DCC206 – Algoritmos 1

Aula 09 – Fluxo em redes – Parte 2

Professor Renato Vimieiro

DCC/ICEx/UFMG

Teorema Max-Flow Min-Cut

- Discutimos anteriormente de forma intuitiva que o fluxo máximo é limitado por um corte s-t no grafo
- Discutiremos esse assunto mais formalmente nessa aula
- Um corte s-t é uma partição dos vértices de um grafo em dois conjuntos A e B tal que $s \in A$ e $t \in B$
- Definimos a capacidade de um corte (A,B) como
  - $c(A, B) = \sum_{u \in A, v \in B} c(u, v)$

Teorema Max-Flow Min-Cut

- **Lema 1:** Seja f um fluxo numa rede G=(V,E) e (A,B) um corte s-t dessa rede. O valor do fluxo $F \; = f^{-}(s) = f^{-}(A) - f^{+}(A)$.
- **Prova:** Intuitivamente essa propriedade diz que o fluxo que chega a t é o que sai de A menos o fluxo que retorna a A.
- Por definição, $f^{+}(s) = 0$. Logo, $\mathrm{F} = f^{-}(s) - f^{+}(s)$.
- Por conservação de fluxo, $f^{-}(v) - f^{+}(v) = 0$ para todo v diferente de s e t
- Portanto, $F = \; \sum_{v \in A} f^{-}(v) - f^{+}(v) = f^{-}(A) - f^{+}(A)$

Teorema Max-Flow Min-Cut

- **Lema 2:** Seja f um fluxo e (A,B) um corte s-t arbitrário. Então, $F = \ f^{-}(s) \leq c(A,B)$.
- **Prova**:
- $F = f^{-}(s) = f^{-}(A) - f^{+}(A)$ (Lema 1)
- $f^{-}(A) - f^{+}(A) \leq f^{-}(A) = \ \sum_{v \in A} f^{-}(v) \ \leq \sum_{v \in A, u \in B} f(v,u) \leq \sum_{v \in A, u \in B} c(v,u) = c(A,B)$ ■

Teorema Max-Flow Min-Cut

- Assim, é provado que, se descobrirmos o valor de um corte (A,B), então o valor do fluxo máximo não é superior à capacidade desse corte
- Analogamente, se encontrarmos um fluxo de valor F, não existe nenhum corte no grafo com capacidade inferior a F
- Isso mostra o quanto esses dois valores estão fortemente relacionados
- Vamos demonstrar como eles estão amarrados no próximo teorema

Teorema Max-Flow Min-Cut

- **Teorema 1**: Se f é um fluxo numa rede G=(V,E) tal que não existe caminho aumentante no grafo residual, então existe um corte s-t (A*,B*) tal que o valor do fluxo F=c(A*,B*). Consequentemente, f é máximo e (A*,B*) é mínimo.
- **Prova:**
- Suponha que f seja um fluxo em que não existe caminho aumentante no grafo residual.
- Seja A* um conjunto de vértices alcançáveis a partir da origem s
- Seja B* = V-A*

Teorema Max-Flow Min-Cut

- Claramente, (A*,B*) é uma partição do conjunto de vértices. Além disso, $s \in A^*$e $t \in B^*$ pois sempre há caminho de s para s, e não existe caminho aumentante no grafo residual.
- Seja (u,v) uma aresta tal que $u \in A^*$ e $v \in B^*$. O fluxo $f(u,v) = c(u,v)$, caso contrário (u,v) seria uma aresta pra frente em G_f e v seria alcançável a partir de s, contradizendo o fato de que $v \in B^*$.
- Seja (u',v') uma aresta tal que $u' \in B^*$e $v' \in A^*$. O fluxo $f(u',v') = 0$, caso contrário a aresta (v',u') seria uma aresta para trás em G_f. Como $v' \in A^*$, existiria um caminho desde s até u'.
- Portanto, as arestas saindo de A* estão saturadas e as entrando em A* não são usadas

Teorema Max-Flow Min-Cut

- Logo, pelo Lema 1, $F = f^{-}(A) - f^{+}(A) = \sum_{u \in A^*, v \in B^*} c(u, v) - 0 = c(A^*, B^*)$
- Como o algoritmo de Ford-Fulkerson termina quando não existe mais caminho aumentante, percebemos que ele é ótimo.
- <u>O teorema também revela que o algoritmo pode ser usado para computar um corte mínimo no grafo</u>
  - Basta realizar uma busca em largura ou profundidade no grafo residual final

Teorema Max-Flow Min-Cut

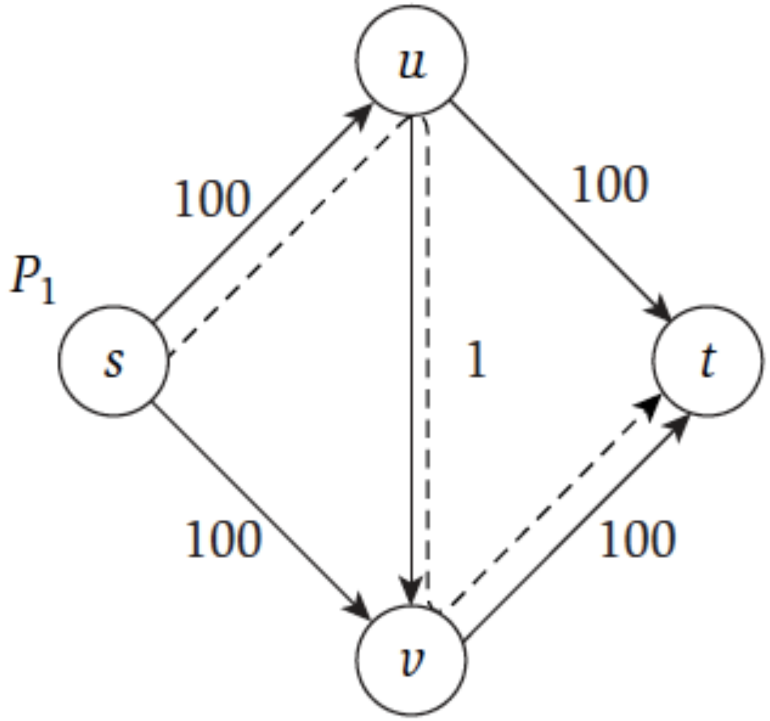
- O Teorema 1 é extremamente relevante para nossa análise.
- Ele mostra que, em todo fluxo em rede, existe um fluxo F e um corte (A,B) tal que F=c(A,B)
- Veja que F tem que ser máximo, senão o fluxo F’ > F infringiria o Lema 2
- Analogamente, c(A,B) tem que ser mínimo, caso contrário F contradiria o Lema 2
- Essas implicações são refraseadas no teorema conhecido como Max-Flow Min-Cut

Teorema Max-Flow Min-Cut

- **Teorema (Max-Flow Min-Cut)**: Em todo fluxo em redes, o valor do fluxo máximo é igual ao valor da menor capacidade de um corte s-t

Recapitulando: desempenho Ford-Fulkerson

- Como analisado anteriormente, o tempo de execução do algoritmo de Ford-Fulkerson é dependente da escolha do caminho aumentante
- Considere o exemplo



Recapitulando: desempenho Ford-Fulkerson

- Se, após a inicialização, o caminho aumentante escolhido fosse s-u-v-t, então o fluxo seria 1
- Em seguida, se escolhêssemos o caminho s-v-u-t, novamente o gargalo seria 1 e o fluxo aumentaria para 2
- O algoritmo poderia alternar entre o primeiro e segundo caminhos para chegar ao fluxo máximo 200, após 200 iterações.
  - Esse mesmo fluxo poderia ser obtido em duas iterações, escolhendo s-u-t e s-v-t, por exemplo.
- De fato, como $(F = \ f^{-}(s)) \leq (\sum c(s,u) = C)$, teremos que realizar O(C) iterações. No entanto, podemos ter C = $2^{f(V)}$. Resultando em um custo bastante alto.
- Na verdade, o problema é como escolher um bom caminho aumentante

Algoritmo Capacity-Scaling

- Uma escolha natural para um bom caminho aumentante é eleger aquele com o maior gargalo
- Contudo, o tempo necessário para escolher tal caminho faria com que o tempo gasto em cada iteração fosse excessivo
- A ideia do algoritmo capacity-scaling é buscar caminhos aumentantes com gargalos, no mínimo, iguais a um parâmetro Δ
  - Isso reduz o custo de se encontrar um caminho exatamente com o maior gargalo, preservando a qualidade de se tentar buscar por caminhos mais efetivos
- Para isso, busca-se um caminho aumentante num subconjunto do grafo residual composto somente das arestas com capacidade maior ou igual a Δ

Algoritmo Capacity-Scaling

- O algoritmo inicia com $\Delta = 2^{\lfloor \lg \max c(s,u) \rfloor}$
- Após atualizar os fluxos com bases em gargalos com tamanho mínimo Δ, o algoritmo reduz o parâmetro pela metade e reinicia o processo
- O ciclo é repetido enquanto $\Delta \geq 1$

Algoritmo Capacity-Scaling

```
Scaling Max-Flow(G,s,t)
   para (u, v) ∈ E
        f[u,v] = 0
   Δ = 2^⌊lg max c(s,u)⌋
   G_f(Δ) = grafoResidual(G,f)
   enquanto Δ > 0
        enquanto existir caminho aumentante em G_f(Δ)
             P = caminho(s,t,G_f(Δ))
             f' = aumentarFluxo(f,P)
             f = f'
             G_f(Δ) = grafoResidual(G,f)
         Δ = Δ/2
   retornar f
```

Análise do algoritmo Capacity-Scaling

- É fácil perceber que o algoritmo termina com o máximo fluxo f
  - $\Delta = 1$ na última iteração, logo o algoritmo é idêntico a Ford-Fulkerson 'tradicional'
- O laço externo controlando o parâmetro de escala é executado no máximo O(lg C)
  - $\sum c(s, u) = C$

Análise do algoritmo Capacity-Scaling

- **Lema 3**: Seja f o fluxo obtido ao fim de uma iteração Δ. Seja (A,B) um corte tal que A contenha os vértices alcançáveis a partir de s em $G_f(\Delta)$ e B os demais. Então, para toda aresta (u,v) tal que $u \in A$ e $v \in B$, $c(u,v) < f(u,v) + \Delta$, e, para toda aresta (v,u), $f(v,u) < \Delta$.
- **Prova**:
- Seja (u,v) a aresta em questão. Suponha que $c(u,v) \geq f(u,v) + \Delta$. Como o gargalo dessa iteração é Δ, existiria uma aresta para frente (u,v) em $G_f(\Delta)$. Dessa forma, v seria alcançável a partir de s. Contradição.
- Seja (v,u) a aresta em questão. Analogamente, se $f(v,u) \geq \Delta$, então existiria uma aresta para trás (u,v) em $G_f(\Delta)$ com capacidade maior que Δ. Logo, v seria alcançável a partir de s. Contradição.

Análise do algoritmo Capacity-Scaling

- **Teorema 3**: Seja f o fluxo obtido ao fim de uma iteração Δ e m=|E|. Existe um corte s-t em G tal que c(A,B) ≤ F + m Δ. Consequentemente, F* ≤ F +m Δ
- **Prova**: Mostraremos que existe um corte que satisfaz tal propriedade.
- Seja (A,B) um corte tal que A contenha os vértices alcançáveis a partir de s em $G_f(\Delta)$ e B os demais. Obviamente, s pertence a A e t pertence a B
- Sabemos que $F = f^{-}(A) - f^{+}(A) \geq \sum_{u \in A, v \in B} (c(u,v) - \Delta) - \sum \Delta \geq c(A,B) - m\Delta$

Análise do algoritmo Capacity-Scaling

- **Teorema 4**: O número de caminhos aumentantes a cada iteração Δ é, no máximo, 2m
- **Prova**:
- Na primeira iteração o teorema é verdadeiro pois qualquer aresta em G_f potencialmente pode ser usada em um caminho simples.
- Considere uma iteração Δ arbitrária e um fluxo fp obtido na iteração anterior. Na iteração anterior, $\Delta' = 2\Delta$. Pelo teorema anterior, sabemos que $F^* \leq Fp + m\,\Delta' = Fp + 2m\,\Delta$
- Como cada caminho aumenta o fluxo em, no mínimo, Δ, temos, no máximo, 2m caminhos

Análise do algoritmo Capacity-Scaling

- Os teoremas anteriores permitem concluir que o custo total do algoritmo é $O(|E|^2 \lg C)$
- O algoritmo original é proporcional às capacidades das arestas, enquanto a modificação é polinomial em função do número de arestas e bits necessários para especificar as capacidades
  - Ele roda em tempo polinomial no tamanho da entrada
- Embora ainda possamos melhorar o custo, esse algoritmo representa uma melhoria significativa do original
  - Se o fluxo máximo fosse 2^{100}, o original poderia levar até 2^{100} iterações para terminar, enquanto o modificado apenas 100.

Algoritmo de Edmonds-Karp

- O algoritmo de Edmonds-Karp é uma modificação do algoritmo de Ford-Fulkerson
- O caminho é recuperado através de uma modificação da busca em largura, ou seja, ele é o menor caminho aumentante (em número de arestas)
- Os caminhos podem ter tamanho $1 \leq k < |V|$
- Cada aresta pode participar de um caminho de tamanho k. Logo, existem no máximo O(VE) caminhos aumentantes
- Essa modificação faz com que o custo seja $O(VE^2)$

Leitura

- Seções 7.2 e 7.3 (Kleinberg e Tardos)