Abordagens Algorítmicas ao Problema do Conjunto Dominante de Peso Mínimo: Estratégias de Procura Exaustiva e Heurística

Eduardo Carvalho Nº113816

Resumo **–Este relatório apresenta duas estratégias algorítmicas distintas para o problema do Conjunto Dominante de Peso Mínimo *(Minimum Weight Dominating Set)*, em outras palavras, determinar um subconjunto de vértices com peso total mínimo, assegurando que cada vértice fora desse subconjunto seja adjacente a, pelo menos, um vértice pertencente a ele. [1] A primeira estratégia trata-se de um algoritmo de procura exaustiva, e a segunda, de uma heurística voraz. É apresentado também uma análise formal e empírica da complexidade computacional do problema, assim como sua comparação. Com os resultados da análise, são apresentadas também algumas estimativas para a duração deste problema para tamanhos superiores. Por fim, são comparados os resultados obtidos, contrastando o desempenho e a qualidade da solução de cada algoritmo.**

Palavras chave **–Grafos, peso mínimo, procura exaustiva, heurística, complexidade**

I. Introdução ao problema

O projeto realizado tem como objetivo a implementação de duas estratégias algorítmicas, uma procura exaustiva e uma heurística voraz/gulosa para a determinação de um conjunto dominante $D \subseteq V$ de peso mínimo num grafo $G(V, E)$ com $n = |V|$ vértices, onde cada vértice $v \epsilon V$ possui um peso $w(v) > 0$. Antes de avançar para a descrição das estratégias utilizadas é importante explicar de que se trata este dito conjunto. Um conjunto dominante é um subconjunto de vértices D tal que todo o vértice v que não pertence a D é adjacente a, pelo menos, um vértice em D. O peso total deste conjunto é a soma dos pesos de todos os vértices que o compõem. O objetivo do problema é, portanto, encontrar o conjunto dominante cujo peso total seja o mais pequeno possível.

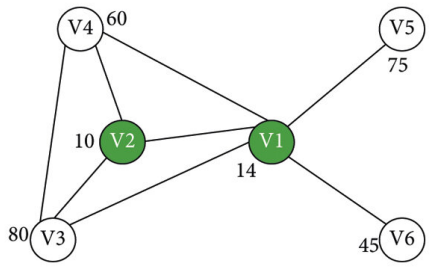

Fig. 1 - Exemplo - Conjunto Dominante de Peso Mínimo, peso total de 10+14=24

O problema em questão destaca-se entre uma vasta gama de problemas *NP-Hard* da teoria de grafos. Esta classificação significa que o problema pertence a uma classe de desafios computacionais para os quais não se conhece nenhuma solução eficiente (em tempo polinomial). Na prática, isto implica que o tempo necessário para encontrar a solução ótima garantida cresce de forma exponencial com o número de vértices (n). Esta intratabilidade computacional é precisamente o que justifica a análise de duas abordagens distintas neste relatório, uma procura exaustiva (como a de 2^n), que se torna impraticável muito rapidamente, e uma heurística, que abdica da garantia de optimalidade em troca de um tempo de execução razoável.

II. Descrição dos Algoritmos

A. Procura Exaustiva

O primeiro algoritmo aplicado ao problema em questão é o de procura exaustiva. Neste método, todos os 2^n subconjuntos de vértices possíveis são verificados iterativamente. A verificação consiste em confirmar se se satisfaz a condição de conjunto dominante. Uma vez satisfeita, o seu peso total do subconjunto é calculado e comparado com o mínimo registado.A análise prossegue até que todos os 2^n subconjuntos tenham sido testados para garantir a optimalidade.

B. Heurística

O segundo algoritmo implementado é uma heurística voraz. Neste método, é criado um conjunto de "vértices por cobrir", que inicialmente contém todos os vértices do grafo. Em seguida, o algoritmo escolhe gulosamente o vértice (ainda não escolhido) com o melhor *score*. Esse *score* é simplesmente o número de nós ainda por cobrir que ele domina, a dividir pelo seu peso. Este vértice é adicionado ao conjunto dominante, e todos os nós que ele passa a dominar (ele próprio e os seus vizinhos) são removidos do conjunto "por cobrir". O algoritmo termina quando este conjunto de vértices "por cobrir"fica vazio.

Fica evidente que o algoritmo não busca a solução óptima. A designação *greedy heuristic* decorre do fato de considerar os vértices com o *score* mais alto os melhores candidatos para a solução final.

III. Análise Formal de Complexidade

A. Procura Exaustiva

Conforme mencionado acima, para cada um dos 2^n subconjuntos, é necessário verificar se cumpre a

condição de conjunto dominante. Esta verificação, no pior caso, tem um custo computacional de $O(n+m)$, pois implica percorrer os nós do subconjunto e todas as suas respetivas ligações.

$$T(n,m) = \sum_{k=0}^{n} \binom{n}{k} \cdot T_{\text{verify}}(k)$$

A expressão $T(n,m)$ formaliza matemáticamente o custo computacional total do algoritmo de procura exaustiva da seguinte forma:

- O somatório $\sum_{k=0}^{n}$ representa a iteração sobre todos os tamanhos k de subconjuntos possíveis, desde o tamanho $k=0$ (o conjunto vazio) até ao tamanho $k=n$ (o conjunto com todos os vértices).
- O termo $\binom{n}{k}$ (coeficiente binomial) quantifica o número exato de subconjuntos distintos que existem para um determinado tamanho k.
- O termo $T_{\text{verify}}(k)$ representa o custo computacional de processar um único subconjunto de tamanho k. Este é o custo da função de verificação (`subset_verifier(graph, subset)`) necessária para determinar se esse subconjunto específico é dominante.

Sabendo que, no nosso caso, $T_{\text{verify}}(k)$ tem custo $O(n+m)$, podemos simplificar passo a passo:

1. Simplificação (Pior Caso): Substituímos $T_{\text{verify}}(k)$ pelo seu limite superior $O(n+m)$:

$$T(n,m) \leq \sum_{k=0}^{n} \binom{n}{k} \cdot O(n+m)$$

2. Isolar o termo constante: Como $O(n+m)$ não depende de k, pode ser retirado da soma:

$$T(n,m) \leq O(n+m) \cdot \left(\sum_{k=0}^{n} \binom{n}{k} \right)$$

3. Simplificação final (Teorema Binomial): Pelo Teorema Binomial, sabemos que:

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n$$

4. Resultado: Substituindo na expressão anterior, obtemos:

$$T(n,m) = O\big((n+m) \cdot 2^n\big)$$

Conclui-se, portanto, que o algoritmo de procura exaustiva possui uma complexidade temporal exponencial, tornando-o computacionalmente impraticável para *inputs* de de maior dimensão.

B. Heurística Voraz

A complexidade da heurística voraz/gulosa é polinomial, sendo determinada pelo seu loop principal. Este loop (`(while nodes_to_cover:)`) executa-se k vezes, onde k é o número de vértices no conjunto dominante final $k \leq n$. Dentro de cada uma dessas k iterações, o algoritmo realiza a sua operação mais custosa, encontrar o vértice com o melhor rácio custo-benefício. Para o fazer, é necessário percorrer todos os $O(n)$ candidatos e, para cada um, analisar os seus vizinhos e executar operações de conjuntos. O custo total de encontrar o "melhor"nó uma vez (o "trabalho interno"de uma iteração) é $O(n+m)$, pois, no total, percorre todos os n nós e as m arestas do grafo. A complexidade total $T(n,m)$ é, portanto, o somatório deste custo, repetido k vezes.

Com isso dito, podemos simplificar a formalização da complexidade computacional do algoritmo implementado como:

1. Simplificação geral:

$$O(k(n+m))$$

.

2. Simplificação (Pior caso): k aproxima-se de n, resultando numa complexidade final de

$$O(n(n+m))$$

ou seja,

$$O(n^2+nm)$$

Facilmente se conclui, a partir da análise formal realizada para as duas estratégias implementadas, que o ritmo de crescimento do algoritmo de procura exaustiva é muito superior ao da heurística implementada. No entanto, esses resultados serão comprovados na secção seguinte.

IV. COMPARAÇÕES DOS RESULTADOS EXPERIMENTAIS COM A ANÁLISE FORMAL

Nesta secção, foram realizadas comparações entre os resultados obtidos experimentalmente e a análise formal conduzida na secção anterior. O processo envolveu a aplicação dos algoritmos (*procura exaustiva* e *heurística voraz*) em diversos grafos gerados especificamente para testar o problema do *Conjunto Dominante de Peso Mínimo*.

O número de arestas em cada grafo foi determinado com base no número total de vértices, n. Para cada n (de 4 a 25), foram gerados quatro grafos, cada um com um número de arestas exatamente igual a uma percentagem do máximo possível. O número de arestas m para cada $P \in \{12.5\%, 25\%, 50\%, 75\%\}$ foi calculado como:

$$m = \text{round}\left(P \times \binom{n}{2} \right)$$

sendo $\binom{n}{2}$ o número máximo de arestas possível num grafo não direcionado simples com n vértices.

É importante notar que este método (escolher m arestas aleatórias de uma lista baralhada) é diferente de um método probabilístico. Além disso, a cada vértice foi atribuído um peso aleatório entre 1 e 100, conforme requerido pelo problema.

Na análise da complexidade computacional, foram calculados três valores: o tempo de execução do algoritmo para as diferentes instâncias, a contagem de operações básicas e o número de configurações testadas. Destaca-se que as operações básicas são aquelas que podem ter maior influência na globalidade do algoritmo.

É importante ainda ressaltar, no entanto, que nem todos os resultados são comentados. Concluímos com exemplos representativos que podem ser generalizados para os demais casos. De todo o modo, todos os gráficos que não foram alvos de análise estão representados no Apêndice.

A. Procura Exaustiva

No que diz respeito ao algoritmo de procura exaustiva, a primeira análise realizada focou na evolução do tempo de execução em função do tamanho do problema (o número de vértices, n). Para isolar o impacto de n, os tempos de execução do algoritmo foram comparados para instâncias de grafos com uma percentagem fixa de arestas, $P = 0.5(50\%)$. Em outras palavras, foi analisado como o tempo de execução escala à medida que n aumenta, mantendo a densidade do grafo constante. Os resultados correspondentes estão representados na Figura 2

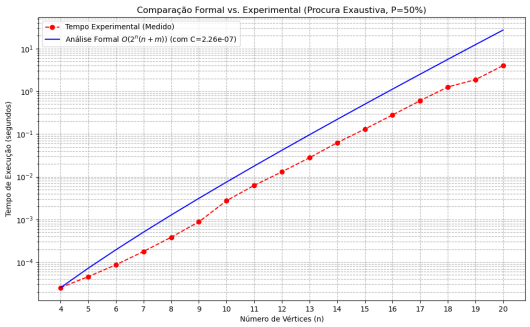

Fig. 2 - Comparação entre o tempo de execução experimental medido da procura exaustiva e a sua complexidade teórica formal $O(2^n(n+m))$, para grafos com $P = 0.5$ e com o eixo Y em escala logarítmica

1. Descrição do Gráfico:
   A Figura 2 apresenta a comparação direta entre os resultados experimentais (tempo de execução real medido) e a análise de complexidade formal (fórmula teórica) para o algoritmo de procura exaustiva. O gráfico representa o tempo de execução em segundos (eixo Y) em função do número de vértices n (eixo X), para uma densidade de arestas fixa de $P = 0.5$. A linha vermelha tracejada corresponde ao tempo experimental medido, enquanto a linha azul representa a função de complexidade teórica $O(2^n(n+m))$, escalada por uma constante $C = 2.26 \times 10^{-7}$ para se ajustar à escala temporal em segundos.
2. Análise da Linha Experimental:
   O aspeto mais relevante deste gráfico é a utilização de uma escala logarítmica no eixo Y. Num gráfico log-linear (onde Y é logarítmico e X é linear), um crescimento exponencial manifesta-se como uma linha reta. Como se pode observar, os pontos experimentais formam uma linha quase perfeitamente reta. Isto comprova empiricamente a análise formal: o tempo de execução cresce exponencialmente com n, validando a complexidade $O(2^n)$ e confirmando que o fator n é o dominante absoluto do custo do algoritmo.
3. Comparação das Curvas (Teórica vs. Experimental):
   A comparação entre as duas curvas reforça esta conclusão. As linhas azul e vermelha são visivelmente paralelas, o que indica que ambas partilham a mesma taxa de crescimento (ou o mesmo declive no gráfico logarítmico). A linha azul representa o limite superior, o pior caso da análise formal, o que explica o facto de estar sistematicamente acima do tempo experimental, que representa o caso médio. O fator C é simplesmente a constante de proporcionalidade que converte o número abstrato de operações teóricas em tempo real (segundos) no máquina utilizada. Este paralelismo constitui a prova visual de que o modelo teórico descreve com precisão o comportamento prático do algoritmo.

A análise da Figura 3 permite avaliar o impacto de ambas as variáveis da complexidade, n e m. Por um lado, é evidente que o número de vértices n é o fator dominante, ditando o crescimento exponencial do algoritmo; isto é comprovado pelo facto de as quatro linhas serem quase retas e paralelas numa escala logarítmica. Por outro lado, o gráfico demonstra que o número variável de arestas m (correspondente às diferentes percentagens P) exerce uma influência clara, sistemática e mensurável no tempo de computação. Observa-se que, para qualquer valor fixo de n no eixo X, a linha correspondente a $P = 0.125$ (menos arestas) é consistentemente a mais rápida, enquanto a linha de $P = 0.75$ (mais arestas) é a mais lenta. Esta observação valida experimentalmente a componente $O(n+m)$ da análise formal, confirmando que o tempo de verificação aumenta com o número de arestas.

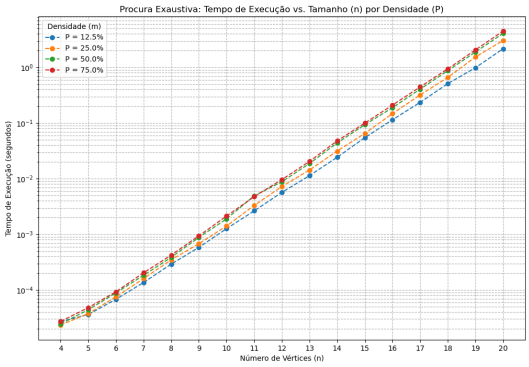

Fig. 3 - Tempo de execução em função do número de vértices para diferentes valores de P

Assim, é notável que o tempo de execução absoluto para os grafos mais densos, com $P = 0.75$, é consistentemente mais elevado (mais lento) do que o dos restantes. No entanto, como as linhas no gráfico logarítmico são paralelas, concluímos que a taxa de crescimento exponencial é a mesma para todas as densidades.

Dado que o tempo de execução não é o melhor indicador da complexidade do problema, realizou-se também uma contagem do número de operações básicas. No nosso caso, este valor é definido pelo custo de verificar se um subconjunto é dominante, o que, no pior caso, implica iterar sobre todos os n nós e todas as m arestas $O(n+m)$ para garantir que todos os vértices estão cobertos. Da mesma forma que se observou previamente a dependência do tempo de execução em relação ao número de arestas, observa-se que esta é mais facilmente notável contabilizando o número de operações.

A Figura 4 apresenta os resultados desta contagem, representando o número de operações em função do número de vértices n. O eixo Y encontra-se em escala logarítmica para permitir uma visualização mais clara do crescimento exponencial.

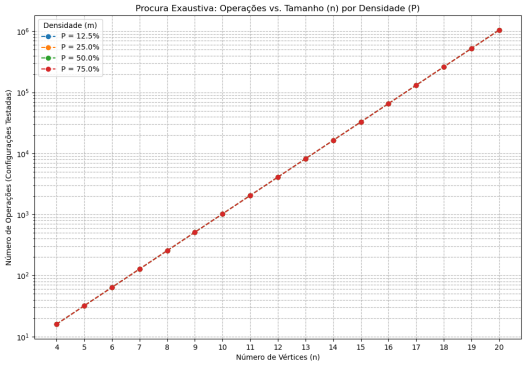

Fig. 4 - Número de operações básicas (configurações testadas) em função de n, para a procura exaustiva em diferentes densidades P, com o eixo Y em escala logarítmica

A observação mais imediata e fundamental deste gráfico é que as quatro linhas, correspondentes às diferentes densidades ($P = 0.125$ a $P = 0.75$), estão perfeitamente sobrepostas, formando uma única linha reta. Isto comprova experimentalmente que o número de configurações testadas, a operação básica dominante do algoritmo é totalmente independente da densidade de arestas m.

Esta análise confirma que o custo total $O(2^n(n+m))$ é composto por duas componentes principais:

1. O número de subconjuntos a testar, $O(2^n)$, que depende apenas de n e é idêntico para todas as densidades (como demonstrado pelas linhas sobrepostas).
2. O custo de verificação de um subconjunto, $O(n+m)$, que depende da densidade m e que, como observado no gráfico de tempo, faz com que os grafos mais densos apresentem tempos de execução ligeiramente superiores.

Este gráfico, portanto, isola e valida o fator $O(2^n)$ como o verdadeiro motor da complexidade exponencial do algoritmo.

Outra análise possível, considerando o número de operações básicas, é a forte relação entre essas operações e o tempo de execução do algoritmo. Esse fenómeno pode ser facilmente observado ao analisar o gráfico da Figura 5.

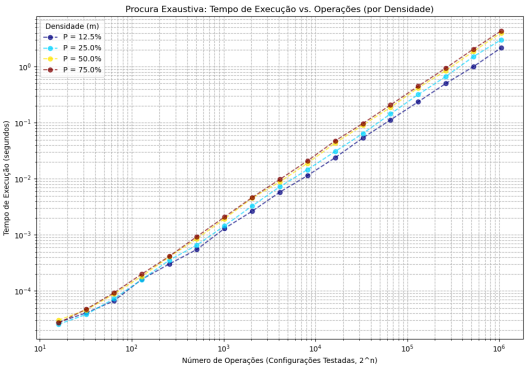

Fig. 5 - Tempo de execução em função do número de operações para as diferentes densidades

Ao observar a Figura 5, retiramos duas conclusões fundamentais que validam perfeitamente a análise de complexidade formal $O(2^n(n+m))$:

1. Proporcionalidade Exponencial:
   O facto de todas as linhas serem retas (e paralelas) num gráfico com ambos os eixos em escala logarítmica prova que o Tempo de Execução (Y) é diretamente proporcional ao Número de Operações 2^n (X). O "motor" exponencial $O(2^n)$ é o fator dominante.
2. Impacto da Densidade (m):
   As linhas estão claramente separadas verticalmente. Para um mesmo número de operações (qualquer ponto no eixo X, que corresponde a um n fixo), o tempo de execução é sistematicamente maior para densidades P mais altas (a linha $P = 0.75$ está no topo) e menor para densidades baixas (a linha $P = 0.125$ está em baixo).

Isto comprova experimentalmente que o custo total é o produto do fator $O(2^n)$ (o eixo X) e de um custo por operação $O(n+m)$ (a separação vertical).

B. Heurística

A segunda abordagem implementada é a heurística voraz, que serve como uma alternativa rápida à procura exaustiva. Este algoritmo abdica da garantia de optimalidade em troca de uma complexidade polinomial $O(n(n+m))$, tornando-o viável para todos os tamanhos de n. A análise experimental seguinte foca-se, portanto, em dois aspetos-chave. Quantificar o seu tempo de execução (que se espera ser insignificante) e avaliar a "qualidade" ou precisão da sua solução, medindo o quão

próximo o `greedy_weight` está do `optimal_weight` encontrado pela procura exaustiva.

Na Figura 6 é possível observar o crescimento do tempo de execução em função do tamanho do problema.

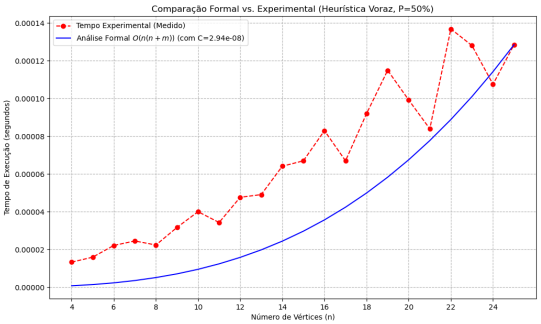

Fig. 6 - Comparação do tempo experimental da heurística com a sua complexidade teórica $O(n(n+m))$ para $P = 0.50$

O gráfico da Figura 6 compara o tempo de execução experimental da heurística voraz com a sua complexidade teórica $O(n(n+m))$, para uma densidade de $P = 0.50$. O eixo Y está em escala linear, e é importante notar que os tempos de execução medidos são extremamente baixos, na ordem dos microssegundos (o máximo é 0.00014 segundos). As flutuações e picos abruptos na linha experimental, como os visíveis em $n = 21$ e $n = 24$, foram particularmente difíceis de explicar. Contudo após alguma revisão de literatura, penso poderem ser explicados pelo facto de estarmos a medir tempos muito pequenos, na ordem dos micro segundos. O "ruído"do sistema operativo e de outros processos não planeados na máquina causam variações que são, por vezes, maiores do que o tempo real que o algoritmo demora a executar, resultando nesta aparência "saltitante". [2] [3] Apesar desta variabilidade, a conclusão principal é que a tendência geral da linha experimental a vermelho segue de perto a curva polinomial da análise formal a azul. O gráfico demonstra que o algoritmo é extraordinariamente rápido e que o seu crescimento não é exponencial, validando a análise de complexidade $O(n(n+m))$.

Nos seguintes dois gráficos (Figuras 7 e 8) observa-se uma análise muito idêntica à feita anteriormente para o algoritmo de procura exaustiva.

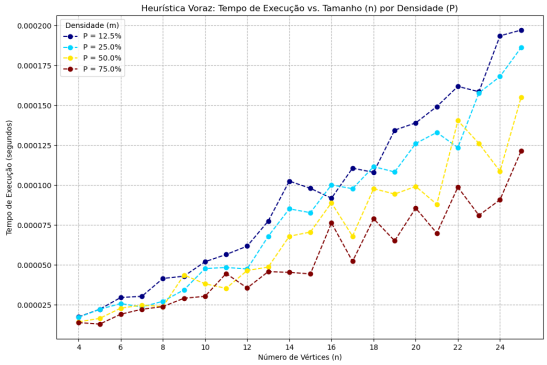

Fig. 7 - Tempo de execução em função do número de vértices para diferentes valores de P

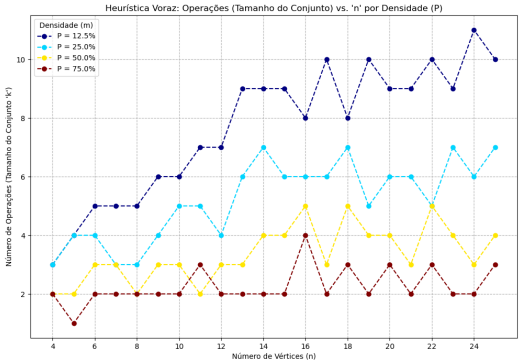

Fig. 8 - Número de operações em função do número de vértices para diferentes valores de P

Novamente, verifica-se uma clara dependência do número de arestas (m) e do número de iterações (k) no tempo de execução, como previsto pela análise formal $O(k(n+m))$. A discrepância mais notável apresenta-se entre os resultados para diferentes valores de P (densidade). Ao contrário do que se poderia intuitivamente esperar, os grafos menos densos ($P = 0.125$) levaram mais tempo a ser analisados, enquanto os grafos mais densos ($P = 0.75$) foram os mais rápidos. A justificação para este fenómeno encontra-se na Figura **??**. Esta figura mostra o número de *operações* (isto é, o tamanho k do conjunto dominante final) em função de n. Em grafos mais densos ($P = 0.75$), um único vértice apresenta um *score* de benefício/custo muito elevado, conseguindo dominar quase todo o grafo. Isto faz com que o algoritmo termine em muito poucas iterações ($k \approx 2$–3). Em grafos esparsos ($P = 0.125$), são necessários muitos vértices (k crescente) para cobrir todos os nós. Conclui-se que, embora o custo de uma iteração ($O(n+m)$) seja maior em grafos densos, o número total de iterações k (o fator dominante) é drasticamente menor, resultando num tempo de execução total mais baixo.

V. ESTIMATIVA DA EVOLUÇÃO DO PROBLEMA

Uma vez feita a análise do ponto de vista da complexidade computacional, foram realizadas estimativas para o tempo de execução para os algoritmos em questão. De forma a realizar a mesma, foram escolhidos os resultados relativos aos grafos com $P = 0.50$. Deste modo, as previsões são apenas aplicáveis a instâncias do problema com a mesma configuração.

A. Procura Exaustiva

O procedimento efetuado para as estimativas realizadas para o algoritmo de procura exaustiva passou pelo ajuste duma função exponencial ao gráfico indicativo do tempo de execução em função do tamanho do problema. A equação resultante do ajuste realizado, isto é, com os parâmetros obtidos, é a seguinte:

$$t(n) \approx (1.29 \times 10^{-6}) \times e^{(0.749 \times n)} \tag{1}$$

Após obtida a equação de ajuste, foi realizada uma extrapolação para vários valores de n. Os tamanhos considerados foram $n \in \{40, 45, 50, 55\}$. Rapidamente se conclui, através da observação da tabela I, um ritmo de crescimento altíssimo. De referência basta fazer uma comparação da estimativa do tempo de execução previsto para um problema de tamanho 40, que está na ordem dos 10^7s e um de tamanho 50, na ordem dos 10^{10}s. Observa-se uma diferença gigante para um aumento de apenas 10 vértices. Considerando um tempo razoável para encontrar uma solução de 30 segundos, o grafo maior encontrado com a configuração descrita ($P = 0.50$) apresenta 22 vértices.

TABELA I
ESTIMATIVA DO TEMPO DE EXECUÇÃO (PROCURA EXAUSTIVA, $P = 0.50$) PARA INSTÂNCIAS DE PROBLEMA DE GRANDE DIMENSÃO, COM BASE NA EQUAÇÃO DE AJUSTE $t(n) = (1.29 \times 10^{-6}) \times e^{(0.749 \times n)}$.

Vértices (n)	Tempo Estimado (segundos)	Tempo Estimado (Dias/Anos)
40	1.31×10^7 s	151.4 dias
45	5.53×10^8 s	17.5 anos
50	2.33×10^{10} s	740.3 anos
55	9.86×10^{11} s	$\approx$ 31275 anos

B. Heurística

Seguidamente, a mesma abordagem foi tida para a heurística desenvolvida. Neste caso foi feito um ajuste linear aos pontos do gráfico do tempo de execução em função do tamanho do problema para a configuração mencionada inicialmente. Da mesma forma, obteve-se a seguinte equação:

$$t(n) \approx (3.54 \times 10^{-8}) \times n(n + m) \tag{2}$$

1. Observação Geral: Novamente, verifica-se uma clara dependência do número de arestas (m) e do número de iterações (k) no tempo de execução, como previsto pela análise formal $O(k(n+m))$. A discrepância mais notável apresenta-se entre os resultados para diferentes valores de P (densidade).
2. Análise do Fenómeno: Ao contrário do que se poderia intuitivamente esperar, os grafos menos densos ($P = 0.125$) levaram mais tempo a ser analisados, enquanto os grafos mais densos ($P = 0.75$) foram os mais rápidos.
   A justificação para este fenómeno encontra-se na Figura **??**. Esta figura mostra o número de *operações* (isto é, o tamanho k do conjunto dominante final) em função de n. Em grafos mais densos ($P = 0.75$), um único vértice apresenta um *score* de benefício/custo muito elevado, conseguindo dominar quase todo o grafo. Isto faz com que o algoritmo termine em muito poucas iterações ($k \approx 2$–3). Em grafos esparsos ($P = 0.125$), são necessários muitos vértices (k crescente) para cobrir todos os nós.
3. Conclusão Experimental
   Conclui-se que, embora o custo de uma iteração ($O(n+m)$) seja maior em grafos densos, o número total de iterações k (o fator dominante) é drasticamente menor, resultando num tempo de execução total mais baixo.

TABELA II
ESTIMATIVA DO TEMPO DE EXECUÇÃO (HEURÍSTICA VORAZ, $P = 0.50$) PARA INSTÂNCIAS DE PROBLEMA DE GRANDE DIMENSÃO, COM BASE NA EQUAÇÃO DE AJUSTE $t(n) \approx (3.54 \times 10^{-8}) \times n(n + m)$.

Vértices (n)	Tempo Estimado (segundos)	Tempo Estimado (Dias/Anos)
40	6.09×10^{-4} s	0.61 milissegundos
45	8.61×10^{-4} s	0.86 milissegundos
50	1.17×10^{-3} s	1.17 milissegundos
55	1.55×10^{-3} s	1.55 milissegundos

VI. COMPARAÇÃO DAS SOLUÇÕES OBTIDAS COM AS DIFERENTES ABORDAGENS

Após a análise computacional, realizou-se uma análise da qualidade da solução. Como esperado, a utilização de uma heurística voraz abdica da garantia de optimalidade. Embora a heurística encontre sempre um conjunto dominante, este não é, em geral, o de peso mínimo. Tendo em conta que a procura exaustiva encontra sempre a solução ótima (o verdadeiro peso mínimo), é possível fazer uma avaliação da qualidade da heurística implementada.

Nesta secção, avaliamos a precisão da heurística (definida como o rácio entre o peso da solução aproximada e o peso da solução ótima) através da comparação dos resultados (`greedy_weight vs. optimal_weight`) para os mesmos grafos.

TABELA III

Resumo da precisão média da heurística voraz por densidade P. O "Rácio"é (Peso Greedy / Peso Ótimo) e o "Overhead"é o aumento percentual médio no peso da solução.

Densidade (P)	Rácio de Precisão (Média)	Overhead (Média) %
12.5%	1.0304	3.04%
25.0%	1.1119	11.19%
50.0%	1.1637	16.37%
75.0%	1.0738	7.38%

TABELA IV

Comparação detalhada da precisão da heurística para as instâncias de $n = 20$, mostrando o peso ótimo (exaustiva) vs. o peso aproximado (greedy).

Densidade (P)	Peso Ótimo (Exaustiva)	Peso Aprox. (Greedy)	Rácio de Precisão
12.5%	271	271	1.0000
25.0%	161	175	1.0870
50.0%	57	69	1.2105
75.0%	25	25	1.0000

1. Análise Geral: As tabelas de resultados demonstram de forma clara o *trade-off* fundamental da heurística voraz: ela abdica da garantia de optimalidade em troca de uma velocidade de execução massivamente superior. A questão central é: "quão boa" é a solução que ela encontra?
2. Precisão Média: A Tabela III oferece uma visão geral. Em média, a heurística voraz é bastante eficaz, com o "overhead" (isto é, o peso extra pago em relação à solução ótima) a situar-se entre 3% e 16%. Curiosamente, a heurística não apresenta um comportamento linear; ela é mais precisa em grafos com densidades extremas (muito esparsos, com $P = 0.125$, ou muito densos, com $P = 0.75$). O seu pior desempenho médio ocorre nas densidades intermédias ($P = 0.25$ e $P = 0.50$), onde o problema é, aparentemente, mais complexo e propenso a "enganar" a lógica gulosa.
3. Estudo de Caso para $n = 20$: A Tabela IV valida perfeitamente esta observação. Para $n = 20$, a heurística voraz foi perfeita, encontrando a solução ótima (Rácio = 1.0000) nos casos de $P = 0.125$ e $P = 0.75$. No entanto, para as densidades intermédias, ela falhou: em $P = 0.25$, a solução foi 8.7% mais pesada (175 vs 161), e em $P = 0.50$, foi 21% mais pesada (69 vs 57).

VII. Conclusões

O projeto realizado permitiu um aprofundamento do conhecimento relativo ao Problema do Conjunto Dominante de Peso Mínimo (MWDS), confirmando experimentalmente a ineficiência computacional da procura exaustiva. A sua complexidade $O(2^n(n + m))$ foi validada pela análise gráfica, e a extrapolação (Tabela I) demonstrou que o tempo de execução se torna impraticável para valores de n superiores a 40 (na ordem de dias ou anos), com o limite prático no nosso sistema a situar-se em $n \approx 22$.

Em contraste, observou-se como a simples implementação de uma heurística voraz (baseada em custo-benefício), de complexidade polinomial $O(n^2 + nm)$, produz resultados em meros milissegundos. Mais importante ainda, a análise de precisão (Tabela III) demonstrou que esta heurística produz resultados "bastante"bons com um *overhead* de peso médio entre 3% e 16% em relação à solução ótima, provando ser um excelente compromisso entre eficiência e qualidade da solução.

Bibliografia

[1] E. Barrena, S. Bermudo, A.G. Hernández-Díaz, A.D. López-Sánchez, e J.A. Zamudio, "Finding the minimum k -weighted dominating sets using heuristic algorithms", vol. 228, pp. 485–497.

[2] Zahir Toufie e Boniface Kabaso, "Os noise mitigations for benchmarking web browser execution environment performance", vol. 27, no. 1, pp. 37.

[3] Carl Staelin, Larry Mcvoy, e Bitmover Inc, "Mhz: Anatomy of a micro-benchmark", 07 de 1998.