Universidade Federal de Roraima
Departamento de Computação
Disciplina: Análise de Algoritmos
Alunos:Joshua Kook Ho Pereira e Rodrigo de Andrade Rolim Bem

PROJETO FINAL A.A. – Network Flow: Task Allocation using Bipartite Graph

Para que o problema de Alocação de Tarefas com Grafos Bipartidos possa ser entendido melhor, devemos ter em mente os seguintes conhecimentos:

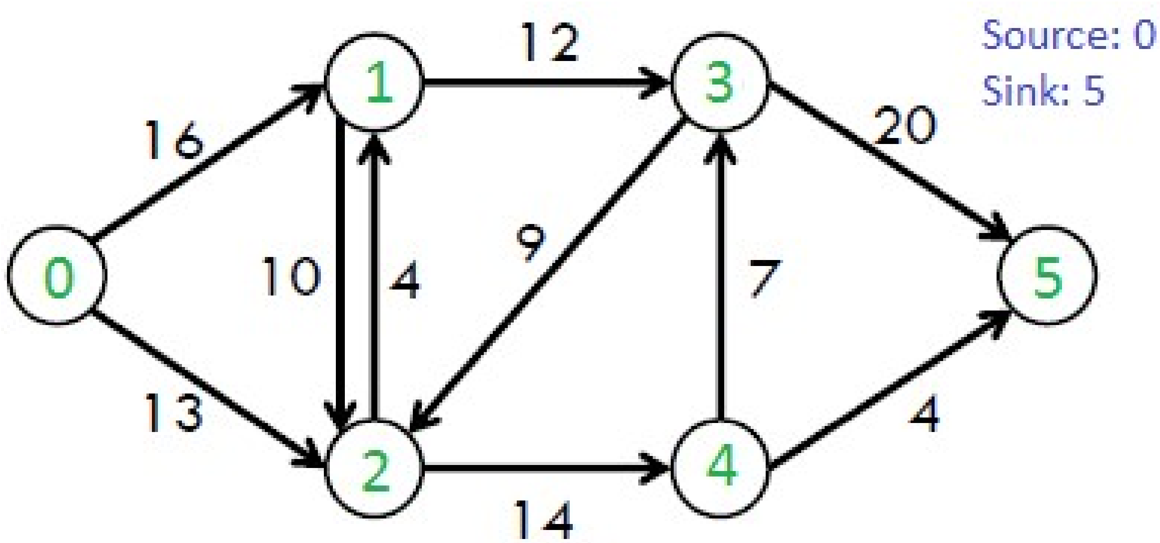
- **Network Flow – Fluxo de Rede**

Um fluxo de rede se refere, em termos simples, à um grafo direcionado cujo os pesos de suas arestas representam a capacidade máxima de fluxo de alguma coisa (definida pelo contexto) que pode passar pela respectiva aresta.

- **Maximum Flow**

Dado um grafo que representa um fluxo de rede, existe um problema de otimização chamado Fluxo Máximo. Este problema consiste em encontrar um fluxo de uma origem (**source S**) para um destino (**sink T**) em uma rede de fluxo, de modo que ele contenha o maior fluxo possível.

É importante considerar que neste problema, todo fluxo que sai de um nodo deve ser equivalente ao fluxo que entra no mesmo, ou seja, um nodo N não pode enviar X unidades de fluxo sendo que ele recebe apenas um valor Y | Y < X.



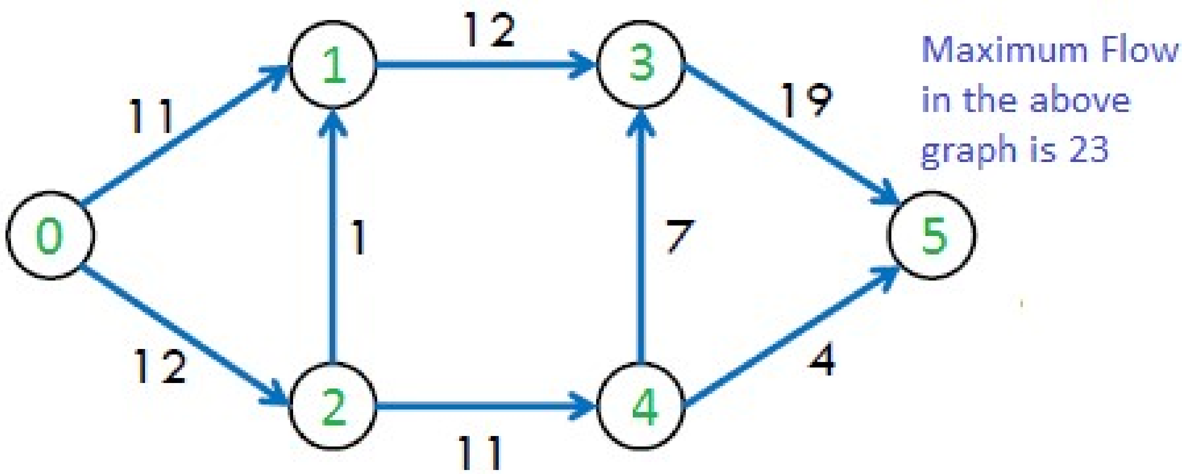
- **Ford-Fullkerson/Edmond-Karp Algorithm**

O algoritmo de Ford-Fulkerson resolve o problema do Fluxo Máximo utilizando Busca em Profundidade (**DFS**) ou Busca em Largura (**BFS**). Caso seja usado **DFS**, o algoritmo será de Ford-Fulkerson, Caso seja usado **BFS**, o algoritmo será de Edmond-Karp.

O algoritmo consiste em copiar o grafo em questão como um grafo residual. A partir deste grafo residual, o algoritmo recursivamente procura se exista algum caminho entre S e T utilizando o DFS. Para cada caminho encontrado, o algoritmo encontra qual a capacidade máxima mínima nas arestas e então a define como o valor do fluxo que passa por aquele caminho.

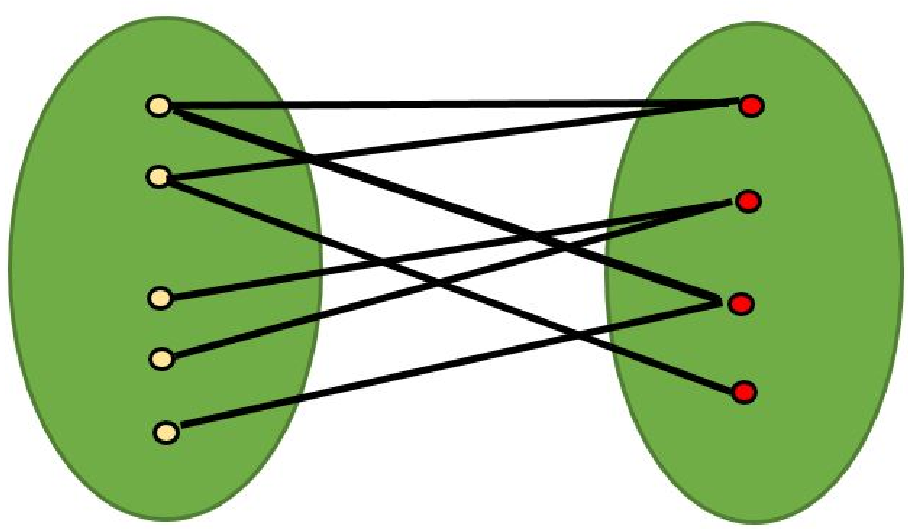
O caminho encontrado é chamado de caminho aumentado (**Augmenting Path**) e com o auxilio do grafo residual do grafo em questão, caminhos que já estão saturados (atingiram sua capacidade máxima) são ou ignorados pelo algoritmo ou possivelmente tem seu fluxo alterado para otimizar o fluxo total.

O algoritmo também é capaz de realocar o fluxo caso necessário (e se possível).



- **Grafo Bipartido**

Um grafo bipartido é um caso específico de grafo onde: dado um grafo V, se este pode ser dividido em 2 **subconjuntos A e B** de tal forma que toda aresta pertencente à V liguem dois nodos de subconjuntos diferentes, ou seja, nenhuma aresta pode ligar dois nodos que pertençam ao mesmo subconjunto.



- **Grafo Residual**

Um grafo residual é uma cópia de um grafo V, porém, com algumas adições: ao encontrar um caminho **S => T** e designar um **fluxo X** a ele, no grafo residual, para cada aresta **U-V** neste caminho, será criada uma aresta **V-U** com peso equivalente ao fluxo que foi alocado na aresta **U-V**. A aresta **U-V** irá então ter seu valor **diminuído em X**.

Assim, quando uma aresta **U-V** estiver saturada (fluxo designado = capacidade máxima), a aresta **U-V** terá valor 0 (o que representará, em uma matriz de adjacência, como se tal aresta não existisse) e logo não será mais considerada pelo algoritmo como um caminho válido.

Dado todos os conceitos acima, podemos elaborar uma solução para nosso problema

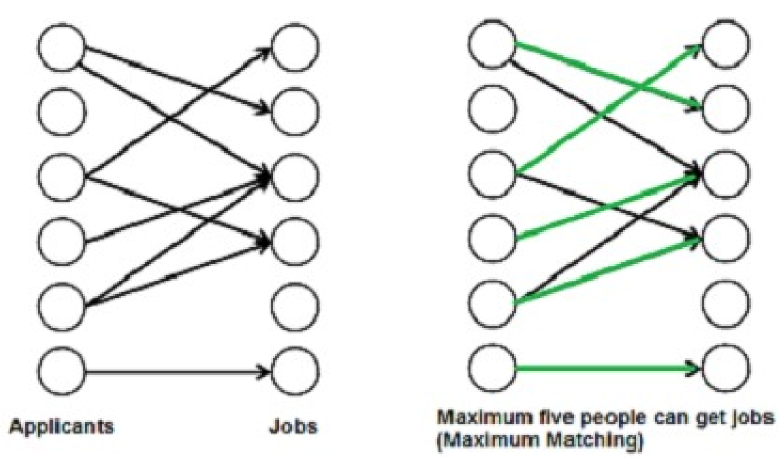
Alocação de Tarefas Utilizando Grafo Bipartido

Primeiramente, consideremos nossa entrada: **M trabalhadores** e **N tarefas**, onde cada trabalhador tem informações de quais tarefas ele pode realizar. Por enquanto, consideremos que cada tarefa só pode ter 1 trabalhador alocado nela.

Podemos modelar esta situação como um grafo bipartido, com **A** sendo o conjunto de trabalhadores e **B** o conjunto de tarefas. Através da utilização de uma matriz de adjacência, caso o trabalhador **i** tenha interesse na tarefa **j**, haverá uma aresta na posição **matriz[i][j].**

1. Pareamento Máximo

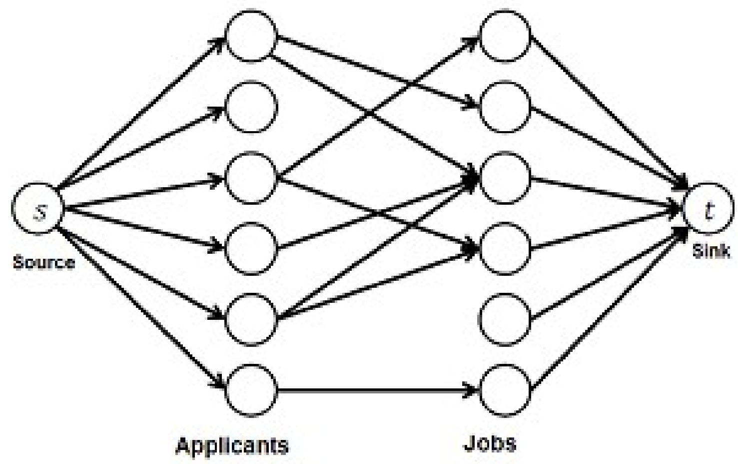
Dado nossa situação acima, queremos, primeiramente, que o número máximo de trabalhadores sejam empregados (indicados à alguma tarefa).

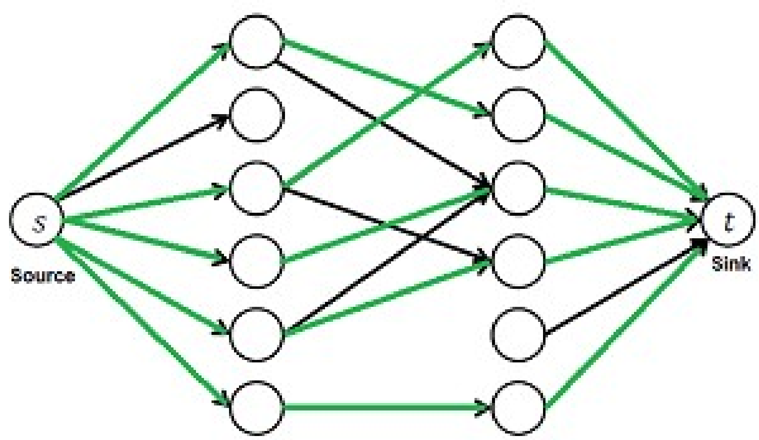


2. Conversão a uma Rede de Fluxo

Nosso problema de pareamento pode ser resolvido convertendo nosso grafo bipartido em uma rede de fluxo, adicionando uma origem S que está ligada à todos os nodos de A, e um destino T ao qual todos os nodos de B estarão ligados. Todas as arestas deste grafo terão valor unitário.

Utilizando o Algoritmo de Ford-Fulkerson, o fluxo máximo equivalerá à quantia máxima de trabalhadores que terão uma tarefa designada.



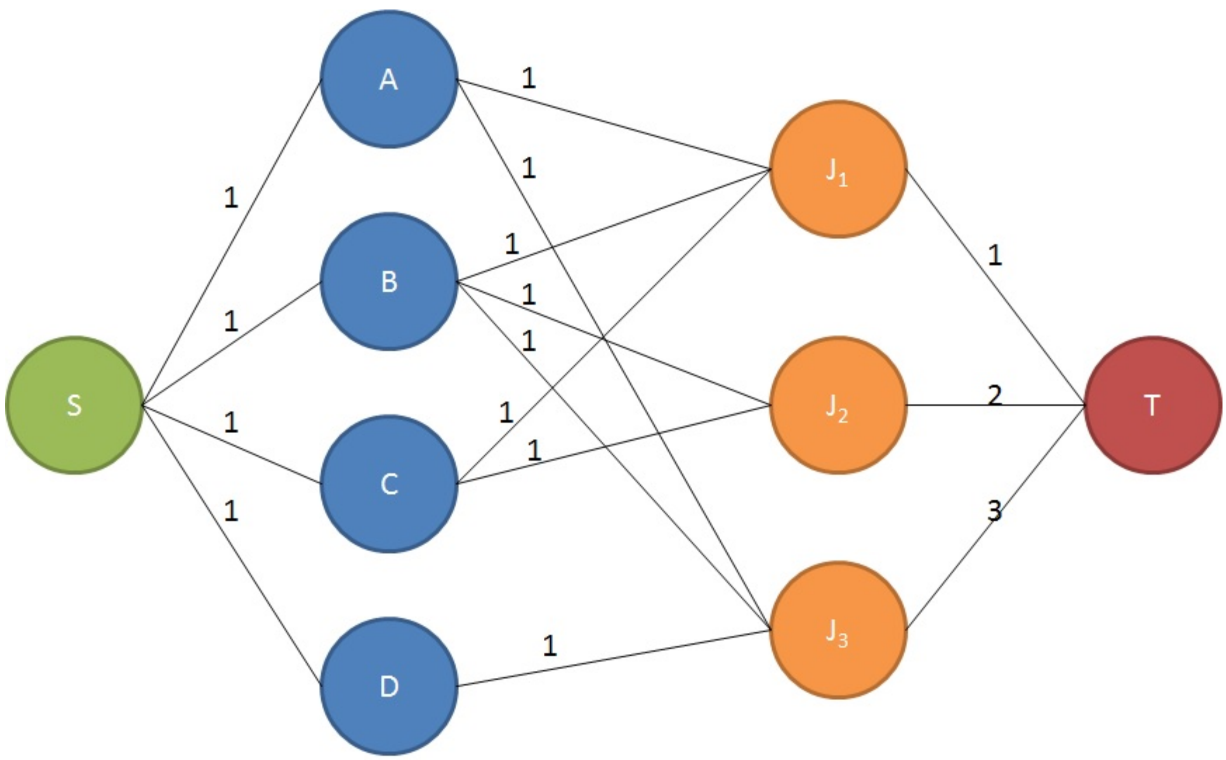


The maximum flow from source to sink is five units. Therefore, maximum five people can get jobs.

3. Considerando Eficiência

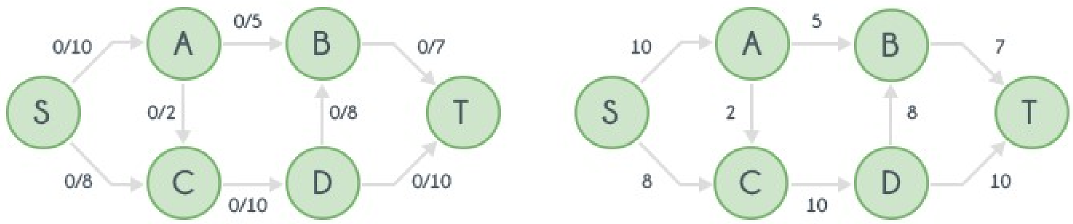
Nosso problema pode ser modelado de forma a considerar eficiência, ou seja, tentaremos fazer com que as tarefas sejam cumpridas mais rapidamente. Para tal, vamos considerar um conceito banal de que se mais de 1 pessoa está realizando uma tarefa, esta será concluída mais rapidamente do que se tivesse apenas 1 pessoa.

Tal problema pode ser facilmente solucionado alterando os pesos das arestas {U-T | U pertence à B}. Assim, caso uma destas arestas tenham um peso maior que 1, significa que ela possibilita que mais de 1 trabalhador seja alocada para ela.



ANEXO – ALGORITMO FORD-FULKERSON

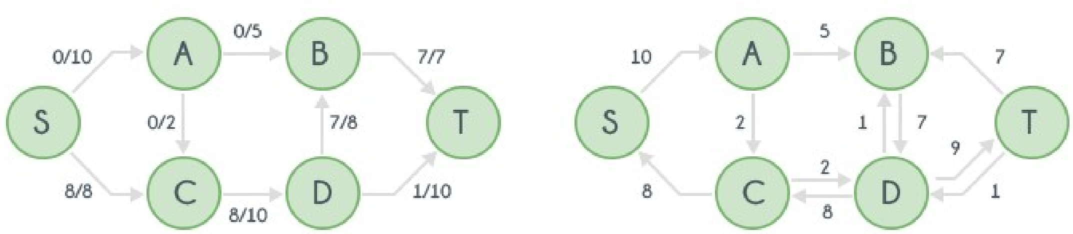
Network (G) Residual Graph (G_R)



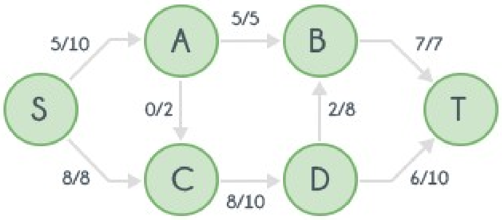
Flow = 0

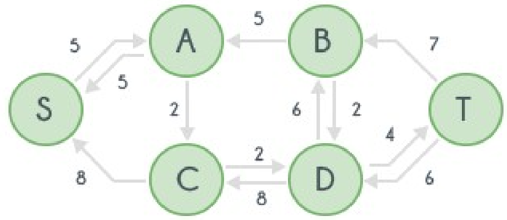
Path 1: S - C - D - B - T ⟶ Flow = Flow + 7

Path 2: S - C - D - T ⟶ Flow = Flow + 1

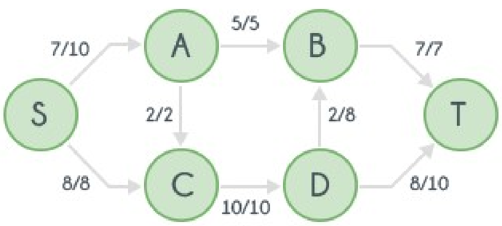


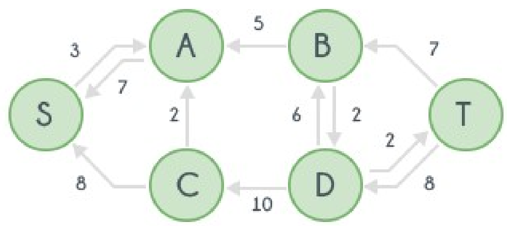
Path 3: S - A - B - T → Flow = Flow + 5





Path 4: S - A - C - D - T → Flow = Flow + 2





No More Paths Left

Max Flow = 15