Trabalho Prático 1

Thaís Ferreira da Silva - 2021092571

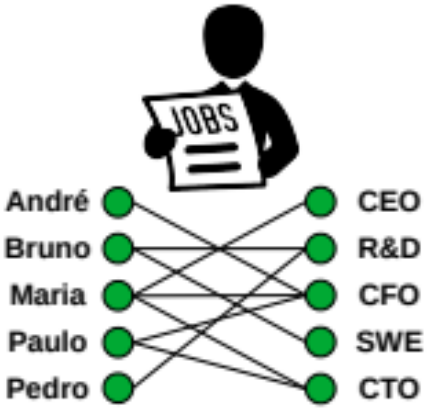
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

thaisfds@ufmg.br

1 Introdução

O problema proposto para esse trabalho prático foi a modelagem de um algoritmo que ajudasse Wille na sua busca do aumento de engajamento na sua rede social, para isso é necessário construir um algoritmo capaz de calcular o emparelhamento máximo em um grafo bipartido separado entre usuários e ofertas de empregos.

Para solucionar esse problema é necessário implementar 2 estruturas diferentes, um algoritmo Guloso e um Exato. O algoritmo exato fará uso do algoritmo ja conhecido ford fulkerson com algumas alterações, além de ser construído com DFS ao invés de BFS padrão.



2 Modelagem

2.1 Estruturas

A estrutura utilizada para a implementação do código foi um Grafo bipartido, e LinkOut. Através dessas estrutura sé possível armazenar usuários e ofertas de emprego como vértices desse garfo, e calcular o emparelhamento máximo das duas formas solicitadas.

- **Grafo:** Para que a estrutura fosse eficiente na busca por caminhos que passam por um número par de estradas, foi necessário a realização de algumas modificações. A logica por trás do desenvolvimento dessa nova estrutura pode ser facilmente comparada com a operação de multiplicação entre números positivos e negativos. ma de dois números de classificação oposta vai gerar um número impar.
- **LinkOut:** Essa estrutura utiliza de um grafo previamente construído para calcular o emparelhamento máximo de duas maneiras diferentes, uma utilizando um algoritmo guloso e a outra um algoritmo exato construído através de uma modificação de Ford-Fulkerson. Ela também possui o algoritmo DFS (Depth First Search) que é utilizado no algoritmo exato.

2.2 Algoritmos

Os 2 métodos implementados para resolução do problema foram um algoritmo guloso, e um algoritmo exato construído com ford fulkerson visto em sala de aula, pois ele é capaz de encontrar o fluxo máximo de um grafo. O objetivo é conectar a maior parte de usuários a pelo menos uma oportunidade de emprego, e para isso será considerado que um usuário só pode ser conectado a uma única oportunidade. Dessa forma será possível agradar o maio número de usuários.

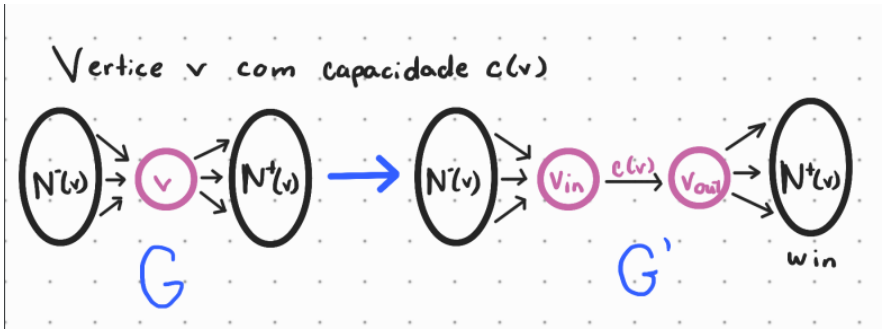
- **algoritmoGuloso:** O algoritmo consiste de sempre pegar a primeira oportunidade de emprego disponível para cada usuário na rede social LinkOut. Desta forma é necessário percorrer toda a matriz de adjacência de possíveis conexões para cada usuário e verificar se essa vaga foi pareada para algum outro usuário. Deve se analisar dois casos possíveis:
  - A oportunidade não foi enviada para nenhum usuário, podendo-se conectar essa oportunidade ao usuários;
  - A oportunidade já foi enviada para algum usuário, dessa forma deve-se analisar a próxima opção de oportunidade.

PSEUDOCODIGO ALGORITMO GULOSO

```
Inicia o método pegando a matriz de adjacência do grafo
Para cada usuário do Linkout
Inicia numeroDeConexoes como 0
  Para cada oportunidade de emprego no Linkout
    Se existe aresta candidato-vaga e a oportunidade não é conectada a nenhum usuário
      Marca a oportunidade como conectada
      numeroDeConexoes++
Retorna numeroDeConexoes
```

- **algoritmoExato:** Para esse algoritmo utilizei a ideia de transformar o problema de emparelhamento máximo em um problema de fluxo máximo adicionando uma origem com arestas para um lado do grafo bipartido e um destino que recebe arestas da outra parte do grafo. Essas novas arestas possuem capacidade igual a 1. Depois utiliza-se o algoritmo de Ford-Fulkerson para achar o fluxo máximo que é igual ao emparelhamento máximo. No entanto é importante enfatizar que o algoritmo foi construído do uma DFS para verificar a existência de um caminho da origem até o destino ao invés de uma BFS.

  Entretanto, na implementação do código não foi necessário adicionar o vesti-se origem e destino, apenas considerar que eles existem. Sendo assim o algoritmo exato utiliza o a mesma matriz do algoritmo guloso e itera sobre cada usuário chamando a função DFS, se existir um caminho até o destino aumenta o contador do fluxo máximo. A DFS será melhor explicada no próximo tópico



PSEUDOCODIGO ALGORITMO EXATO

```
Copia o grafo bipartido da rede para a função
Cria um vetor de candidatos para armazenar o numero de oportunidade de empregos,
e inicializa os seus valores como -1
Inicia o numeroDeConexoes = 0
Para cada usuário
   Inicia o vetor visitado como falso
   Chama BFS
   Se achar um caminho na BFS
      numeroDeConexoes++
retorna numeroDeConexoes
```

- **DFS:** O algoritmo é parecido com o estudado em sala de aula, um DFS recursivo, mas com pequenas alterações para se adaptar ao problema proposto. Ele procura um emparelhamento para um determinado usuário no grafo bipartido. A função DFS percorre todas as oportunidades possíveis no grafo, e para cada oportunidade ele verifica se o usuário posse uma conexão com ele e se a oportunidade não foi visitada/marcada. Se essas condições forem verdadeiras o DFS é chamado novamente para a oportunidade, se isso for possível, atualiza-se o emparelhamento e retorna verdadeiro. Caso contrário, retorna falso.

PSEUDOCODIGO DFS
recebe o vetor bipartido, u, vetor de visitado e vetor de candidatos
para cada vizinho v de u
  se candidato-vaga = 1 e v não foi visitado
    marca o v como visitado
    se candidatos[v] ¡ 0 ou DFS = true
      candidatos[v] = u
      retorna verdadeiro
retorna falso

---

3 Análise

3.1 Comparativa

Para a analise comparativa foi analisado o tamanho da entrada e o tempo de execução para cada um dos algoritmos implementados, na tabela a seguir é possível analisar o teste executado e os tempos para os dois algoritmos em milissegundos.

test_case	Algoritmo Guloso	Algoritmo Exato
0 a 8	0ms	0ms
9	0ms	15ms
10	1ms	186ms
11	2ms	55ms
12	5ms	52ms
13	17ms	322ms

Podemos concluir ao analisar a tabela que o algoritmo Guloso é extremamente mais rápido do que o algoritmo Exato, e isso pode ser relacionado a diversos fatores. A implementação do Guloso é mais simples e rápida, mas fornece resultados mais imprecisos, podendo ser igual ou inferior ao emparelhamento máximo possível. Isso ocorre pois ele sempre leva em consideração apenas o primeiro emparelhamento possível de se realizar, e caso exista um outro usuário que possa se relacionar apenas com a oportunidade já pareada não é possível repensar a escolha de pareamento previamente realizada.

Já o algoritmo Exato é mais complexo e preciso, consequentemente também mais demorado. Ele leva em consideração todas as possibilidade possíveis de emparelhamento, calculando esse número diversas vezes pare tentar achar algum emparelhamento que seja maior do que o encontrado anteriormente.

Por fim, ambos os algoritmos são impactados pelo tamanho da entrada. O tamanho da matriz de adjacência que será construída e o número de arestas que devem ser adicionadas ao grafo, geram um aumento que pode ser facilmente analisado na imagem. Os caos de 0 a 8 possuem entradas relativamente pequenas, onde o teste 8 possui a maior entrada nesse intervalo, com uma matriz 153 x 125 e 250 arestas a serem adicionadas, o seu tempo de execução ainda fica abaixo de 1ms. Comparando esse tempo com os dos casos 10 e 13 é possível perceber esse aumento, onde o caso 10 possui matriz 954 x 864 e 3400 arestas e o caso 13 possui uma matriz 5037 x 1627 com 11000 arestas.

3.2 Complexidade de Tempo

Em relação a complexidade de tempo da estrutura grafo, como suas operações são lineares a sua complexidade será O(1). Já em relação ao LinkOut todas as suas funções possuem complexidade O(Usuário * Oportunidade) por ser necessário iterar duas vezes para percorrer toda matriz de adjacência.

3.3 Complexidade de Espaço

Sobre a complexidade de Espaço da estrutura grafo implementado, a maioria de suas funções possui complexidade linear O(1) exceto o seu construtor que é O(Usuários*Empregos) deviso a inicialização do vetor. Já em relação ao LinkOut O algoritmo Guloso possui complexidade O(1) por utilizar a matriz já alocada na estrutura Grafo, e O algoritmo Exato possui complexidade O(Usuário * Oportunidade) pois preferi realocar a matriz e facilitar o manuseio dos dados.