Network Flow: Task allocation using Bipartite Graph

Ilem Lima dos Santos
Departamento de Ciência da Computação
Universidade Federal de Roraima, UFRR
Boa Vista, Brasil
ilemlima@gmail.com

***Abstract*—This report in article format aims to understand the operation of the Ford-Fulkerson and Edmonds-Karp algorithms, which are used to solve the problem of maximum flow in a flow network. Analyze its complexity and its applications, specifically in task allocation.**

***Keywords*—Bipartite graph, Graph theory, Network flow, Task allocation.**

***Resumo*—Este relatório em formato de artigo tem como objetivo buscar entender o funcionamento dos algoritmos de Ford-Fulkerson e Edmonds-Karp, que são utilizados para resolver o problema do fluxo máximo em uma rede de fluxo. Analisar sua complexidade e suas aplicações, em específico na alocação de tarefas.**

***Palavras-chave*—Alocação de tarefas, Fluxo de rede, Grafo bipartido, Teoria dos grafos.**

I. INTRODUÇÃO

A. Grafo

Um grafo é uma tripla ordenada (V, E, ψ) formado por um conjunto não vazio V de elementos chamados vértices, um conjunto E de elementos chamados arestas e uma função de incidência ψ que associa a cada aresta um par de vértices ordenado (distintos ou não) de V [2].

Diversos problemas do mundo real podem ser modelados e estudados por meio de grafos. Considere, por exemplo, um trecho do mapa rodoviário do estado de Roraima que inclui a cidade de Boa Vista. Suponha que em uma viagem tenha que sair de Boa Vista e visitar as cidades de Caracaraí, Rorainópolis e Caroebe. Este problema pode ser representado através de um grafo.

B. Grafo Bipartido

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.

C. Alocação de Tarefas

O problema da alocação de tarefas é o problema de atribuir tarefas a máquinas de modo a satisfazer algum objetivo geral, o problema é bastante conhecido nos contextos de produção industrial e de sistemas operacionais.

A ideia da alocação de tarefas também pode ser utilizada em outros casos, como no nosso caso em que uma organização quer alocar as tarefas entre os funcionários de forma que o trabalho seja concluído o mais rápido possível.

A Figura 1 ilustra a ideia da alocação de tarefas entre funcionários e tarefas usando um grafo bipartido.

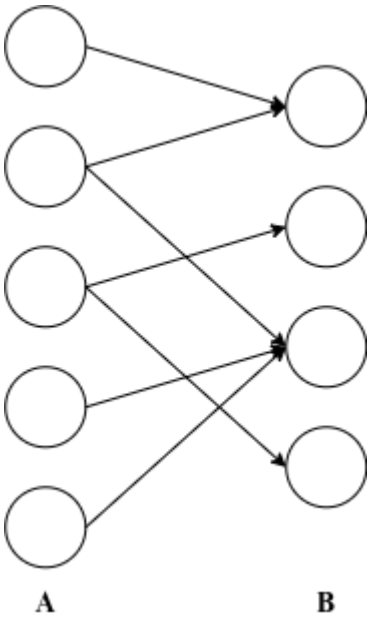

Fig. 1. Grafo bipartido.

D. 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 $c(u,\ v) \geq 0$ [1]. 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. Iremos classificar os vértices de G como:

$$s \in V : source$$
$$t \in V : terminal\ ou\ sink$$
$$\forall v \in V - \{s,\ t\} : nós\ internos$$

De um modo mais formal, dizemos um fluxo em G como uma função $f : E \to \mathbb{N}^+$, que satisfaz as três propriedades:

i) Para todos $u, v \in V$, seja $0 \leq f(u,\ v) \leq c(u,\ v)$
ii) Para todo $u \in V - \{s,\ t\}$, seja
$$\sum_{v \in V} f(u,\ v) = 0$$
iii) Para todo $u, v \in V$, seja $f(u,\ v) = -f(v,\ u)$

A redes de fluxo podem modelar muitos problemas, entre eles líquidos que fluem por tubos, peças que percorrem linhas de montagem, correntes que passam por redes elétricas e informações transmitidas por redes de comunicação.

II. FLUXO MÁXIMO

O problema do Fluxo Máximo consiste em encontrar um fluxo f do vértice s ao vértice t através de uma rede capacitiva em que o fluxo f seja máximo[1].

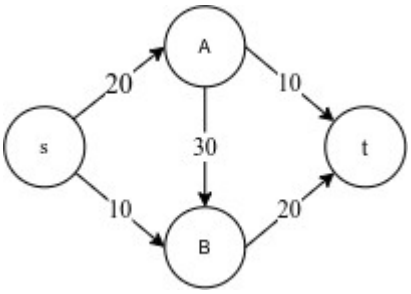

Fig. 2. Rede de fluxo.

A. Ford-Fulkerson

O método de Ford-Fulkerson tem por objetivo encontrar um fluxo máximo para uma rede de fluxos. É chamado de método por englobar diversas implementações com diferentes tempos de execução.

O algoritmo de Ford-Fulkerson é iterativo, começando com $f(u,\ v) = 0$ para todo $u,\ v \in V$ 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 aumentados.

FORD-FULKERSON(G, s, t)
1: **for** $cada\ aresta\ (u,\ v) \in G.E$
2: $\quad (u,\ v).f = 0$
3: **while** $existir\ um\ caminho\ p\ de\ s\ a\ t\ na\ rede\ residual G_f$
4: $\quad c_f(p) = min\{c_f(u,\ v) : (u,\ v)\ está\ em\ p\}$
5: $\quad$ **for** $cada\ aresta\ (u,\ v)\ em\ p$
6: $\quad\quad$ **if** $(u,\ v) \in E$
7: $\quad\quad\quad (u,\ v).f = (u,\ v).f + c_f(p)$
8: $\quad\quad$ **else** $(v,u).f = (v,u).f - c_f(p)$

O caminho encontrado é chamado de caminho

aumentado (Augmenting Path) e com o auxílio do grafo residual do grafo em questão, caminhos que já estão saturados (atingiram sua capacidade máxima) são ou ignorados pelo algoritmo ou possivelmente tem seu fluxo alterado para otimizar o fluxo total.

B. 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.

O diferencial entre os algoritmos de Ford-Fulkerson e Edmond-Karp se dá no método de descoberta do caminho aumentado, pois enquanto Ford-Fulkerson não define um padrão para a busca, Edmond-Karp tenta encontrar o menor caminho com a busca em largura. Tal característica irá influenciar no tempo de execução dos algoritmos.

III. 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.

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].

Dado nossa situação acima, queremos, primeiramente, que o número máximo de funcionários sejam alocados (indicados à alguma tarefa) de forma que o trabalho seja concluído o mais rápido possível.

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.

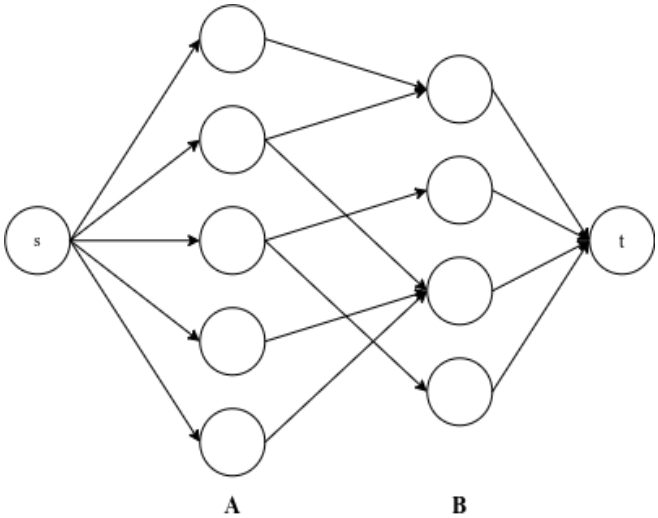

Fig. 3. Rede de fluxo com s e t.

Utilizando o algoritmo de Ford-Fulkerson e Edmond-Karp, o fluxo máximo equivalerá à quantia máxima de funcionários que terão uma tarefa designada.

IV. ANÁLISE DE COMPLEXIDADE

A. Ford-Fulkerson

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 de $O(|E| f^{max})$, onde E é o número de arestas e f^{max} é o fluxo máximo da rede.

Quando as capacidades são números inteiros e o valor de fluxo f^{max} é pequeno, o tempo de execução do algoritmo de Ford-Fulkerson também é pequeno.

B. Edmond-Karp

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

$O\left(|V|\ |E|^2\right)$, onde V é o número de vértices e E é número de arestas do grafo.

V. CONCLUSÃO

Neste trabalho definimos e apresentamos uma solução para o problema de alocação de tarefas, utilizando o algoritmo de Edmonds-Karp. Infelizmente não consegui fazer a experimentação dos algoritmos para uma avaliação mais aprofundada, somente a análise de complexidade.

Os códigos desenvolvidos para o projeto estão presentes no Github, o link está na seção de apêndice.

APÊNDICE

Link para o repositório: https://github.com/IlemSantos/IlemSantos_FinalProject_AA_RR_2022.git

REFERÊNCIAS

[1] Thomas H. Cormem, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Algoritmos: Teoria e Prática. 3a edição. Elsevier, 2012, pp. 579-599.

[2] Judith L. Gersting. Fundamentos Matemáticos para a Ciência da Computação: Matemática Discreta e suas aplicações. 7a edição. LTC, 2012, cap. 6.