UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE CIÊNCIA E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BEATRIZ DA SILVA RANGEL

Um estudo de heurísticas para o Problema da k-Dominação

RIO DAS OSTRAS

2022

UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE CIÊNCIA E TECNOLOGIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BEATRIZ DA SILVA RANGEL

Um estudo de heurísticas para o Problema da k-Dominação

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação do Instituto de Ciência e Tecnologia da Universidade Federal Fluminense, como parte dos requisitos necessários à obtenção do Grau de Bacharel em Ciências. Área de Concentração: Algoritmos e Combinatória

Orientadora:
Profa. Dra. Maise Dantas da Silva

Co-orientador:
Prof. Dr. Rodrigo Lamblet Mafort

RIO DAS OSTRAS

2022

Ficha catalográfica automática - SDC/BRO
Gerada com informações fornecidas pelo autor

R196e Rangel, Beatriz da Silva
Um estudo de heurísticas para o Problema da k-Dominação / Beatriz da Silva Rangel ; Maise Dantas da Silva, orientadora ; Rodrigo Lamblet Mafort, coorientador. Niterói, 2022.
57 f.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação)-Universidade Federal Fluminense, Instituto de Ciência e Tecnologia, Rio das Ostras, 2022.

1. Otimização combinatória (Computação). 2. Heurística. 3. Grafo. 4. Produção intelectual. I. Silva, Maise Dantas da, orientadora. II. Mafort, Rodrigo Lamblet, coorientador. III. Universidade Federal Fluminense. Instituto de Ciência e Tecnologia. IV. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

BEATRIZ DA SILVA RANGEL

Um estudo de heurísticas para o Problema da k-Dominação

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação do Instituto de Ciência e Tecnologia da Universidade Federal Fluminense, como parte dos requisitos necessários à obtenção do Grau de Bacharel em Ciências. Área de Concentração: Algoritmos e Combinatória.

Aprovada em 29 de julho de 2022.

BANCA EXAMINADORA

Maise Dantas da Silva

Profa. Dra. Maise Dantas da Silva - Orientadora, UFF

Rodrigo L. Mafort

Prof. Dr. Rodrigo Lamblet Mafort - Orientador, UFF

Andre Renato Villela da Silva arvsilva@id.uff.br:09528761755

Assinado de forma digital por Andre Renato Villela da Silva arvsilva@id.uff.br:09528761755
Dados: 2022.07.31 14:41:52 -03'00'

Prof. Dr. André Renato Villela da Silva, UFF

Danilo Artigas

Prof. Dr. Danilo Artigas da Rocha, UFF

ASSINADO DIGITALMENTE
MARCOS RIBEIRO QUINET DE ANDRADE
DATA
11/08/2022
A conformidade com a assinatura pode ser verificada em:
http://serpro.gov.br/assinador-digital
SERPRO

Prof. Dr. Marcos Ribeiro Quinet de Andrade, UFF

Rio das Ostras

2022

Aos meus pais, Salvador (in memoriam) e Nelcy.
À minha irmã, Kíssila.

Agradecimentos

À minha família,
em especial à minha irmã Kíssila pelo apoio, compreensão e carinho constante ao longo dos últimos anos.

Aos meus orientadores, Professora Maise Dantas e Professor Rodrigo Mafort,
por toda dedicação, apoio e paciência. Meus sinceros agradecimentos pelos aprendizados e pela oportunidade de tê-los como orientadores.

Aos amigos,
pela compreensão nos momentos de ausência e todo o suporte a cada desafio.

Aos professores e funcionários do Pólo Universitário de Rio das Ostras da UFF,
pelas oportunidades, ensinamentos e dedicação.

Resumo

O Problema da k-Dominação Mínima consiste em encontrar, para um grafo $G = (V, E)$, um conjunto mínimo $S \subseteq V$ de forma que cada vértice $v \in V$ pertence a S ou possui pelo menos k vizinhos em S. Por se tratar de um problema NP-difícil, encontrar uma solução exata para este problema tende a ser custoso computacionalmente. Neste trabalho, são apresentadas seis heurísticas gulosas, que exploram os requisitos e os graus dos vértices, bem como combinações dos dois fatores. Os experimentos realizados foram divididos em duas etapas. A primeira consiste na comparação dos resultados das heurísticas com as soluções exatas, para grafos com vinte, cinquenta e cem vértices. Já na segunda etapa, que considera grafos com até dez mil vértices, as heurísticas são comparadas entre si, considerando o tamanho de cada solução obtida e seu tempo de execução.

Palavras-chave: Problema da k-Dominação Mínima; Heurísticas; Método Guloso; Otimização em Grafos.

Abstract

For a graph $G = (V, E)$, the Minimum k-Dominating Set Problem consists of finding a minimum set $S \subseteq V$ where each vertex $v \in V$ belongs to S or has at least k neighbors in S. Since the problem is NP-hard, finding an exact solution has a high computational cost. This work presents six greedy heuristics based on vertices requirements, degrees, and combinations of both factors. The experiments were divided into two parts. The first one compares heuristic results with exact solutions of graphs with twenty, fifty, and a hundred vertices. The second part considers graphs with up to ten thousand vertices and compares each heuristic to the others, considering the costs of the solutions and the execution times.

Keywords: Minimum k-Dominating Set Problem; Heuristics; Greedy Method; Graph Optimization.

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Sumário

Capítulo 1

Introdução

Em grafos, os problemas de dominação constituem uma família amplamente difundida por sua aplicabilidade em diversos campos de estudo, variando da computação às ciências sociais. Essa família inclui, além do problema clássico do Conjunto Dominante, variantes como Dominação Conexa, Dominação Total e k-Dominação [13].

Para um grafo de entrada $G = (V, E)$, um subconjunto $S \subseteq V$ é considerado um *conjunto dominante* se, para cada $v \in V$, ou v está em S ou v tem pelo menos um vizinho em S [3]. A Figura 1.1 ilustra, respectivamente, um grafo G e um possível conjunto dominante $S = \{a, b, d\}$ para G.

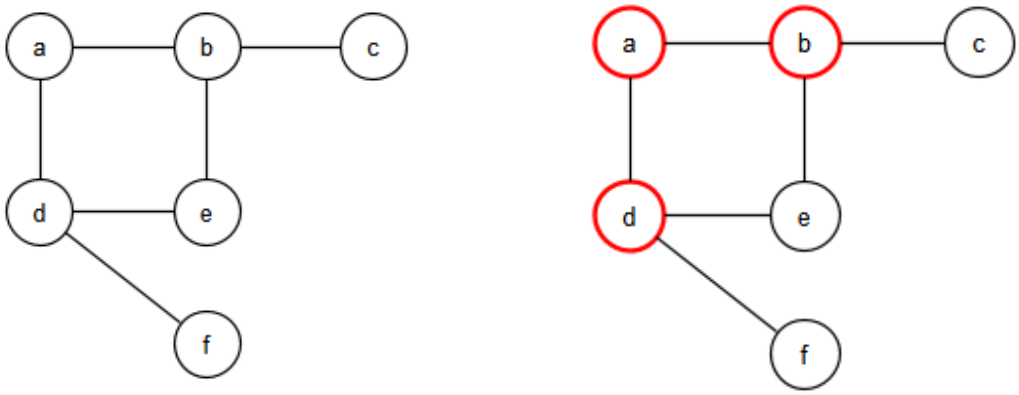


Figura 1.1: Representação geométrica de um grafo $G = (V, E)$ e um possível conjunto dominante $S = \{a, b, d\}$.

Para o Problema do Conjunto Dominante, temos como entrada um grafo G e um inteiro ℓ. O objetivo consiste em verificar se existe um conjunto dominante de tamanho ℓ para G. Este problema é classificado como NP-completo para grafos em geral [7]. Sua versão de otimização, o Problema do Conjunto Dominante Mínimo, também tem grande relevância para a computação por suas aplicações em áreas práticas, como mineração de dados e projeto de redes sem fio, sendo um problema NP-difícil [1].

Como uma generalização do Problema do Conjunto Dominante, temos o Problema da k-Dominação, que recebe como entrada um grafo G e dois inteiros k e ℓ, e tem por objetivo verificar se existe um conjunto k-dominante de tamanho ℓ. Para um grafo $G = (V, E)$, um conjunto $S \subseteq V$ é dito *k-dominante* se, para todo vértice $v \in V$, v está em S ou v tem pelo menos k vizinhos em S.

Considerando que o Problema do Conjunto Dominante pode ser visto como uma restrição do Problema da k-Dominação, onde $k = 1$, e que o Problema do Conjunto Dominante é NP-completo, conclui-se que o Problema da k-Dominação também é NP-completo.

Neste trabalho, será abordado o Problema da k-Dominação Mínima, que busca encontrar um conjunto de cardinalidade mínima capaz de k-dominar o grafo. O *custo* de uma solução para uma instância do problema corresponde à cardinalidade dessa solução.

1.1 Motivação e Objetivos

A família dos problemas de dominação tem sido extensivamente estudada na literatura, com diferentes sugestões de abordagens para contornar sua NP-dificuldade.

O Problema da k-Dominação possui aplicação em diversas áreas, como, por exemplo, saúde (detecção de surtos de doenças infecciosas) e computação (determinação do número de roteadores necessários para o funcionamento de uma rede sem fio) [13].

Esta monografia está focada no estudo do Problema da k-Dominação Mínima. Neste contexto, são realizados o estudo e a implementação de métodos heurísticos para a resolução deste problema. O objetivo principal deste trabalho é fazer uma análise comparativa, onde será medida a eficiência das heurísticas implementadas, usando como um dos critérios a distância entre a solução ótima e as soluções encontradas por elas.

1.2 Estruturação do Trabalho

No Capítulo 2, são apresentados os conceitos teóricos de grafos e classes de complexidade, estabelecendo o foco do estudo no Problema da k-Dominação.

Com a fundamentação teórica definida, o Capítulo 3 aborda as heurísticas desenvolvidas ao longo do trabalho. Neste capítulo, os algoritmos utilizados para a implementação são apresentados e seu funcionamento explicado.

O Capítulo 4 traz os resultados obtidos com o estudo comparativo entre as heurísticas implementadas e a solução ótima (para instâncias com até cem vértices).

Por fim, no Capítulo 5, são apresentadas as conclusões obtidas por meio da análise dos resultados.

Capítulo 2

Fundamentação Teórica

2.1 Grafos

Um grafo simples $G = (V, E)$ consiste em um conjunto finito de vértices V e um conjunto de arestas E. Uma aresta é formada por um par não ordenado $v, u \in V$, onde os vértices u e v são suas *extremidades* [12]. A Figura 2.1 é uma representação geométrica de um grafo $G = (V, E)$, onde $V = \{a, b, c, d, e, f\}$ e $E = \{(a, b), (a, c), (a, d), (b, c), (b, e), (b, f), (c, f), (d, e), (d, f)\}$, sendo os vértices de V representados por pontos e as arestas de E representadas por linhas que conectam esses pontos.

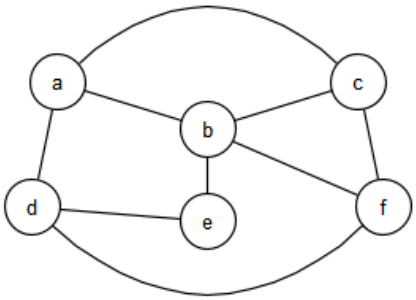


Figura 2.1: Representação geométrica de um grafo G

Seja $e = (u, v)$ uma aresta do grafo. A aresta e é *incidente* a u e a v. Os vértices u e v são ditos *adjacentes* ou *vizinhos*. O conjunto de todos os vizinhos de um vértice v é chamado de *vizinhança* de v, denotado por $N(v)$. O número de arestas incidentes em v determina o *grau* de v, denotado por $d(v)$ [2].

2.2 Método Guloso

O método guloso é um modelo de desenvolvimento de algoritmos que pode ser caracterizado como imediatista, sendo um dos métodos mais simples e diretos para busca de soluções. Durante a execução, o método guloso faz escolhas ótimas locais a cada iteração, sem considerar como cada escolha afeta a solução geral [5].

Em alguns cenários, o método guloso é capaz de encontrar uma solução ótima. Porém, na maior parte dos casos, são encontradas soluções aproximadas.

Quando o método guloso é aplicado a um problema de otimização, o conjunto solução deve atender a um critério de minimização (ou maximização) e satisfazer uma propriedade P. Dessa forma, dado um conjunto S, constrói-se um conjunto solução $S' \subseteq S$ por meio de um processo iterativo onde, a cada etapa, S' deve satisfazer a uma dada propriedade P [14].

2.3 Complexidade Computacional

A Complexidade Computacional tem como objetivo fazer a classificação da dificuldade de resolução dos problemas, medindo a quantidade de recursos computacionais necessários para resolvê-los. Ela é focada em problemas de decisão, onde apenas duas respostas são possíveis: SIM ou NÃO.

Nessa classificação, a classe P consiste dos problemas para os quais existem algoritmos polinomiais. Um algoritmo é dito polinomial se sua complexidade é $O(n^c)$ para todas as entradas de tamanho n, sendo c uma constante.

A classe NP é formada pelos problemas de decisão que admitem um algoritmo capaz de verificar, em tempo polinomial, se uma resposta SIM é válida [6].

Uma das formas mais usadas para provar que um problema integra uma classe é por meio da *transformação polinomial* entre problemas. Um problema Π pode ser transformado polinomialmente (ou reduzido) em um problema Π' se, para qualquer instância I de Π, é possível construir uma instância I' de Π' em tempo polinomial, de tal forma que I tem resposta SIM para Π se, e somente se, I' tiver resposta SIM para Π'.

Um problema Π é NP-completo se Π pertencer a NP e todos os problemas pertencentes a NP forem redutíveis a Π. Devido à transitividade de transformações entre problemas,

para provar que Π é NP-completo, não é necessário transformar todos os demais problemas de NP em Π; basta que um problema NP-completo seja transformado em Π. Alguns exemplos de problemas NP-completos são o Problema do Conjunto Dominante e o Problema do Ciclo Hamiltoniano (cujo objetivo é determinar se existe um ciclo que passa por cada vértice do grafo exatamente uma vez).

Um problema de decisão Π é NP-difícil se todo problema em NP puder ser transformado em Π. Esta classe consiste de problemas potencialmente mais difíceis que os NP-completos, uma vez que não necessariamente pertencem a NP.

Quando o problema em questão é de otimização, diz-se que o problema é NP-difícil quando sua versão de decisão for um problema NP-completo.

Capítulo 3

Heurísticas

Considerando que o Problema da k-Dominação Mínima (versão de otimização do Problema da k-Dominação) é NP-Difícil, as soluções ótimas não podem aparentemente ser obtidas em um limite de tempo viável para grandes instâncias. Dessa forma, uma das alternativas é o uso dos métodos heurísticos, como os algoritmos gulosos.

Os métodos heurísticos são uma técnica utilizada para encontrar soluções mais rapidamente, em casos onde outros métodos exatos são muito custosos computacionalmente.

Este capítulo aborda as heurísticas criadas e implementadas ao longo do trabalho, construídas para encontrar soluções aproximadas para o Problema da k-Dominação Mínima.

Foram desenvolvidos quatro algoritmos gulosos e duas variações deles para tratar o problema em um tempo computacional aceitável. Como será visto no Capítulo 4, esses seis algoritmos foram implementados e comparados com o objetivo de determinar uma heurística com melhor desempenho, isto é, a que encontra conjuntos solução mais próximos das soluções exatas.

3.1 Heurística gulosa baseada em graus

Nesta seção, é apresentada a primeira heurística desenvolvida no trabalho. O Algoritmo 3.1 recebe como entrada um grafo $G = (V, E)$ e uma constante k, que representa o requisito necessário para os vértices serem dominados. A cada iteração, o algoritmo escolhe, dentre os vértices ainda não dominados, um vértice $v \in V$ de maior grau $D[v]$.

O vetor de requisitos R é inicializado com o valor k para cada vértice v. Ao longo do algoritmo, este vetor será utilizado para manter o controle sobre a quantidade de vizinhos

Algoritmo 3.1: Heurística gulosa para k-dominação, considerando vértices com maior grau.

Entrada: Grafo $G = (V, E)$ e constante k.
Saída: Conjunto S capaz de k-dominar G.

```
1  para cada v ∈ V faça
2      R[v] ← k          // Inicialização do vetor de requisitos.
3      D[v] ← d(v)       // Inicialização do vetor de graus.
4  S ← ∅                 // S é o conjunto solução em construção.
5  F ← V    // O conjunto dos vértices ainda não dominados.
6  enquanto F ≠ ∅ faça
7      Seja v o vértice de maior grau D[v] em F.
8      S ← S ∪ {v}
9      para cada u ∈ N(v) faça
10         se u ∈ F então
11             R[u] ← R[u] − 1
12             D[u] ← D[u] − 1
13             se R[u] = 0 então
14                 F ← F \ {u}                    // u foi dominado.
15     F ← F \ {v}
16 retorne S
```

que já foram adicionados ao conjunto solução S. Se um dos vizinhos de v pertencer a S, $R[v]$ é atualizado tal que $R[v] = R[v]-1$. Quando $R[v] = 0$, o vértice v foi dominado, pois já possui pelo menos k vizinhos em S. A cada iteração, o vértice v e eventuais vizinhos dominados são removidos de F.

A dominação de todos os vértices é garantida pelo comando de parada, que ocorre quando $F = \emptyset$, sinalizando que não há mais vértices não dominados no grafo.

A Figura 3.1 ilustra a aplicação do Algoritmo 3.1 em um grafo. Normalmente, uma heurística tem melhor aproveitamento quando é aplicada a grafos de dimensões elevadas, onde encontrar uma solução ótima é um processo custoso. Neste caso, em específico, a heurística encontrou a solução ótima, mas esta ocorrência é incomum em instâncias maiores.

O grafo $G = (V, E)$ do exemplo é dado por $V = \{a, b, c, d, e, f\}$ e $E = \{(a, b), (a, c), (a, d), (b, e), (c, f), (d, f), (e, f)\}$. Além disso, o valor da constante k será 2.

Durante as iterações do algoritmo, os vértices com borda vermelha indicam aqueles que foram incluídos no conjunto solução S e as arestas coloridas de vermelho indicam como a dominação do vértice v será propagada. Após o processamento de v, as arestas in-

cidentes a v são grafadas como linhas cinzas traçadas. Os vértices dominados pelo critério de pelo menos k vizinhos dentro do conjunto k-dominante são marcados com borda azul, para diferenciá-los dos vértices contidos em S.

Na primeira iteração, ocorre empate entre os vértices a e f ($D[a] = D[f] = 3$). Como o Algoritmo 3.1 escolhe o vértice de maior grau ainda não dominado na iteração para incluí-lo no conjunto solução S e não há critério de desempate, o vértice a é selecionado arbitrariamente. Em seguida, o algoritmo propaga no grafo os efeitos da inclusão do vértice a no conjunto solução.

Como resultado da primeira iteração do algoritmo, temos o grafo ilustrado na Figura 3.1(d). Os requisitos dos vértices vizinhos de a foram atualizados após a inclusão de a em S.

Na segunda iteração, novamente o vértice de maior grau atualizado é escolhido. Desta vez, como não houve empate, o vértice f é selecionado, e os requisitos dos vértices adjacentes são alterados. Por consequência da atualização dos requisitos, pelo segundo critério da k-dominação, os vértices c e d são considerados dominados (dois de seus vizinhos fazem parte do conjunto solução).

Na última iteração, há outro caso de empate: os vértices b e e que tem $D[b] = D[e] = 1$. Considerando a escolha de b pelo algoritmo, o único ainda não dominado no grafo é o vértice e. Portanto, a propagação pela vizinhança, de acordo com o algoritmo, só afetará este vértice, pois os demais já foram dominados. Após a atualização do requisito de e, o grafo é completamente k-dominado, com $S = \{a, b, f\}$ sendo o conjunto solução retornado.

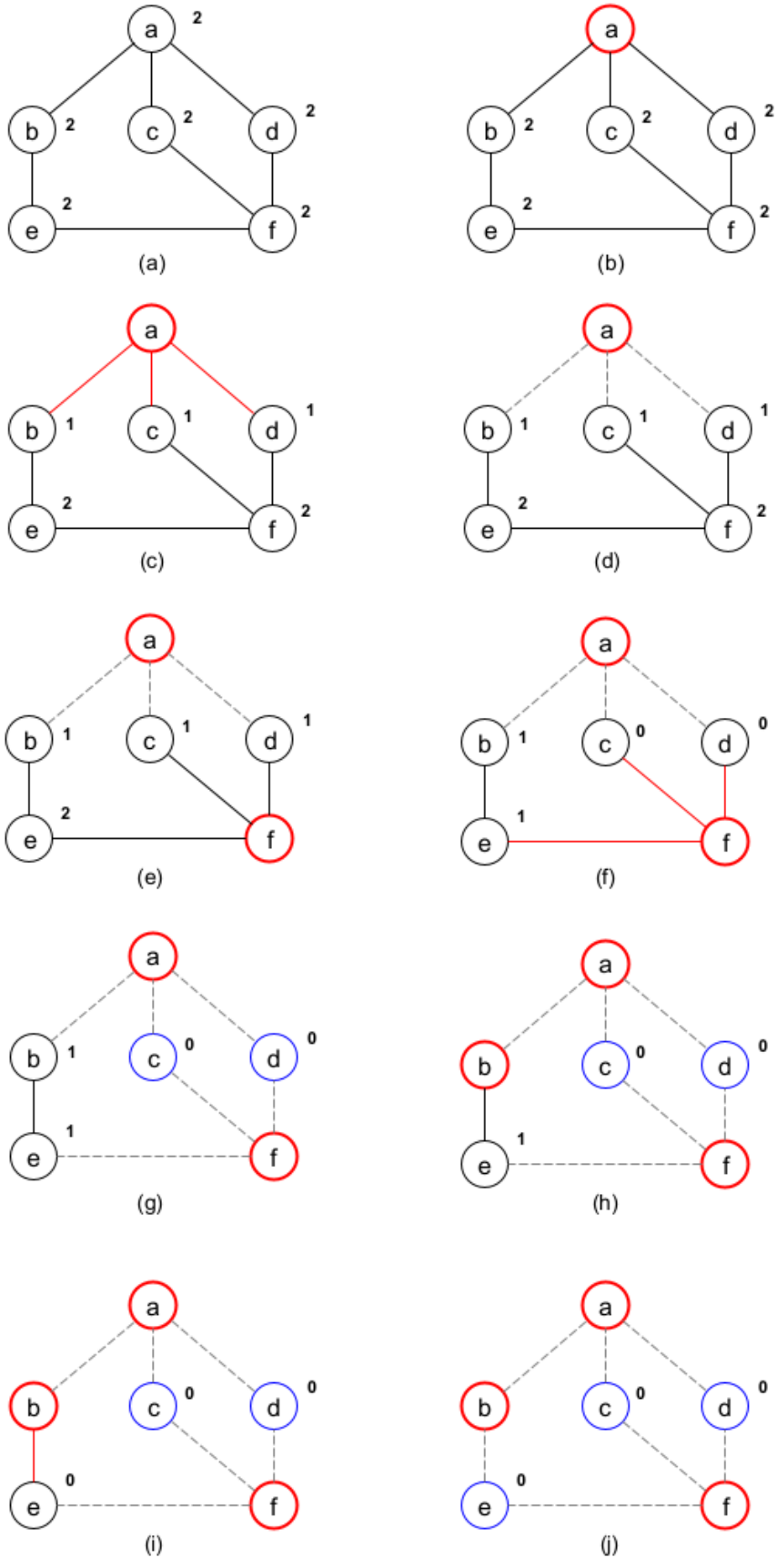


Figura 3.1: Exemplo de execução do Algoritmo 3.1.

3.2 Heurística gulosa baseada em requisitos

Assim como o Algoritmo 3.1, o algoritmo apresentado a seguir é uma heurística gulosa. Entretanto, os dois algoritmos adotam critérios de escolha distintos com o intuito de estudar como diferentes critérios de decisão afetam a eficiência da heurística.

Algoritmo 3.2: Heurística gulosa para k-dominação, considerando vértices com maior requisito.

Entrada: Grafo $G = (V, E)$ e constante k.
Saída: Conjunto S capaz de k-dominar G.

```
1  para cada v ∈ V faça
2      R[v] ← k
3  S ← ∅                          // S é o conjunto a ser construído.
4  F ← V      // Conjunto dos vértices ainda não dominados.
5  enquanto F ≠ ∅ faça
6      Seja v o vértice com maior requisito R[v] em F.
7      S ← S ∪ {v}
8      para cada u ∈ N(v) faça
9          se u ∈ F então
10             R[u] ← R[u] − 1
11             se R[u] = 0 então
12                 F ← F \ {u}                    // u foi dominado.
13     F ← F \ {v}
14 retorne S
```

O Algoritmo 3.2 recebe como entrada um grafo $G = (V, E)$ e uma constante k, que consiste no requisito para que cada vértice seja k-dominado. Como o critério de escolha entre os vértices é o requisito, o algoritmo seleciona o vértice de maior requisito que ainda não tenha sido dominado.

Vale observar que, na iteração inicial, todos os requisitos tem valor k. No entanto, ao longo da execução, conforme vértices são acrescentados ao conjunto S, os requisitos de seus vizinhos são alterados.

Assim como no Algoritmo 3.1, a nova heurística mantém as informações sobre os vértices ainda não dominados em um conjunto auxiliar F, que é atualizado a cada iteração, com a retirada do vértice v adicionado a S e dos vértices que passaram a ser k-dominados pela inclusão de v em S. O algoritmo termina quando $F = \emptyset$, indicando que todos os vértices foram k-dominados.

Para ilustrar o funcionamento do Algoritmo 3.2, será utilizado o grafo da Figura 3.2, com $k = 2$. Para este grafo, uma solução ótima é formada pelos vértices $\{b, e, d\}$ e a heurística encontra uma solução com quatro vértices.

Assim como na Figura 3.1, as etapas ocorridas durante as iterações do algoritmo serão marcadas por um código de cores. Os vértices acrescentados ao conjunto solução S terão suas bordas coloridas de vermelho e a sinalização dos efeitos da inclusão de um vértice v em S sobre sua vizinhana será feita utilizando arestas vermelhas. Posteriormente, ao término do processamento de v, as arestas serão indicadas com linhas cinzas traçadas. Os vértices k-dominados por seus vizinhos serão marcados com bordas azuis para distinguí-los dos vértices que fazem parte de S.

Além disso, ao longo destas iterações, os valores dos requisitos $R[v]$ serão atualizados para os vértices v afetados por aqueles adicionados ao conjunto k-dominante.

O Algoritmo 3.2 escolhe o vértice com maior requisito para dar início à iteração. Contudo, inicialmente, todos os vértices têm o mesmo valor de requisito k e, por isso, o algoritmo escolhe qualquer um deles. Para o exemplo, o primeiro escolhido será o vértice a e sua adição ao conjunto k-dominante S faz com que os requisitos dos vértices adjacentes sejam decrementados.

Após a atualização dos requisitos, dá-se início à segunda iteração, que parte do grafo ilustrado na Figura 3.2 (d). Nesta iteração, há três vértices empatados: c, e, f. Como não há critério de desempate, qualquer vértice dentre os empatados pode ser escolhido. O algoritmo escolhe, então, o vértice c.

Os requisitos dos vértices adjacentes a c são alterados após a sua inclusão ao conjunto S. Por consequência, de acordo com o segundo critério de k-dominação, o vértice d foi k-dominado, pois dois de seus vizinhos fazem parte do conjunto solução.

Na terceira iteração, o vértice f é o escolhido por ter o maior requisito e é acrescentado a S. Em seguida, sua vizinhança tem seus requisitos atualizados. Como o vértice d já foi k-dominado em iterações anteriores, ele não é afetado nesta etapa da execução. Já o vértice e, que precisava de somente mais um vizinho em S, passou a ser k-dominado.

Como última etapa da execução, o vértice b é adicionado ao conjunto S, encerrando o processamento do algoritmo ($F = \emptyset$).

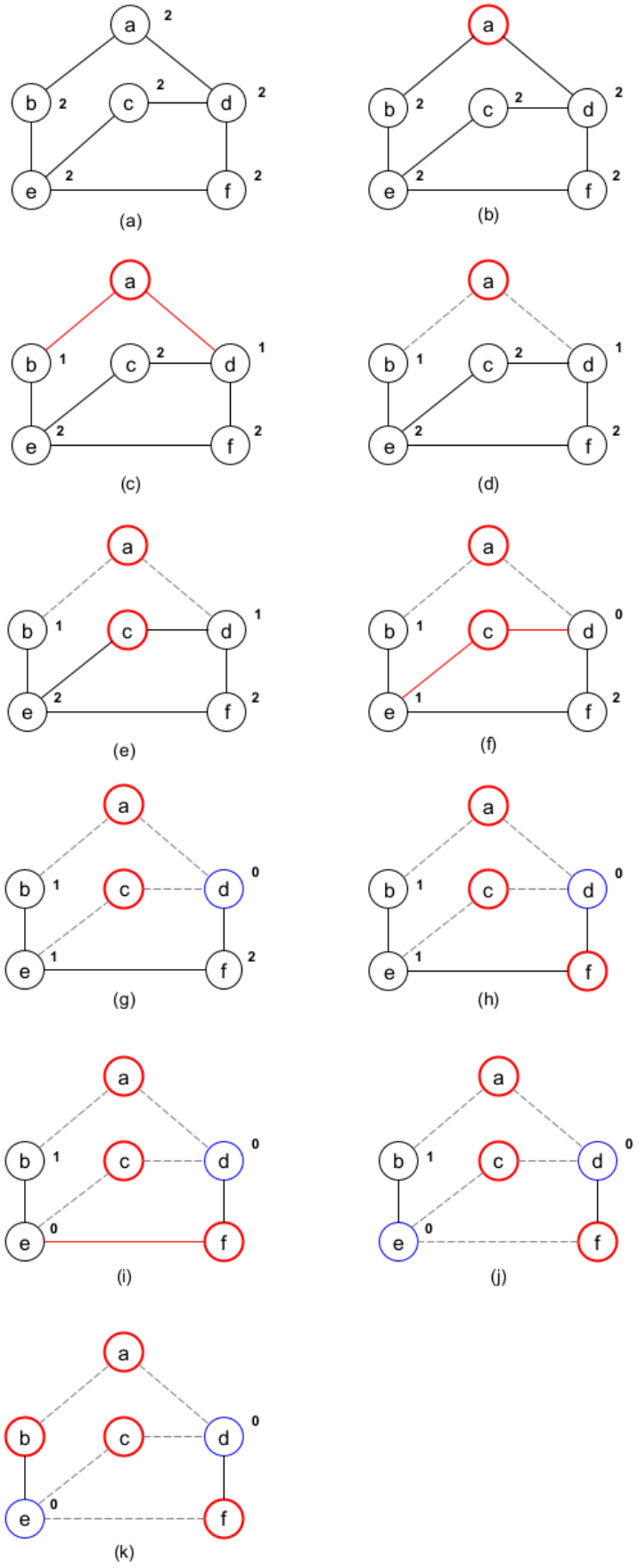


Figura 3.2: Exemplo de execução do Algoritmo 3.2.

3.3 Heurística gulosa baseada em graus e requisitos

Após o desenvolvimento de heurísticas utilizando o grau e o requisito do vértice separadamente, a criação de um algoritmo híbrido auxilia no entendimento de como cada uma dessas características afetam o desempenho da heurística. Para criar uma relação entre o grau e o requisito foi desenvolvida uma equação que considera ambos, dando maior peso ao que está no numerador, permitindo a variação para testes.

O algoritmo apresentado a seguir, inspirado no método proposto por Cordasco *et al.* [4], tem como critério de escolha a equação $\varphi_R(v) = \frac{R[v]^2}{n-D[v]}$ que recebe o quadrado do requisito de um vértice como numerador e número total de vértices do grafo menos o grau do vértice como denominador. Dessa forma, leva-se em consideração os dois critérios de escolha analisados nas heurísticas anteriores, destacando o requisito como fator de maior peso na equação.

Algoritmo 3.3: Heurística gulosa para k-dominação, considerando os resultados da equação φ_R.

Entrada: Grafo $G = (V, E)$ e constante k.
Saída: Conjunto S capaz de k-dominar G.

```
1  para cada v ∈ V faça
2      R[v] ← k
3      D[v] ← d(v)
4  S ← ∅              // S é o conjunto solução em construção.
5  F ← V    // O conjunto dos vértices ainda não dominados.
6  enquanto F ≠ ∅ faça
7      Seja v o vértice em F que maximiza φ_R(v) = R[v]²/(n−D[v]).
8      S ← S ∪ {v}
9      para cada u ∈ N(v) faça
10         se u ∈ F então
11             R[u] ← R[u] − 1
12             D[u] ← D[u] − 1
13             se R[u] = 0 então
14                 F ← F \ {u}                    // u foi dominado.
15     F ← F \ {v}
16 retorne S
```

O Algoritmo 3.3 recebe como entrada um grafo $G = (V, E)$ e uma constante k correspondente ao valor dos requisitos necessários para a k-dominação do vértice. Os vértices são inseridos no conjunto solução S de acordo com o critério de decisão apresentado, es-

colhendo a cada iteração o vértice não dominado que maximiza a equação.

Na primeira iteração, todos os vértices tem o mesmo requisito k. Por isso, neste caso, o critério que influencia a escolha do maior valor de $\varphi_R(v)$ é o grau.

Como nos casos anteriores, o controle sobre a parada do algoritmo é feito utilizando o conjunto auxiliar F, que é atualizado a cada iteração com a retirada dos vértices k-dominados.

O exemplo da Figura 3.3 ilustra o funcionamento do Algoritmo 3.3. Neste caso, em particular, a heurística encontra uma solução ótima com três vértices.

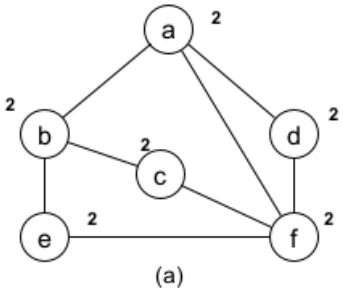
Para sinalizar as iterações do algoritmo no exemplo, os vértices adicionados ao conjunto k-dominante serão destacados com bordas vermelhas, assim como as arestas vizinhas, que serão coloridas de vermelho para sinalizar a propagação do efeito da inclusão do vértice v ao conjunto solução S. Após a iteração, as arestas, antes vermelhas, serão grafadas como linhas cinzas traçadas. Já os vértices k-dominados pelos seus vizinhos serão assinalados com bordas azuis, para diferenciá-los dos vértices contidos em S. A Figura 3.3 apresenta em tabelas o valor da equação φ_R para cada vértice do grafo.

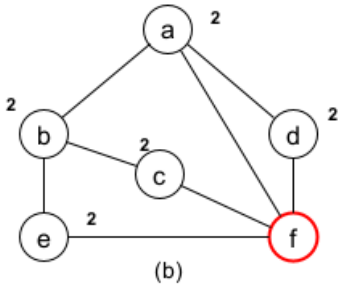
Na iteração inicial, o vértice f é o que maximiza a equação e é, portanto, adicionado ao conjunto k-dominante S. Sua inclusão ao conjunto é propagada para sua vizinhança, atualizando os requisitos dos vértices a, c, d e e. Com as mudanças nos graus, requisitos e tamanho do grafo, os resultados da equação para cada vértice também serão alterados.

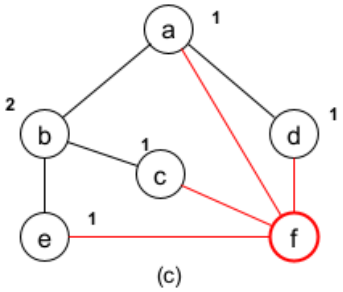
De acordo com os valores atualizados, o próximo adicionado a S é o vértice b. Pelo segundo critério da k-dominação, a propagação do acréscimo de b a S afeta seus vértices adjacentes. Os vértices a, c e e passam a ser k-dominados por terem dois vizinhos em S.

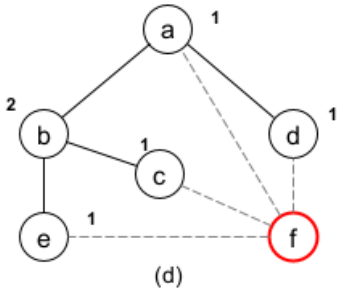
O único vértice restante para a última iteração é o vértice d. Assim, a única alternativa para a k-dominação do vértice é acrescentá-lo a S. O conjunto resultante das iterações do algoritmo é $S = \{b, f, d\}$.

v	$\varphi_R(v)$
a	1,33
b	1,33
c	1
d	1
e	1
f	**2**

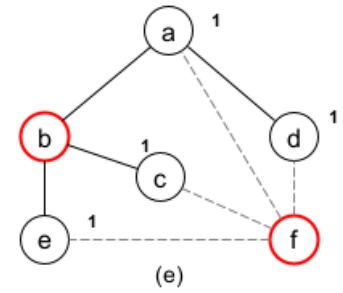


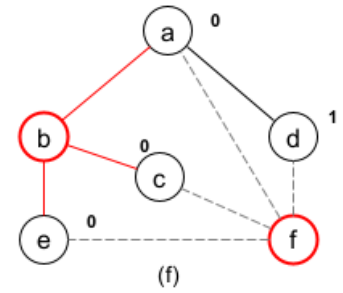


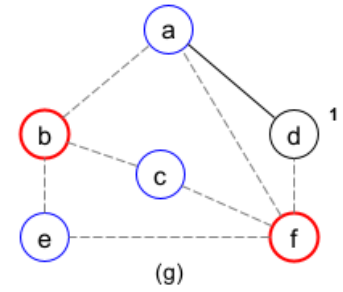




v	$\varphi_R(v)$
a	0,33
b	**2**
c	0,25
d	0,25
e	0,25







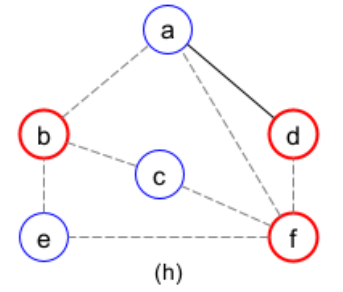


Figura 3.3: Exemplo de execução do Algoritmo 3.3.

Além do Algoritmo 3.3, também foi desenvolvida uma variante utilizando a maximização da equação $\varphi_G(v) = \frac{D[v]^2}{n-R[v]}$, que prioriza o grau dos vértices. A motivação para o desenvolvimento e teste da variante foi uma melhor visualização da influência de cada um dos fatores sobre os resultados. O Algoritmo 3.3 deixa os requisitos dos vértices como o fator mais influente da equação, enquanto sua variante substitui os requisitos pelos graus. A Figura 3.4 mostra um exemplo das diferenças nos resultados de acordo com a equação usada.

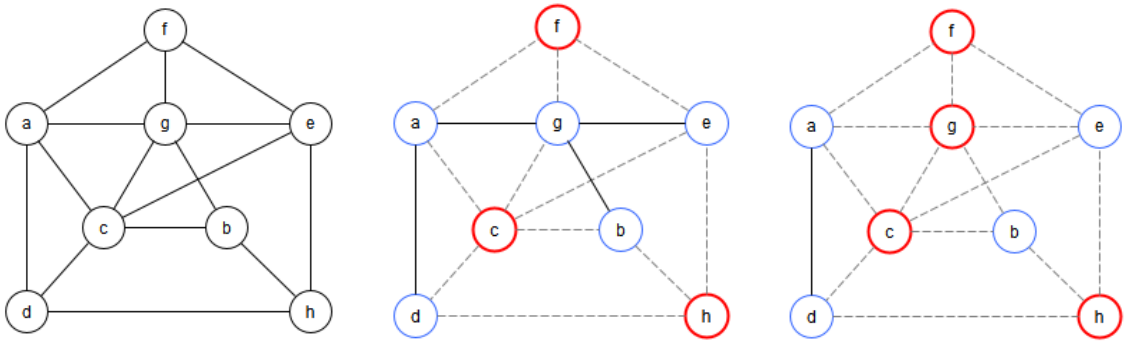


Figura 3.4: Exemplo de grafo que ilustra a diferença entre os resultados do Algoritmo 3.3, considerando φ_R (b) e φ_G (c).

Com o resultado da variante apresentado na Figura 3.4(c), onde é utilizada a maximização de $\varphi_G(v)$ como critério de escolha, temos que, inicialmente, os vértices c e g resultam no maior valor possível para a equação, pois os requisitos são iguais para todos os vértices, e c e g tem grau máximo no grafo. O algoritmo seleciona o vértice c de forma arbitrária.

Na segunda iteração, o algoritmo escolhe o vértice g, que agora tem seus atributos $R[v] = 1$ e $D[v] = 4$. Em seguida, seleciona o vértice h, que após duas iterações tem o maior grau. Por fim, o vértice f, que é o último ainda não dominado, é inserido em S. Assim, o conjunto k-dominante resultante é $S = \{c, g, h, f\}$.

Enquanto isso, aplicando o Algoritmo 3.3 com a maximização de $\varphi_R(v)$ como critério de escolha, a iteração inicial é a mesma apresentada anteriormente, e o vértice c é selecionado. Porém, os vértices de maior grau tem desvantagem neste caso, eliminando o vértice g das opções que maximizam a equação, diferente do que ocorre na variante.

Os valores da equação para a segunda iteração apresentam um empate entre f e h. O algoritmo seleciona o vértice f, e esta escolha não afeta os valores de $R[v]$ e $D[v]$ para o vértice h, que, por consequência, é o escolhido na iteração seguinte. O conjunto k-dominante S resultante destas iterações é $S = \{c, f, h\}$, conforme a Figura 3.4(b).

Nos exemplos apresentados, os conjunto resultantes tem diferença de tamanho um. Mas, com grafos maiores, como os utilizados ao longo dos testes, as diferenças tendem a se tornar mais significativas.

3.4 Heurística baseada em requisitos com desempate

O Algoritmo 3.4 utiliza o mesmo critério de decisão do Algoritmo 3.2, com o acréscimo de uma regra de desempate, que visa otimizar suas escolhas. Neste caso, quando mais de um vértice tiver o mesmo requisito, o vértice escolhido será o de maior grau.

Algoritmo 3.4: Heurística gulosa para k-dominação, considerando vértices com maior requisito e desempate considerando o maior grau.

Entrada: Grafo $G = (V, E)$ e constante k.
Saída: Conjunto S capaz de k-dominar G.

```
1  para cada v ∈ V faça
2      R[v] ← k
3      D[v] ← d(v)
4  S ← ∅                          // S é o conjunto a ser construído.
5  F ← V      // Conjunto dos vértices ainda não dominados.
6  enquanto F ≠ ∅ faça
7      Seja r_max o maior requisito k dos vértices em F.
8      P = {u ∈ F | R[u] = r_max}
9      Seja v o vértice de maior grau atualizado D[v] em P.
10     S ← S ∪ {v}
11     para cada u ∈ N(v) faça
12         se u ∈ F então
13             R[u] ← R[u] − 1
14             D[u] ← D[u] − 1
15             se R[u] = 0 então
16                 F ← F \ {u}                    // u foi dominado.
17     F ← F \ {v}
18 retorne S
```

Como nos algoritmos anteriores, o Algoritmo 3.4 recebe como entrada um grafo $G = (V, E)$ e uma constante k, representando os requisitos para que um vértice seja dominado por seus vizinhos, sem que seja necessária sua inclusão no conjunto solução S. O vetor de requisitos $R[v]$ armazena o valor de k associado a cada vértice, monitorando os vizinhos k-dominados.

Na iteração inicial, todos os vértices tem o mesmo requisito $R[v] = k$, fazendo com que o vértice escolhido seja aquele com maior grau, pelo critério de desempate. Ao longo das iterações, os requisitos serão atualizados, tornando a escolha dos vértices um sistema de duas etapas: os vértices com maior requisito são selecionados e, em caso de empate, o de maior grau será escolhido.

A Figura 3.5 ilustra o funcionamento do Algoritmo 3.4. Neste exemplo, como ocorrido em algoritmos anteriores, a heurística encontra uma solução ótima com três vértices. Para demonstrar as iterações do algoritmo no exemplo, será utilizado o mesmo esquema de cores que nas seções anteriores.

Na primeira iteração do algoritmo, todos os vértices têm o mesmo valor de requisito, porém o critério de desempate pelo maior grau reduz as opções de escolha para duas: os vértices b e e. No exemplo, o algoritmo escolhe b e sua inclusão ao conjunto S é propagada para sua vizinhança, alterando os requisitos dos vértices a, d e e.

Na segunda iteração, novamente ocorre empate após o processamento dos critérios de escolha, e o vértice c é selecionado. A propagação da inclusão de c em S altera os requisitos dos vértices a e f. No caso de a, seu requisito se torna zero e ele é considerado k-dominado, pois tem dois vizinhos que fazem parte do conjunto k-dominante.

Para a terceira iteração, os vértices d, e e f têm o mesmo requisito. Contudo, o vértice e tem o maior grau e por isso é escolhido pelo critério de desempate. Sua inclusão no conjunto S torna os vértices d e f também k-dominados. Assim, o conjunto resultante das iterações do Algoritmo 3.4 é $S = \{b, c, e\}$.

Como no Algoritmo 3.3, a criação do critério de desempate para o Algoritmo 3.4 permite o desenvolvimento de uma variante. Nesta, o desempate foi feito de acordo com os vértices de menor grau. A criação desta segunda versão teve como objetivo a visualização de como uma escolha contraintuitiva poderia afetar os conjuntos solução obtidos pela heurística. A Figura 3.6 ilustra um caso onde existe uma diferença entre as soluções de acordo com o critério utilizado.

Aplicando o Algoritmo 3.4, onde o critério de escolha é o vértice com maior requisito e o critério de desempate é o vertice de maior grau temos o melhor resultado para o grafo ilustrado na Figura 3.6(a).

Inicialmente, com os requisitos empatados, temos como primeira escolha o vértice de maior grau g e a sua inclusão em S se propaga para os vértices adjacentes, alterando seus valores de requisito. Por consequência, os únicos vértices com requisitos inalterados são d e h. Como o vértice h tem o maior grau, ele é o escolhido na segunda iteração.

Com a adição de h ao conjunto solução, os vértices b e e são dominados pelo segundo critério da k-dominação. Novamente, todos os vértices têm o mesmo valor de requisito, fazendo com que o critério de desempate sirva como o único para decidir o próximo vértice

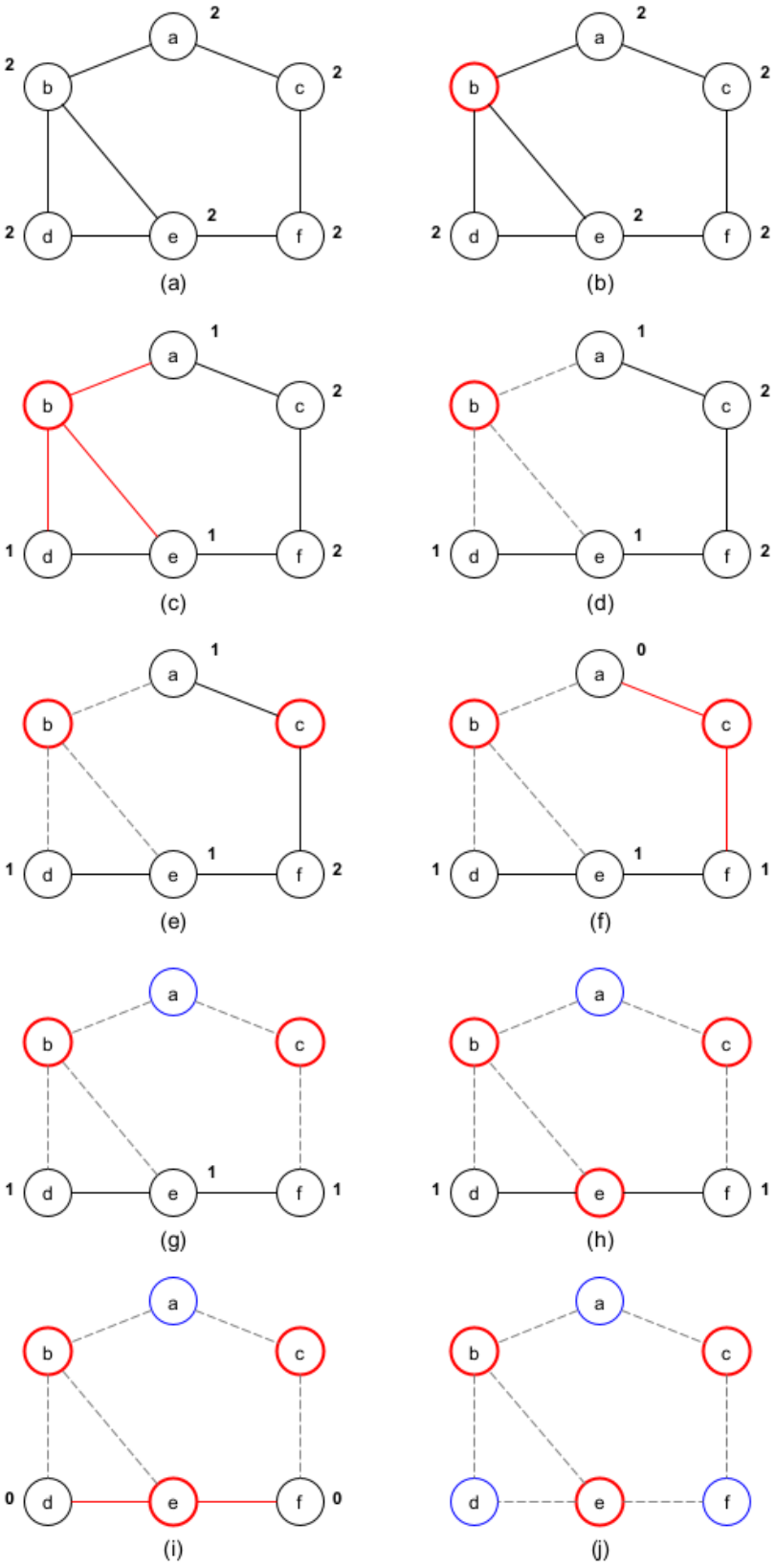
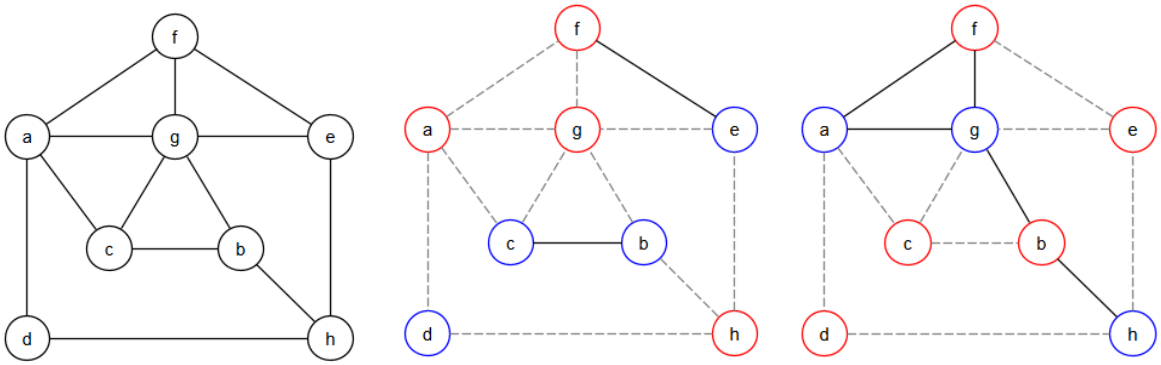


Figura 3.5: Exemplo de execução do Algoritmo 3.4.



(a) (b) (c)

Figura 3.6: Exemplo de grafo que apresenta a diferença entre os resultados do Algoritmo 3.4 considerando o grau maior e o grau menor como critérios de desempate, respectivamente.

adicionado ao conjunto solução.

O vértice a, com o maior grau, é o escolhido na terceira iteração, dominando os vértices c e d. Por último, o vértice f é adicionado ao conjunto k-dominante por ser o único ainda não k-dominado no grafo. O conjunto k-dominante resultante das iterações com critério de desempate de maior grau é $S = \{g, h, a, f\}$.

Utilizando o critério de desempate da variante, que leva em consideração o menor grau quando há empate nos requisitos, as escolhas mudam desde o início das iterações.

Na primeira iteração, o escolhido é vértice d, por ter o menor grau. Em seguida, os vértices b, c, e, f e g têm requisitos e graus iguais entre si. O algoritmo escolhe o vértice e para dar início à segunda iteração. O vértice e é acrescentado ao conjunto solução, consequentemente dominando h. Em seguida, os vértices b e c estão empatados, e o algoritmo escolhe c, que domina g e a pelo segundo critério da k-dominação.

Na quarta iteração, os vértices restantes estão empatados. Como não há conexão entre eles, a ordem de adição ao conjunto solução não altera o resultado final. Sendo assim, o algoritmo escolhe o vértice f e, na última iteração, adiciona b, que é último vértice ainda não k-dominado por qualquer critério. O conjunto resultante é $S = \{d, e, c, f, b\}$.

Nos exemplos comparativos, os conjuntos k-dominantes têm diferença de tamanho um. Mas, esse foi um resultado para o grafo usado no exemplo; as vantagens e desvantagens de cada algoritmo ficam mais evidentes com o aumento do tamanho dos grafos utilizados, como será demonstrado no Capítulo 4.

Capítulo 4

Resultados obtidos

Este capítulo apresenta os resultados dos testes realizados com as seis heurísticas detalhadas anteriormente. Além disso, o ambiente computacional utilizado para a execução dos testes será detalhado.

O estudo de desempenho das heurísticas foi dividido em duas partes. Na Seção 4.1, o estudo que compara as soluções dos métodos heurísticos com a solução ótima é detalhada. Já na Seção 4.2, o objetivo é comparar as soluções encontradas para cada heurística, além do seu tempo de execução.

Nos gráficos e tabelas apresentados ao longo deste capítulo, as heurísticas implementadas serão representadas por siglas. Assim, o Algoritmo 3.1 é representado por HG, o Algoritmo 3.2 por HR, o Algoritmo 3.3 por $H\varphi_R$ e a variante deste algoritmo por $H\varphi_G$. Adicionalmente, o Algoritmo 3.4 é representado por $H\Delta$ e sua variante, que considera o menor grau como critério de desempate, por $H\delta$.

Para padronizar os testes e minimizar possíveis oscilações de desempenho devido ao uso de máquina local, os métodos heurísticos e o modelo utilizado para a obtenção de soluções exatas foram executados em máquina virtual no ambiente de demonstração disponibilizado pelo *Google Cloud* [8]. A máquina virtual escolhida para os testes foi a *e2-highmem-4*, com 4 CPUS virtuais, 32GB de memória RAM e sistema operacional Ubuntu.

As instâncias utilizadas nos testes foram geradas de forma randômica: dado o número n de vértices desejados e um número real p, cada uma das $\frac{n^2-n}{2}$ arestas tem probabilidade p de existir. O valor de p corresponde aproximadamente à densidade do grafo [1].

[1] A densidade de um grafo corresponde à sua quantidade de arestas dividida pelo número máximo de arestas que ele pode conter.

Todas as instâncias foram salvas em arquivos binários. Esses arquivos são compostos por $n+m+1$ registros, formados por dois inteiros cada. O primeiro registro do arquivo tem o número de vértices n e o número de arestas m do grafo. Os n registros seguintes armazenam dois inteiros referentes ao rótulo original do vértice e ao seu identificador em números sequenciais no intervalo de $[1, n]$. Nos grafos gerados aleatoriamente, o identificador e o rótulo são equivalentes. Os últimos m registros armazenam os identificadores das extremidades de cada aresta. Os arquivos binários com as instâncias foram disponibilizados no GitHub (`https://github.com/RangelBeatriz/instancias-tcc`).

Os métodos heurísticos foram implementados em C++, fazendo uso da biblioteca BibGrafos [10] para auxiliar nas operações realizadas nos vértices dos grafos. Para sua execução na máquina virtual, o compilador GCC (GNU Compiler Collection) foi usado.

4.1 Comparação entre heurísticas e a solução exata

Para a comparação dos resultados das heurísticas com as soluções ótimas, o modelo de programação linear apresentado em "Propagação em Redes: Dominação Vetorial e Seleção de Alvos" [11] foi adaptado para o Problema da k-Dominação. Esta adaptação consiste em adotar o valor de k como o requisito de todos os vértices.

O modelo recebe como entrada um grafo $G = (V, E)$, através de sua representação por matriz de adjacências ADJ, e uma constante k. No modelo implementado, a saída, que corresponde ao conjunto k-dominante, é dada por um vetor de inteiros S, onde cada posição representa um vértice do grafo e é preenchida com 1 ou 0. Se a posição contém o número 1, o vértice que ela representa está no conjunto k-dominante. Caso contrário, o vértice será dominado pelos vértices adjacentes.

Sejam v um vértice do grafo e k o seu requisito para a dominação. A expressão matemática capaz de representar a verificação da k-dominação desse vértice por um conjunto solução sugerido é dada por:

Função Objetivo:

$$\min \sum_{v \in V} S[v] \tag{4.1.1}$$

Sujeito à:

$$(S[v] \cdot k) + (\sum_{u \in V} ADJ[v,u] \cdot S[u]) \geq k,\ \forall v \in V \tag{4.1.2}$$

$$ADJ[v,u] \in \{0,1\}, \forall v \in V,\ \forall u \in V \tag{4.1.3}$$

$$S[v] \in \{0,1\}, \forall v \in V \tag{4.1.4}$$

$$k \leq \delta(G) \tag{4.1.5}$$

A Expressão 4.1.2 estabelece as condições necessárias para que um vértice v seja considerado k-dominado. Se $S[v] = 1$, então $S[v] \cdot k = k$, tornando a expressão verdadeira. Caso contrário, a k-dominação de v deverá ser atendida por meio de seus vizinhos. Cada vizinho u de v ($ADJ[v,u] = 1$) contribuirá no somatório se fizer parte do conjunto k-dominante ($S[u] = 1$). O vetor S é capaz de k-dominar o grafo G se a expressão for verdadeira para todos os vértices.

As soluções ótimas foram obtidas através da execução do modelo no *CPLEX Optimization Studio*, versão 20.10 [9]. Uma vez que um método capaz de encontrar soluções ótimas para instâncias pequenas foi definido, foi possível prosseguir para as comparações entre os resultados dos métodos heurísticos e as soluções exatas.

Para comparar os resultados de heurísticas com as soluções exatas, foram utilizados grafos de vinte, cinquenta e cem vértices, processados por cada uma das heurísticas e pelo CPLEX para obtenção dos resultados exatos.

Todos os testes foram realizados no mesmo ambiente, para garantir a precisão dos resultados e manter a confiabilidade das comparações. Com isso, foi possível obter dados comparativos que podem ser utilizados como ferramentas para entender as distinções entre o comportamento das heurísticas e das soluções exatas.

Quando as heurísticas desenvolvidas neste trabalho são usadas em grafos pequenos, como acontece na primeira etapa de testes, seus resultados chegam ao valor ótimo para

algumas instâncias. Gradualmente, os resultados se afastam da solução ótima à medida que os grafos aumentam de tamanho e densidade, até que a diferença de resultado entre as heurísticas e o valor exato se estabilize, mantendo uma distância similar para cada uma das densidades, independente do tamanho de cada grafo.

A Tabela 4.1 demonstra os resultados observados ao executarmos o CPLEX e as heuristicas em grafos com vinte vértices.

				Custos das Soluções Heurísticas					
Densidade	ID	k	CPLEX	HG	HR	$H\varphi_G$	$H\varphi_R$	$H\Delta$	$H\delta$
25%	1	2	6	8	8	7	7	7	9
	2	2	6	7	8	8	7	7	9
	3	2	6	7	**6**	7	7	7	8
	4	2	6	8	8	9	8	7	8
	5	2	6	7	7	8	7	7	9
50%	1	3	6	8	9	9	8	8	8
	2	3	5	6	8	6	8	8	9
	3	3	6	8	7	8	8	7	7
	4	3	5	6	7	6	6	6	9
	5	3	6	7	8	7	7	7	8
75%	1	6	8	9	**8**	9	**8**	**8**	9
	2	5	7	8	8	8	9	8	8
	3	6	8	9	**8**	**8**	**8**	**8**	9
	4	5	6	8	7	7	7	7	7
	5	3	4	6	5	6	5	5	5

Tabela 4.1: Comparação entre os resultados do método exato e das heurísticas para grafos com vinte vértices.

Por meio dos dados obtidos para os grafos de vinte vértices, o desempenho das heurísticas pode ser analisado para grafos pequenos. Neste caso, destaca-se que apenas dois algoritmos não chegaram à solução ótima em nenhuma das quinze instâncias analisadas: o Algoritmo 3.1 e a variante do Algoritmo 3.4, $H\delta$.

A visão inicial dos resultados dos algoritmos e a determinação de dois com desempenho menos favorável em grafos pequenos fundamenta a observação dos resultados para outras instâncias, a fim de analisar se o comportamento das heurísticas se mantém com o aumento do número de vértices.

Após os grafos de tamanho vinte, as instâncias de cinquenta vértices foram testadas. A Tabela 4.2 apresenta os resultados da execução do CPLEX e das heurísticas para os grafos gerados com cinquenta vértices.

				Custos das Soluções Heurísticas					
Densidade	ID	k	CPLEX	HG	HR	$H\varphi_G$	$H\varphi_R$	$H\Delta$	$H\delta$
25%	1	2	10	12	15	13	13	12	15
	2	3	13	17	17	20	17	15	17
	3	3	13	14	15	16	15	17	17
	4	3	12	17	17	16	15	14	19
	5	2	9	12	12	13	13	13	15
50%	1	9	18	24	19	22	20	21	21
	2	9	17	22	19	20	20	19	19
	3	9	17	20	21	20	20	20	20
	4	8	17	23	19	20	19	18	19
	5	9	17	21	19	21	21	20	20
75%	1	16	20	22	22	21	23	24	22
	2	14	19	22	21	21	21	20	21
	3	16	21	24	23	23	24	22	23
	4	13	18	21	**18**	21	20	**18**	20
	5	16	21	25	23	23	24	23	23

Tabela 4.2: Comparação entre os resultados do método exato e das heurísticas para grafos com cinquenta vértices.

O aparecimento de soluções ótimas nas heurísticas foi menos frequente em grafos de cinquenta vértices, ocorrendo somente em uma instância de grafo em dois algoritmos distintos, o Algoritmo 3.2 e o Algoritmo 3.4 (colunas HR e $H\Delta$). Aparentemente, essas heurísticas apresentam soluções mais próximas das ótimas, no quadro geral. Porém, como os grafos utilizados são pequenos, ainda não é possível concluir que essa seja a realidade para todos os casos.

A Tabela 4.3 apresenta os resultados para instâncias geradas com cem vértices, onde a diferença de tamanho entre as soluções heurísticas e a solução ótima fica mais evidente.

Nas instâncias de cem vértices, há apenas um caso onde a heurística retorna o custo ótimo. Além disso, algumas tendências observadas nos resultados anteriores acontecem com mais clareza.

				Custos das Soluções Heurísticas					
Densidade	ID	k	CPLEX	HG	HR	$H\varphi_G$	$H\varphi_R$	$H\Delta$	$H\delta$
25%	1	7	28	35	39	35	35	34	36
	2	8	31	40	37	39	39	39	39
	3	7	29	37	35	34	33	32	36
	4	6	25	33	33	35	35	32	34
	5	8	31	41	36	39	38	37	39
50%	1	18	35	44	39	42	41	40	40
	2	16	32	39	37	38	36	37	38
	3	20	39	50	42	46	43	41	44
	4	20	35	43	42	40	39	39	40
	5	18	36	44	40	42	41	40	42
75%	1	32	42	46	44	46	46	44	45
	2	33	43	48	46	47	47	47	45
	3	30	40	47	43	44	45	41	42
	4	31	41	46	43	44	45	**41**	44
	5	32	42	49	45	45	46	44	46

Tabela 4.3: Comparação entre os resultados do método exato e das heurísticas para grafos com cem vértices.

O Algoritmo 3.1 (coluna HG) encontrou soluções piores para todos os grafos de tamanho cem, com exceção de três deles, onde ficou empatado com outras heurísticas. Já a variante do Algoritmo 3.4 (coluna $H\delta$) apresentou resultados com custos próximos aos retornados pelas outras heurísticas para a maior parte das instâncias.

Com os resultados iniciais, comparados à solução ótima obtida com o CPLEX, foi possível analisar o desempenho geral das heurísticas e apontar aquelas que se destacam por apresentarem os melhores ou piores resultados. Contudo, o objetivo do desenvolvimento de heurísticas é ter soluções aproximadas para instâncias onde seria impraticável obter soluções ótimas em tempo computacional viável, o que ainda não é o caso dos testes realizados nesta seção.

4.2 Comparação entre heurísticas

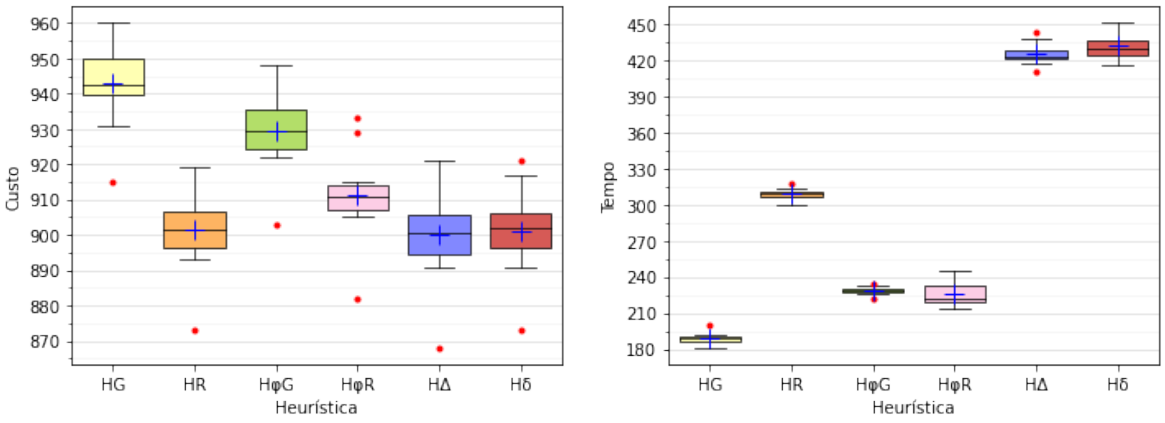
Como mencionado nos capítulos anteriores, heurísticas são criadas visando a obtenção de resultados aproximados para grafos onde buscar a solução ótima é inviável computacionalmente. Portanto, foram feitos testes com instâncias de grafos com dois, cinco e dez mil vértices, com densidades 25%, 50% e 75%, para avaliar o desempenho de todas as heurísticas implementadas neste trabalho.

Para cada combinação de tamanho e densidade, foram realizados testes com dez grafos distintos gerados aleatoriamente. Dessa forma, os resultados apresentados nesta seção são baseados na execução dos seis algoritmos apresentados no Capítulo 3 em noventa instâncias distintas.

Nesta seção, o estudo terá foco em analisar as diferenças entre as heurísticas, comparando o custo das soluções e o tempo de execução de cada uma delas.

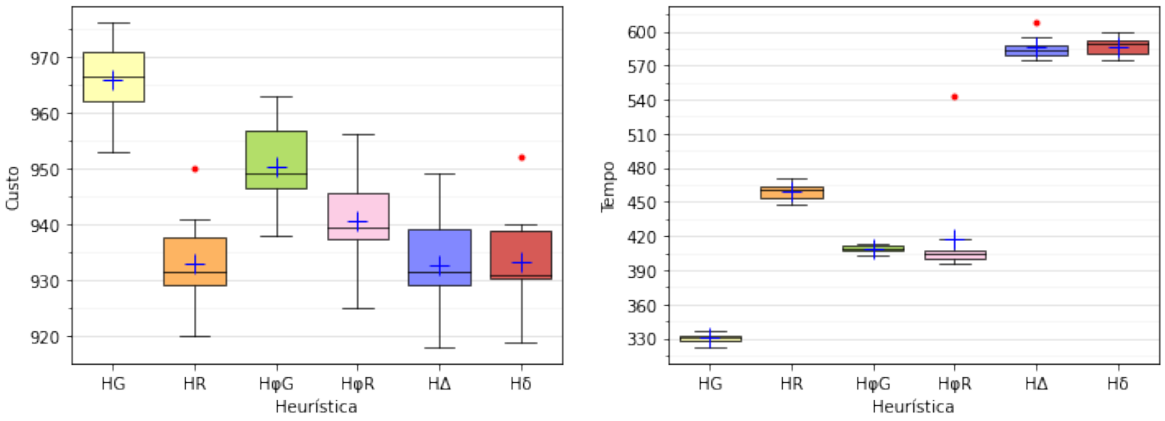
Para melhor visualização das distinções, serão usados gráficos do tipo *boxplot*, gerados através de um programa implementado em Python 3.9, utilizando a API PyPlot dentro da biblioteca Matplotlib. Nesse caso, gráficos *boxplot* são utilizados para mostrar a média, a amplitude dos dados(valores máximo e mínimo) e a presença de *outliers*. A média dos valores é representada pelo símbolo "+", a amplitude dos dados pelas barras acima e abaixo das caixas e os *outliers* por pontos vermelhos fora do alcance da amplitude. Além disso, cada caixa do gráfico tem uma linha horizontal marcada para representar a mediana dos dados. O eixo custo se refere ao tamanho dos conjuntos solução encontrados por cada heurística e o eixo tempo ao tempo de execução dos algoritmos em milisegundos. Os dados completos dos experimentos estão disponibilizados no Apêndice A.

Observando os gráficos apresentados ao longo desta seção, é possível afirmar que os custos são influenciados pelo tamanho e pela densidade dos grafos em todos os casos, mas o tempo só apresenta uma variação significativa devido à mudança de densidade a partir dos grafos de cinco mil vértices.



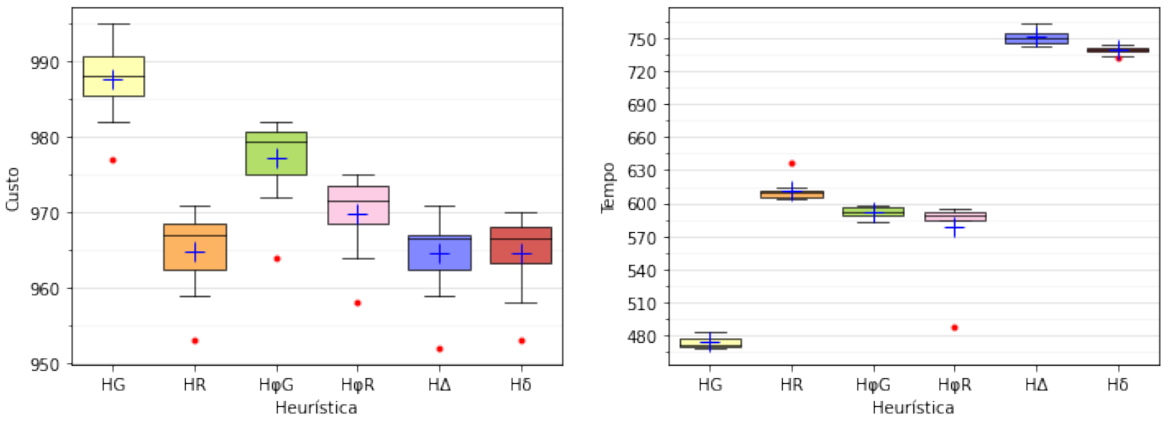
(a) (b)

Figura 4.1: Comparação dos resultados de custo e tempo das heurísticas para grafos de dois mil vértices com 25% de densidade.



(a) (b)

Figura 4.2: Comparação dos resultados de custo e tempo das heurísticas para grafos de dois mil vértices com 50% de densidade.



(a) (b)

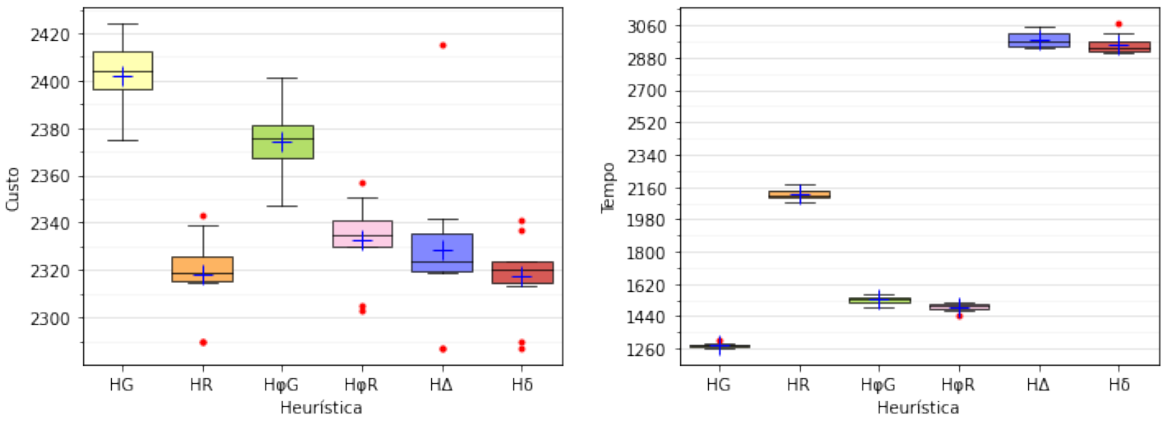
Figura 4.3: Comparação dos resultados de custo e tempo das heurísticas para grafos de dois mil vértices com 75% de densidade.

Para as instâncias de dois mil vértices, cujos resultados são apresentados nas Figuras 4.1, 4.2 e 4.3, a Heurística de Grau (HG) mostra o pior desempenho relacionado a custos, mas o melhor dos tempos de execução.

As heurísticas HR, $H\Delta$ e $H\delta$ têm desempenho similar na análise dos custos, manifestando diferenças mínimas de acordo com a densidade. Contudo, enquanto o tempo de execução da heurística HR ficou mediano em relação às outras, os tempos das heurísticas $H\Delta$ e $H\delta$ são desfavoráveis em todos os casos (provavelmente devido à segunda ordenação atrelada ao critério de desempate).

Os resultados de $H\varphi_G$ e $H\varphi_R$ não são significativos quando comparados aos de outras heurísticas, mas uma comparação entre as duas mostra que o desempenho de $H\varphi_R$ é superior ao de sua variante $H\varphi_G$, tanto para tempo, quanto para custo.

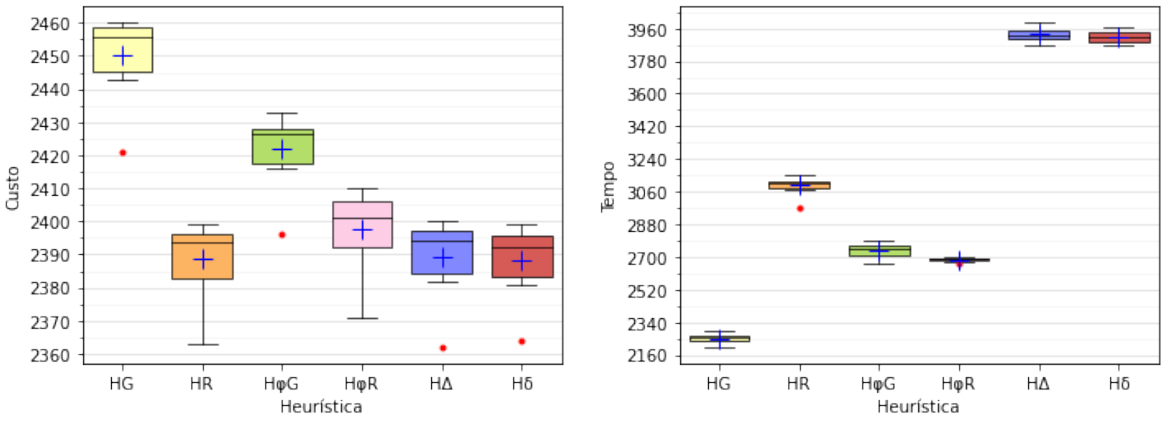
Os gráficos das Figuras 4.4, 4.5 e 4.6 para os grafos de cinco mil vértices reforçam a maioria dos pontos levantados para as instâncias de dois mil vértices com alguns detalhes melhor esclarecidos.



(a) (b)

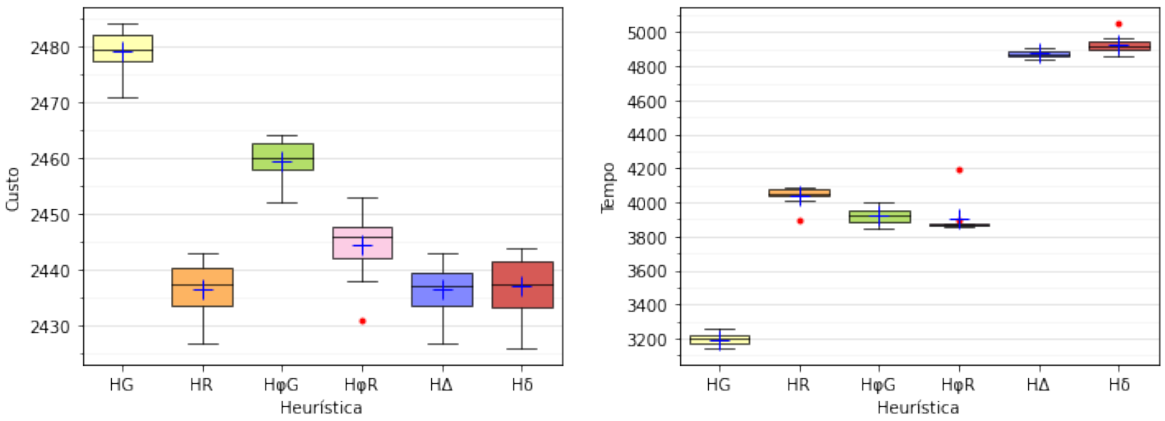
Figura 4.4: Comparação dos resultados de custo e tempo das heurísticas para grafos de cinco mil vértices com 25% de densidade.

As heurísticas HR e $H\delta$ obtiveram os melhores desempenhos no quesito custo. Já a heurística HG apresenta o menor tempo de execução dentre todas as heurísticas. Também fica evidente a influência das densidades no desempenho de tempo das heurísticas $H\varphi_G$ e $H\varphi_R$; à medida que os grafos ficam mais densos, os tempos de execução se aproximam dos tempos de HR.



(a) (b)

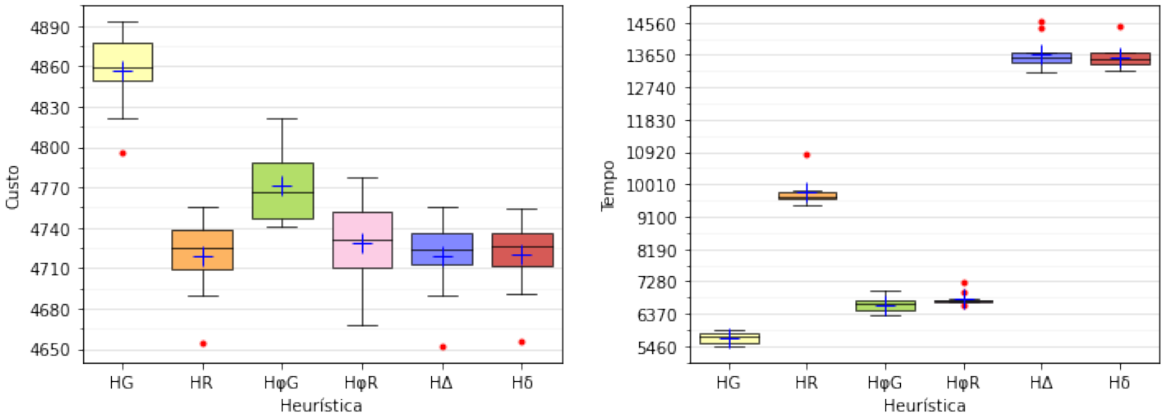
Figura 4.5: Comparação dos resultados de custo e tempo das heurísticas para grafos de cinco mil vértices com 50% de densidade.



(a) (b)

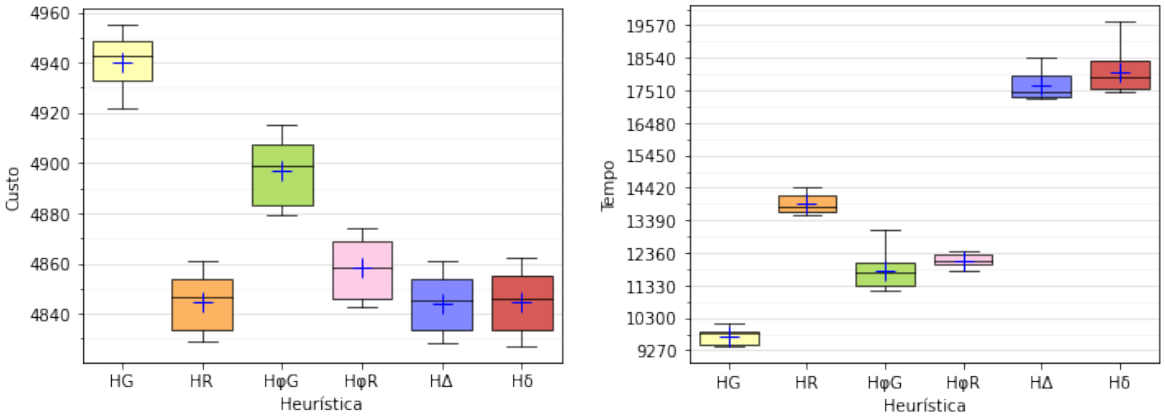
Figura 4.6: Comparação dos resultados de custo e tempo das heurísticas para grafos de cinco mil vértices com 75% de densidade.

Os resultados dos testes para as instâncias de grafos com dez mil vértices, apresentados nas Figuras 4.7, 4.8 e 4.9, mantém o padrão observado nos casos anteriores, inclusive na análise da influência da densidade sobre o tempo observada nas instâncias de cinco mil vértices.



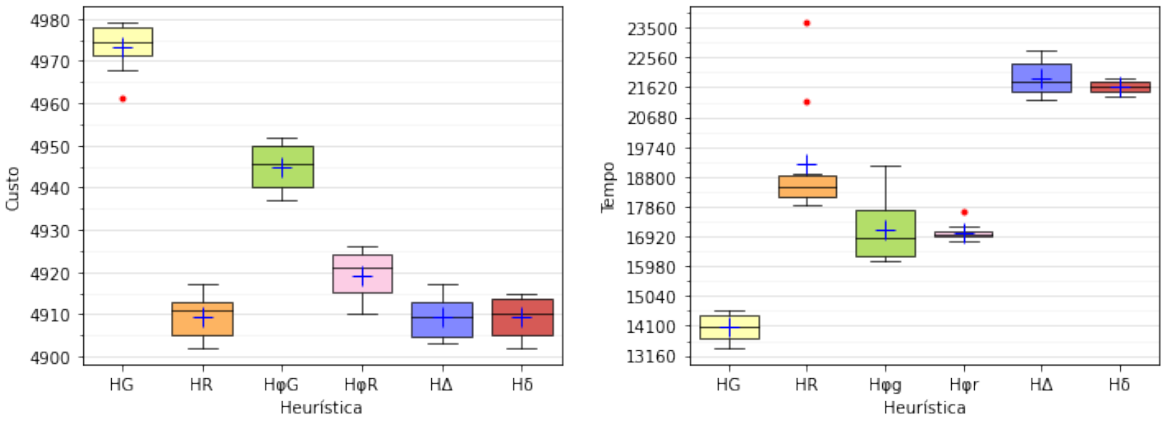
(a) (b)

Figura 4.7: Comparação dos resultados de custo e tempo das heurísticas para grafos de dez mil vértices com 25% de densidade.



(a) (b)

Figura 4.8: Comparação dos resultados de custo e tempo das heurísticas para grafos de dez mil vértices com 50% de densidade.



(a) (b)

Figura 4.9: Comparação dos resultados de custo e tempo das heurísticas para grafos de dez mil vértices com 75% de densidade.

Capítulo 5

Conclusão

Este trabalho abordou o Problema da k-Dominação Mínima com o foco no desenvolvimento de heurísticas para grafos simples, ou seja, não orientados e sem peso em suas arestas.

Para o desenvolvimento das heurísticas, o grau e o requisito para a k-dominação dos vértices foram levados em consideração, sendo ambos características atreladas ao grafo e ao problema, respectivamente. As diferentes combinações dos fatores levaram à criação de seis heurísticas, como abordadas nos Capítulos 3 e 4.

Cada um dos fatores considerados teve um nível de influência sobre os resultados. Porém, como evidenciado ao longo do trabalho, o requisito para k-dominação do vértice se mostrou mais significativo, mesmo combinado a outros fatores, quando comparado ao grau. Além disso, quando o grau é usado em conjunto com o requisito, nota-se uma queda no desempenho em relação à heurística que faz uso apenas do requisito.

Os experimentos realizados evidenciaram alguns comportamentos contraintuitivos, principalmente nas heurísticas que incluíram um critério de desempate ao Algoritmo 3.2. Em determinados casos, utilizar o menor grau para escolher entre vértices com o mesmo valor de requisito se mostrou mais vantajoso que escolher o maior. Curiosamente, o desempenho geral (uma análise conjunta dos resultados) das heurísticas com desempate foi pior que o da heurística que fazia uso somente dos requisitos para a seleção dos vértices.

A partir dos resultados obtidos neste trabalho, podemos confirmar que a escolha de cada vértice importa para o resultado final e que o critério aplicado para a escolha de um vértice não será necessariamente o melhor critério sempre.

5.1 Trabalhos Futuros

O primeiro ponto que pode ser apontado como trabalho futuro é a complementação dos experimentos realizados, comparando também os tempos de execução das heurísticas e do método exato (CPLEX).

Outras possibilidades a serem consideradas são a melhoraria do desempenho das heurísticas apresentadas e o desenvolvimento de novos métodos que obtenham melhor resultado da relação entre grau e requisito. Dentre as possiveis melhorias destacam-se o uso de um *heap* para reduzir o custo das ordenações empregadas e a criação de um valor linear que combine graus e requisitos, atribuindo maior peso ao primeiro. A utilização de um único valor combinado evitaria o custo computacional associado à dupla ordenação executada nos algoritmos com critério de desempate.

Os algoritmos desenvolvidos nesta dissertação podem ser aplicados a outros problemas, como, por exemplo, o Problema da Seleção de Alvos (difere do problema apresentado ao permitir que vértices dominados colaborem na dominação de seus vizinhos).

Referências

[1] Mayra C. Albuquerque**. "Matheuristics for Variants of the Dominating Set Problem"**. Tese de Doutorado. Pontífica Universidade Católica do Rio de Janeiro, 2018.

[2] Christiane N. Campos**. "O problema da coloração total em classes de grafos"**. Tese de Doutorado. Universidade Estatual de Campinas, 2006.

[3] David Chalupa**. "An order-based algorithm for minimum dominating set with application in graph mining"**. Em: *Information Sciences* 426 (2018), pp. 101–116.

[4] Gennaro Cordasco, Luisa Gargano, Marco Mecchia, Adele A. Rescigno e Ugo Vaccaro**. "Discovering Small Target Sets in Social Networks: A Fast and Effective Algorithm"**. Em: *Algorithmica* 80.6 (jun. de 2018), pp. 1804–1833. ISSN: 0178-4617. DOI: 10.1007/s00453-017-0390-5.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest e Clifford Stein**. *Introduction to Algorithms***. 3ª edição. Mit Press, 2009.

[6] Jeff Erickson**. *Algorithms***. Independently published, 2019, pp. 1–472. URL: https://jeffe.cs.illinois.edu/teaching/algorithms/book/Algorithms-JeffE.pdf.

[7] Michael R. Garey e David S. Johnson**. *Computers and Intractability: A Guide to the Theory of NP-Completeness***. Worth Publishers, 2011.

[8] Google Cloud Platform – Compute Engine. 2008. URL: https://cloud.google.com/compute/?hl=en-us.

[9] IBM**. *IBM ILOG CPLEX Optimization Studio***. 2009. URL: %5Curl%7Bhttps://www.ibm.com/products/ilog-cplex-optimization-studio%7D (acesso em 10/06/2022).

[10] Rodrigo Mafort**. *BibGrafos***. Jun. de 2020. URL: https://github.com/rodrigomafort/BibGrafos (acesso em 29/06/2022).

[11] Rodrigo Lamblet Mafort**. "Propagação em Redes: Dominação Vetorial e Seleção de Alvos"**. Tese de Doutorado. Universidade Federal Fluminense, 2020, p. 205. URL: http://www.ic.uff.br/PosGraduacao/frontend-tesesdissertacoes/download.php?id=997.pdf&tipo=trabalho.

[12] Lilian Markezon e Oswaldo Vernet**.** ***Representações Computacionais de Grafos***. 1ª edição. Vol. 24. Notas em Matemática Aplicada. SBMAC, 2012.

[13] Rommel T. Oliveira**. "Sobre Conjuntos Dominantes Eficientes em Grafos"**. Dissertação de Mestrado. Universidade Federal de Goiás, 2009.

[14] Jayme L. Szwarcfiter**.** ***Teoria Computacional de Grafos***. 1ª edição. Vol. Único. SBC. Elsevier, 2018.

APÊNDICE A – Resultados completos das heurísticas

As Tabelas A.1, A.2 e A.3 apresentam, respectivamente, os resultados dos experimentos realizados com as heurísticas do Capítulo 3 em grafos com dois, cinco e dez mil vértices. As colunas Dens., ID e m indicam respectivamente a densidade aproximada, o identificador e o número de arestas de cada instância. As demais colunas apresentam o custo e o tempo de execução em milissegundos de cada algoritmo. Os valores em negrito apresentam os melhores resultados encontrados para cada instância.

				HG		HR		$H\varphi_G$		$H\varphi_R$		$H\Delta$		$H\delta$	
Dens.	ID	m	k	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
25%	1	498921	219	950	**186**	907	310	933	229	915	234	**906**	424	**906**	425
	2	498995	223	960	**190**	918	318	945	233	929	246	921	438	**917**	448
	3	500978	218	941	**200**	**896**	306	926	229	912	232	898	422	900	424
	4	499663	211	915	**181**	873	300	903	221	882	214	**868**	411	873	416
	5	500265	217	942	**184**	898	309	925	226	908	219	**894**	424	897	427
	6	499813	216	931	**187**	893	307	922	227	905	219	**891**	422	**891**	432
	7	500005	223	960	**191**	**919**	314	948	235	933	225	**919**	430	921	436
	8	500166	219	943	**189**	905	310	933	229	912	219	**903**	423	906	451
	9	499989	217	939	**189**	898	308	924	227	907	219	**896**	417	**896**	425
	10	499411	219	950	**191**	906	306	936	230	910	236	**904**	443	**904**	437
50%	1	999699	458	962	**328**	931	464	948	409	938	408	**929**	589	931	593
	2	999662	467	976	**334**	950	471	963	413	956	417	**949**	594	952	600
	3	1000026	453	953	**322**	922	453	939	403	929	396	**921**	574	922	575
	4	1000086	458	968	**328**	932	462	948	408	940	403	**929**	584	931	590
	5	999840	463	976	**333**	**938**	462	959	413	954	405	942	608	940	592
	6	999950	458	965	**332**	**929**	455	950	407	937	399	932	579	931	580
	7	998796	452	954	**329**	920	463	938	405	925	397	**918**	583	919	579
	8	999485	457	962	**331**	**930**	452	946	407	939	543	931	577	**930**	581
	9	998673	461	971	**333**	941	460	958	412	947	403	**940**	585	**940**	592
	10	1000195	460	970	**337**	936	447	953	409	941	405	936	582	**935**	589
75%	1	1499752	709	995	**484**	**970**	610	982	593	974	595	971	751	**970**	741
	2	1499811	720	991	**470**	**967**	611	979	597	972	592	**967**	758	968	738
	3	1498652	715	987	**471**	**962**	604	974	589	968	585	**962**	750	963	734
	4	1499578	708	977	**468**	953	604	964	583	958	587	**952**	744	953	732
	5	1498718	717	985	**469**	**964**	614	978	591	970	586	**964**	756	**964**	742
	6	1499397	720	989	**481**	969	610	980	598	974	591	**967**	744	968	740
	7	1499252	713	982	**472**	959	605	972	588	964	584	959	763	**958**	738
	8	1500148	719	987	**474**	967	609	980	591	971	487	966	742	**965**	740
	9	1499647	722	994	**478**	971	636	982	593	975	592	971	752	**970**	744
	10	1499258	719	990	**472**	**967**	606	981	597	972	592	**967**	749	968	741

Tabela A.1: Resultados das heurísticas para instâncias com dois mil vértices

				HG		HR		$H\varphi_G$		$H\varphi_R$		$H\Delta$		$H\delta$	
Dens.	ID	m	k	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
25%	1	3124077	574	2420	**1278**	2339	2117	2398	1555	2351	1496	2339	2980	**2337**	2934
	2	3126980	562	2375	**1262**	2290	2093	2347	1487	2303	1484	**2287**	2952	**2287**	2909
	3	3126050	569	2399	**1266**	**2315**	2114	2368	1550	2332	1487	2320	2941	2320	2925
	4	3124550	571	2412	**1278**	2327	2127	2382	1515	2342	1440	**2324**	2991	**2324**	2907
	5	3124364	570	2409	**1291**	**2319**	2137	2380	1534	2337	1497	2326	2968	2322	2945
	6	3121850	569	2413	**1283**	**2319**	2103	2375	1537	2330	1505	2323	3020	2321	2982
	7	3125548	569	2399	**1279**	2322	2108	2376	1536	2337	1509	**2319**	3040	2320	3013
	8	3127071	562	2376	**1272**	2290	2071	2351	1516	2305	1471	**2287**	2930	2290	2938
	9	3122715	575	2424	**1308**	2343	2178	2401	1565	2357	1518	2342	2938	**2341**	2919
	10	3125867	568	2396	**1272**	2317	2142	2367	1536	2330	1514	2415	3054	**2313**	3069
50%	1	6250580	1188	2460	**2234**	**2399**	3115	2433	2723	2407	2689	2400	3971	**2399**	3947
	2	6245959	1179	2445	**2266**	**2381**	3111	2417	2668	2390	2679	2382	3900	**2381**	3898
	3	6248861	1180	2443	**2203**	2385	3070	2416	2707	2392	2682	2384	3920	**2383**	3877
	4	6248820	1186	2457	**2259**	2396	3152	2427	2777	2401	2689	**2394**	3991	**2394**	3968
	5	6248910	1185	2458	**2229**	2395	3113	2428	2714	2404	2675	**2394**	3926	**2394**	3943
	6	6251982	1188	2459	**2258**	**2396**	3132	2427	2758	2410	2690	2398	3902	2397	3926
	7	6251504	1181	2446	**2250**	**2382**	3072	2419	2761	2392	2697	2386	3936	2384	3886
	8	6252116	1188	2454	**2295**	2397	3096	2430	2765	2408	2702	2400	3955	**2396**	3938
	9	6251411	1185	2460	**2267**	2392	3107	2426	2786	2401	2687	2394	3912	**2390**	3883
	10	6249366	1170	2421	**2238**	2363	2972	2396	2696	2371	2659	**2362**	3863	2364	3870
75%	1	9374031	1819	2482	**3158**	2439	4037	2461	3872	2447	3876	**2437**	4873	2438	4904
	2	9372032	1815	2477	**3197**	2433	4040	2459	3945	2441	3893	2433	4862	**2432**	4929
	3	9375228	1823	2482	**3179**	**2443**	4063	2463	3954	2450	3866	**2443**	4906	2444	4946
	4	9373775	1822	2482	**3165**	**2442**	4043	2462	3878	2453	3874	2443	4865	**2442**	4915
	5	9372288	1810	2471	**3144**	2427	4011	2452	3850	2431	3852	2427	4842	**2426**	4856
	6	9373872	1818	2480	**3254**	**2435**	4079	2458	3912	2445	4194	2437	4862	2437	4892
	7	9373097	1817	2478	**3212**	2436	4082	2458	4000	2445	3868	**2435**	4893	2437	4911
	8	9375392	1815	2475	**3217**	**2431**	4089	2453	3987	2438	3854	**2431**	4912	2432	4885
	9	9374159	1820	2479	**3217**	2439	4053	2464	3932	2448	3867	**2438**	4869	2440	5050
	10	9376631	1844	2484	**3201**	2441	3890	2463	3922	2447	3868	**2440**	4888	2444	4968

Tabela A.2: Resultados das heurísticas para instâncias com cinco mil vértices

				HG		HR		$H\varphi_G$		$H\varphi_R$		$H\Delta$		$H\delta$	
Dens.	ID	m	k	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo	Custo	Tempo
25%	1	12496864	1147	4796	**5489**	4654	9406	4741	6670	4668	6607	**4652**	13153	4655	13194
	2	12502761	1158	4822	**5461**	**4690**	9619	4782	6781	4709	6687	**4690**	13436	4691	13544
	3	12498801	1170	4878	**5665**	4739	9634	4821	7017	4756	6760	**4736**	13376	4737	13410
	4	12505710	1163	4848	**5545**	**4707**	9584	4791	6749	4727	6775	4711	13443	4710	13401
	5	12500989	1166	4861	**5633**	4724	10877	4806	6777	4738	6758	**4721**	14396	4726	14473
	6	12501761	1165	4854	**5813**	**4716**	9835	4741	6347	4778	6762	4719	14580	4719	13392
	7	12497171	1170	4878	**5872**	4739	9786	4758	6317	**4716**	6727	4736	13730	4735	13722
	8	12496662	1170	4881	**5814**	**4740**	9614	4761	6431	4768	7235	**4740**	13492	4742	13617
	9	12499199	1166	4858	**5800**	**4725**	9672	4744	6617	4736	6708	4726	13665	4727	13497
	10	12504576	1174	4893	**5938**	4756	9755	4773	6546	**4687**	6996	4755	13648	4754	13735
50%	1	25002597	2411	4948	**9734**	4853	13567	4906	12347	4866	12196	**4852**	17368	**4852**	17484
	2	24993397	2406	4938	**9401**	4848	14425	4902	11884	4857	12334	**4846**	18068	4848	19682
	3	24995390	2411	4949	**9431**	**4854**	13594	4908	12115	4872	12165	**4854**	17250	4856	17461
	4	25002391	2398	4922	**9472**	4830	14271	4879	11904	4845	12345	4828	18233	**4827**	18531
	5	24992319	2401	4932	**9434**	4835	13957	4889	13071	4845	12058	**4834**	17329	**4834**	18235
	6	24999275	2414	4950	**10108**	4861	14232	4913	11355	4874	12417	**4859**	18551	4862	18526
	7	24990168	2398	4935	**9845**	**4829**	13676	4880	11172	4843	11773	4831	17275	4831	17651
	8	25001181	2406	4948	**9843**	4845	13664	4896	11151	4860	11998	4845	17601	**4844**	17809
	9	25003489	2401	4924	**10122**	**4833**	13742	4881	11584	4849	11784	**4833**	17318	**4833**	17525
	10	24994776	2414	4955	**9847**	4861	13892	4915	11300	4870	12073	4861	17659	**4860**	18013
75%	1	37500164	3670	4977	**14564**	4912	18693	4946	17818	4921	16918	4911	22072	**4910**	21605
	2	37493443	3666	4972	**13489**	4906	17923	4940	17727	4915	16751	4907	22773	**4905**	21356
	3	37499238	3672	4979	**14108**	**4912**	18824	4947	17748	4924	17246	4913	22484	4915	21896
	4	37497757	3669	4971	**13404**	4910	17984	4945	17309	4921	17092	**4908**	21210	4910	21319
	5	37498420	3672	4978	**13566**	4914	18146	4951	19153	4924	16894	**4912**	21733	4913	21424
	6	37497124	3673	4979	**14050**	**4913**	18257	4951	16156	4924	17088	4914	21512	4914	21626
	7	37498731	3663	4968	**14579**	**4902**	18885	4937	16228	4910	16964	4903	21257	4903	21610
	8	37493723	3664	4961	**14122**	4903	23651	4940	16264	4910	16859	4904	22776	**4902**	21665
	9	37499091	3674	4977	**14601**	4917	21160	4952	16461	4926	17697	4917	21498	**4915**	21921
	10	37494451	3665	4972	**14086**	4905	18316	4938	16347	4916	17031	**4904**	21886	4906	21868

Tabela A.3: Resultados das heurísticas para instâncias com dez mil vértices