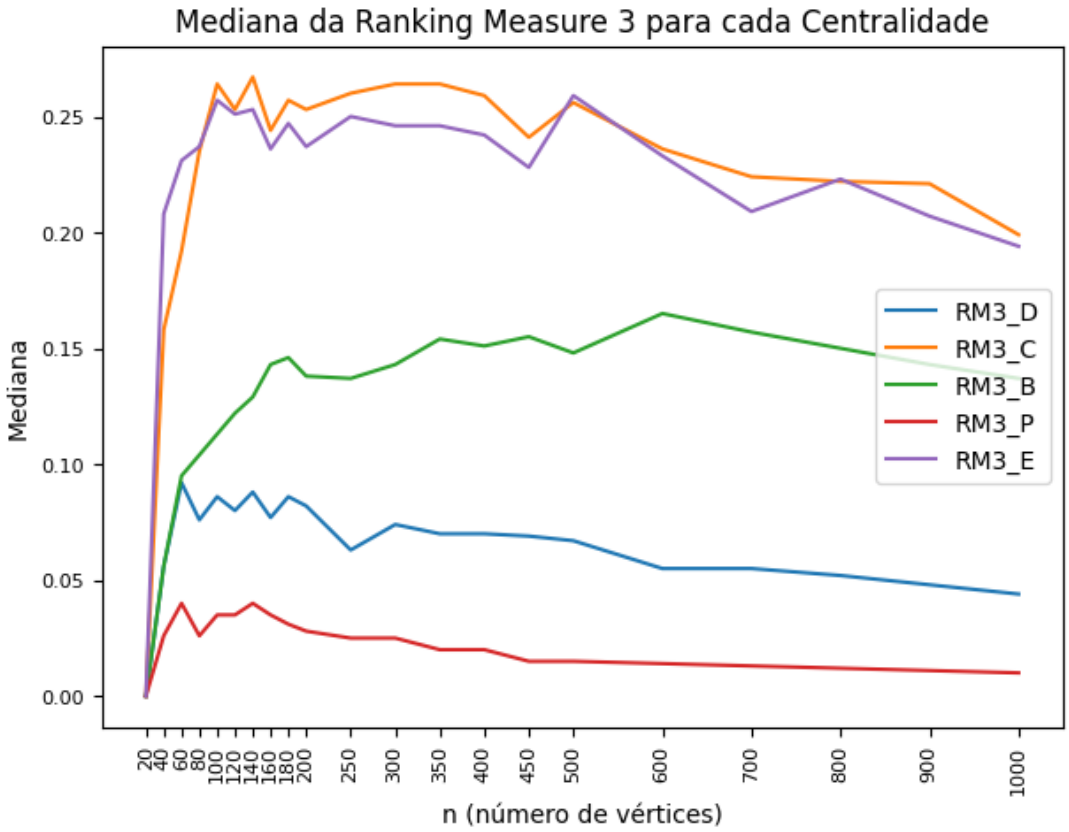


Figura 12. Mediana da Ranking Measure 3 dentro de cada conjunto de instâncias com $|V| = n$, para cada centralidade.

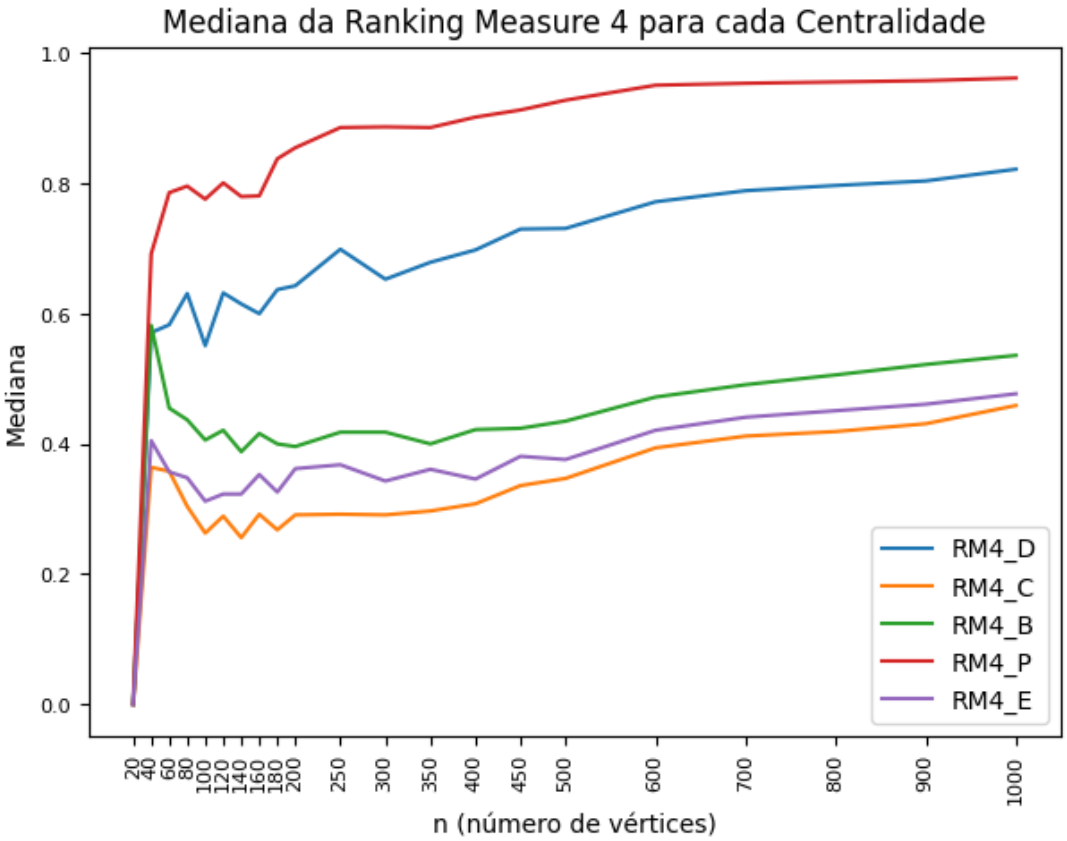


Figura 13. Mediana da Ranking Measure 4 dentro de cada conjunto de instâncias com $|V| = n$, para cada centralidade.

Uma centralidade se mostrará boa identificadora de nós d-branch se apresentar os maiores valores de F-Score e Ranking Measure 4, e os menores nas outras Rankings Measures. Observando os gráficos acima percebe-se que, seguindo esses padrões, as centralidades que mais se destacam são as de Grau e PageRank, sendo esta a que melhor desempenhou em todas as métricas, especialmente em relação à F-Score. Para confirmar tais resultados analisando pelo Gephi, escolhemos uma instância aleatória em SPD, com $n = 40$ e $m = 50$.

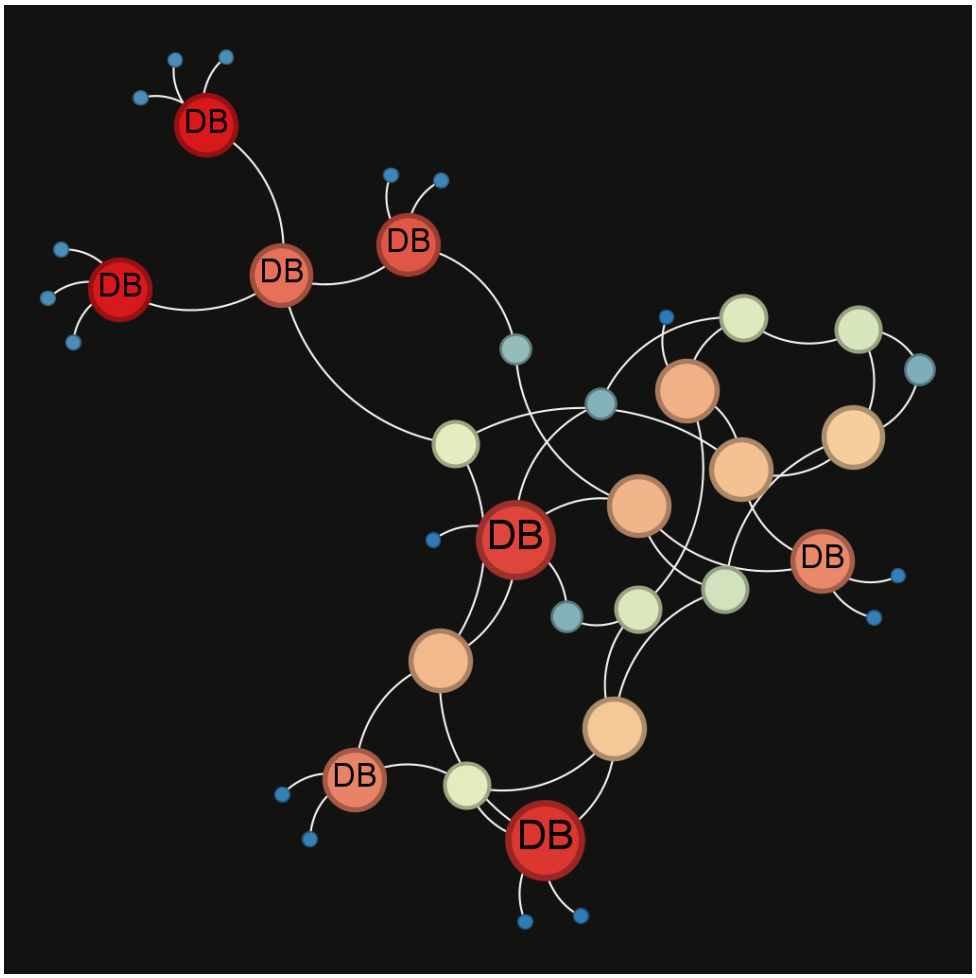


Figura 14. Imagem de análise produzida no Gephi de instância de SPD com $n = 40$ e $m = 50$. O tamanho dos nós é proporcional ao grau, a cor indo do azul ao vermelho é proporcional ao PageRank, e os vértices d-branch são demarcados por label DB.

Observando a análise exposta na Figura 14, visualizamos que há uma boa quantidade de vértices de mesmo grau que são tanto d-branch ou não, indicando que esta centralidade não se mostra tão adequada; enquanto isso, os nós de maior PageRank (mais vermelhos) concentram os d-branches, confirmando a qualidade desta centralidade para a tarefa de identificá-los.

Avaliação do Modelo Heurístico. Fazendo uso dos resultados produzidos pelo modelo ótimo, avaliamos os do heurístico gerados usando as centralidades de Grau, e após, PageRank. Estas centralidades se mostraram as que melhor desempenharam nas análises da seção anterior, sendo a de PageRank a que mais se destacou. Para nossa avaliação fizemos uso das métricas de distância para cada instância, que servirá para medir o erro do modelo heurístico em relação ao ótimo, e de taxa de sucesso percentual para cada conjunto de instâncias onde a quantidade de vértices ($|V|$) é igual a um n

(e.g. 100, 500, 600):

$$distance_i = \frac{db_{h_i} - db_i}{db_i} \qquad \text{success rate}_n = \frac{hits_n}{N_n} \times 100 \tag{12}$$

Onde db_i e db_{h_i} são, respectivamente, a quantidade de nós d-branch encontrada pelo modelo ótimo, e pelo modelo heurístico, para uma instância i; $hits_n$ é o número de instâncias em que a heurística acertou o total de nós d-branch, dentro de um conjunto de instâncias onde $|V| = n$, e N_n é o total de instâncias deste conjunto. Observando melhor a fórmula da distância percebe-se que ela pode ser vista como o erro do modelo heurístico. Tendo as métricas definidas, geramos os gráficos abaixo:

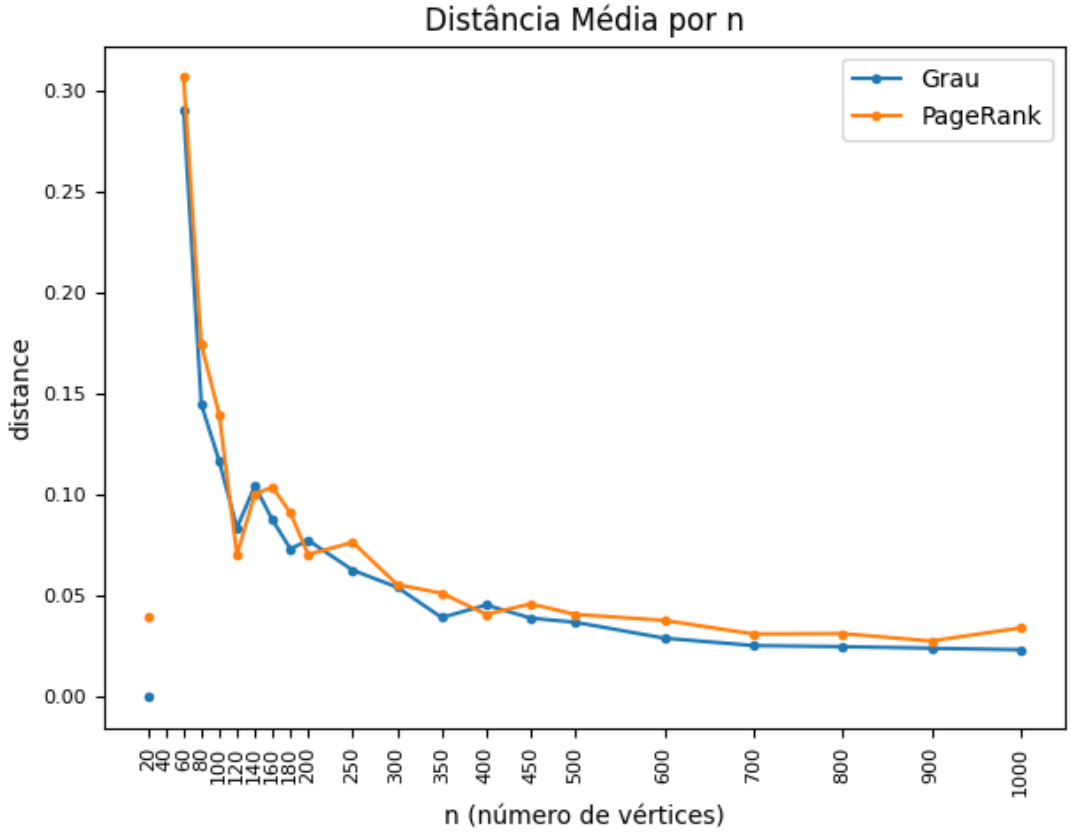


Figura 15. Distância média dentro de cada conjunto de instâncias com $|V| = n$, obtida usando o modelo heurístico com as centralidades de Grau e PageRank.

Observando a Figura 15 percebemos que as curvas para a distância média, relativas a Grau e PageRank, pouco se distanciam para cada n e se cruzam diversas vezes até $n = 450$, onde a partir deste ponto o valores para PageRank se mantém um pouco acima dos para Grau. Consequentemente, o erro (distância relativa ao ótimo) médio do modelo heurístico não se diferencia muito quando se compara o resultados usando as centralidades de Grau e PageRank, porém, para $n \geq 450$ percebe-se que esta apresenta maior erro médio.

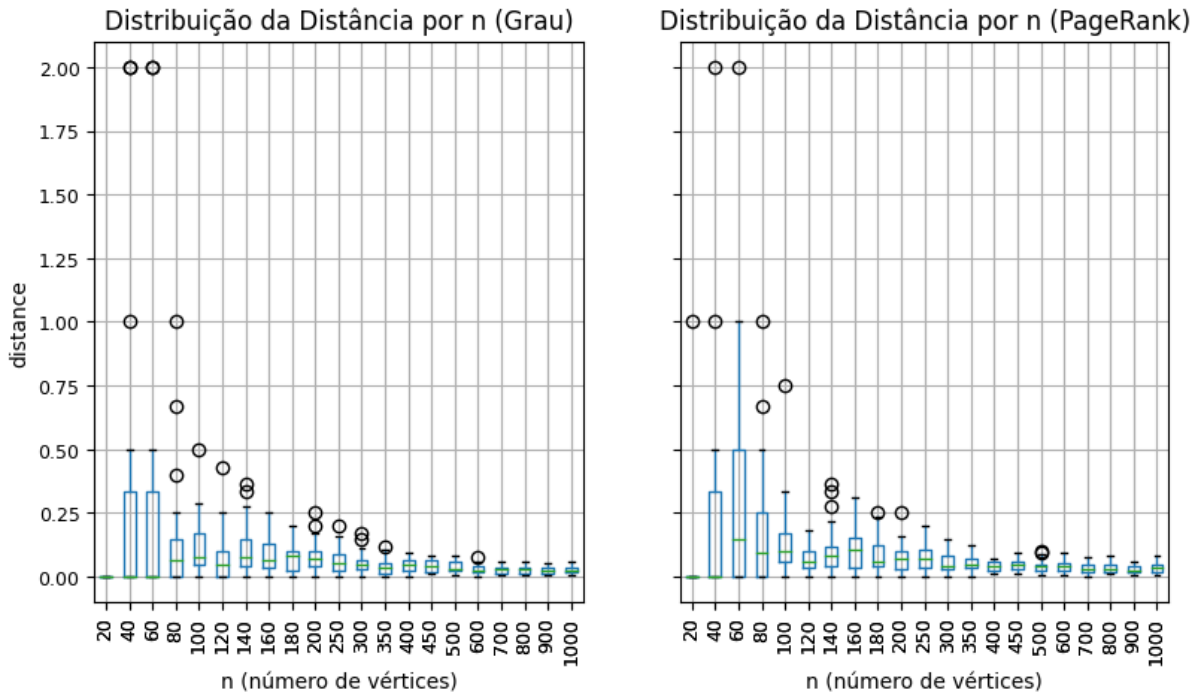


Figura 16. Boxplots com a distribuição das distâncias dentro de cada conjunto de instâncias com $|V| = n$, obtidas usando o modelo heurístico com as centralidades de Grau e PageRank.

A Figura 16 mostra o quanto a distribuição das distâncias calculadas para cada conjunto de instâncias com $|V| = n$ se assemelha entre os resultados da heurística usando tanto Grau como PageRank. Um lado positivo em comum é que a variância, proporcional à "altura"dos boxplots, diminui conforme n aumenta, além de ao mesmo tempo praticamente sumir com os outliers, indicando que ao n aumentar seu erro passa a se tornar mais comportado, de magnitude menor, e menos variante.

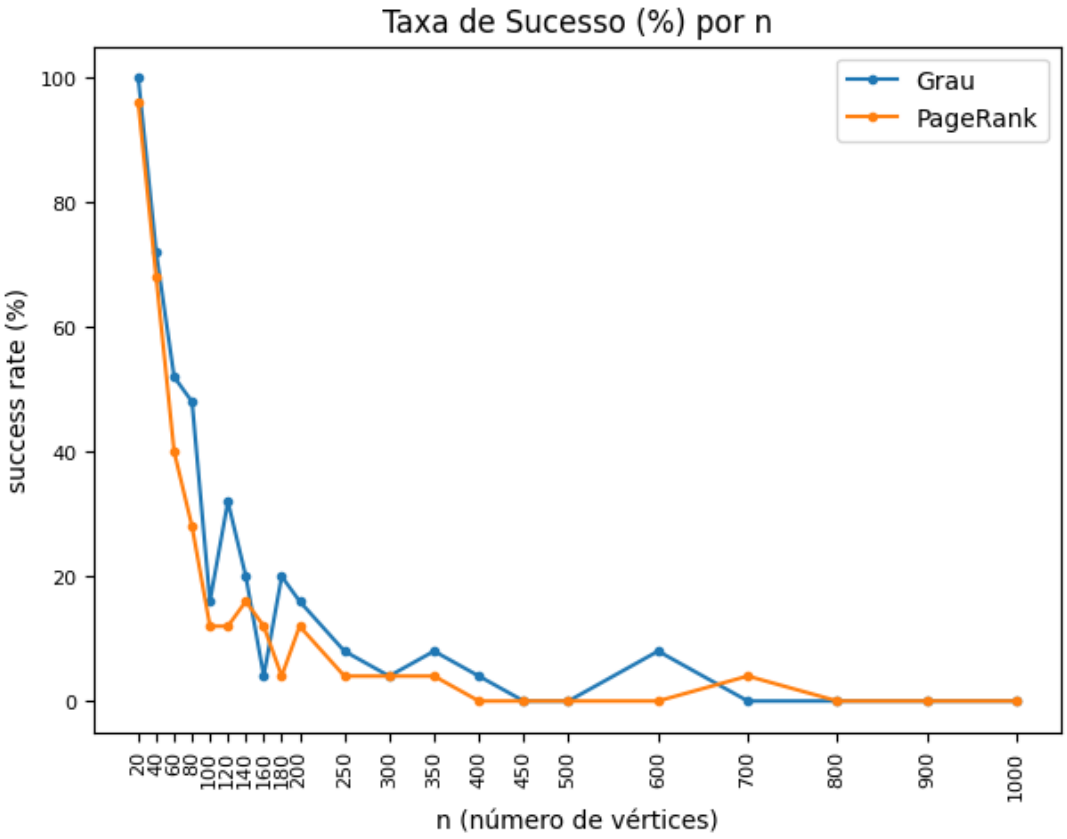


Figura 17. Taxa de Sucesso percentual dentro de cada conjunto de instâncias com $|V| = n$, obtida usando o modelo heurístico com as centralidades de Grau e PageRank.

Assim como nas Figuras 15 e 16, observamos em 17 que os valores para Grau e PageRank pouco se distanciam quando avaliamos a Taxa de Sucesso percentual. No entanto, é interessante notar que para os menores valores de n a heurística usando Grau demonstra entregar melhora Taxa de Sucesso, enquanto que para os maiores, Grau e PageRank tendem a se igualarem quanto a essa medida de desempenho.

Ao observar os gráficos percebemos que, apesar da expectativa de observar melhores resultados usando o PageRank, estes se aproximam muito dos obtidos usando a centralidade de Grau, chegando a indicar para boa parte das instâncias um desempenho abaixo do que se teve usando o Grau, como pode ser visto com nitidez nas taxas de sucesso entre 60 e 600, por exemplo. No entanto, devido aos resultados tão expressivos à favor do uso do PageRank como centralidade identificadora de nós d-branch, abre-se a possibilidade de que a forma que este está sendo aplicado pelo algoritmo precisa de revisão. Afinal, a forma como é calculado o PageRank traz informações mais globais do que o Grau, o que pode implicar na necessidade de numa abordagem menos gulosa.

Focando agora apenas em avaliar o modelo heurístico em si, apesar deste apresentar uma taxa de sucesso inferior a 20% logo a partir de $n = 100$, vemos que os

valores da distância se mostram interessantes, uma vez que, além de indicarem um erro médio inferior a 10% após $n = 100$, possuem um comportamento de magnitude e variância decrescentes conforme n aumenta, ou seja, conforme o problema se torna mais custoso, o que torna o algoritmo atraente para a aplicação.

UM ESTUDO DA RELAÇÃO ENTRE A QUANTIDADE DE NÓS D-BRANCH E A DENSIDADE DO GRAFO

Faz sentido imaginar que quanto mais arestas, mais completo o grafo, e consequentemente menor a chance de nós d-branch surgirem. A densidade, apresentada na Equação 13, metrifica o quão completo um grafo é, portanto, partindo da reflexão inicial, seria possível relacionar a quantidade de nós d-branch à densidade?

$$density(G) = \frac{2m}{n(n-1)} \tag{13}$$

Diante da inquietação apresentada acima, utilizamos os resultados exatos obtidos pela resolução do modelo matemático para todas as instâncias do conjunto de dados SPD para poder gerar o gráfico mostrado na Figura 18, onde os valores das quantidades de nós d-branch para cada instância são exibidos em função da densidade.

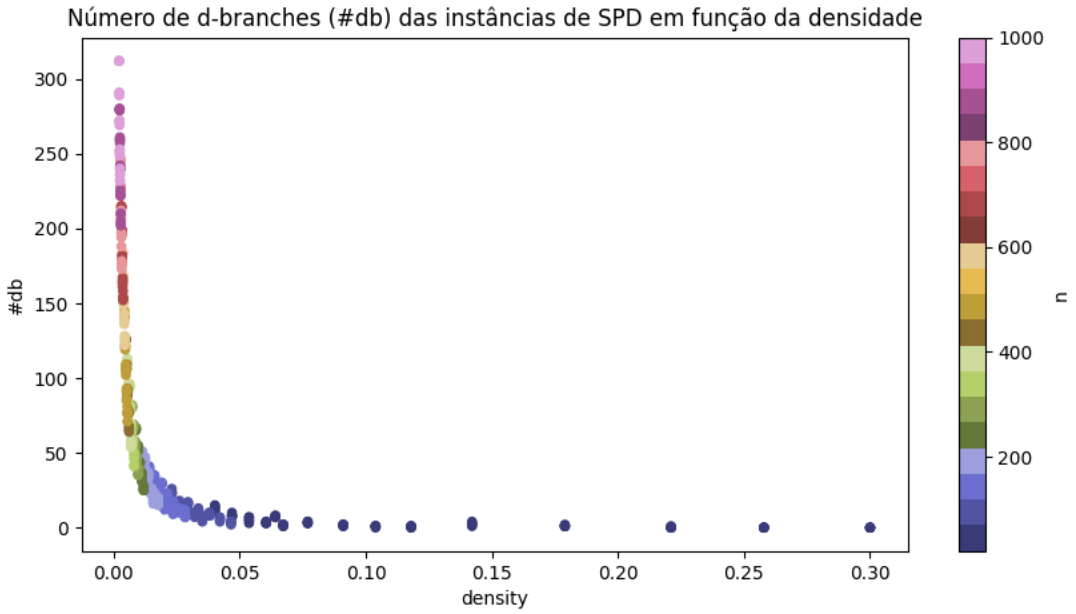


Figura 18. Quantidade de d-branches de cada instância em função da densidade de cada uma. Cada instância é identificada pelo padrão de cores associado a seu número de vértices n, exibido à direita do gráfico.

Inicialmente observa-se um comportamento semelhante à função $f(x) = \frac{1}{x}$, onde $x \in \mathbb{R}$, sugerindo uma relação não-linear do número de vértices d-branch ($\#db$) em função da densidade ($density$). Para uma melhor investigação, propomos a seguinte modelagem aproximativa:

$$\#db \approx \frac{1}{\alpha \times density} \tag{14}$$

Onde $\alpha \in \mathbb{R}$ é uma constante arbitrária, definida para poder tornar a fórmula mais generalizável. Aplicando $\log(x)$ aos dois lados da equação obtemos

$$\log(\#db) \approx -\log(\alpha) - \log(density) \tag{15}$$

Indicando que, num gráfico log-log devemos esperar ver retas de coeficiente angular negativo, uma vez que $density \geq 0$, e não estamos trabalhando com grafos nulos (sem arestas). Verificamos nossas observações pelos gráficos log-log mostrados nas Figuras 19 e 20.

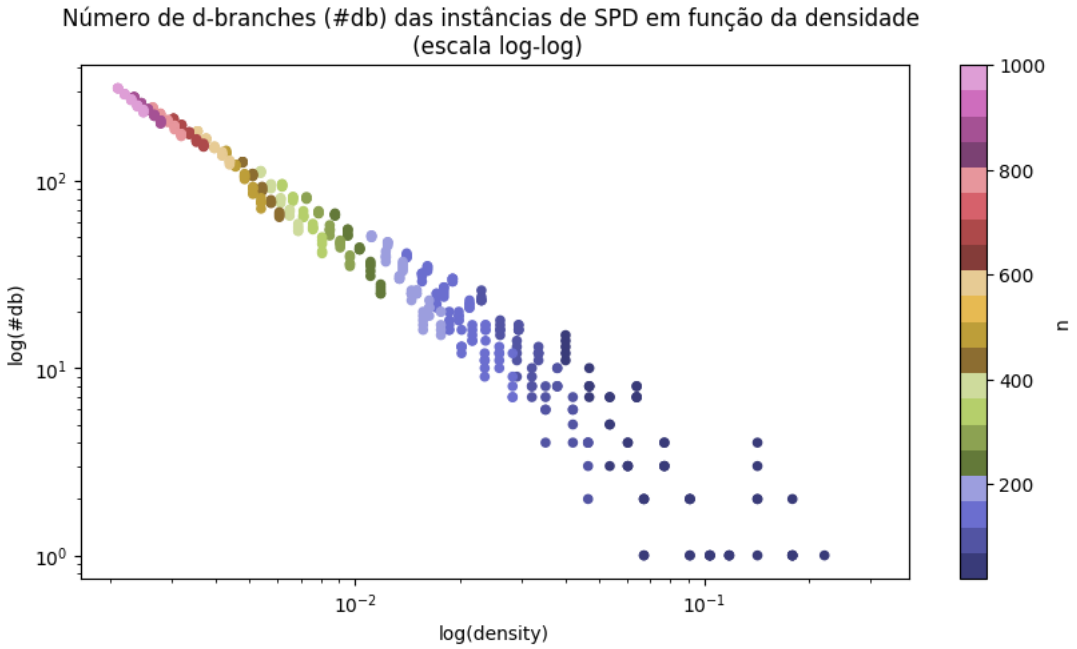


Figura 19. Quantidade de d-branches de cada instância em função da densidade de cada uma, apresentadas em escala log-log. Cada instância é identificada pelo padrão de cores associado a seu número de vértices n, exibido à direita do gráfico.

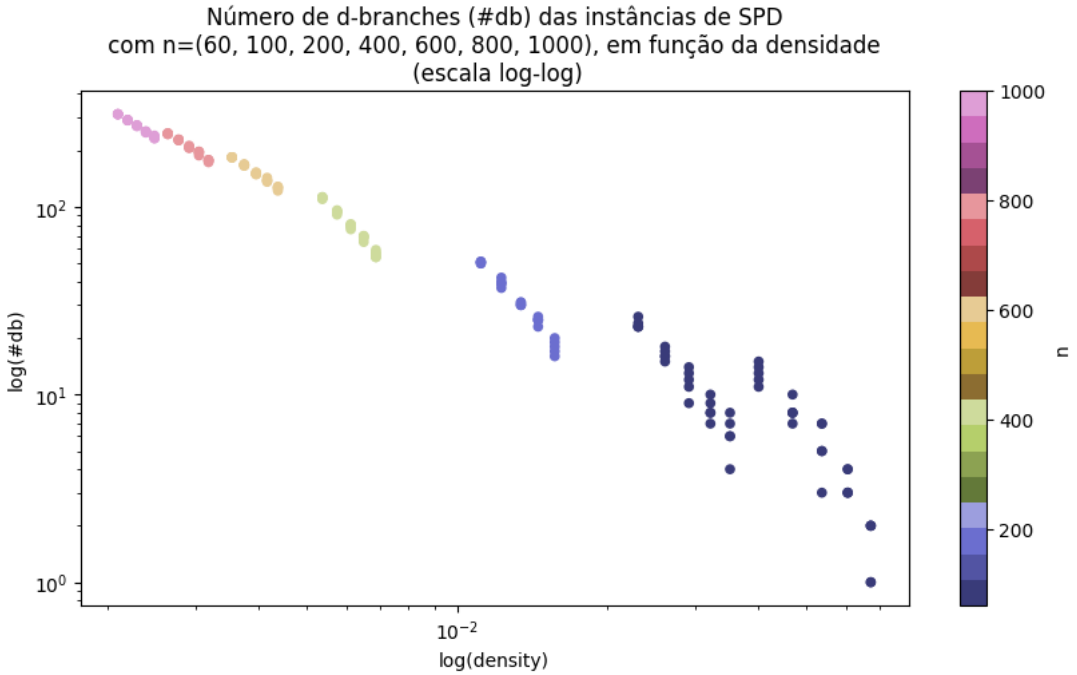


Figura 20. Quantidade de d-branches das instâncias com n=(60, 100, 200, 400, 600, 800, 1000) em função da densidade de cada uma, apresentadas em escala log-log. Cada instância é identificada pelo padrão de cores associado a seu número de vértices n, exibido à direita do gráfico.

Analisando a Figura 19 vemos surgir, para cada n, retas bem definidas conforme

o que estávamos prevendo, especialmente para $n \geq 200$. Para explicitá-las melhor, vendo as mesmas separadas, geramos a Figura 20, onde temos instâncias com n's específicos e com bom espaçamento entre si. Diante do observado, a relação da quantidade de vértices d-branch em função da densidade de grafos passa a se tornar uma possibilidade interessante que merece ser confirmada por meio de investigações em outros conjuntos de dados, dado que, por prover uma estimativa direta, derivada de uma fórmula, fornece uma alternativa rápida a algoritmos complexos, e que pode ser usada por estes para diminuir espaços de busca, por exemplo. Outra consequência interessante é que, aliando este modelo aproximativo ao fato de que grande parte dos vértices d-branch ocupa as primeiras opções de um ranqueamento por centralidades, como vimos pelo desempenho do PageRank nas Ranking Measures, um método que surge para encontrar os vértices d-branch consiste em apenas ordenar os vértices de um grafo G por uma centralidade (ou uma combinação delas) como o PageRank, e selecionar os k melhores ranqueados, onde $k = \#db(G) \approx \frac{1}{\alpha \times density(G)}$. Mesmo sendo uma estratégia com margem para erros, sua complexidade computacional relativamente baixa, especialmente considerando que estamos lidando com um problema NP-difícil, não deixa de ser um atrativo para estudos futuros.

CONCLUSÕES

As análises realizadas neste trabalho demonstram que as medidas de centralidade podem auxiliar na descoberta de vértices d-branch, colocando-as para serem consideradas no desenvolvimento de modelos heurísticos destinados à solução de problemas de Otimização Combinatória como o d-MBV. A meta-heurística ILS fazendo uso das centralidades gerou bons resultados. Uma possível relação aproximando a quantidade de nós d-branch em função da densidade de grafos mostrou-se como uma abordagem interessante para a definição de métodos de solução menos complexos.

No entanto, como desafios, ainda é preciso avaliar o desempenho do algoritmo para outros conjuntos de dados, readaptá-lo para tirar proveito de todo o potencial do PageRank, desenvolver implementações fazendo uso de outras meta-heurísticas, como multi-start e GRASP, e investigar e confirmar a relação exposta na seção anterior em outros conjuntos de grafos com maior variância de densidades.

Devido ao sucesso da aplicação de técnicas de análise de dados para a descoberta de informações relevantes ao desenvolvimento de soluções para problemas complexos, acreditamos que este é um caminho que deve ser mais desbravado, trazendo mais elementos, métricas, e ferramentas que possam, a partir desta interdisciplinaridade, abrir o olhar para novas abordagens de solução para problemas de Otimização Combinatória.

AGRADECIMENTOS

Aos meus pais, que sempre me apoiaram, e continuam apoiando, na minha jornada, seja acadêmica, profissional, ou pessoal. Se hoje estou aqui, foi porque vocês foram a minha base.

Às professoras Simone de Lima Martins (orientadora), e Maria Claudia S. Boeres, que me acompanharam e orientaram durante todo este trajeto, além de me trazerem novos saberes que serão essenciais para o meu futuro, seja acadêmico ou profissional. Obrigado também pela paciência e compreensão ao longo dos momentos difíceis.

À Bárbara Emily R. de Morais, que foi bolsista de IC neste projeto junto comigo e com quem dividi tarefas e trabalhei em equipe.

À Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ), por me conceder a bolsa para poder trabalhar nesta pesquisa.

REFERÊNCIAS

[1] GARGANO, Luisa *et al.* Spanning Trees with Bounded Number of Branch Vertices. *In*: WIDMAYER, Peter *et al.* [Ed.]. *Automata, Languages and Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. P. 355–365. ISBN 978-3-540-45465-6.

[2] MERABET, Massinissa, DESAI, Jitamitra e MOLNAR, Miklos. A Generalization of the Minimum Branch Vertices Spanning Tree Problem. *In*: LEE, Jon, RINALDI, Giovanni e MAHJOUB, A. Ridha [Ed.]. *Combinatorial Optimization*. Cham: Springer International Publishing, 2018. P. 338–351. ISBN 978-3-319-96151-4.

[3] MORENO RAMÍREZ, Jorge, FROTA, Yuri e MARTINS, Simone. An exact and heuristic approach for the d-minimum branch vertices problem. *Computational Optimization and Applications*, v. 71, dez. 2018. DOI: 10.1007/s10589-018-0027-x.

[4] STÜTZLE, Thomas e RUIZ, Rubén. Iterated Local Search. *In*: [*s. l.: s. n.*], jan. 2017. P. 1–27. DOI: 10.1007/978-3-319-07153-4_8-1.

[5] BASTIAN, Mathieu, HEYMANN, Sebastien e JACOMY, Mathieu. *Gephi: An Open Source Software for Exploring and Manipulating Networks*. [*S. l.: s. n.*], 2009.

Disponível em:
http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.

[6] CSARDI, Gabor e NEPUSZ, Tamas. The igraph software package for complex network research. *InterJournal*, Complex Systems, p. 1695, 2006. Disponível em: https://igraph.org.

[7] LIMA MARTINS; ALEXANDRE PLASTINO, Maria Claudia Silva Boeres; Simone de. Identifying elements of solutions of combinatorial optimization problems using measures of graph centrality. *ANAIS DO SIMPóSIO BRASILEIRO DE PESQUISA OPERACIONAL, 2023, São José dos Campos. Anais eletrônicos... Campinas, Galoá, 2023.*, 2023. Disponível em: https://proceedings.science/sbpo-2023/trabalhos/identifying-elements-of-solutions-of-combinatorial-optimization-problems-using-m?lang=pt-br.

[8] ZHAO, Zheng *et al.* Advancing feature selection research. *ASU Feature Selection Repository Arizona State University*, p. 1–28, jan. 2010.

[9] PEREIRA, Rafael *et al.* Correlation analysis of performance measures for multi-label classification. *Information Processing and Management*, v. 54, p. 359–369, mai. 2018. DOI: 10.1016/j.ipm.2018.01.002.

Documento assinado digitalmente
gov.br VICTOR PEREIRA DE LIMA
Data: 23/01/2024 16:26:29-0300
Verifique em https://validar.iti.gov.br

Simone de Lima Martins

Victor Pereira de Lima