Felipe Machado Cordeiro

Universidade Federal de Minas Gerais

Matemática Computacional

Introdução

O problema consiste em dado um grafo ponderado direcionado G(V,A), onde as arestas representam ciclovias em uma cidade, o peso a quantidade de ciclistas que podem trafegar naquela ciclovia por hora com segurança e os vértices representam intersecções entre as ciclovias. Nesta cidade estão distribuídas diversas franquias de uma loja que faz entregas usando ciclistas e seus clientes. A partir desses dados, precisa-se encontrar a quantidade máxima de ciclistas que podem circular com segurança nas ciclovias.

Solução do problema

A solução usada neste problema foi baseada na solução de Ford Fulkerson adaptada para o caso de várias fontes, as franquias, e terminais, os clientes.

O algoritmo de Fulkerson consiste em realizar buscas em largura no grafo começando da fonte enquanto um terminal for encontrado, ou seja, enquanto houver um caminho entre fonte e terminal. Para cada caminho encontrado o algoritmo encontra o fluxo capaz de passar por este, definido pela menor capacidade das arestas pertencente a este. Depois de encontrado o valor fluxo, é retirado da capacidade de cada aresta do caminho este valor.

A adaptação do problema proposto para o algoritmo de Fulkerson foi criar uma fonte e terminal equivalente a todas fontes e a todos terminais. A fonte criada possui uma aresta que liga a todas as outras fontes originais e tem capacidade infinita. O terminal criado segue o mesmo conceito, sendo que em seu caso todos os terminais originais possuem uma aresta com capacidade infinita que os ligam com o terminal criado. Dessa forma toda busca iniciada na fonte criada passa necessariamente por uma das fontes originais e caso a busca termine em um terminal no caminho sempre estará um dos terminais originais.

Representação em alto nível do algoritmo.

```
BFS(grafo, pai) {
    fonte = franquia(grafo)
    terminal = cliente(grafo)
    fila f
    vistou[vértices(grafo)] = {0,0...0}
    pai[fonte] = -1
    push(f, terminal)
    fluxo_total = 0
    enquanto( !vazia(p) )
        u = frente(p)
        pop(f)
        v = vizinho(u)
        enquanto(v != -1)
```

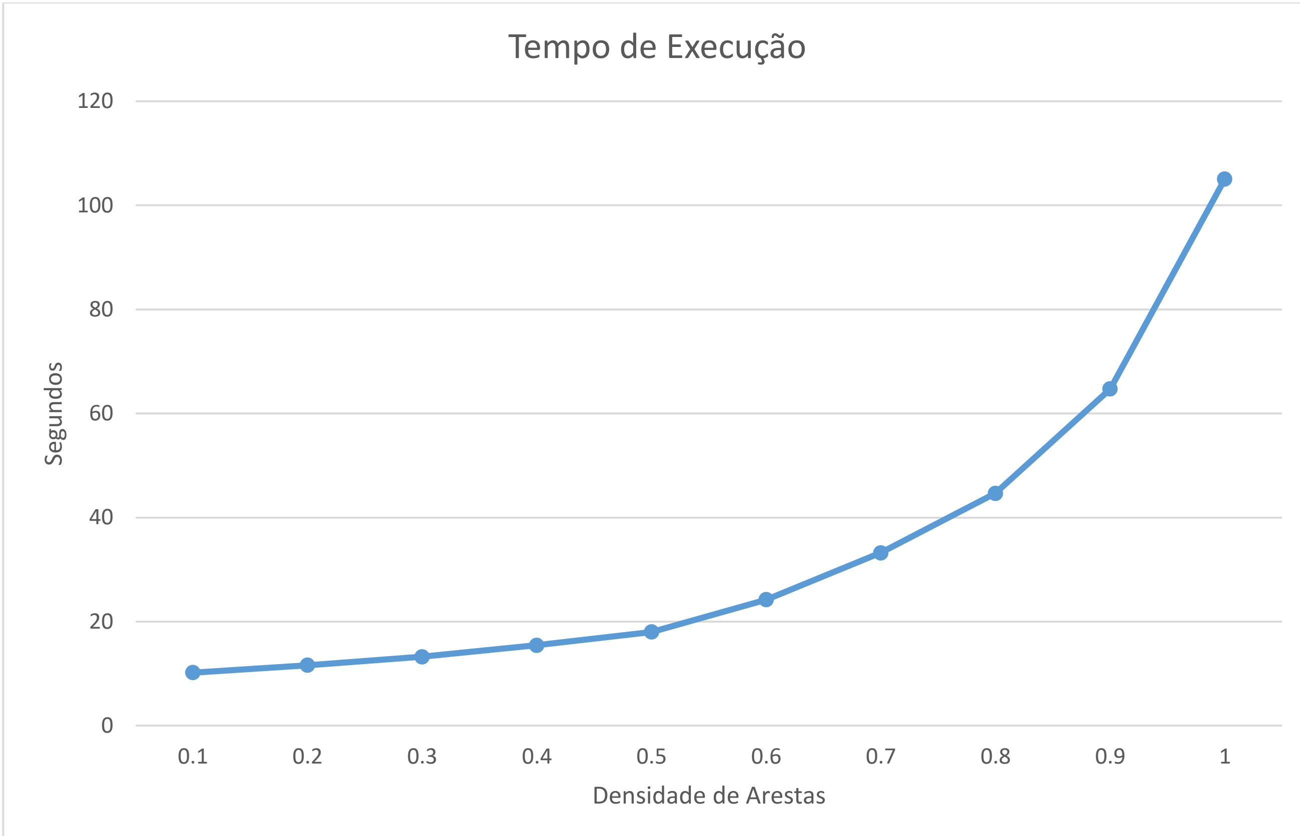
```
                se(visitou[v] != 1)
                    visitou[v] = 1
                    pai[v] = u
                    push(f, v)
                fim_se
                v = próximo_vizinho(u, v)
            fim_enquanto
        fim_enquanto
        retornar visitou[cliente] == 1
fim
encontrar_fluxo_maximo(grafo) {
        pai[vértices(grafo)]
        fonte = franquia(grafo)
        terminal = cliente(grafo)
        enquanto( BFS(grafo, pai)  )
            fluxo_caminho = encontrar_fluxo_caminho(grafo, pai, fonte, terminal)
            adicionar_fluxo_caminho(grafo, pai, fonte, terminal, fluxo_caminho)
            fluxo_total += fluxo_caminho
        fim_enquanto
        retornar fluxo_total
fim
```

Análise de Complexidade

Para a solução definida, toda vez que um caminho for encontrado, uma aresta vai se tornar inutilizada, pois o fluxo que passa por um caminho e definido pela menor capacidade das arestas. Essa aresta, porém, pode voltar a ser reutilizável caso seja um caminho hipotético que passa em seu sentido contrário for encontrado. Toda vez que isso acontecer, porém, o caminho da fonte a até vértice já terá aumentado pelo menos +2, e como o caminho de maior tamanho e limitado por |V|, isso pode acontecer no máximo |V|/2 vezes. Como há no máximo O (E) pares de vértices que usam esta aresta, o número de arestas que se tornam inutilizáveis é limitado por O (|V||E|), que limita a quantidade de caminho encontrados. Como o caminho encontrado tem no máximo O (E) arestas, encontrar o fluxo de um caminho e adiciona-lo a cada aresta é O (E). O algoritmo então ao total é então O (|V||E|^2).

Análise Prática

Para realizar a análise pratica do algoritmo, fixou-se um número de vértices e aumentou a densidade de arestas no grafo, guardando o tempo de execução para cada quantia. Pode perceber no gráfico que o aumento é polinomial.


Tempo de Execução
120
100
80
60
40
20
0
Segundos
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Densidade de Arestas