Prob.1 – Clique of size k – 2nd Assigment

Sobral, Pedro – 98491 – sobral@ua.pt

***Resumo* – Este relatório procura mostrar como o algoritmo "Clique of size k" funciona, neste caso com um algoritmo randomizado. Todos os algoritmos foram escritos em Python (3.10), e serão feitas analises sobre a complexidade computacional dos mesm**

***Abstract* – This report seeks to show how the Clique of size k algorithm works, with a randomized algorithm. All the algorithms were written in Python (3.10), and will be made analizes about the computational complexity of the algorithms.**

I. INTRODUÇÃO

O problema apresentado, é o 1 – Cliques of size k, ou seja, dado um grafo G, um clique de G é um sub-grafo completo de tamanho k, sendo k o número de vértices desse mesmo sub-grafo. [2]

Além deste relatório, foram criados os ficheiros de código em Python, e os ficheiros que guardam os resultados dos testes dos algoritmos dos grafos gerados através da seed correspondente ao número mecanográfico. Como a geração de grafos é um processo rápido, não foi necessário guarda-los num ficheiro como sugerido no enunciado.

O problema irá ser resolvido por um algoritml randomizado, baseado no algoritmo de Las Vegas, onde o mesmo tentará dentro de um intervalo de tempo encontrar a solução esperada (uma clique de tamanho k) dentro do grafo original.

De modo a correr o algoritmo, basta correr o seguinte comando dentro da pasta /src, ou ler o README do projeto:

```
$ python3 main.py
```

II. PROBLEMA

O problema consiste na resulução e na procura de sub-grafos de tamanho k (k corresponde ao número de vértices), de um determinado grafo G. O problema passa por começar a procura num determinado vértice e ver se é possível completar um sub-grafo de tamanho k. [3][4]

O problema adaptado ao mundo real, passa pelo seguinte exemplo: O grafo representa uma rede social, e os vértices desse mesmo grafo, as pessoas que utilizam essa rede social, sendo que as arestas são as conecções das pessoas, como por exemplo serem amigos. Um clique neste contexto pode ser caraterizado como os amigos em comum, pois todos se conhecem entre todos. Este tipo de problema tem mais aplicações práticas, nomeadamente em matérias de bioinformática, e de engenharia eletrítécnica.[1]

III. GERAÇÃO DE GRAFOS

Os grafos são gerados, criando os vértices que podem estar compreendidos entre 1 e 20, e os mesmos são identificados por dígitos, sendo posteriormente feito de forma random a atribuição das arestas com os respetivos vértices criados anteriormente. Para o contexto da resolução do problema é criada uma lista de adjacência, que nada mais é que um dicionário com o vértice como chave e com os vértices a que está ligado como valor, como o exemplo seguinte demonstra:

{0: [3, 4, 9], 3: [0, 1, 5, 7, 8], 4: [0, 5], (...)

Para exemplo de demonstração, a seguinte figura demonstra um grafo gerado:

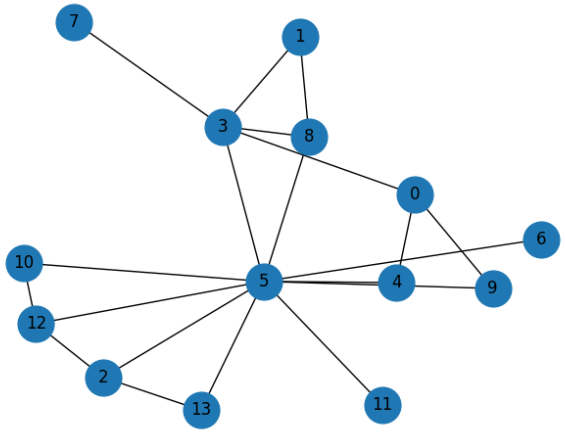


Figura 1 - Exemplo de um grafo gerado

III. ALGORITMO RANDOMIZADO

A abordagem para a criação deste algoritmo foi seguindo a lógica do algoritmo de Las Vegas [5]. Uma implementação baseada no algoritmo de Las Vegas conduz a um resultado correto porém o tempo de execução é probabilístico e varia muito de acordo com o input que é feito e com a "sorte" dos vértices escolhidos em formar uma clique ou não. Sendo que o algoritmo desenvolvido

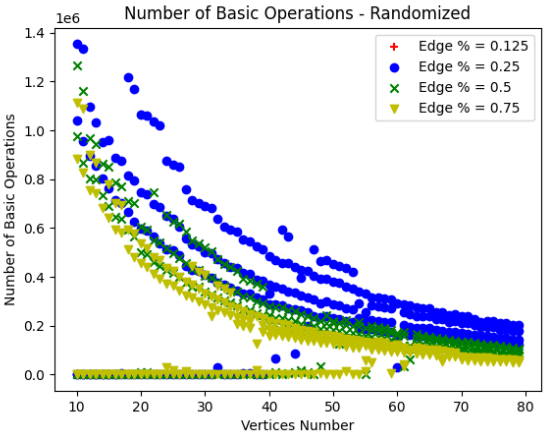
começa por gerar de forma aleatória um conjunto de vértices de tamanho k, e vai verificando se esse conjunto de vértices forma uma clique, enquanto não formar será sempre gerado um novo conjunto de vértices de tamanho k (sempre diferente dos anteriores gerados) até ser encontrado um conjunto que forme uma clique ou até se passarem 5 segundos após ter começado a correr o algoritmo, para tentar amenizar os tempos de execução das centenas de grafos analisados.

V. RESULTADOS

Foi tida em consideração a seguinte abordagem, com grafos diferentes com o número de vértices a variar entre 10 e 80, sendo que para cada valor de vértices foram gerados 4 grafos com [12.5, 25, 50, 75] % de arestas em relação ao número de vértices, onde procurou-se por uma clique de tamanho k (sendo k o valor calculado pelas percentagens [12.5 25 50 75] com o número de vértices). Foram feitas as analises de grafos entre 10 e 80 vértices pois foi o mesmo número para o trabalho anterior, deste modo fazer com os mesmos grafos é uma comparação justa.

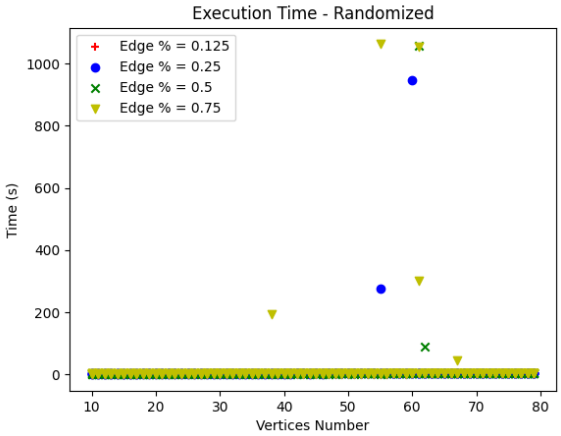
Para analise dos algoritmos desenvolvidos, serão feitas comparações da nova implementação deste trabalho com a implementação do trabalho anterior, e será feita uma analise com os grafos fornecidos pelo Docente no eLearning.

A. *Número de Operações Básicas*



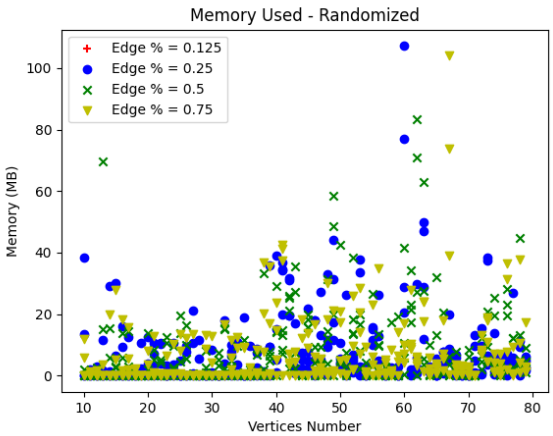
Um dado curioso em relação à visualização do gráfico é que quanto maior o grafo fica em relação ao número de vértices, menor é o número de operações básicas necessárias. Um possível explicação para este acontecimento, será porque como valor de k depende do valor dos vértices, o valor será maior, portanto são mais vértices a verificar se formam uma clique de cada vez, logo menos soluções tem tempo de verificar, dái o número de operações básicas ir diminuindo à medida que o número de vértices aumenta.

B. *Tempos de Execução*



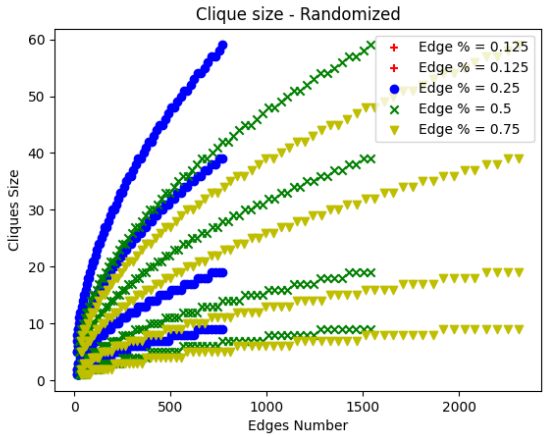
Como já mencionado anteriormente cada grafo tem um tempo de analise de 5 segundos, para encontrar uma clique de tamanho k, este valor foi definido assim de modo que não demorasse muito mais que entre uma hora a duas horas a obter os resultados de todos os grafos analisados. Posto isto a maioria dos casos são 5 segundos de tempo de execução, sendo que há alguns outliers, estes outliers têm uma explicação, quando o computador suspendia, o processamento do algoritmo ficava em stand-by voltado só a funcionar novamente depois do computador entrar na sessão. Tirando os outliears da analise temporal, é possível chegar à conclusão que a complexidade será em torno de **O(n^2*k).** A parte de n^2 por causa dos 2 ciclos for a verificar se é uma clique ou não, e a parte de multiplicar por k porque é feito tantas vezes quantas soluções o algoritmo testar até encontrar uma solução correta.

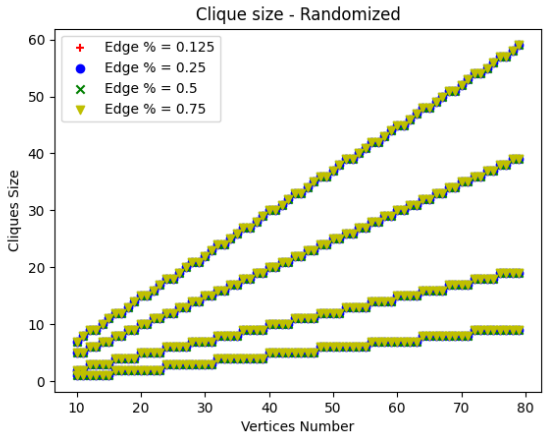
C. *Memória usada para a resolução do algoritmo*



Relativamente à analise da memoria usada, como enquanto não é encontrada uma solução, são guardadas todas as soluções testadas de modo a que não haja nenhuma solução testada mais que uma fez, a memorias ainda é bastante usada como se pode verificar. Assim a quantidade de memória usada é maior quanto mais tempo demorar o algoritmo a encontrar uma solução.

D. Tamanho da Clique comparado com o número de vértices e de arestas





De modo a comparar visualmente o número das cliques conforme o número de vértices e de arestas aumenta. Através dos gráficos é possível ver que ao nível do crescimento do número de arestas o tamanho da clique mesmo cresce logaritmicamente e o nível do crescimento do número de vértices o crescimento é tendencialmente uma porporção direta.

E. Análise estatística de Resultados

Fazendo uma comparação de resultados obtivos com o algoritmo de pesquisa exaustiva do trabalho anterior, podemos observar que dos 1120 grafos gerados e analisados, apenas há 61 resultados diferentes. Ou seja, o algoritmo não é 100% certo, mas apresenta uma taxa de acerto de 94.55%, comparando os resultados com o algoritmo de pesquisa exaustiva.

Quanto à correlação dos resultados, se esta for menor que zero significa que a correlação é inversamente proporcional, igual a zero que não existe relação, e se for maior que zero, que é diretamente proporcional. Neste caso, como mostra a tabela abaixo, as variáveis apresentam um correlação diretamente proporcional uma fez com apresentam um valor de 0.822635.

	Pesquisa exaustiva	Algoritmo Randomizado
Pesquisa exaustiva	1.000000	0.822635
Algoritmo Randomizado	0.822635	1.000000

F. GRAFOS FORNECIDOS

Foram feitas analises dos grafos fornecidos no eLearning, o valor de k continuou a ser calculado com o número de vértices e as seguintes percentagens [12.5, 25, 50, 75], e foram obtidos resultados negativos (que não havia clique de tamanho k) com a exceção do grafo “tiny” onde para as percentagens 12.5 e 25, foram encontrados cliques de tamanho k.

Comparando os resultados com a versão de pesquisa exaustiva do trabalho 1, os resultados foram bastante semelhantes. Como podemos ver nas seguintes tabelas, onde se vê os resultados de cada algoritmo com as percentagens que cada valor de k tomou de acordo com o número de vértices.

Tiny Version

	12.5	25	50	75
Pesquisa Exaustiva	True	True	False	False
Algoritmo Randomizado	True	True	False	False

Medium Version

	12.5	25	50	75

Pesquisa Exaustiva	False	False	False	False
Algoritmo Randomizado	False	False	False	False

Large Version

	12.5	25	50	75
Pesquisa Exaustiva	False	False	False	False
Algoritmo Randomizado	False	False	False	False

Como podemos observar os resultados são os mesmos, comparando com os algoritmos desenvolvidos no trabalho 1, o que permite dizer que o algoritmo randomizado tal como já se viu tem uma grande taxa de sucesso.

VI. CONCLUSÃO

Em tom de conclusão, um pouco como esperado, o algoritmo randomizado baseado no algoritmo de Las Vegas, obtém uma resposta certa (nas vezes, que são poucas que não obtem, poderá dever-se ao limite máximo imposto de 5 segundos para analise do grafo), porém os tempos de execução são probabilísticos e dependem muito do número de operações que são feitas, pois como a solução a analisar pode ser ou não uma solução válida o número de operações vária e muito consuante isso.

É importante realçar que comprando os resultados com o algoritmo de pesquisa exaustiva do trabalho anterior a precisão foi de 94,55%, ou seja comparando o mesmo grafo para o mesmo clique de tamanho k, em 94,55% das vezes o resultado é o mesmo, o que demonstra uma taxa de sucesso alta do algoritmo.

VII. REFERÊNCIAS

[1] - https://en.wikipedia.org/wiki/Clique_problem
[2] - https://pt.wikipedia.org/wiki/Clique
[3] - https://iq.opengenus.org/algorithm-to-find-cliques-of-a-given-size-k/
[4] - https://www.geeksforgeeks.org/find-all-cliques-of-size-k-in-an-undirected-graph/
[5]- https://en.wikipedia.org/wiki/Las_Vegas_algorithm