

CAPÍTULO I

I.1) INTRODUÇÃO: (Problema de Fluxo de Custo Mínimo)

Os problemas de fluxo em rede são, inquestionavelmente de suma importância em vários setores da investigação científica. Podemos citar por exemplo, aplicações em ciência da computação, engenharia, pesquisa operacional, matemática aplicada, gerenciamento, entre outras. Abordamos aqui os problemas de fluxo de custo mínimo, ou seja, procuramos enviar um fluxo constante em uma rede G com n nós e m arestas ao menor custo possível.

Discutiremos nesse trabalho alguns dos algoritmos de fluxo de custo mínimo, fazendo, paralelamente a isso, uma análise da "eficiência" computacional destes algoritmos. Para tanto, abordaremos inicialmente sobre a complexidade de algoritmos onde enfatizamos a análise de pior caso. Utilizaremos um modelo teórico de computador denominado RAM (Random Access Machine). Na verdade este modelo se equivale a outros modelos teóricos como máquinas RASP ou máquinas de TURING.

Abordamos também o problema de caminho mínimo (capítulo II), utilizado frequentemente como subproblema do problema de fluxo de custo mínimo. Estudamos os algoritmos de *correção de rótulos* (label correcting) mais apropriados para redes com ciclo de custo negativo. Outros métodos mais específicos podem ser utilizados dependendo dos dados do problema. Caso não tenhamos a presença de ciclos de custo negativo, algoritmos de *atualização de rótulos* (label setting) poderiam ser utilizados. No capítulo II não estaremos preocupados com aspectos especiais de implementação dos algoritmos de caminho mínimo, faremos apenas uma abordagem genérica.

Estudaremos posteriormente (no capítulo III), as condições de otimalidade dos problemas de fluxo de custo mínimo. Como veremos, estas condições irão sugerir metodologias para elaboração de algoritmos além de serem úteis como critério de parada. As condições de otimalidade servem como um "certificado" atestando a otimalidade ou não de uma solução.

Apresentamos posteriormente alguns dos algoritmos de fluxo de custo mínimo (capítulos IV e V). Estudaremos os algoritmos de ciclos de custo negativo, algoritmo de caminhos mínimos sucessivos e o algoritmo com escala de capacidade, sendo os dois primeiros pseudo-polinomiais e o último polinomial.

I.2) ANÁLISE DE COMPLEXIDADE: (Análise de Pior Caso)

O tempo de execução de um algoritmo depende basicamente da "natureza" e "tamanho" dos dados. Problemas maiores exigem mais tempo de processamento, e diferentes dados de mesmo tamanho, frequentemente requerem tempos distintos de processamento.

A *função de complexidade* (de pior caso) de um algoritmo é uma função definida no "tamanho" do problema (definido na próxima seção) e que retorna o maior tempo necessário pelo algoritmo na solução de qualquer instância do problema para este tamanho. Em outras palavras, a função de complexidade mede a velocidade de processamento do algoritmo (no pior caso quando o "tamanho" do problema aumenta indefinidamente. Por exemplo, se o tempo necessário na resolução de um algoritmo de fluxo em redes for cnm para $c \geq 0$ (onde n é igual ao número de nós e m o número de arestas) o tempo necessário para se resolver qualquer instância com n nós e m arestas (tamanho do problema) será no máximo cnm . Chamamos $T(n,m) = cnm$ (onde $c \geq 0$), de *função de complexidade local*.

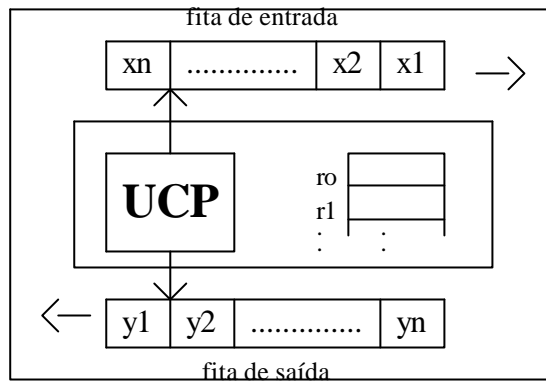
Em nosso trabalho faremos uso frequente do conceito de *complexidade assintótica*, como definido abaixo:

Definição I.1: Um algoritmo tem *complexidade assintótica* $O(f(n))$ se para alguma constante c e n_0 , o tempo necessário pelo algoritmo será no máximo $cf(n) \forall n \geq n_0$. Em outras palavras, se $T(n)$ é a complexidade local de um algoritmo, dizemos que $T(n)$ é de ordem $O(f(n))$ se e somente se $\exists c, n_0$ tal que $T(n) \leq cf(n) \forall n \geq n_0$. •

A complexidade assintótica de um algoritmo é um limite superior para o tempo de processamento para valores suficientemente grandes de n .

I.2.1) Tamanho de um problema : (Máquina RAM)

Para definir de forma mais precisa o tamanho de um problema consideremos inicialmente o seguinte modelo matemático de computador conhecido como RAM (Random Access Machine). Trata-se de um modelo hipotético de máquina constituído de duas unidades de fita, uma unidade de controle e processamento e uma memória. Em uma das fitas só a operação de leitura é permitida, enquanto que na outra só é permitida a operação de leitura:



FiguraI.1: Modelo de uma máquina RAM

Cada unidade de fita (assim como a memória) é formada por uma sequência de células cada uma delas podendo armazenar um número inteiro de tamanho arbitrário. Não há limite para o número de células na memória. Esta abstração é válida quando:

- ### o tamanho do problema for suficientemente pequeno para ser armazenado na memória principal de um computador.
- ### os inteiros utilizados nos cálculos são suficientemente pequenos para serem armazenados em uma única palavra do computador.

Para determinarmos a complexidade em tempo (ou espaço) devemos especificar o tempo necessário para executar cada instrução e o espaço utilizado por cada registro. São 2 os critérios usuais:

- Critério de Custo Logaritmo
- Critério de Custo Uniforme

O *critério de custo uniforme* assume que cada instrução RAM consome uma unidade de tempo e cada célula uma unidade de espaço. Se definirmos "passo" de um algoritmo como sendo a execução de cada instrução que o implementa, a complexidade de tempo é o maior número de passos necessários para a computação completa de qualquer problema para o qual se aplicar o algoritmo.

No *critério de custo logaritmico*, levaremos em conta o fato de que para representar um inteiro ### em uma célula necessitaremos $w(\varphi)$ bits, onde:

$$w(\mathbf{j}) = \begin{cases} \lceil \log |\mathbf{j}| \rceil + 1, & \text{se } \mathbf{j} \neq 0 \\ 1 & \text{se } \mathbf{j} = 0 \end{cases}$$

Portanto, no critério logaritmico, faz-se a hipótese de que o tempo e espaço necessários para executar uma instrução são proporcionais ao "comprimento" ou "tamanho" dos dados (número de bits necessários para codificar os dados).

Suponha que queiramos discutir o tamanho de um inteiro x . Podemos analisar segundo as duas abordagens acima: (1) supor que o tamanho do dado seja x ou (2), assumir que o tamanho de x seja aproximadamente $\log x$. A segunda forma é mais conveniente pois reflete de forma mais exata o funcionamento do computador. Assim, o número de operações executadas pela máquina (tempo de processamento) será proporcional ao número de bits dos dados envolvidos. A representação binária de x requer $\log x$ bits, e portanto o espaço requerido para se armazenar x é proporcional a $\log x$. É importante lembrar que a complexidade assintótica calculada em ambos os critérios difere apenas de uma constante sendo portanto equivalentes.

O tamanho de um problema de fluxo em rede será função de como os dados são armazenados no computador. Por exemplo, se utilizarmos a representação de lista de adjacência (estrutura de dados mais conveniente p/ o problema de fluxo) teremos p/ o grafo G abaixo a seguinte configuração:

c_{ij} = custo u_{ij} = capacidade

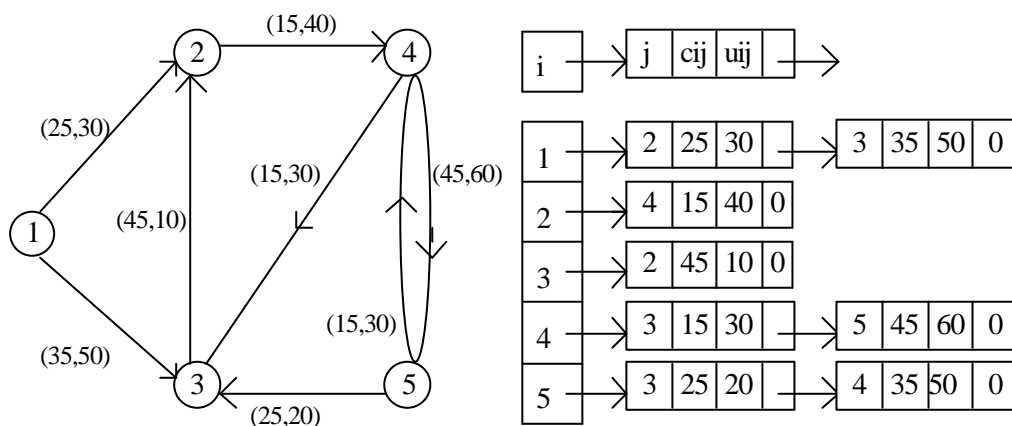


Figura I.2: Lista de Adjacência

Caso utilizemos o critério de custo uniforme teremos que armazenar n nós (cada um ocupando uma unidade de espaço), m arestas, m custos e m capacidades. Assim $n+3m$ células seriam necessárias.

No critério de custo logaritmico o tamanho do problema será dado pelo número de bits necessários para se armazenar sua lista de adjacência. Como a lista de adjacência armazena um ponteiro para cada nó e arco, além de custos e capacidades associados a estes arcos, teremos

aproximadamente $n \log n + n \log m + m \log C + m \log U$ bits para armazenar todos os dados do problema de fluxo de custo mínimo (verificação trivial), onde C representa o maior dos custos envolvidos e U a maior capacidade dos arcos envolvidos na rede. Observe que estamos fazendo uma majoração quando supomos que cada um dos custos e capacidades envolvidos terão $\log C$ e $\log U$ bits respectivamente.

I.2.2: Algoritmos Polinomiais e Exponenciais:

Em princípio poderíamos expressar o tempo de processamento de um algoritmo como função do tamanho do problema (critério de custo logaritmico), isto entretanto seria mais trabalhoso e desnecessário. Com o objetivo de simplificar a idéia acima expressaremos o tempo de processamento equivalentemente como função dos parâmetros n , m , $\log C$ e $\log U$.

Consideramos "bom", um algoritmo de fluxo, se sua complexidade de pior caso for limitada por uma função polinomial nos parâmetros do problema (ou seja: n , m , $\log C$ e $\log U$). Este algoritmo será chamado *polinomial*. Como exemplo temos $O(n^2)$, $O(n + m \log C)$, $O(nm \log(n^2 / m))$, etc.

Um algoritmo tem complexidade assintótica *exponencial* se sua função de complexidade não for limitada por um polinômio no tamanho do problema. Por exemplo: $O(2^n)$, $O(n^{###})$, $O(nC)$ são limites exponenciais.

Dizemos que um algoritmo tem complexidade *pseudo-polinomial* se a função de complexidade for definida por um polinômio nos parâmetros n, m, C e U . Note que podemos ter $C = O(2^n)$ e $U = O(2^m)$, desta forma $O(nC)$, $O(mU)$ são limites pseudo-polinomiais.

Um algoritmo será *fortemente polinomial* se sua complexidade depender apenas dos parâmetros n e m , e *fracamente polinomial* se depender de m , n , $\log C$ e $\log U$. A vantagem de termos um algoritmo fortemente polinomial é que ele poderá trabalhar também com dados arbitrariamente grandes para o custo e a capacidade!

Importante! Na discussão acima dissemos que um algoritmo de fluxo tem tempo polinomial se for limitado por um polinômio nos parâmetros n , m , $\log C$ e $\log U$. Na seção anterior dissemos que a complexidade computacional de um algoritmo era definida no tamanho do problema (no caso de fluxo: $n \log n + m \log m + n \log C + m \log U$ bits). Desta forma um algoritmo de fluxo será dito polinomial, se for limitado por um polinômio no tamanho do problema (por exemplo: $(n \log n + m \log m + n \log C + m \log U)^2$). Entretanto, é fácil ver que o tempo de processamento de um algoritmo de fluxo é limitado polinomialmente no tamanho do problema (número de bits necessários para se armazenar os dados do problema) se e somente se ele for limitado polinomialmente nos parâmetros do problema. De fato, se o tempo de processamento de

um algoritmo de fluxo for limitado por n^2 ele será estritamente limitado por $(n \log n + m \log m + n \log C + m \log U)^2$, reciprocamente se um algoritmo de fluxo for limitado por $(n \log n + m \log m + n \log C + m \log U)^2$ ele será limitado por $(n^2 + m^2 + n \log C + m \log U)^2$, polinômio nos parâmetros n , m , $\log C$ e $\log U$.