Algoritmos Avançados
2nd Project — Randomized Algorithms for Combinatorial Problems
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 três métodos de resoluções diferentes, o Exhaustive Search e o Greedy Heuristic (já implementados no primeiro projeto) e o Randomized Algorithm.**

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 three different resolution methods, the Exhaustive Search and the Greedy Heuristic (already implemented in the first project) and the Randomized Algorithm.**

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 Randomized Algorithm, para resolver o problema proposto.

O Randomized Algorithm [2] utiliza a aleatoriedade como parte de sua lógica ou como uma ferramenta para resolver um problema. Este algoritmo resolve problemas difíceis/impossíveis de resolver à mão. Pode também melhorar a eficiência e a robustez do algoritmo.

Após o desenvolvimento do algoritmo realizar uma análise experimental e formal da complexidade computacional do algoritmo desenvolvido, comparando-os depois.

Por fim, analisar a precisão das soluções obtidas comparando-as com as soluções obtidas com os algoritmos do primeiro projeto.

Os objetivos deste relatório são:

- Implementar o Randomized Algorithm, como resolução ao problema;
- Executar os algoritmos desenvolvidos para grafos de grande dimensão;
- Efetuar uma análise formal e experimental do algoritmo desenvolvido neste projeto;
- Comparar as soluções obtidas no Randomized Algorithm com as soluções atingidas nos algoritmos Exhaustive Search e o Greedy Heuristic;
- Comparar a análise formal e experimental.

II. ANÁLISE FORMAL DO RANDOMIZED ALGORITHM

O Randomized Algorithm desenvolvido é do tipo Monte Carlo. O Monte Carlo tem como objetivo explorar de forma aleatória todas as soluções possíveis de um problema, para assim fazer estimativas ou escolhas. Estas estimativas e escolhas podem ser melhoradas fazendo muitas iterações ao método e calculando a média dos resultados.

O algoritmo vai gerar o seguinte número de soluções candidatas, para n arestas:

$$Num(n) = max(\,2,\ 0.23 * \log(2^n - 1)\,) \qquad (1)$$

guardando sempre a melhor solução que obteve (o menor edge dominating set que obteve).

Para gerar cada solução candidata, o algoritmo cria uma variável temporária com as arestas do grafo que está a analisar. Esta variável é um dicionário de listas, que representa a lista de adjacência do grafo.

Além disso, cria também uma variável temporária, que vai ser um conjunto de uma solução candidata.

De seguida é realizado um ciclo até que o dicionário temporário de arestas fique vazio. Dentro de cada ciclo, escolhemos de forma aleatória uma aresta, e removemos a aresta e as arestas adjacentes do dicionário. Depois, guardamos a aresta escolhida aleatoriamente no conjunto que representa a solução candidata.

No fim do ciclo, verificamos se a solução candidata é melhor que a solução que temos guardada, ou seja, se tem um tamanho mais pequeno, se sim, atualizamos a solução guardada para a solução candidata.

No fim do algoritmo, retornamos a solução guardada (a melhor solução que se obteve) [3].

A complexidade computacional do Randomized Algorithm, é de $O(n^3)$, uma vez que o algoritmo apresenta três ciclos, um ciclo para as iterações, outro ciclo que se realiza enquanto a solução candidata ainda não é um edge dominating set e um ciclo para remover as arestas adjacentes à variável temporária dicionário.

III. Geração de Grafos e exemplos de Grafos

Para testar o Randomized Algorithm foram usados os grafos provenientes da geração de grafos que implementei no projeto anterior. A explicação do mesmo pode ser encontrada no relatório do primeiro projeto.

Além disso, foram usados também grafos propostos pelo professor.

IV. Análise experimental dos algoritmos

Foram realizadas várias experiências aos algoritmos desenvolvidos em ambos os projetos, 1 e 2.

A. Soluções

O Exhaustive Search encontra sempre uma solução possível, já o Greedy Heuristic e o Randomized Algorithm não garantem isso, devolvem sempre um edge dominating set, mas nem sempre o do tamanho mais pequeno.

Em 48 grafos testados, apenas 9 soluções do Greedy estavam erradas, tinham um tamanho superior a 1 das soluções possíveis. Para o Randomized Algorithm foram encontradas 18 soluções erradas, geralmente com um tamanho superior a 1 das soluções possíveis.

O Accuracy do Randomized Algorithm para esta experiência foi calculado, dividindo o número de soluções corretas (30) pelo número de grafos testados (48), o que dá um Accuracy de 62.5%.

As soluções erradas do Greedy e do Randomized Algorithm tem sempre tamanho superior ao das soluções possíveis.

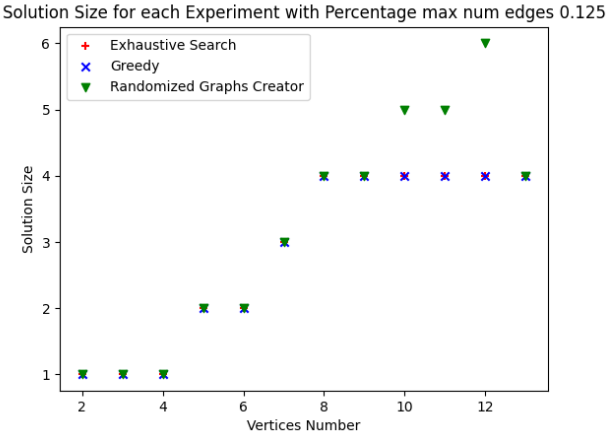

Fig. 1 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 12.5%.

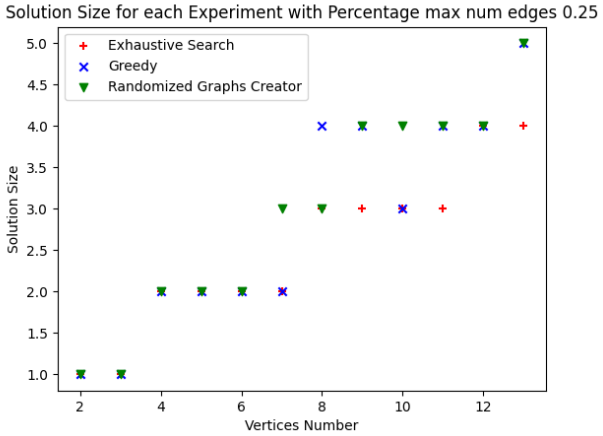

Fig. 2 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 25%.

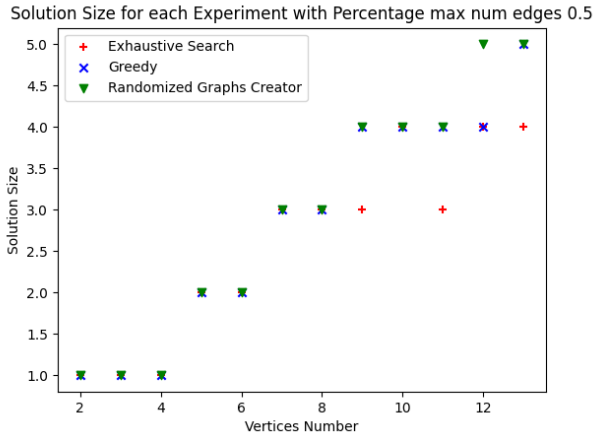

Fig. 3 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 50%.

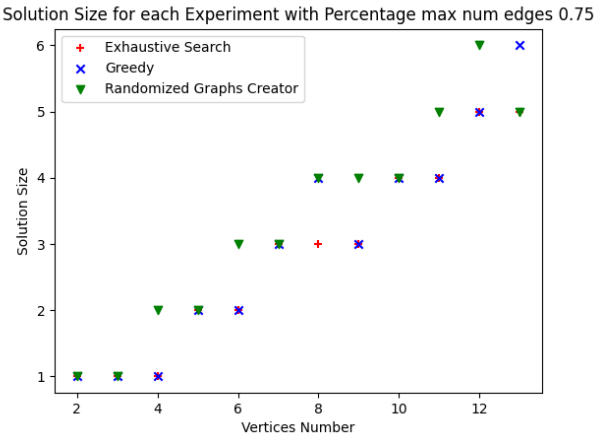

Fig. 4 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 75%.

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.

O Randomized Algorithm dá-nos soluções mais corretas para grafos pouco densos, pois a probabilidade de escolher uma aresta que está na solução é maior, uma vez que os vértices têm poucas arestas. 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 Randomized Algorithm apresenta piores resultados em grafos de elevada densidade.

Analisando agora apenas o Greedy Heuristic e o Randomized Algorithm, através das seguintes figuras:

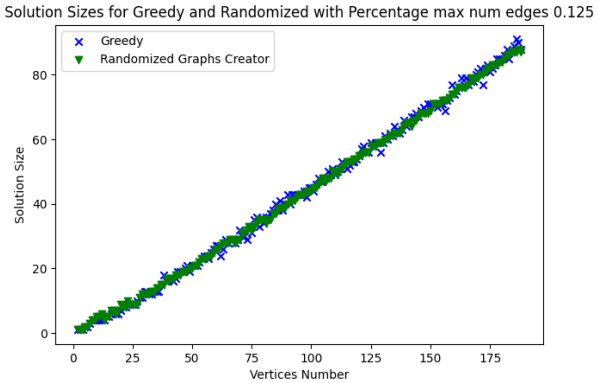

Fig. 5 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 75%.

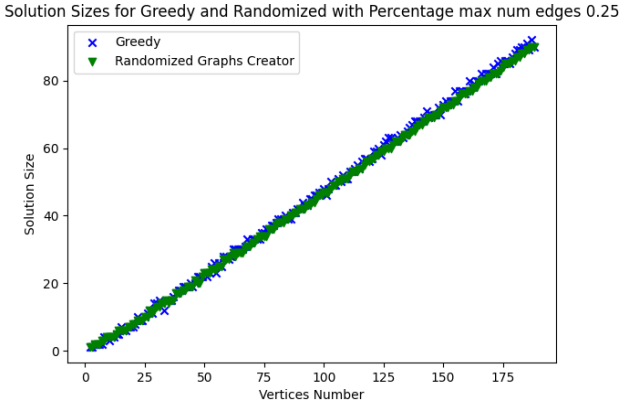

Fig. 6 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 75%.

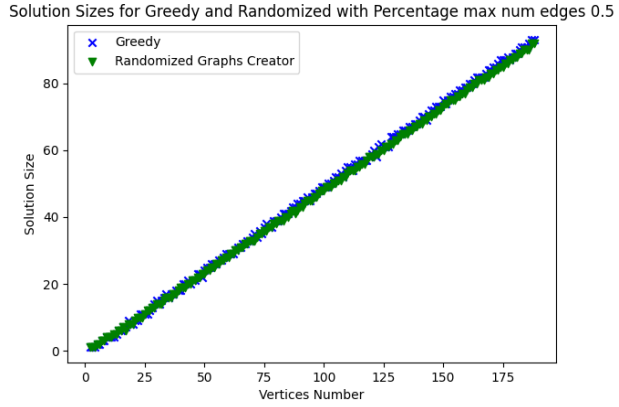

Fig. 7 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 75%.

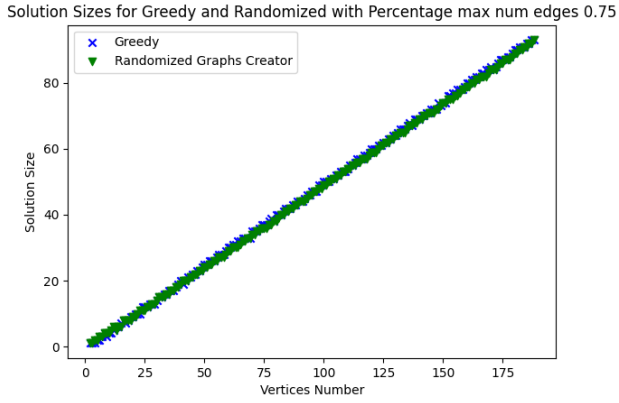

Fig. 8 - Tamanho da solução de cada experiência com percentagem do número máximo de arestas de 75%.

Podemos observar que ambos os algoritmos dão resultados praticamente idênticos, havendo resultados um pouco mais diferentes em grafos mais esparsos.

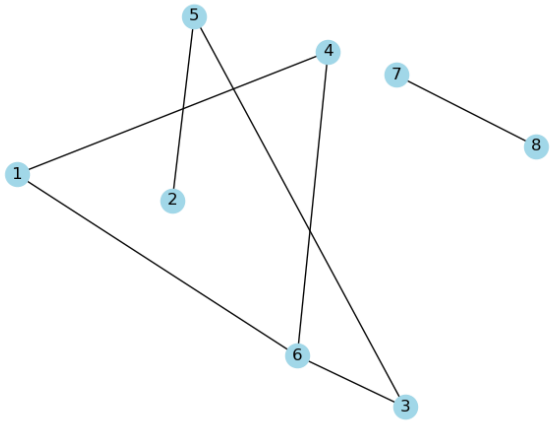

Fig. 9 - Grafo gerado para 8 vértices e densidade de arestas de 25% .

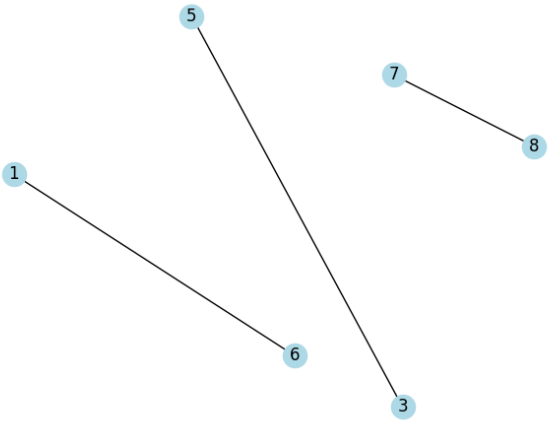

Fig. 12 - Solução encontrada pelo o Randomized Algorithm do grafo de 8 vértices e densidade de arestas de 25% .

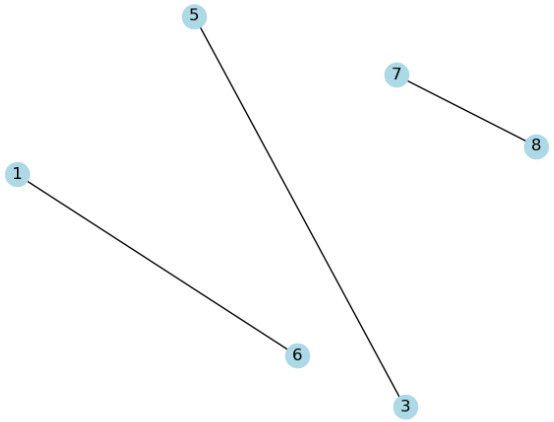

Fig. 10 - Solução encontrada pelo o Exhaustive Search do grafo de 8 vértices e densidade de arestas de 25% .

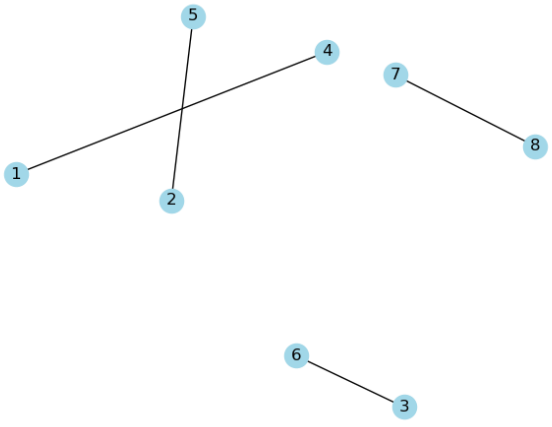

Fig. 11 - Solução encontrada pelo o Greedy do grafo de 8 vértices e densidade de arestas de 25% .

B. Número de operações básicas

No gráfico da Figura 13, podemos encontrar o número de operações básicas no algoritmo em várias experiências com diferentes números de vértices, e diferentes números de arestas.

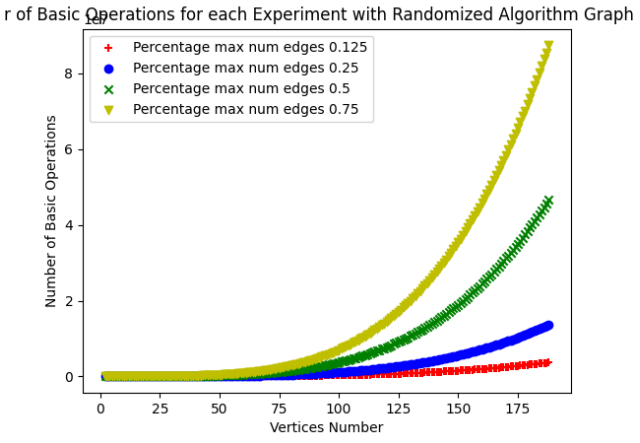

Fig. 13 - Número de operações básicas em cada experiência com o Randomized Algorithm.

Com este gráfico, podemos concluir que quanto maior for o números de vértices e arestas, mais operações básicas são efetuadas.

Os valores do número de operações básicas do Randomized Algorithm, para grafos densos e de 175 vértices, estão na ordem dos 10^6, já para grafos esparsos e de 175 vértices estão na ordem dos 10^5, ou seja, há muitas mais operações básicas para grafos densos.

Estas conclusões, podem se justificar pelo facto de o número de iterações dependerem do número de arestas (tal como já vimos em cima, quando é explicado o algoritmo), logo quanto mais arestas tiver (quanto mais vértices, que implica um maior número

de arestas) e quanto mais denso, mais iterações são efetuadas, e portanto mais operações são efetuadas.

C. Tempos de execução

No gráfico da Figura 14, podemos encontrar os tempos de execução no algoritmo em várias experiências com diferentes números de vértices, e diferentes números de arestas.

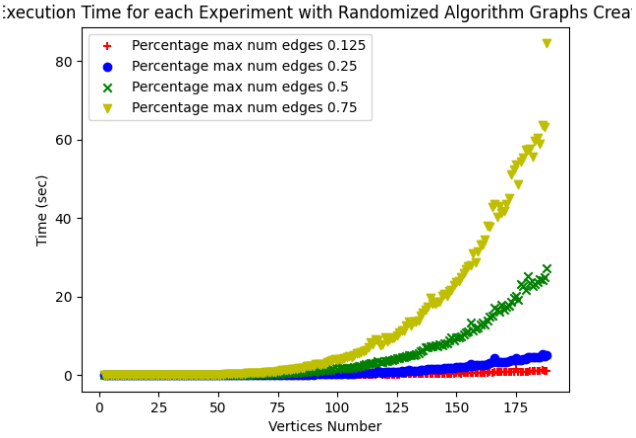

Fig. 14 - Tempo de execução em cada experiência com o Randomized Algorithm.

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.

Os valores dos tempos de execução do Randomized Algorithm, para grafos densos e com números de vértices por volta dos 175, estão com durações de 53, 52 segundos, já para grafos esparsos estão a 0.8, 1 segundos, ou seja, os tempos de execução para grafos densos é muito maior do que para grafos esparsos.

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

O gráfico da Figura 15 apresenta o número de configurações testadas, em diferentes números de vértices, e diferentes números de arestas, no Randomized Algorithm.

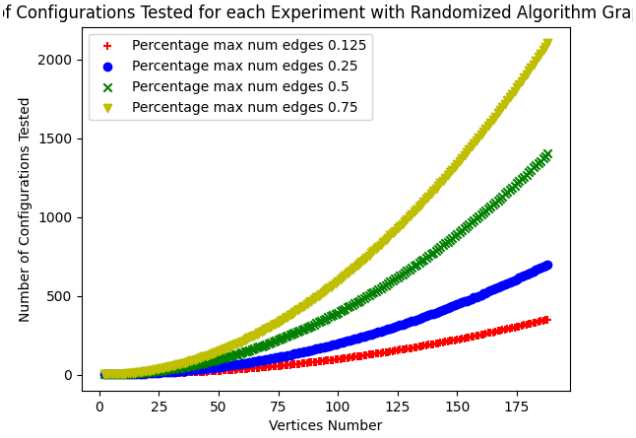

Fig. 15 - Número de configurações testadas em cada experiência com o Randomized Algorithm.

No Randomized Algorithm o número de configurações testadas é o número de iterações efetuadas, determinada pela seguinte fórmula (tal como já vimos anteriormente), n número de arestas:

$$Num(n) = max(2,\ 0.23 * \log(2^n - 1)) \qquad (1)$$

Quanto maior forem os números de vértices e arestas, maior o número de configurações testadas no Randomized Algorithm. O número de configurações testadas cresce, isto porque, quantas mais arestas existirem (e mais vértices existirem, que implica um maior número de arestas), maior o número de iterações a realizar devida a fórmula de cima.

IV. ANÁLISE DO RANDOMIZED ALGORITHM COM OS GRAFOS SW FORNECIDO PELO PROFESSOR

Executamos também o Randomized Algorithm com os grafos SW fornecidos pelo professor, como podemos observar na figura 16.

```
1  vertices_num edges_num solution_size basic_operations_num configurations_tested execution_time
2  250 1273 112 1015274 203 0.349610
3  13 13 4 218 3 0.000123
```

Fig. 16 - Resultados para os grafos SW através do Randomized Algorithm.

Para um grafo de 250 vértices, 1273 arestas obteve-se uma solução de tamanho 112, com um número de operações básicas de 1015274, com 20 configurações testadas e com uma duração de 0.349610s. E para um grafo de 13 vértices, 13 arestas obteve-se uma solução de tamanho 4, com um número de operações básicas de 218, com 3 configurações testadas e com uma duração de 0.000123s.

V. Comparação da análise experimental e formal

A análise formal vai de encontro à análise experimental. Vimos que na análise experimental do Randomized Algorithm os números de operações básicas, os tempos de execução e as configurações testadas crescem com complexidade de $O(n^3)$, tal como concluímos na análise formal.

VI. 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. O Randomized Algorithm é um algoritmo que resolve um problema extremamente difícil em uma resolução bastante simples e fácil de entender, o que não acontece com o Greedy, pois a resolução deste é muito mais complexa. Em termos de accuracy, tempos de execução e de número de operações básicas, o Greedy comporta-se melhor que este.

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. Por fim, o Randomized Algorithm é adequado em problemas difíceis/impossíveis de resolver à mão, são indicados também para quando queremos melhorar a eficiência e a robustez do algoritmo.

References

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

[2] https://en.wikipedia.org/wiki/Randomized_algorithm

[3] https://www.nature.com/articles/srep01736