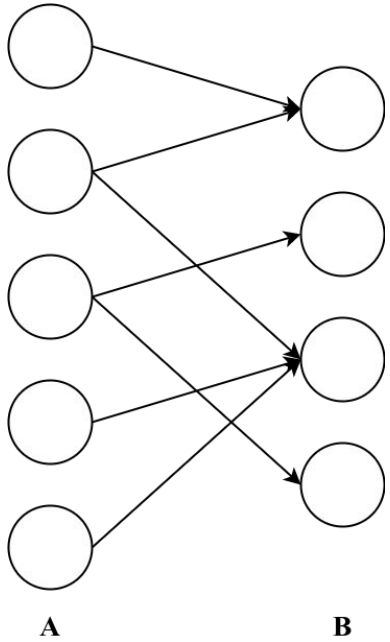
Network Flow: Task allocation using Bipartite Graph

Integrantes: Ilem Lima dos Santos
Professor: Herbert Oliveira Rocha

Grafo e Grafo Bipartido

Um grafo é uma tripla ordenada (V, E, φ) formado por um conjunto V de elementos chamados vértices, um conjunto E de elementos chamados arestas e uma função de incidência φ que associa a cada aresta um par de vértices ordenado de V.

Dentro da Teoria dos Grafos, existe um tipo específico de grafo chamado bipartido. Os grafos bipartidos são aqueles em que é possível separar os vértices em dois conjuntos distintos tal que nenhuma aresta do grafo conecta dois vértices de um mesmo conjunto, ou seja, os dois vértices de uma aresta estão sempre em conjuntos distintos.


A
B

Redes de Fluxo

Uma rede de fluxo $G = (V, E, c)$ é um grafo dirigido no qual cada aresta $(u, v) \in E$ tem uma capacidade não negativa $c(u, v) >= 0$. Numa rede de fluxo temos dois vértices especiais, source s e um terminal t, também chamado de sink, e para todo vértice v do grafo existe um caminho a partir de s passando por v que chega em t.

$$s \in V : source$$

$$t \in V : terminal\ ou\ sink$$

$$\forall v \in V - \{s,\ t\} : nos\ internos$$

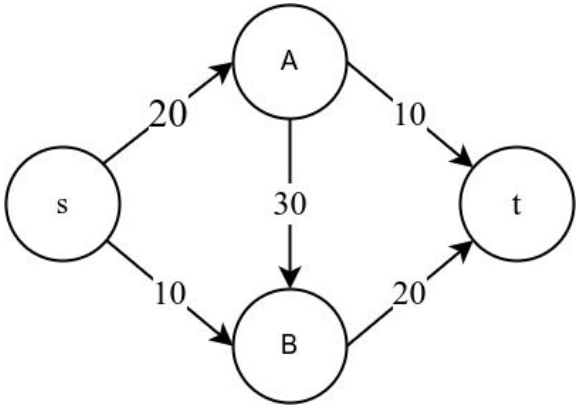
Redes de Fluxo

Diversos problemas do mundo real que podem ser formulados como fluxo de redes:

- Planejamento de voos em linhas aéreas;
- Escoamento de produção;
- Alocação de recursos a tarefas.
- Correntes que passam por redes elétricas
- Informações transmitidas por redes de comunicação.

Fluxo Máximo

O problema do Fluxo Máximo consiste em encontrar um fluxo f do vértice s (source) ao vértice t (terminal) através de uma rede capacitiva em que o fluxo f seja máximo.



Algoritmo de Ford-Fullkerson

O método de Ford-Fulkerson tem por objetivo encontrar um fluxo máximo para uma rede de fluxos.

O método é iterativo, começando com f (u, v) = 0 para todo u, v ∈ V dando um fluxo inicial o valor 0. A cada iteração aumentamos o valor do fluxo, encontrando um caminho aumentante, que podemos imaginar como um caminho de s a t onde podemos empurrar mais fluxo e depois aumentar o fluxo ao longo desse caminho. O processo é repetido até que não sejam encontrados mais caminhos aumentantes.

Algoritmo de Edmond-Karp

O algoritmo de Edmonds-Karp consegue melhorar o limite do algoritmo de Ford-Fulkerson escolhendo sempre a distância do caminho mais curto, desde s até t na rede residual.

Implementação

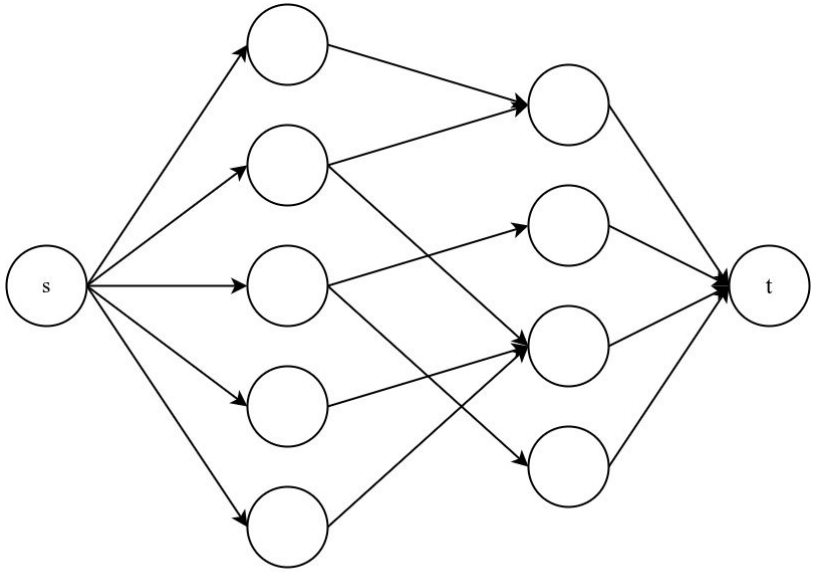
A implementação dos algoritmos de Ford-Fulkerson e Edmond-Karp se deram na linguagem de programação C++.

Primeiramente consideramos, a entrada do programa os dois conjuntos: conjunto de funcionários e tarefas. Para cada tarefa, o programa também recebe a lista de funcionários que podem concluir a tarefa. Por enquanto, consideramos que cada tarefa só pode ter um funcionário alocado nela.

Implementação

Podemos modelar esse cenário com um grafo bipartido, sendo o conjunto A os funcionários e B conjunto de tarefas. Através da utilização de uma matriz de adjacência, caso o funcionário i possa fazer a tarefa j, haverá uma aresta na posição matriz[i][j].

O próximo passo é converter nosso grafo bipartido em uma rede de fluxo, adicionando um vértice source s que se ligará a todos os vértices do conjunto A, e um terminal t ao qual todos os vértices do conjunto B estarão ligados. Todas as arestas deste grafo terão valor unitário.


s
t
A
B

Análise de Complexidade

O tempo de execução do algoritmo de Ford-Fulkerson depende de como determinamos o caminho aumentador p. Em alguns casos, o algoritmo pode executar infinitamente (se as arestas forem números irracionais) ou demorar muito tempo até encontrar o valor de fluxo máximo.

Se as capacidades das arestas consistirem em números inteiros, no pior caso o caminho será aumentando de uma em uma unidade a cada iteração.

O algoritmo de Ford-Fulkerson, a ordem de complexidade desse algoritmo é $O(|E| f^*)$ de , onde E é o número de arestas e f^* é o fluxo máximo da rede.

Análise de Complexidade

Como cada iteração do algoritmo de Ford-Fulkerson leva tempo $O(E)$ quando se utiliza Busca em Largura na procura do caminho aumentador, o algoritmo de Edmonds-Karp pode resolver o problema com complexidade de pior caso $O(|V| |E|^2)$, onde V é o número de vértices e E é número de arestas do grafo.

Obrigado!