Algoritmos Avançados
1st Project — Exhaustive Search
Minimum edge dominating set

Eva Pomposo Bartolomeu

Resumo **- Neste relatório apresenta-se análise formal e experimental do problema proposto na cadeira Algoritmos Avançados, o Conjunto Mínimo Dominante de Arestas. As análises são realizadas a dois métodos de resoluções diferentes, o Exhaustive Search e o Greedy Heuristic.**

Abstract **- This report presents a formal and experimental analysis of the problem proposed in the Advanced Algorithms course, the Minimum Edge Dominating Set. The analyses are performed using two different resolution methods, the Exhaustive Search and the Greedy Heuristic.**

I. INTRODUCTION

No âmbito da cadeira de Algoritmos Avançados, foi-me proposto um trabalho acerca do problema de grafos minimum Edge Dominating Set [1]. Este problema consiste em encontrar um Minimum edge dominating set (conjunto mínimo dominante de arestas) para um dado grafo não direcionado G(V, E), com n vértices e m arestas. Um edge dominating set (conjunto dominante de arestas) de G é um subconjunto D de arestas, tal que toda aresta que não está em D é adjacente a, pelo menos, uma aresta em D. Um minimum edge dominating set é um conjunto de arestas dominantes de menor tamanho possível.

Perante este problema foi pedido a implementação e a testagem do algoritmo Exhaustive Search, e de um outro método usando o Greedy heuristic, para resolver o problema proposto.

O Exhaustive Search [2] avalia todas as possibilidades, cada combinação de níveis permitidos de todos os atributos.

O Greedy Heuristic [3] vai construindo a solução, efetuando em cada etapa a escolha ótima de acordo com as heurísticas.

Após o desenvolvimento dos algoritmos realizar uma análise experimental e formal da complexidade computacional dos algoritmos desenvolvidos, comparando-os depois.

Os objetivos deste relatório são:

- Desenvolver uma geração de grafos adequada ao problema;
- Implementar o Exhaustive Search, e o método Greedy Heuristic, como resolução ao problema;
- Efetuar uma análise formal e experimental dos algoritmos desenvolvidos;
- Comparar a análise formal e experimental.

II. ANÁLISE FORMAL DOS ALGORITMOS

A. Exhaustive Search

O Exhaustive Search a desenvolver vai fazer um ciclo de 1 até ao número de arestas do grafo que está a analisar. Em cada iteração vai ser gerado todos os subconjuntos de arestas possíveis do conjunto de arestas do grafo. Depois, através de um outro ciclo é analisado se algum subconjunto pode ser a solução ou seja, se todas as arestas que não estão no subconjunto são adjacentes a, pelo menos, uma aresta do subconjunto.

Para isso, vai se determinar as arestas que não estão no subconjunto. De seguida, é executado um for sobre estas arestas e verificamos se cada aresta, não pertencente ao conjunto, tem um vértice em comum com alguma aresta do subconjunto. Se não tem significa que a aresta em questão não é adjacente a nenhuma aresta do subconjunto e saltamos para o próximo subconjunto. Se tem, continuamos a analisar as próximas arestas não pertencentes ao subconjunto, se chegarmos ao fim sem encontrar nenhuma aresta que não seja adjacente, paramos o algoritmo e retornamos o subconjunto como solução.

No pior dos casos, analisar todas as combinações de arestas, a complexidade computacional deste algoritmo é o número de combinações possíveis de n elementos (número de arestas) tomados de k a k, tal como é descrito na seguinte fórmula:

$$\sum_{k=1}^{n} (n\ k) = 2^n - 1 \qquad (1)$$

Com base na fórmula anterior, podemos dizer que a complexidade computacional do Exhaustive Search é $O(2^n)$, sendo n o número de arestas no grafo [4].

B. Greedy Heuristic

O Greedy Heuristic vai ordenar o dicionário de arestas, onde as chaves são os vértices não isolados, e os valores

são uma lista dos vértices aos quais a chave está ligada por uma aresta. Este dicionário vai ser ordenado pelo tamanho das listas de cada chave, e além disso os elementos das listas também vão ser ordenados pela ordem a qual estão como chaves no dicionário. Ou seja, ordena-se os vértices pelo número de arestas que têm.

De seguida, é realizado um ciclo até este dicionário ficar vazio. Dentro de cada ciclo, pegamos na primeira chave (vértice 1), e no primeiro valor da lista da chave (vértice 2), e adicionamos ao resultado (um conjunto de arestas). Além disso, criamos uma lista com as listas de adjacências do vértice 1 e do vértice 2. Eliminamos do dicionário as chaves correspondentes ao vértice 1 e ao vértice 2.

Ainda dentro do ciclo, percorremos a lista de vértices anteriormente criada, em cada iteração verificamos se o vértice é chave no dicionário, se não for passamos para a próxima iteração. Se for chave, pegamos na sua respectiva lista de adjacência, verificamos se o vértice 1 e o vértice 2 estão na lista, se estiverem eliminamos ambos os vértices. Caso a lista não esteja vazia, atualizamos esta lista no dicionário, se estiver vazia eliminamos a chave em questão do dicionário [5].

A complexidade computacional do Greedy Heuristic é O(n log n), uma vez que, ordenar os vértices de um dicionário pelo número de arestas é O(n log n), sendo n o número de arestas.

III. GERAÇÃO DE GRAFOS

A primeira etapa a ser implementada foi a geração de grafos, onde foram criados grafos de 2 vértices até 350. Os vértices de cada grafo são pontos 2D no plano XOY, com coordenadas aleatórias de valor inteiro entre 1 e 355. Para efeitos de visualização, deve ser evitado vértices muito próximos, por isso, estabeleceu-se que a distância entre dois vértices tem de ser superior a 1.

Para cada número fixo de vértices, foi gerado 4 grafos distintos com diferentes números de arestas, um com 12.5% do número máximo de arestas, outro com 25% do número máximo de arestas, 50% e 75%, de modo a ter grafos mais esparsos (com apenas algumas arestas) e grafos mais densos (número de arestas está próximo do número máximo de arestas).

Considerando v o número de vértices de um grafo não orientado, podemos dizer que o número máximo de arestas do mesmo é dado pela seguinte fórmula:

$$Max(v) = v(v - 1) \div 2 \qquad (1)$$

As arestas foram também geradas aleatoriamente, ou seja, os vértices de cada aresta que se iam criando, foram escolhidos de forma arbitrária, mas evitando vértices isolados.

Foram guardadas informações de cada grafo em um ficheiro de texto, e numa imagem. No ficheiro de texto, na primeira linha é apresentado um dicionário, em que as chaves são os nomes dos vértices, e os valores das respectivas chaves são as coordenadas do mesmo (apresentadas por um tuplo). Já na segunda linha do ficheiro pode se encontrar as listas de adjacências, isto é, um dicionário onde cada entrada representa uma aresta do grafo, as chaves são os nomes de todos os vértices não isolados, e os valores são uma lista dos vértices a que a chave está ligada por uma aresta.

Na imagem é guardado uma visualização gráfica do grafo. Nas próximas duas figuras podemos ver um exemplo dos dois ficheiros gerados para um grafo:

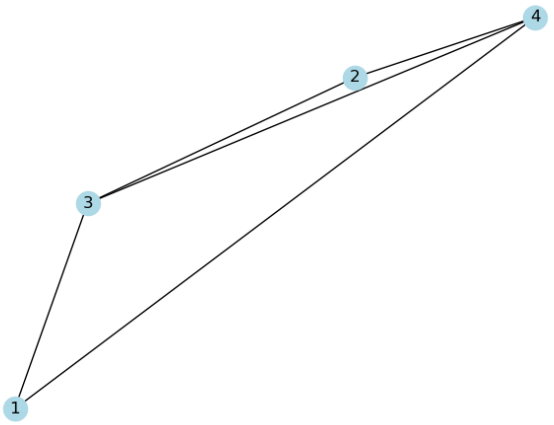

Fig. 1 - Imagem gerada para um grafo com 4 vértices e densidade de arestas de 75%.

```
1  {'1': (84, 25), '2': (252, 146), '3': (120, 100), '4': (341, 168)}
2  {'1': [4, 3], '4': [1, 3, 2], '2': [3, 4], '3': [2, 4, 1]}
```

Fig. 2 - Ficheiro de texto gerado para um grafo com 4 vértices e densidade de arestas de 75%.

IV. ANÁLISE EXPERIMENTAL DOS ALGORITMOS

Foram realizadas várias experiências aos algoritmos, com os grafos anteriormente explicados.

A. Soluções

O Exhaustive Search encontra sempre uma solução possível, já o Greedy Heuristic não garante isso, devolve sempre um edge dominating set, mas nem sempre o do tamanho mais pequeno. Em 52 grafos testados, apenas 9 soluções do Greedy estavam erradas, tinham um tamanho superior a 1 das soluções possíveis. As soluções erradas do Greedy tem sempre tamanho superior ao das soluções possíveis.

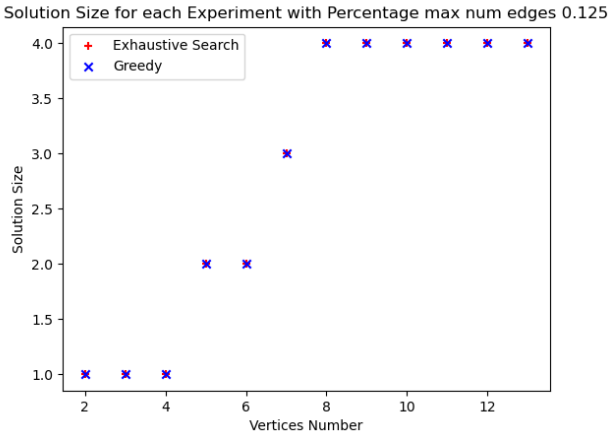

Fig. 3 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 12.5%.

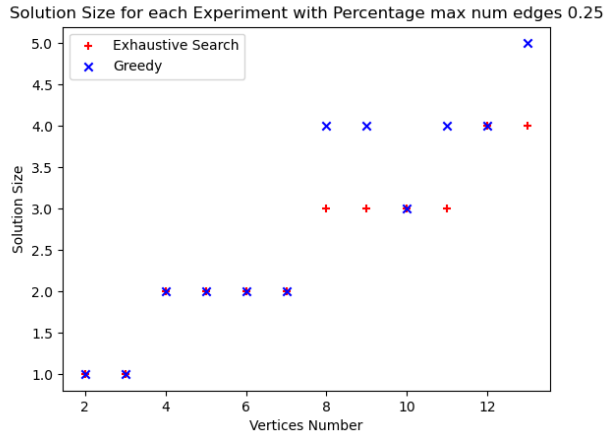

Fig. 4 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 25%.

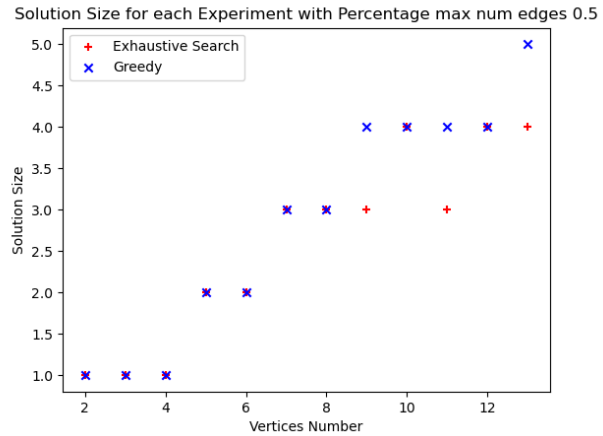

Fig. 5 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 50%.

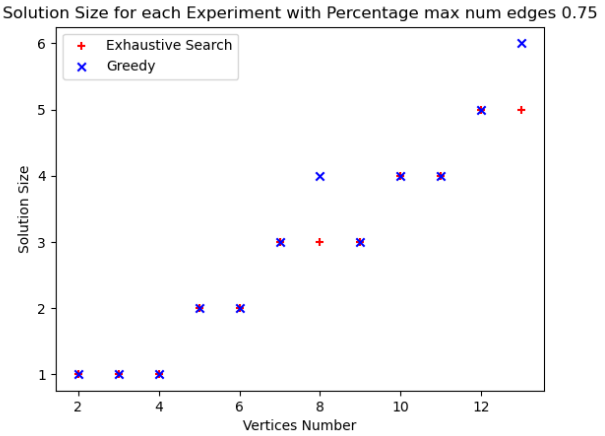

Fig. 6 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 75%.

O Greedy Heuristic dá-nos soluções mais corretas para grafos muito densos, pois a probabilidade de o vértice com mais arestas estar nas arestas da solução é maior Além disso, este algoritmo dá-nos também melhores soluções com grafos muito esparsos, pois neste caso a probabilidade da solução ser o próprio grafo é grande. Ou seja, o Greedy apresenta piores resultados em grafos intermédios em relação à densidade.

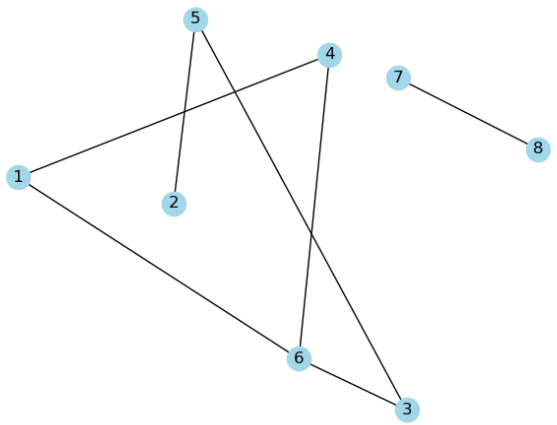

Fig. 7 - Grafo gerado para 8 vértices e densidade de arestas de 25% .

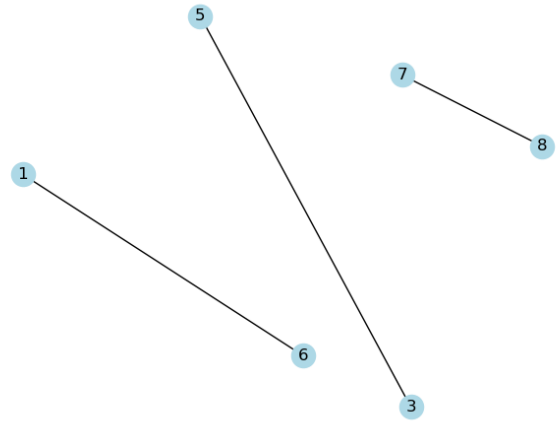

Fig. 8 - Solução encontrada pelo o Exhaustive Search do grafo de 8 vértices e densidade de arestas de 25% .

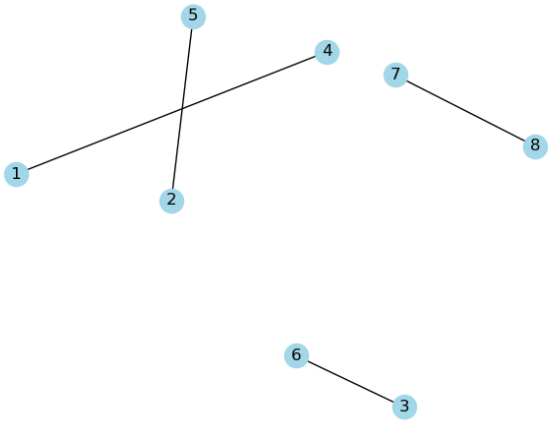

Fig. 9 - Solução encontrada pelo o Greedy do grafo de 8 vértices e densidade de arestas de 25% .

B. Número de operações básicas

Nos gráficos das Figuras 10 e 11, podemos encontrar o número de operações básicas nos dois algoritmos em várias experiências com diferentes números de vértices, e diferentes números de arestas.

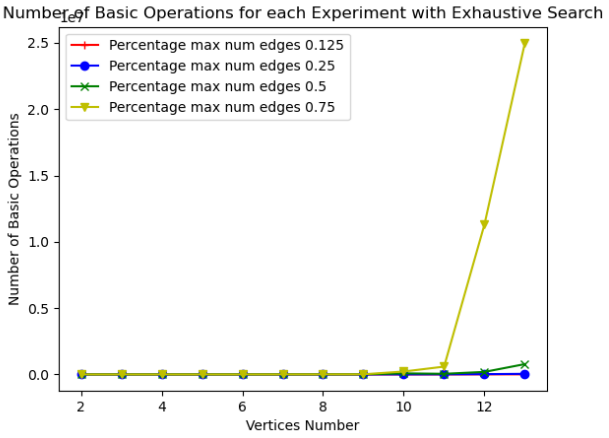

Fig. 10 - Número de operações básicas em cada experiência com o Exhaustive Search.

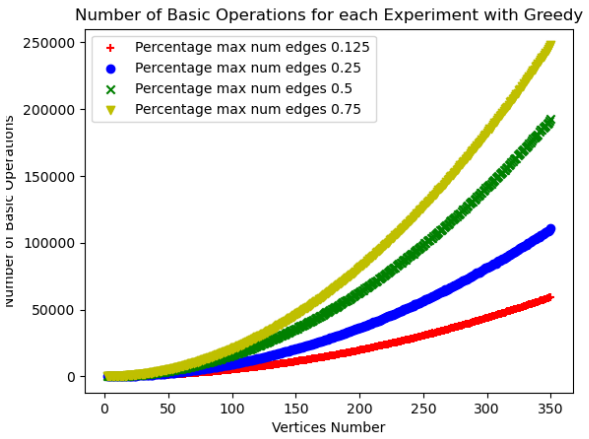

Fig. 11 - Número de operações básicas em cada experiência com o Greedy Heuristic.

Com estes gráficos, podemos concluir que quanto maior for o números de vértices e arestas, mais operações básicas são efetuadas, seja qual for o algoritmo. Apesar disso, é notório que no Exhaustive Search este crescimento, para o mesmo número de vértices e arestas, é muito mais acentuado que no Greedy.

Os valores do número de operações básicas do Exhaustive Search, para grafos densos e de 13 vértices, estão na ordem dos 10^6, já no Greedy estão na ordem dos 10^2, ou seja, o número de operações básicas no Exhaustive é muito maior que no Greedy.

No Exhaustive Search, o maior número de arestas (e o maior número de vértices, que implica um maior número de arestas), leva a um número de operações básicas maior, porque quantas mais arestas existirem, maior o conjunto de arestas, e mais combinações de subconjuntos vão existir para serem testadas.

No Greedy Heuristic, o maior número de arestas leva a um número de operações básicas maior, pois quanto mais arestas um dado grafo tem, quanto mais denso é, mais arestas a serem removidas do dicionário ordenado. Quanto

mais vértices um grafo tiver, mais arestas tem, e provavelmente mais arestas vão estar na solução.

O Exhaustive Search apresenta mais operações básicas que o Greedy, porque no Exhaustive Search testamos vários subconjuntos até encontrarmos uma solução possível, já no Greedy vamos construindo o subconjunto desde o início, vamos construindo a solução, não construímos várias.

C. Tempos de execução

Nos gráficos das Figuras 12 e 13, podemos encontrar os tempos de execução nos dois algoritmos em várias experiências com diferentes números de vértices, e diferentes números de arestas.

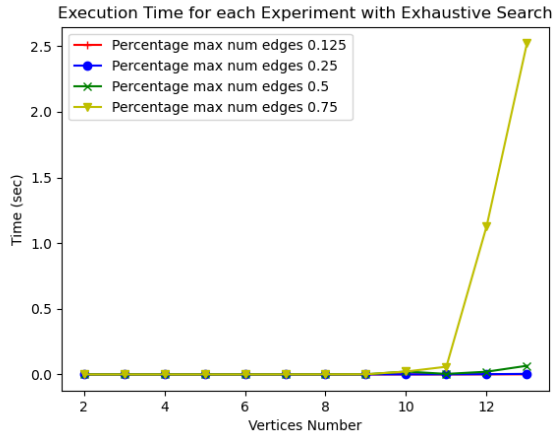

Fig. 12 - Tempo de execução em cada experiência com o Exhaustive Search.

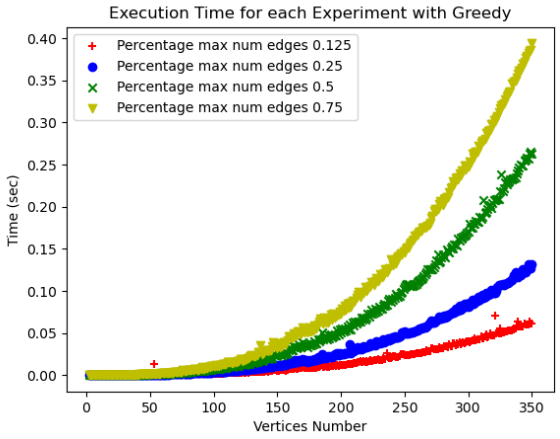

Fig. 13 - Tempo de execução em cada experiência com o Greedy Heuristic.

Tal e qual como no número de operações básicas, podemos dizer que quanto maior for o números de vértices e arestas, maior o tempo de execução, seja qual for o algoritmo. Apesar disso, é notório que no Exhaustive Search este crescimento, para o mesmo número de vértices e arestas, é muito mais acentuado que no Greedy.

Os valores dos tempos de execução do Exhaustive Search, para grafos densos e com números de vértices por volta dos 13, estão com durações de 1, 2 segundos , já no Greedy estão a 0.00008 segundos, ou seja, o número de operações básicas no Exhaustive é muito maior que no Greedy.

As razões para as quais isto acontece são as mesmas que as do número de operações básicas, pois quanto mais operações básicas efetuarmos maior o tempo de execução.

D. Configurações testadas

Passando agora para o número de configurações testadas, podemos dizer que no Greedy o número de configurações testadas é sempre 1, uma vez que neste algoritmo vamos construindo o subconjunto desde o início, isto é, vamos construindo a solução, ou seja, uma única configuração testada. Por isso, foi desenvolvido apenas para o algoritmo Exhaustive Search o gráfico com o número de configurações testadas, em diferentes números de vértices, e diferentes números de arestas.

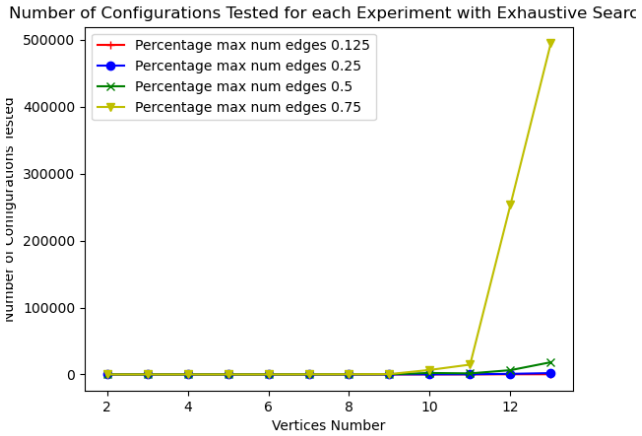

Fig. 14 - Tempo de execução em cada experiência com o Greedy Heuristic.

No Exhaustive Search o número de configurações testadas é o número de subconjuntos testados até encontrar a solução.

Quanto maior forem os números de vértices e arestas, maior o número de configurações testadas no Exhaustive Search. O número de configurações testadas cresce exponencialmente. Isto porque, quantas mais arestas existirem (e mais vértices existirem, que implica um maior número de arestas), maior o conjunto de arestas, e mais combinações de subconjuntos vão existir para serem testadas.

IV. Comparação da análise experimental e formal

A análise formal vai de encontro à análise experimental. Vimos que na análise experimental do Exhaustive Search tanto os números básicos de operações como os tempos de execução, crescem de forma exponencial, tal como concluímos na análise formal. Em relação ao Greedy, em ambas as análises retiramos que os números básicos de operações e os tempos de execução crescem com complexidade de O(n log n).

IV. Conclusion

Os tempos de execução, e o número de operações básicas são valores extremamente grandes na abordagem Exhaustive Search, crescem de forma exponencial, sendo mesmo impossível encontrar uma solução num tempo de vida para problemas de grafos grandes. Em resposta a este problema temos o método Greedy Heuristic, que apresenta respostas muito rapidamente, com um número de operações básicas muito mais pequeno, no entanto, esta abordagem não nos dá sempre a solução correta, às vezes dá-nos uma solução aproximada.

É de realçar que não há um algoritmo melhor que o outro, apenas em certas situações uma abordagem pode ser mais adequada que a outra.

O algoritmo Exhaustive Search é adequado para problemas de grafos muito pequenos, e quando queremos saber respostas 100% corretas. Já o Greedy é apropriado para problemas de grafos enormes, e quando queremos uma resposta rápida e com um valor aproximado à solução.

References

[1] https://en.wikipedia.org/wiki/Edge_dominating_set

[2] https://en.wikipedia.org/wiki/Brute-force_search

[3] https://en.wikipedia.org/wiki/Greedy_algorithm

[4] https://www.educative.io/m/find-all-subsets

[5]https://reader.elsevier.com/reader/sd/pii/S0304397508009110?token=43FB500F80651AEC12EE56442F0C186FA9540EF7738023C0F5F5F193D92F37B48CA555178C7D1BDFD0C9FB6DFBA3A520&originRegion=eu-west-1&originCreation=20221105174510