Trabalho Prático II - Algoritmos I

Augusto Guerra de Lima
2022101086

Departamento de Ciência da Computação; Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais; Primavera de 2024
augustoguerra@dcc.ufmg.br

1 Introdução

Este trabalho aborda um problema algorítmico relacionado a uma rede elétrica de uma siderúrgica, composta por geradores e consumidores de energia. O principal objetivo é computar o fluxo máximo de energia que a rede pode suportar, bem como determinar a quantidade de energia faltante em equipamentos específicos, a energia perdida ao longo do processo e a identificação de conexões críticas, ou seja, aquelas que operam em sua capacidade máxima. Como será demonstrado ao longo deste trabalho, é possível modelar este problema utilizando uma rede de fluxos. Uma vez computado o fluxo máximo por meio do algoritmo de *Ford-Fulkerson*, em particular utilizando o algoritmo de *Edmonds-Karp*, será possível avaliar de maneira eficiente as demais quantidades requisitadas.

Na segunda seção, será apresentada a modelagem do problema. A terceira seção abordará a solução computacional proposta para calcular as respostas desejadas. A quarta seção incluirá uma análise da complexidade assintótica do programa desenvolvido. Por fim, o trabalho será concluído com as considerações finais e as referências utilizadas.

2 Modelagem

Em primeira análise, observa-se que o problema pode ser modelado por meio de uma **rede de fluxo** $G := (V, E)$, um grafo direcionado no qual cada aresta $e = (u, v) \in E$ está associada a uma capacidade $c_{u,v} \geq 0$. No entanto, algumas diferenças em relação a uma rede de fluxo convencional devem ser destacadas. Primeiramente, cada vértice apresenta um consumo, o qual é denotado por k_v. Além disso, o grafo original conta com múltiplas **fontes** (geradores) e múltiplos **sorvedouros** (consumidores).

É possível transformar a instância original em uma instância para a qual o método de *Ford-Fulkerson* pode ser aplicado normalmente [CLRS674]. A ideia consiste em criar uma nova **fonte** s e, para cada uma das fontes da instância original, adicionar uma aresta de capacidade infinita conectando-a à nova fonte. Além disso, deve-se criar um **sorvedouro** t e, para cada consumidor, adicionar uma aresta até o sorvedouro com capacidade igual ao seu consumo.

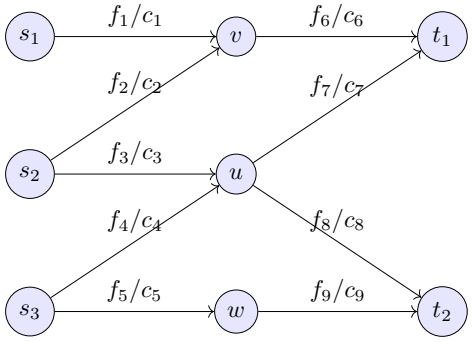

Figura 1: Instância original do problema com múltiplas fontes e sorvedouros.

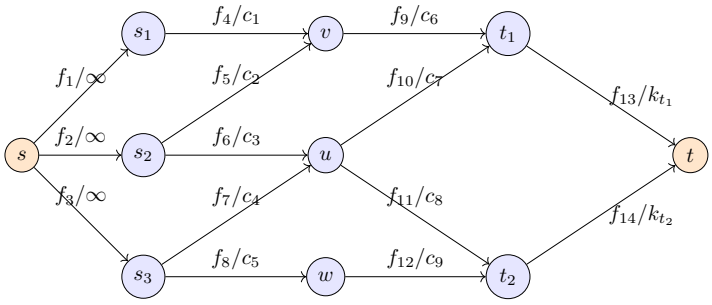

Figura 2: Instância traduzida para o algoritmo com fontes e sorvedouros unificados.

É evidente que, para calcular um dos itens requisitados, é necessário estabelecer uma definição da modelagem matemático-computacional do problema:

Definição 2.1: A **energia máxima** que a rede pode comportar é o fluxo máximo da rede de fluxos, denotada por f^*.

Definição 2.2: A **energia faltante** E_m é a diferença entre a soma do consumo de todos os sorvedouros e do fluxo máximo computado; i.e.

$$E_m = (\sum_{v \in V} k_v) - |f^*|.$$

Definição 2.3: A **energia perdida** E_l é a diferença entre o que pode ser fornecido em potencial pelas fontes e o valor do fluxo máximo; i.e.

$$E_l = (\sum_{e \in \mathcal{E} \subset E} c_e) - |f^*|.$$

Onde $\mathcal{E}$ é o subconjunto de E de arestas que saem de uma fonte.

Definição 2.4: Uma **conexão crítica** é uma aresta tal que o valor de fluxo é igual ao valor de sua capacidade ($f_e = c_e$), ou seja, uma aresta saturada.

3 Solução

Certamente, o ponto central é o cálculo do fluxo máximo da rede, pois, uma vez computado, os demais itens são diretamente deriváveis. A solução computacional, como mencionado anteriormente, baseia-se no algoritmo de *Edmonds-Karp*. Esse algoritmo utiliza uma busca em largura para encontrar caminhos aumentantes na rede residual $G_r := (V, E_r)$, incrementando o fluxo ao longo desses caminhos até que nenhum outro caminho aumentante possa ser encontrado. O processo converge para f^*.

Algorithm 1 Algoritmo de Edmonds-Karp

Entrada: Grafo $G := (V, E)$ com capacidades c_e $\forall e \in E$, fonte s, sorvedouro t
Saída: Fluxo máximo f^* da rede
Inicialize $f_e = 0$ $\forall e \in E$
while existir um caminho aumentante P de s a t em G_e (utilizando busca em largura) **do**
    Determine a capacidade residual $c_e(P) = \min\{c_e(u, v) : (u, v) \in P\}$
    for cada aresta $e = (u, v) \in P$ **do**
        Aumente o fluxo: $f_{(u,v)} \leftarrow f_{(u,v)} + c_e(P)$
        Ajuste o fluxo reverso: $f_{(v,u)} \leftarrow f(v, u) - c_e(P)$
    end for
end while
Retorne $f^* = \sum_{(s,v) \in E} f_{(s,v)}$ (fluxo total que sai de s)

Os demais itens foram computados utilizando exclusivamente as definições mencionadas. Para exibir as conexões críticas em ordem, foi empregada uma fila de prioridade com ordenação personalizada, baseada nas capacidades das arestas. Durante o processamento da entrada, à medida que as fontes e os sorvedouros eram definidos, os valores correspondentes ao consumo total, bem como a listagem completa de todas as fontes, foram pré-computados.

Algorithm 2 Computar energia faltante

Entrada: Grafo $G := (V, E)$ com fluxo máximo f^* computado
Saída: Energia faltante nos consumidores E_m
Retorne $(\sum_{v \in V} k_v) - |f^*|$

Algorithm 3 Computar energia perdida

Entrada: Grafo $G := (V, E)$ com fluxo máximo f^* computado
Saída: Energia perdida na rede E_l
for cada aresta $e = (s_i, v) \in \mathcal{E} \subset E$ **do**
    some o valor da capacidade
end for
Retorne $(\sum_{e \in \mathcal{E} \subset E} c_e) - |f^*|$

Algorithm 4 Listar conexões críticas

Entrada: Grafo $G := (V, E)$ com fluxo máximo f^* computado
Saída: Listagem ordenada as arestas onde $f_e = c_e$
Inicie uma fila de prioridade $Q = \emptyset$
for $\forall e \in E$ **do**
    if $f_e = c_e$ **then** $Q \leftarrow Q \cup \{e\}$
    end if
end for
Realize um *Heapsort* em Q
Liste as conexões críticas

4 Análise de complexidade

4.1 Edmonds-Karp

Proposição 4.1: A complexidade do *tempo de execução* do algoritmo de *Edmonds-Karp* pertence a $\mathbf{O}(|V| \cdot |E|^2)$.

Prova: Cada iteração do algoritmo envolve a execução de uma busca em largura para encontrar um caminho aumentante na rede residual. A complexidade da busca em largura é bem conhecida e pertence ao conjunto $\mathbf{O}(|V| + |E|)$.

O número total de iterações, por sua vez, está relacionado ao número de caminhos aumentantes encontrados durante a execução do algoritmo. Em *Edmonds-Karp*, cada caminho aumentante melhora o fluxo em pelo menos uma unidade, saturando no máximo uma aresta em cada execução. Como cada aresta pode ser saturada no máximo $|V|$ vezes, o número total de iterações do algoritmo é limitado superiormente por $\mathbf{O}(|V| \cdot |E|)$.

Somando esses dois fatores, temos que o custo total do algoritmo é dado pelo produto do número de iterações pelo custo de cada iteração. Assim, a complexidade de tempo do algoritmo é $\mathbf{O}(|V| \cdot |E| \cdot (|V| + |E|))$. Para grafos densos, onde $|E| \approx |V|^2$, $\mathbf{O}(|V| \cdot |E|^2)$. ■

4.2 Energias faltante e perdida

Proposição 4.2: A complexidade do *tempo de execução* para computar as energias faltante e perdida pertencem respectivamente a $\mathbf{O}(1)$ e $\mathbf{O}(|\mathcal{E}|)$.

Prova: Para o cálculo da energia faltante, o consumo total foi previamente computado durante o processamento da entrada, assim como o fluxo máximo. Portanto, apenas uma operação adicional é necessária para obter esse valor. Por outro lado, o cálculo da energia perdida requer uma iteração sobre o conjunto de arestas saturadas $\mathcal{E}$. ■

4.3 Conexões críticas

Proposição 4.3: A complexidade do *tempo de execução* para listas as conexões críticas está em $\mathbf{O}(|E| + |\mathcal{C}| \cdot \lg(|\mathcal{C}|))$, onde $\mathcal{C}$ é o conjunto de arestas saturadas.

Prova: Primeiramente, itera-se por todas as arestas do grafo, o que está em $\mathbf{O}(|E|)$. Seja $\mathcal{C} \subseteq E$ o subconjunto de arestas saturadas; é bem verdade que realizar uma ordenação utilizando um *heap* pertence a $\mathbf{O}(|\mathcal{C}| \cdot \lg(|\mathcal{C}|))$. Assim, a complexidade total do procedimento é dada por $\mathbf{O}(|E| + |\mathcal{C}| \cdot \lg(|\mathcal{C}|))$. ■

4.4 Complexidade espacial

Proposição 4.4: A complexidade de *espaço* do programa implementado pertence ao conjunto $\mathbf{O}(|V| + |E|)$.

Prova: A estrutura de dados mais relevante utilizada é a lista de adjacência, empregada para representar a rede de fluxo. Essa estrutura requer um espaço da ordem de $\mathbf{O}(|V| + |E|)$. Além disso, outras estruturas, como a lista de fontes e a fila de prioridade, também são utilizadas, mas possuem uma ordem de complexidade inferior. ■

5 Considerações finais

O estudo e implementação de redes de fluxo, como realizado neste trabalho, destacam a importância desses conceitos na resolução de problemas práticos, como o gerenciamento de redes elétricas. A abordagem apresentada demonstrou como problemas reais podem ser modelados matematicamente e solucionados de forma eficiente por meio de algoritmos consagrados, como o de *Edmonds-Karp*.

Foi especialmente importante observar como o algoritmo converge para a solução do fluxo máximo, que, conforme discutido em aula, é equivalente ao corte mínimo do grafo. Essa equivalência abre espaço para discussões mais amplas, como a relação entre problemas algorítmicos e a possibilidade de redutibilidade entre eles. Esses tópicos, amplamente abordados na última parte da disciplina de *Algoritmos I*, ressaltam a importância de compreender como problemas distintos podem ser transformados e resolvidos utilizando abordagens comuns, fortalecendo a base teórica e prática no estudo de algoritmos.

6 Referências

[1] CORMEN, Thomas H. et al. *"Introduction to Algorithms"*. 3rd ed. Cambridge, Massachusetts: MIT Press, 2009.

[2] KLEINBERG, Jon; TARDOS, Éva. *"Algorithm Design"*. 1st ed. Boston: Pearson, 2005.