Algoritmos de Pesquisa Exaustiva & Greedy Heuristic para encontrar um conjunto dominante com k vértices de um grafo

Gonçalo Freitas
Mestrado em Engenharia Computacional

***Abstract* –Given a undirected graph G(V, E), use an exhaustive search and a greedy heuristic approach to find if the graph as a dominating set with k vertices. An analysis about the computational complexity, time execution and number of basic operations of the problem will also be made.**

***Resumo* –Dado um grafo não direcionado G(V, E), é usado um algoritmo de pesquisa exaustiva e um algoritmo heurístico para descobrir se o grafo têm um conjunto dominante com k vértices. Também será feita uma análise sobre a complexidade computacional, o tempo de execução e o número de operações básicas do problema.**

***Palavras chave* –Grafos, Conjunto Dominante, Complexidade Computacional, Pesquisa Exaustiva, Greedy Heuristic**

I. Introdução

A. Conjunto Dominante

Na teoria dos grafos, um conjunto dominante de um grafo G, com V vértices e E arestas, $G = (V, E)$, é um subconjunto D de V de tal forma que cada vértice que não está em D é adjacente a pelo menos um membro de D.

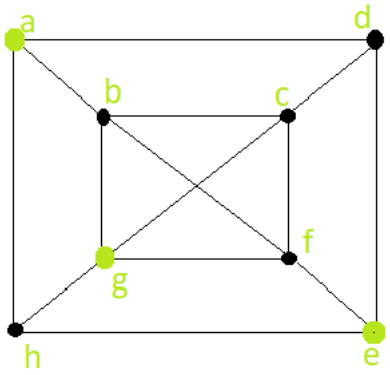

Fig. 1: Exemplo de um grafo com 8 vértices com conjunto dominante. [1]

Analisando o grafo da Fig.1 podemos afirmar que um conjunto dominante deste grafo poderá ser $D = \{a, g, e\}$.

Posto isto, o principal objetivo de ambos os algoritmos é encontrar, se possível, um conjunto dominante com $K \times V$ vértices, onde K corresponde a uma percentagem do número total de vértices, V. Para este problema iremos considerar $K \in [0.125, 0.25, 0.50, 0.75]$. Contudo, como o resultado de $K \times V$ podem não ser um valor inteiro irá ser usada a função $math.floor$ de forma a obter o número inteiro igual ou inferior ao resultado desta operação.

B. Pesquisa Exaustiva

A pesquisa exaustiva baseia-se numa abordagem de força bruta, geralmente aplicada a problemas combinatórios, cuja estratégia pode ser descrita pelos seguintes passos. [2]

1. Enumerar todos os candidatos para a solução
2. Verificar se satisfazem as condições do problema
3. Se necessário, escolher uma solução do conjunto de candidatos viáveis (que são solução)

Neste caso, apenas queremos saber se existe algum conjunto dominante de K vértices, não quantos existem. Uma melhor explicação do algoritmo encontra-se em II-A.

C. Greedy Heuristic

Um Greedy Heuristic algortihtm é uma abordagem para resolver um problema através da seleção da melhor opção disponível no momento, não se preocupando se esta seleção levará ao melhor resultado global. O algoritmo nunca inverte a decisão anterior, mesmo que a esta seleção esteja errada. Funciona numa abordagem 'top-down'. [3]

Este algoritmo pode não produzir o melhor resultado para todos os problemas mas, quando consegue, geralmente tem uma performance melhor. Outra vantagem destes algoritmos é a maior facilidade de descrever o algoritmo. Uma melhor explicação do algoritmo desenvolvido encontra-se em II-B.

II. Algoritmos

A. Pesquisa Exaustiva

No nosso caso, para implementar este algoritmo, começamos por gerar todas as combinações de vértices possíveis. Por exemplo, para $V = 3$, as combinações possíveis seriam:

$$[(0), (1), (2), (0, 1), (0, 2), (1, 2), (0, 1, 2)]$$

Contundo, assumindo que, por exemplo, $K = 0.50$, as

soluções possíveis seriam apenas:

$$[(0),(1),(2)]$$

No código, de maneira a tornar o algoritmo mais eficiente, esta lista com os conjuntos candidatos a solução não é guardada mas sim calculada a cada iteração pois, à medida que o tamanho do grafo aumenta o número de conjuntos aumenta, pelo que, guardar a lista tornaria-se muito penalizador em termos de memória e rapidez.
Após ter uma lista com os conjuntos que podem ser solução basta iterar e procurar se existe algum conjunto que verifica as condições de conjunto dominante. Caso não exista nenhum é retornado um conjunto vazio. Caso exista, o código retorna imediatamente esse conjunto e finaliza a execução.

B. Greedy Heuristic

Quanto ao Greedy Heuristic, este algoritmo pode ser realizado de várias maneiras, dependendo de como se decide abordar o problema. Após uma análise de vários grafos gerados pelo package NetworkX e fazendo uso da função dominating_set(G), foi possível observar que o vértice 0 faz parte de um dos conjuntos dominantes aproximadamente 90% das vezes. Posto isto, decidiu-se seguir uma abordagem em que, apenas se vai analisar os conjuntos em que esteja presente o vértice 0. Assim, analisando o exemplo demonstrado em II-A só existira um conjunto possível de ser solução, o conjunto (0).
Posto isto, este código é bastante semelhante aquele da pesquisa exaustiva apenas com a adição da condição de que o vértice 0 tem que fazer parte dos conjuntos a analisar, o que, por sua vez, diminui o número de conjuntos candidatos a solução e torna o algoritmo menos exigente em termos computacionais, como iremos ver mais a frente.

III. Análise

A package NetworkX permite-nos gerar um grafo aleatório com V vértices e atribuir a uma aresta uma certa probabilidade, P, de ela existir. Para este problema iremos considerar $V \in [4,5,6,...,25]$ e $P \in [0.125, 0.25, 0.50, 0.75]$. Os valores de K considerados serão aqueles já referidos na introdução.

A. Análise Formal

O problema apresentado retrata-se num tradicional problema resolvido por um método de pesquisa exaustiva de verificar, de todas as combinações de n vértices do grafo, com tamanho k, quais satisfazem a condição. O número de combinações de n vértices com tamanho k pode ser traduzido no seguinte somatório:

$$\sum_{k=1}^{\binom{n}{k}} 1$$

Dado uma combinação, de forma a verificar se se traduz num conjunto dominante, é necessário verificar todos os vértices que não estão na combinação e todos que se encontram na combinação. No total temos $n-k$ vértices que não pertencem à combinação, o que se traduz num somatório de $j=1$ até $n-k$. Por outro lado, temos k vértices que pertencem a combinação, o que leva a um somatório de $g=1$ até k. Assim sendo, esta verificação é traduzida em dois somatórios como os de seguida:

$$\sum_{j=1}^{n-k}\sum_{g=1}^{k} 1$$

Juntando os 3 somatórios apresentados temos:

$$\sum_{k=1}^{\binom{n}{k}}\sum_{j=1}^{n-k}\sum_{g=1}^{k} 1 = k(n-k)\binom{n}{k} = k(n-k)\frac{n!}{k!(n-k)!}$$

Simplificando vem,

$$k(n-k)\frac{n!}{k!(n-k)!} = n!\frac{k}{k(k-1)!}\frac{(n-k)}{(n-k)(n-k-1)!} =$$

$$= \frac{1}{k(k-1)!}\frac{n!}{(n-k-1)!}$$

Como $n!$ cresce mais rapidamente do que $(n-k-1)!$ podemos então concluir que este problema possui uma complexidade computacional $n!$, $\mathcal{O}(n!)$, onde $\mathcal{O}$ representa a notação de *Big-O*.
Não foi possível realizar um fit fatorial aos nossos dados com o programa Matlab contudo, analisando as curvas representadas na Fig.2, podemos confirmar que os nossos dados seguem uma distribuição do tipo $a \times (n-b)! + c$, o que nos ajuda a confirmar a complexidade computacional obtida através da análise formal realizada anteriormente.

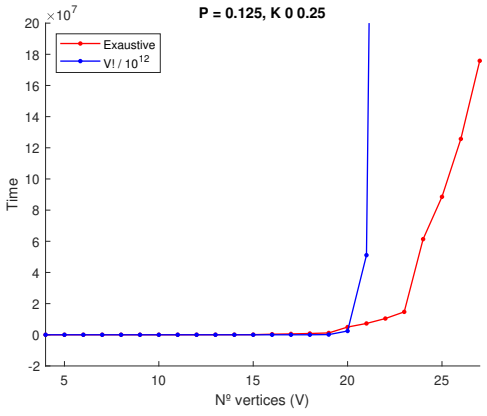

Fig. 2: Comparação do tempo de execução do algoritmo exaustivo com $V!$, em função de V

B. Resultados

B.1 Exemplos

Na Fig.3 podemos ver um exemplo de um grafo gerado pela package NetworkX, para valores de $V = 6$, $K =$

0.50, $P = 0.75$. Neste exemplo, ambos os algoritmos retornaram, como resposta, o conjunto {0,1,2}. Com base no explicado em I-A, podemos concluir que este é realmente um conjunto dominante e com o tamanho pretendido ($math.floor(K \times V)$).

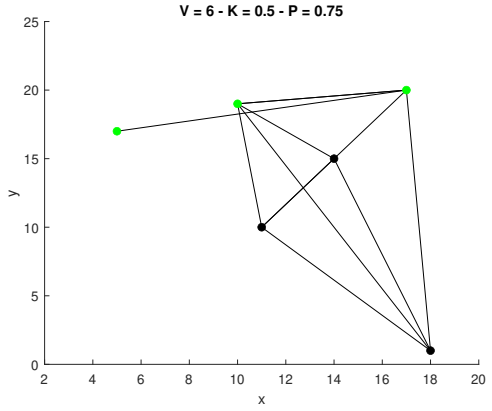

Fig. 3: Exemplo de um grafo gerado fazendo uso da package NetworkX, para $V = 6$, $K = 0.50$, $P = 0.75$, onde ambos os algoritmos encontraram solução

Já nas Figuras 4 e 5 podemos ver os outros dois casos, isto é, quando apenas o algoritmo Exaustivo e quando nenhum encontra uma solução de tamanho pretendido, respetivamente.

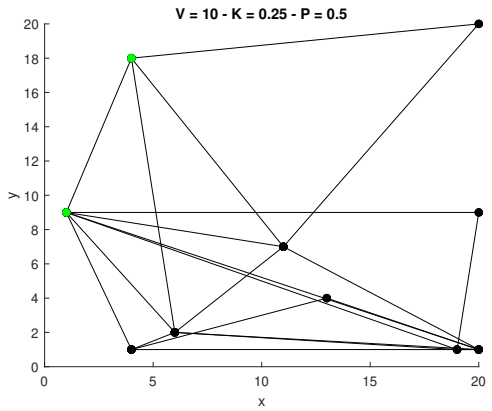

Fig. 4: Exemplo de um grafo gerado fazendo uso da package NetworkX, para $V = 6$, $K = 0.25$, $P = 0.5$, onde apenas o algoritmo Exaustivo encontrou solução

Após uma análise de 'papel e caneta' foi possível confirmar que no exemplo da Figura 4 não existe nenhuma solução que cumpra os requisitos do algoritmo Heurístico, isto é, o vértice 0 pertencer ao conjunto dominante e o conjunto ter tamanho 2. Já no exemplo da Figura 5 foi possível confirmar que não existe nenhuma solução possível de tamanho 1, o que nos dá mais certeza que os algoritmos estão a funcionar corretamente.

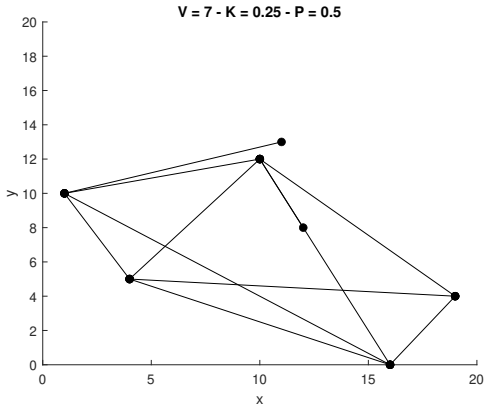

Fig. 5: Exemplo de um grafo gerado fazendo uso da package NetworkX, para $V = 7$, $K = 0.25$, $P = 0.5$, onde nenhum algoritmo encontrou solução

B.2 $P = 0.125$

Para os outros valores de P os gráficos representativos aos tempos de execução e ao número de operações de cada algoritmo encontram-se no Appendix A. Estes não foram alvo de análise pois esta seria igual a que será feita para um valor de $P = 0.125$.

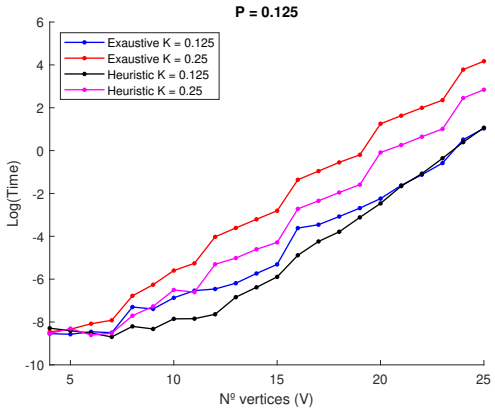

Fig. 6: Logaritmo do tempo de execução dos algoritmos em função do número de vértices - $P = 0.125$ - $K = 0.125, 0.25$

Analisando primeiramente os tempos de execução reparamos que, tal como esperado, estes crescem de uma maneira exponencial, como se pode ver pelas Fig.6 e Fig.7. Reparamos também que o algoritmo Heurístico apresenta quase sempre melhores tempos. Foi feito uma análise mais profunda para perceber quando isto não acontecia e foi descoberto que apenas acontece quando o algoritmo Heurístico não funciona, isto é, quando o algoritmo de pesquisa Exaustiva encontra um conjunto dominante mas o Heurístico não. No entanto, o tempo de execução não é a melhor maneira de comparar algoritmos pois este depende de vários fatores e pode induzir em erro. No entanto, uma boa maneira de comparar algoritmos é comparar o número de operações básicas que cada algoritmo re-

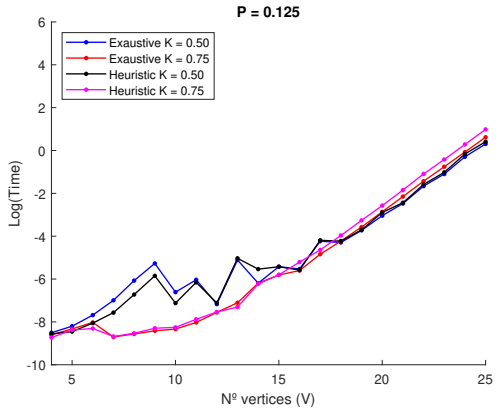

Fig. 7: Logaritmo do tempo de execução dos algoritmos em função do número de vértices - $P = 0.125$ - $K = 0.50, 0.75$

aliza.

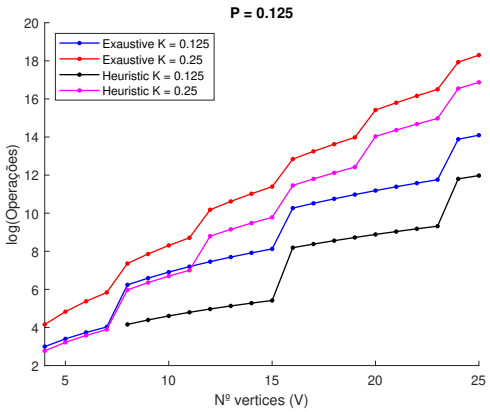

Fig. 8: Logaritmo do número de operações realizadas pelos algoritmos em função do número de vértices - $P = 0.125$ - $K = 0.125, 0.25$

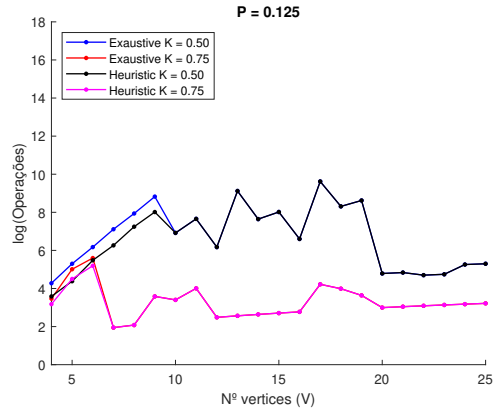

Fig. 9: Logaritmo do número de operações realizadas pelos algoritmos em função do número de vértices - $P = 0.125$ - $K = 0.50, 0.75$

Analisando então o número de operações realizadas por cada algoritmo para cada valor de K e para grafos de vários tamanho, Fig. 8 e 9, rapidamente reparamos que o algoritmo Exaustivo realiza mais ou o mesmo número de operações mas nunca menos que o algoritmo Heurístico. Isto permite-nos concluir, tal como esperado, que o algoritmo Heurístico traz vantagens no estudo deste problema.

C. Número de configurações

Também poderíamos estudar o número de configurações testadas, isto é, o número de conjuntos testados até encontrar uma solução ou chegar ao fim. Contudo, este número seria proporcional ao número de comparações, já analisado anteriormente, Fig.8 e 9, e que podemos confirmar pelas Fig.10 e 11 que realmente o são.

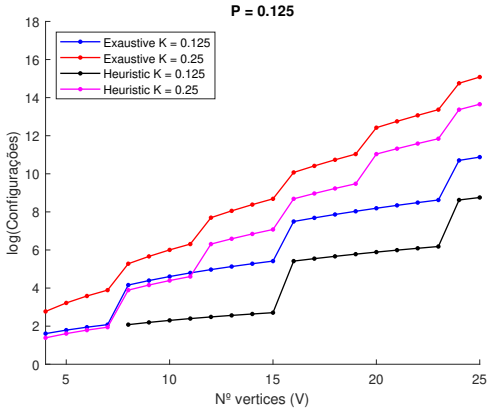

Fig. 10: Logaritmo do número de configurações testadas pelos algoritmos em função do número de vértices - $P = 0.125$ - $K = 0.125, 0.25$

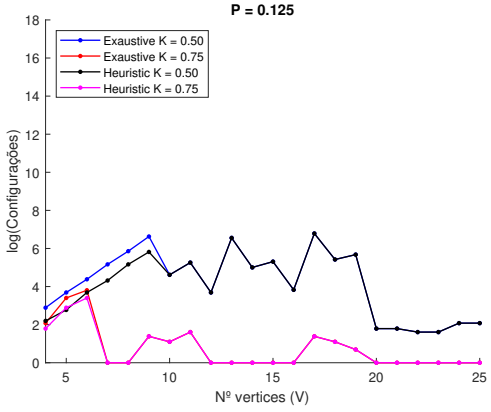

Fig. 11: Logaritmo do número de configurações testadas pelos algoritmos em função do número de vértices - $P = 0.125$ - $K = 0.50, 0.75$

D. Grafos de dimensões maiores

Como o tempo de execução cresce de forma exponencial e o número de operações de forma $n!$, resolver

este problema, para grafos de grandes dimensões, exigiria um grande poder computacional e demoraria um grande período de tempo. No entanto, podemos estimar quanto tempo um problema de dimensão N demoraria. Para isso, basta usar os nossos dados e realizar uma extrapolação (exponencial) destes de forma a obter uma estimativa do tempo pretendido.

O máximo número de vértices calculado foi de $V = 30$ (embora apenas tenha sido feito uma análise até $V = 25$). Em média, o algoritmo Exaustivo tem um tempo de execução de aproximadamente 6 minutos por par (K, P), para $V = 30$. Já para o algoritmo Heurístico este tempo médio é de aproximadamente 2 minutos. Assumindo $P = 0.125$, $K = [0.125, 0.25, 0.5, 0.75]$ e $V \in [4, 5, ..., 30]$ no total, isto é, $Time_{exaustive} + Time_{hueristic}$, foi preciso um tempo de execução de aproximadamente 53 minutos. Vamos apenas considerar $P = 0.125$ e $K = 0.25$ para estudar o tempo necessário para grafos de dimensões maiores, visto que para outros valores de P e de K o procedimento seria o mesmo.

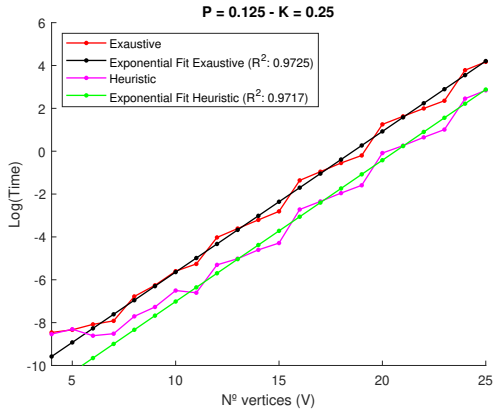

Fig. 12: Logaritmo do número de configurações testadas pelos algoritmos em função do número de vértices + Exponential Fits - $P = 0.125$ - $K = 0.25$

Na Fig.12 estão representados os fits realizados aos dados do algoritmo Exaustivo e do algoritmo Heurístico, cujas equação são as seguintes, respetivamente:

$$f(x) = (4.992 \times 10^{-6}) \times e^{0.6566x}$$

$$f(x) = (1.226 \times 10^{-6}) \times e^{0.6596x}$$

Podemos então usar estas equações para estimar quanto tempo seria necessário para executar os nossos algoritmos para grafos de tamanho $V = N$, cujos valores se encontram representados na Tabela I, para alguns valores de N.

Esta tabela permite-nos também perceber que à medida que o tamanho do grafo aumenta, a diferença entre o tempo de execução dos algoritmos, $diff$, vai aumentando, pelo que, o algoritmo Exaustivo começa a ser menos viável, sendo preferível optar pelo algoritmo Heurístico.

TABLE I: Estimativas do tempo de execução dos algoritmos para $V = N$

$(V, Time(s))$	$Heuristic$	$Exaustive$	$\|diff\|$
$V = 35$	1.3020×10^4	4.7731×10^4	3.4711
$V = 40$	3.5230×10^5	1.2723×10^6	920000
$V = 45$	9.5328×10^6	3.3914×10^7	24381200
$V = 50$	2.5794×10^8	9.0399×10^8	646050000

IV. Soluções

Outra análise que pode ser feita é o número de vezes que o nosso algoritmo Heurístico encontra uma solução, versus o número de vezes que o algoritmo de pesquisa Exaustiva encontra. É de esperar que sempre que o algoritmo Heurístico encontre uma solução o algoritmo Exaustivo também encontre. Isto é exatamente o que acontece como se pode ver comparando as Tabelas II e III, onde vemos que o número de soluções obtidas pelo algoritmo Heurístico, por par (K, P), é sempre menor ou igual ao número de soluções obtidas pelo algoritmo Exaustivo. No Appendix B, podemos ver mais detalhadamente quando é que cada algoritmo encontra uma solução, isto é, para que valor/valores de V é que existe uma solução dado um par (K, P), onde verificamos que o algoritmo Heurístico nunca encontra uma solução para um grafo de certo tamanho sem o algoritmo Exaustivo também encontrar. Aqui vamos apenas analisar a diferença de vezes que o algoritmo Heurístico encontra uma solução perante o algoritmo Exaustivo, para $V = [4, 5, ..., 30]$.

TABLE II: Total número de soluções encontradas para o algoritmo Heurístico

(K, P)	$P = 0.125$	$P = 0.25$	$P = 0.50$	$P = 0.75$
$K = 0.125$	0	0	7	15
$K = 0.25$	0	14	20	23
$K = 0.50$	21	24	27	27
$K = 0.75$	24	25	27	27
Total	45	63	81	92

TABLE III: Total número de soluções encontradas para o algoritmo de Pesquisa Exaustiva

(K, P)	$P = 0.125$	$P = 0.25$	$P = 0.50$	$P = 0.75$
$K = 0.125$	0	0	15	23
$K = 0.25$	2	14	22	27
$K = 0.50$	21	24	27	27
$K = 0.75$	24	26	27	27
Total	47	74	91	104

Sabendo que no máximo, por conjunto de (K, P), temos 27 soluções, comparando o número de soluções encontradas pelos algoritmos vemos que, em média, o algoritmo Heurístico encontra 89.58% do número de soluções do algoritmo Exaustivo, pelo que podemos concluir que o algoritmo Heurístico desenvolvido é uma boa proposta de abordagem ao problema. Em relação ao número total de soluções possíveis vemos que, em média, o algoritmo Heurístico encontra um conjunto

dominante 65.05% das vezes enquanto que o algoritmo Exaustivo encontra 73.15 % das vezes.

V. Aplicações na vida real

Grafos são objeto de vários estudos e têm inúmeras aplicações na vida real como, redes sociais, redes biológicas, recomendações de produtos, mapas/gps entre muitas outras, [4]. Para este problema em específico, uma boa aplicação seria, por exemplo, junto da polícia para descobrir os chefes por detrás de uma rede criminosa, isto pois, os chefes contactam com vários trabalhadores que, em principio, têm pouco contacto entre sim, ou seja, dado N suspeitos e as suas relações poderíamos estudar o conjunto dominante para encontrar quem estaria por detrás da rede criminosa.

VI. Conclusão

Este trabalho permitiu por em prática vários tópicos, desde os mais teóricos, como a análise formal para descobrir a complexidade computacional, aos mais práticos, nomeadamente a escrita de código.

Foi possível também reparar nas vantagens e desvantagens dos algoritmos Heurísticos, em que vimos que é possível obter uma solução correta mais rapidamente mas no entanto também pode não ser obtida nenhuma solução, apesar desta existir.

Quanto a pesquisa Exaustiva é possível concluir que, apesar de ser um algoritmo de fácil escrita e compreensão, para casos de dimensões maiores não é uma opção viável devido a ser um algoritmo que requer um grande poder computacional devido à sua grande complexidade.

References

[1] GeeksforGeeks, "Dominant set of a graph", Aug 2022. **URL:** https://www.geeksforgeeks.org/dominant-set-of-a-graph/

[2] Joaquim Madeira, *Slides Teóricos - Algoritmos Avançados (40751)*, Universidade de Aveiro.

[3] "Greedy algorithm". **URL:** https://www.programiz.com/dsa/greedy-algorithm

[4] "Applications of graphs in computer science", Jun 2022. **URL:** https://unacademy.com/content/nta-ugc/study-material/computer-science/applications-of-graphs-in-computer-science/

Appendix

I. Análise - Resultados

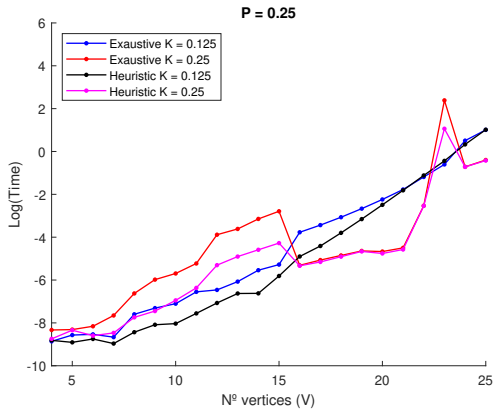

Fig. 13: Logaritmo do tempo de execução dos algoritmos em função do número de vértices - $P = 0.25$ - $K = 0.125, 0.25$

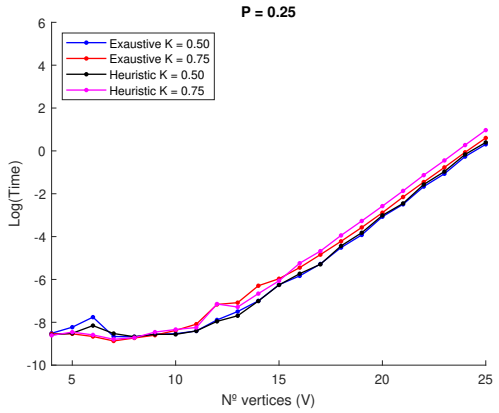

Fig. 14: Logaritmo do tempo de execução do algoritmo Exaustivo em função do número de vértices - $P = 0.25$ - $K = 0.50, 0.75$

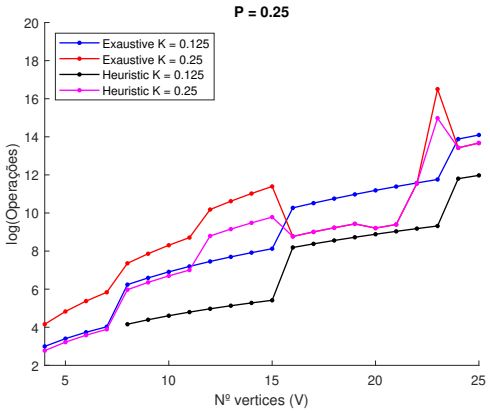

Fig. 15: Logaritmo do número de operações realizadas pelos algoritmos em função do número de vértices - $P = 0.75$ - $K = 0.125, 0.25$

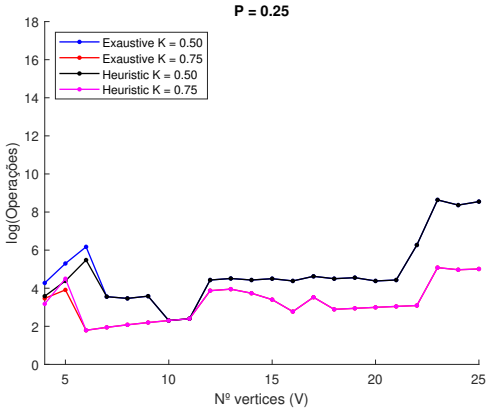

Fig. 16: Logaritmo do número de operações realizadas pelos algoritmos em função do número de vértices - $P = 0.25$ - $K = 0.50, 0.75$

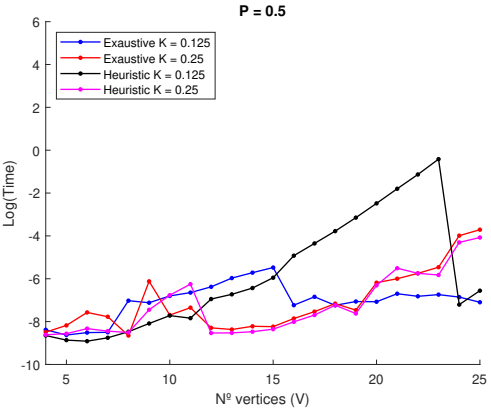

Fig. 17: Logaritmo do tempo de execução dos algoritmos em função do número de vértices - $P = 0.25$ - $K = 0.125, 0.25$

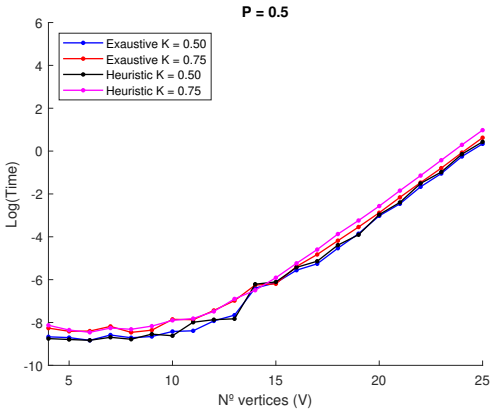

Fig. 18: Logaritmo do tempo de execução dos algoritmos em função do número de vértices - $P = 0.50$ - $K = 0.50, 0.75$

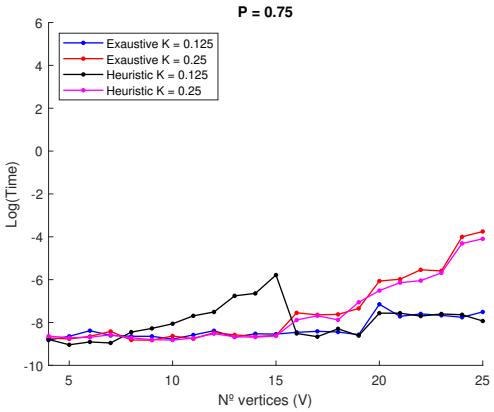

Fig. 21: Logaritmo do tempo de execução dos algoritmos em função do número de vértices - $P = 0.75$ - $K = 0.125, 0.25$

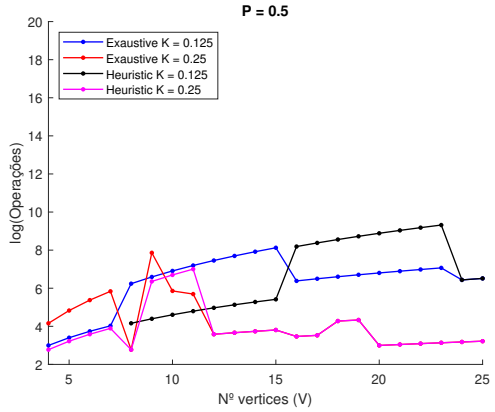

Fig. 19: Logaritmo do número de operações realizadas pelos algoritmos em função do número de vértices - $P = 0.50$ - $K = 0.125, 0.25$

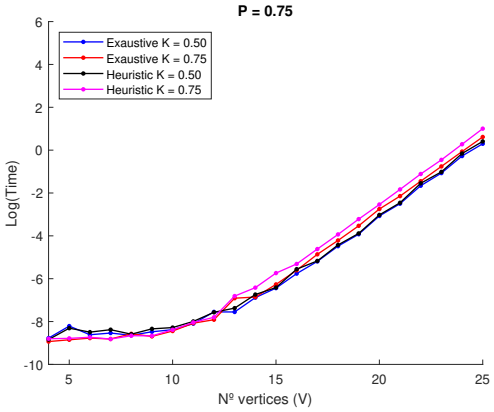

Fig. 22: Logaritmo do tempo de execução dos algoritmos em função do número de vértices - $P = 0.75$ - $K = 0.50, 0.75$

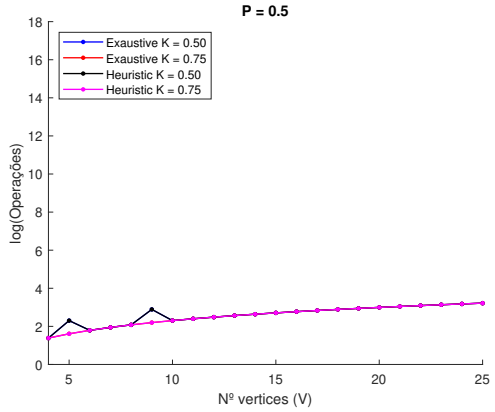

Fig. 20: Logaritmo do número de operações realizadas pelos algoritmos em função do número de vértices - $P = 0.50$ - $K = 0.50, 0.75$

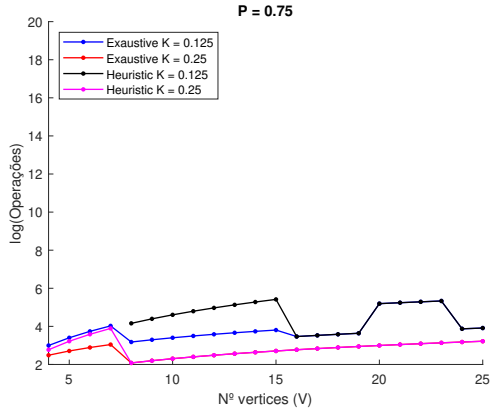

Fig. 23: Logaritmo do número de operações realizadas pelos algoritmos em função do número de vértices - $P = 0.75$ - $K = 0.125, 0.25$

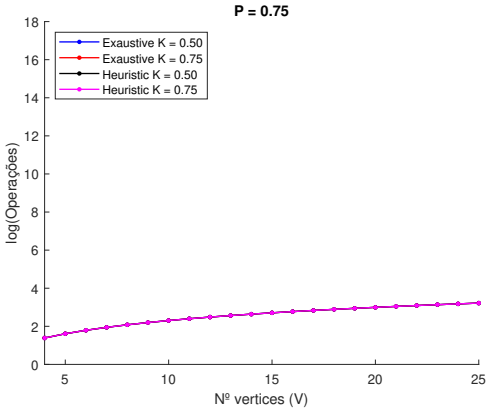

Fig. 24: Logaritmo do número de operações realizadas pelos algoritmos em função do número de vértices - $P = 0.75$ - $K = 0.50, 0.75$

II. Soluções

Dado um par (K, P) o/s vértice/s em que os algoritmos encontram solução, $V \in [4, 5, ..., 30]$

A. $P = 0.125$

A.1 $K = 0.125$

Nenhuma solução encontrada para ambos os algoritmos.

A.2 $K = 0.25$

Nenhuma solução encontrada para o algoritmo Heurístico.
Solução Exaustiva: $V = [28, 29]$

A.3 $K = 0.50$

Solução Heurística e Exaustiva: $V = [10, 11, 12, ..., 30]$

A.4 $K = 0.75$

Solução Heurística e Exaustiva: $V = [7, 8, 9, ..., 30]$

B. $P = 0.25$

B.1 $K = 0.125$

Nenhuma solução encontrada para ambos os algoritmos.

B.2 $K = 0.25$

Solução Heurística e Exaustiva: $V = [16, 17, 18, ..., 22, 24, 25, ..., 30]$

B.3 $K = 0.50$

Solução Heurística e Exaustiva: $V = [7, 8, 9, ..., 30]$

B.4 $K = 0.75$

Solução Heurística: $V = [7, 8, 9, ..., 30]$
Solução Exaustiva: $V = [6, 7, 8, ..., 30]$

C. $P = 0.50$

C.1 $K = 0.125$

Solução Heurística: $V = [24, 25, 26, ..., 30]$
Solução Exaustiva: $V = [16, 17, 18, ..., 30]$

C.2 $K = 0.25$

Solução Heurística: $V = [8, 10, 11, 12, ..., 30]$
Solução Exaustiva: $V = [8, 12, 13, 14, ..., 30]$

C.3 $K = 0.50$

Solução Heurística e Exaustiva: $V = [4, 5, 6, ..., 30]$

C.4 $K = 0.75$

Solução Heurística e Exaustiva: $V = [4, 5, 6, ..., 30]$

D. $P = 0.75$

D.1 $K = 0.125$

Solução Heurística: $V = [16, 17, 18, ..., 30]$
Solução Exaustiva: $V = [8, 10, 11, 12, ..., 30]$

D.2 $K = 0.25$

Solução Heurística: $V = [8, 10, 11, 12, ..., 30]$
Solução Exaustiva: $V = [4, 5, 6, ..., 30]$

D.3 $K = 0.50$

Solução Heurística e Exaustiva: $V = [4, 5, 6, ..., 30]$

D.4 $K = 0.75$

Solução Heurística e Exaustiva: $V = [4, 5, 6, ..., 30]$