Problema do Fluxo Máximo

Aulas 21 e 22

Conteúdo

Conteúdo.

- Problema do Fluxo Máximo.
- Problema do Corte Mínimo.
- Teorema Max-fluxo min-corte
- Algoritmo de caminho aumentante.
- capacidade-escala.
- Caminho aumentante Mínimo.

Fluxo Máximo e Corte Mínimo

Fluxo Máximo e Corte Mínimo.

- Dois riquíssimos problemas algoritmicos
- Problema clássico em otimização combinatória.
- Dualidade matemática.

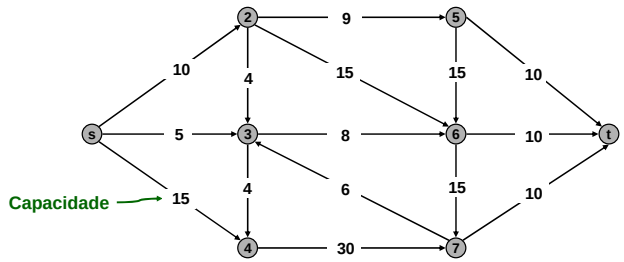
Aplicações / reduções.

- Conectividade em redes.
- Emparelhamento bipartido.
- Data mining.
- Open-pit mining.
- Escalonamento de vôos.
- Processamento de imagens.
- rede reliability.
- Segurança de dados estatísticos.
- Computação Distribuída.
- Egalitarian stable matching.
- Muito mais . . .

Rede de Fluxo Máximo

Rede de Fluxo Máximo: G = (V, E, s, t, u) .

- (V, E) = Grafo dirigido, sem arcos paralelos
- Dois nós distintos: s = source, t = sink.
- u(e) = capacidade do arco e.

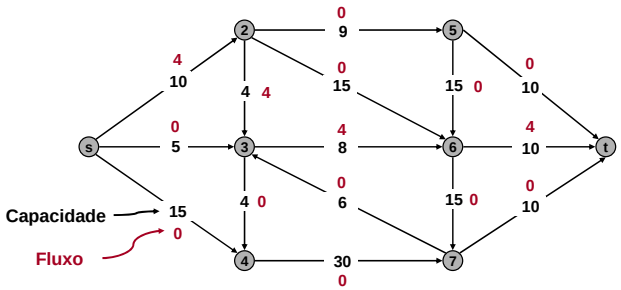


Fluxos

s-t fluxo é uma função f: E → ℜ satisfazendo:

- Para cada e ∈ E: $0 \le f(e) \le u(e)$ (capacidade)
- Para cada v ∈ V – {s, t}: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservação)

$$\sum_{e \text{ in to } v} f(e) := \sum_{w:(w,v)\in E} f(w,v) \qquad \sum_{e \text{ out of } v} f(e) := \sum_{w:(v,w)\in E} f(v,w)$$



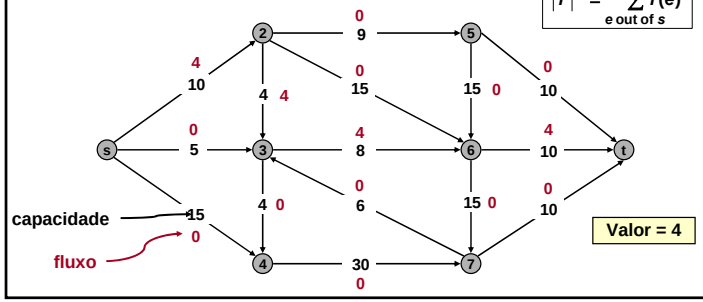
Fluxos

s-t fluxo é uma função f: E → ℜ satisfazendo:

- para cada e ∈ E: $0 \le f(e) \le u(e)$ (capacidade)
- para cada v ∈ V – {s, t}: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservação)

Fluxo Máximo: encontrar um s-t fluxo que maximiza o fluxo que sai da fonte.

$$|f| = \sum_{e \text{ out of } s} f(e)$$

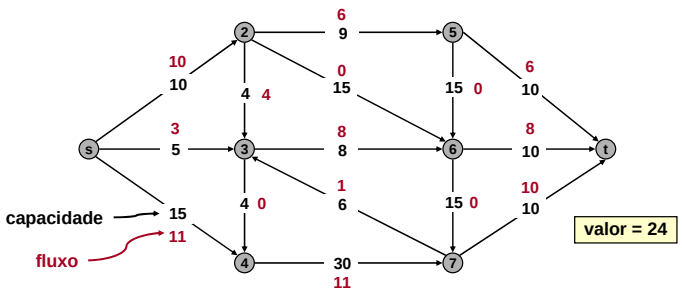


Fluxos

s-t fluxo é uma função f: E → ℜ satisfazendo:

- para cada e ∈ E: $0 \le f(e) \le u(e)$ (capacidade)
- para cada v ∈ V – {s, t}: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservação)

Fluxo Máximo: encontrar um s-t fluxo que maximiza o fluxo que sai da fonte.

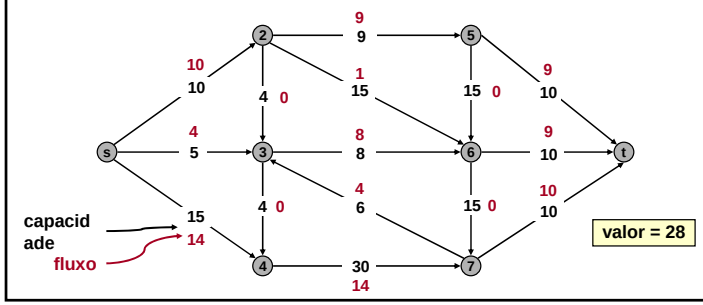


Fluxos

An s-t fluxo is a function f: E → ℜ that satisfies:

- para cada e ∈ E: $0 \le f(e) \le u(e)$ (capacidade)
- para cada v ∈ V – {s, t}: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservação)

Fluxo Máximo: encontrar um s-t fluxo que maximiza o fluxo que sai da fonte.



Redes

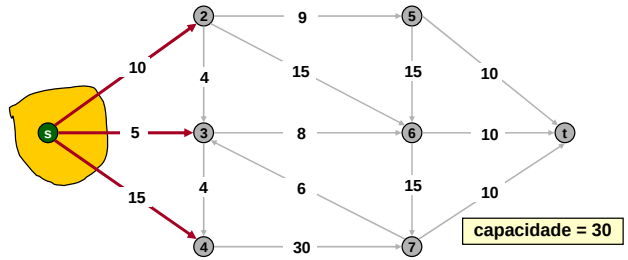
Rede	Nós	Arcos	Fluxo
comunicação	telefone, computador, satélite	cabos, fibra ótica, microondas	voz, video, pacotes
circuitos	portas, registradores, processodores	fios	corrente elétrica
mecânica	junções	estacas, molas	energia
hidráulica	reservatórios, estações de tratamento	canos	fluído, óleo
finanças	ações, dinheiro	transações	dinheiro
transporte	aeroportos, intersecção de ruas	estradas, rotas aéreas	vôos, veículos, passageiros
química	locais	limites	energia

Cortes

s-t corte é uma partição de nós (S, T) tal que $s \in S$, $t \in T$.

- **capacidade** de um s-t corte (S, T) é: $\sum_{e \text{ out of } S} u(e) := \sum_{\substack{(v,w) \in E \\ v \in S,\ w \in T}} u(v,w).$

s-t Corte Mínimo: Encontrar um s-t corte de capacidade mínima.

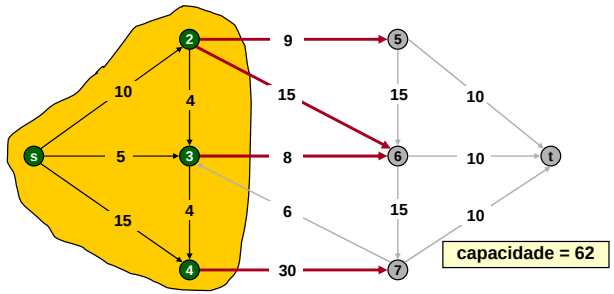


Cortes

s-t corte é uma partição de nós (S, T) tal que $s \in S$, $t \in T$.

- **capacidade** de um s-t corte (S, T) é: $\sum_{e \text{ out of } S} u(e) := \sum_{\substack{(v,w) \in E \\ v \in S,\ w \in T}} u(v,w).$

s-t Corte Mínimo: Encontrar um s-t corte de capacidade mínima.

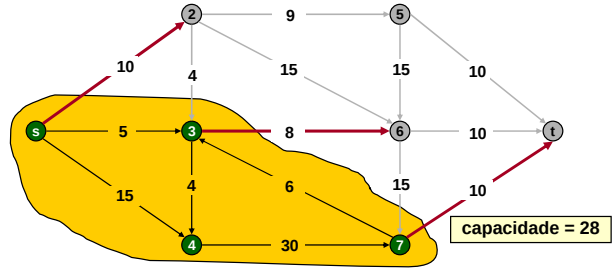


Cortes

s-t corte é uma partição de nós (S, T) tal que $s \in S$, $t \in T$.

- **capacidade** de um s-t corte (S, T) é: $\sum_{e \text{ out of } S} u(e) := \sum_{\substack{(v,w) \in E \\ v \in S,\ w \in T}} u(v,w).$

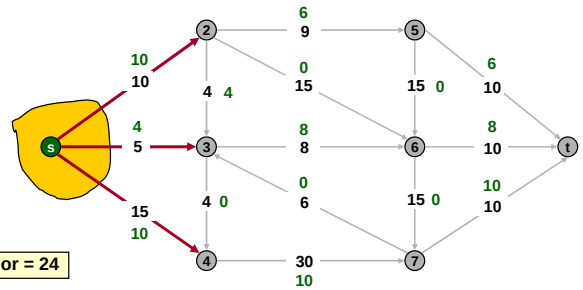
s-t Corte Mínimo: Encontrar um s-t corte de capacidade mínima.



Fluxos e Cortes

L1. Seja f um fluxo, e (S, T) um corte. Então, o fluxo enviado através do corte é igual a quantidade chegando ab t.

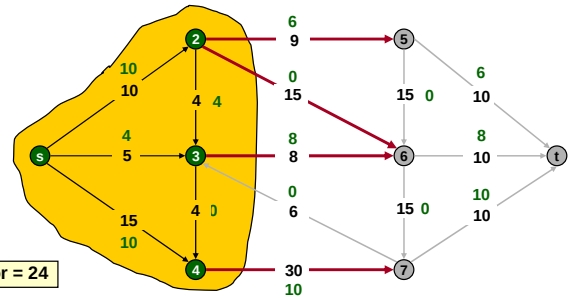
$$\sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e) = \sum_{e \text{ out of } s} f(e) = |f|$$



Fluxos e Cortes

L1. Seja f um fluxo, e (S, T) um corte. Então, o fluxo enviado através do corte é igual a quantidade chegando ab t.

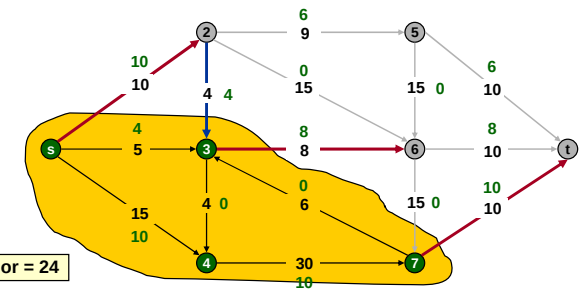
$$\sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e) = \sum_{e \text{ out of } s} f(e) = |f|$$



Fluxos e Cortes

L1. Seja f um fluxo, e (S, T) um corte. Então, o fluxo enviado através do corte é igual a quantidade chegando ab t.

$$\sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e) = \sum_{e \text{ out of } s} f(e) = |f|$$



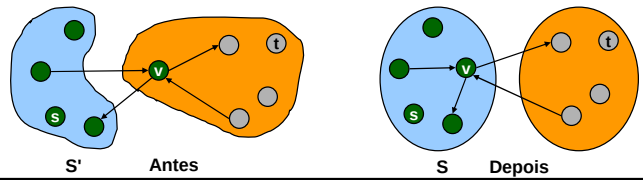
Fluxos e Cortes

Seja f um fluxo, e (S, T) um corte. Então, $\sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e) = |f|$.

Prova por indução em |S|.

- Caso Base: S = { s }.
- Hípotese Indutiva: suponha que vale para |S| < k.
  - Considere um corte (S, T) with |S| = k
  - S = S' ∪ { v } para algum v ≠ s, t, |S' | = k-1 ⇒ cap(S', T') = | f |.
  - incluir v a S' aumenta a capacidade do corte por

$$\sum_{e \text{ out of } v} f(e) - \sum_{e \text{ in to } v} f(e) = 0$$



Fluxos e Cortes

L2. Seja f um fluxo, e (S, T) um corte. Então, $|f| \le cap(S, T)$.

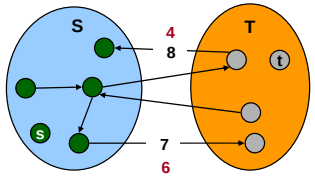
Prova.

$$|f| = \sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e)$$

$$\le \sum_{e \text{ out of } S} f(e)$$

$$\le \sum_{e \text{ out of } S} u(e)$$

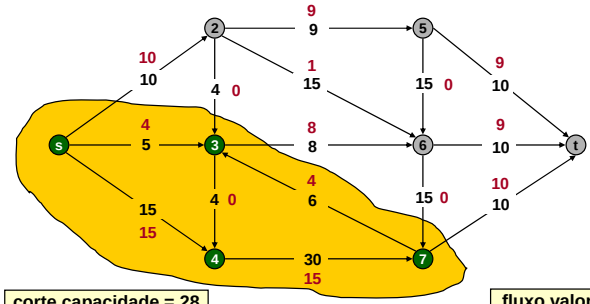
$$= cap(S, T)$$



Corolário. Seja f um fluxo, e (S, T) um corte. Se |f| = cap(S, T), então f é um Fluxo Máximo e (S, T) de um Corte Mínimo.

Fluxo Máximo e Corte Mínimo

Corolário. Seja f um fluxo, e (S, T) um corte. Se |f| = cap(S, T), então f é um Fluxo Máximo e (S, T) é um Corte Mínimo.



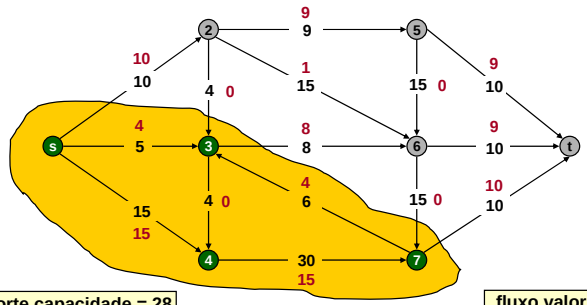
corte capacidade = 28

fluxo valor = 28

Teorema Max-flow Min-cut

Teorema MAX-flow MIN-cut (Ford-Fulkerson, 1956): Qualquer rede, o valor do Fluxo Máximo é igual ao valor do corte mínimo.

- "Boa caracterização."
- Prova IOU.

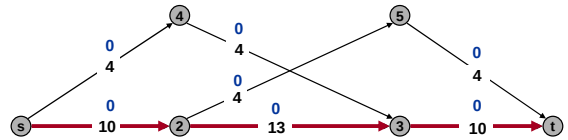


corte capacidade = 28

fluxo valor = 28

Construindo um Algoritmo

Encontre um s-t caminho onde cada arco tem u(e) > f(e) e "aumente" o fluxo ao longo do caminho.

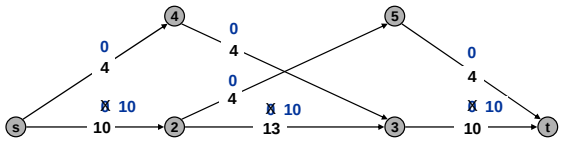


fluxo valor = 0

Construindo um Algoritmo

Encontre um s-t caminho onde cada arco tem u(e) > f(e) e "aumente" o fluxo ao longo do caminho.

- **Repita até não poder mais.**



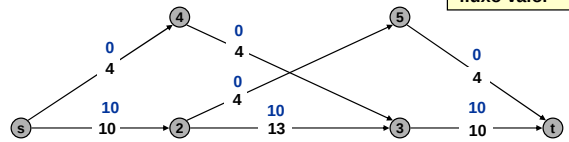
fluxo valor = 10

Construindo um Algoritmo

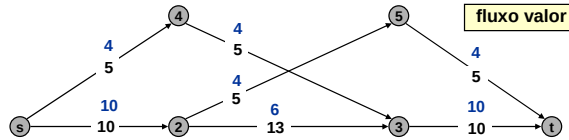
Encontre um s-t caminho onde cada arco tem u(e) > f(e) e "aumente" o fluxo ao longo do caminho.

- **Repita até não poder mais.**
- **Algoritmo Greedy falha.**

fluxo valor = 10



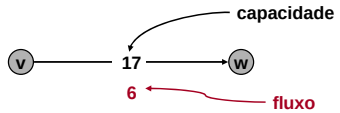
fluxo valor = 14



Arco Residual

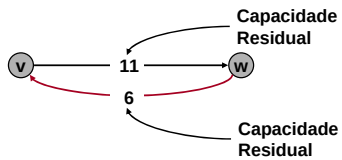
Grafo original G = (V, E).

- **fluxo f(e).**
- **Arco e = (v, w) ∈ E.**



Grafo Residual: $G_f = (V, E_f)$.

- **Arcos Residuais e = (v, w) e e^R = (w, v).**
- **"Desfaz" fluxo enviado.**



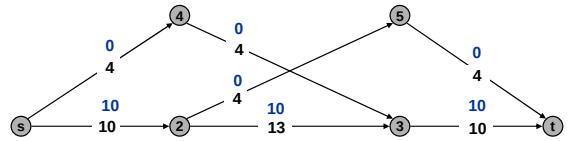
Grafo Residual e Caminho Aumentante

Grafo Residual : $G_f = (V, E_f)$.

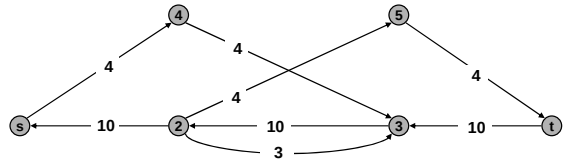
- **$E_f = \{e : f(e) < u(e)\} \cup \{e^R : f(e) > 0\}$.**

$$u_f(e) = \begin{cases} u(e) - f(e) & \text{if } e \in E \\ f(e) & \text{if } e^R \in E \end{cases}$$

G

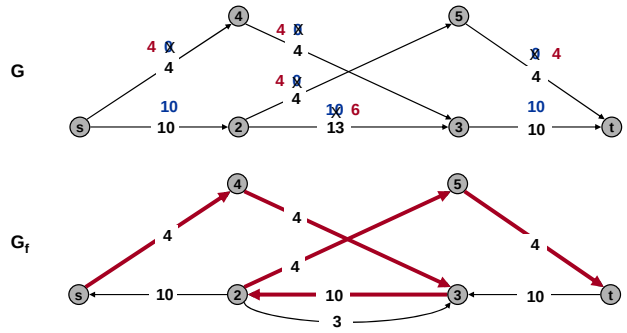


G_f



Caminho Aumentante

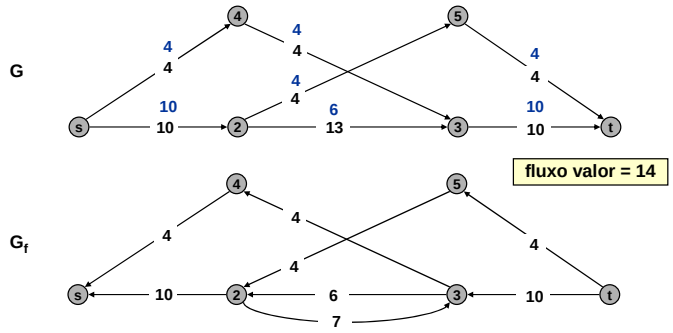
Caminho aumentante = Caminho no Grafo Residual.



Caminho Aumentante

Caminho aumentante = Caminho no Grafo Residual.

- **Fluxo Máximo ⇔ sem caminhos aumentantes ???**



Teorema Max-flow Min-cut

Teorema dos Caminhos Aumentantes (Ford-Fulkerson, 1956): U m fluxo f é Fluxo Máximo se e só sem não há caminhos aumentantes.

Teorema MAX-flow MIN-cut (Ford-Fulkerson, 1956): o valor do Fluxo Máximo é igual ao valor do corte mínimo.

Provamos ambos simultaneamente mostrando :

(i) **f é um Fluxo Máximo.**

(ii) **Não há caminhos aumentantes relativos a f.**

(iii) **Há um a corte (S, T) tal que |f| = cap(S, T).**

Prova do Teorema Max-flow Min-cut

Provamos ambos simultaneamente mostrando :

(i) **f é um Fluxo Máximo.**

(ii) **Não há caminhos aumentantes relativos a f.**

(iii) **Há um a corte (S, T) tal que |f| = cap(S, T).**

(i) ⇒ (ii)

- **Mostramos a contrapositiva.**
- **Seja f um fluxo. Se há um caminho aumentante, então podemos aumentar f enviando fluxo ao longo do caminho.**

(iii) ⇒ (i)

- **Próximo slide.**

(iii) ⇒ (i)

- **Foi o Corolário do Lema 2.**

Prova do Teorema Max-flow Min-cut

Provamos ambos simultaneamente mostrando :

(i) f é um Fluxo Máximo.

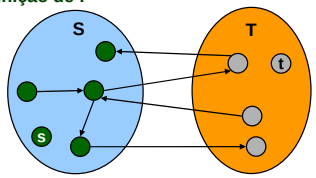
(ii) Não há caminhos aumentantes relativos a f.

(iii) Há um a corte (S, T) tal que |f| = cap(S, T).

(ii) ⇒ (iii)

- Seja f um fluxo sem caminhos aumentantes.
- Seja S um conj. de vertices atingíveis de s no grafo residual.
  - claramente s ∈ S, e t ∉ S pela definição de f

$$|f| = \sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e)$$
$$= \sum_{e \text{ out of } S} u(e)$$
$$= cap(S, T)$$



Rede Original

Algoritmo Caminho Aumentante

Aumenta (f, P)

```
b ← bottleneck(P)
FOREACH e ∈ P
   IF (e ∈ E) // arco indo
      f(e) ← f(e) + b
   ELSE       // arco voltando
      f(e^R) ← f(e) - b
RETURN f
```

FordFulkerson (V, E, s, t)

```
FOREACH e ∈ E
   f(e) ← 0
G_f ← grafo residual

WHILE (houver caimho aumentante P)
   f ← aumenta(f, P)
   atualiza G_f
RETURN f
```


Tempo de execução

Suposição: todas as capacidades são inteiros entre 0 e U.

Invariante: todo valor de fluxo f(e) e toda capacidade residual $u_f(e)$ permanece um inteiro durante o algoritmo.

Teorema: o algoritmo termina em no máximo | f * | ≤ nU iterações.

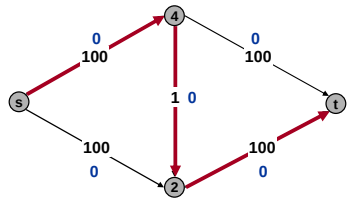
Corolário: se U = 1, então o algorithm é O(mn).

Teorema da integralidade: Se todas as capacidades são inteiras, então há um Fluxo Máximo f para o qual todo valor de fluxo f(e) é um inteiro.

Nota: o algoritmo pode não terminar em instâncias patológicas (com capacidades irracionais). Além disso, o valor de fluxo pode nem mesmo convergir para a resposta correta.

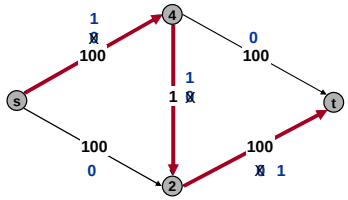
Escolha de Bons Caminhos Aumentantes

Tome cuidado na escolha dos caminhos aumentantes.



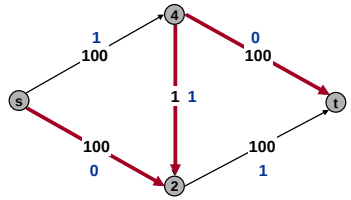
Escolha de Bons Caminhos Aumentantes

Tome cuidado na escolha dos caminhos aumentantes.



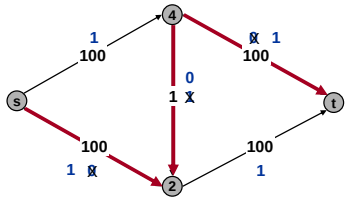
Escolha de Bons Caminhos Aumentantes

Tome cuidado na escolha dos caminhos aumentantes.



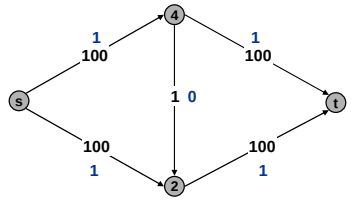
Escolha de Bons Caminhos Aumentantes

Tome cuidado na escolha dos caminhos aumentantes.



Escolha de Bons Caminhos Aumentantes

Tome cuidado na escolha dos caminhos aumentantes.



200 iterações possíveis.

Escolha de Bons Caminhos Aumentantes

. Tome cuidado na escolha dos caminhos aumentantes

- Algumas escolhas levam a algoritmos exponenciais.
- Boas escolhas levam a algoritmos polinomiais.

Objetivo: Escolher caminhos aumentantes tal que:

- Escontre caminhos aumentantes eficientemente.
- Poucas iterações.

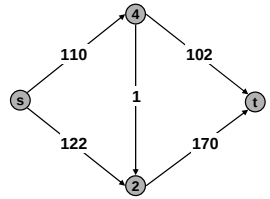
Edmonds-Karp (1972): Escolher caminho aumentante com

- Capacidade Máxima. (caminho gordo)
- Capacidade suficientemente grande . (escala da capacidade)
- Menor número de arcos. (caminho mínimo)

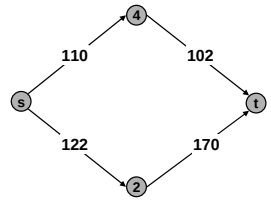
Capacidade de Escala

Intuição: Escolher caminho de maior capacidade aumenta o fluxo na quantidade máxima possível.

- Não precisa encontrar o caminho com a maior capacidade exata.
- Mantenha o parâmetro de escala Δ.
- Seja $G_f(\Delta)$ o subgrafo do grafo residual consistindo só de arcos com capacidade pelo menos Δ.



G_f



$G_f(100)$

Capacidade de Escala

Intuição: Escolher caminho de maior capacidade aumenta o fluxo na quantidade máxima possível.

- Não precisa encontrar o caminho com a maior capacidade exata.
- Mantenha o parâmetro de escala Δ.
- Seja $G_f(\Delta)$ o subgrafo do grafo residual consistindo só de arcos com capacidade pelo menos Δ.

escalaMaxfluxo(V, E, s, t)

```
FOREACH e ∈ E, f(e) ← 0
Δ ← menor potência de 2 maior ou igual a U

WHILE (Δ ≥ 1)
   G_f(Δ) ← Δ-grafo residual
   WHILE (houver caminho aumentante P em G_f(Δ))
      f ← aumenta(f, P)
      atualize G_f(Δ)
   Δ ← Δ / 2
RETURN f
```


Capacidade de Escala: Análise

L1. Se todas as capacidades são inteiras, então durante o algoritmo, todo valor de fluxo e de capacidade residual permanecem inteiros.

- Assim, $\Delta = 1 \Rightarrow G_f(\Delta) = G_f$, e no término f é um Fluxo Máximo.

L2. O while externo repete $1 + \lfloor \log_2 U \rfloor$ vezes.

- Inicialmente $U \le \Delta < 2U$, e Δ diminui por um fator de 2 a cada iteração.

L3. Seja f o fluxo no final de uma fase de Δ-escala. Então o valor do fluxo máximo é no máximo $|f| + m\Delta$.

L4. Há no máximo 2m aumentos por fase de escala.

- Seja f o fluxo no final da fase de escala anterior.
- L3 $\Rightarrow$ $|f^*| \le |f| + m(2\Delta)$.
- Cada aumento em uma Δ-fase aumenta $|f|$ de pelo menos Δ.

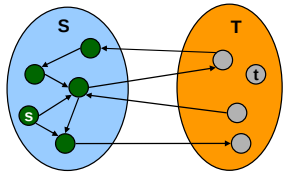
Teorema. O algoritmo roda em tempo $O(m^2 \log(2U))$.

Capacidade de Escala: Análise

L3. Seja f o fluxo no final de uma fase de Δ-escala. Então o valor do fluxo máximo é no máximo | f | + m Δ.

- Mostramos que no final de uma Δ-fase, há um a corte (S, T) tal que cap(S, T) ≤ | f | + m Δ.
- Escolha S para ser o conj. de nós atingíveis de s em $G_f(\Delta)$.
  - claramente s ∈ S, e t ∉ S pela definição de S

$$
\begin{aligned}
|f| &= \sum_{e \text{ out of } S} f(e) - \sum_{e \text{ in to } S} f(e) \\
&\geq \sum_{e \text{ out of } S} (u(e) - \Delta) - \sum_{e \text{ in to } S} \Delta \\
&= \sum_{e \text{ out of } S} u(e) - \sum_{e \text{ out of } S} \Delta - \sum_{e \text{ in to } S} \Delta \\
&= cap(S, T) - m\Delta
\end{aligned}
$$



Rede Original

Escolha de Bons Caminho Aumentantes

. Tome cuidado na escolha dos caminhos aumentantes

- Algumas escolhas levam a algoritmos exponenciais.
- Boas escolhas levam a algoritmos polinomiais.

Objetivo: Escolher caminhos aumentantes tal que:

- Escontre caminhos aumentantes eficientemente.
- Poucas iterações.

Edmonds-Karp (1972): escolha caminho aumentante com

- Capacidade Máxima. (Caminho gordo)
- Capacidade suficientemente grande. (escada de capacidade)
- Menor número de arcos. (menor caminho)

Menor caminho aumentante

Intuição: escolha o caminho via breadth first search.

- Fácil de implementar.
  - Podem implementar para ver!
- Encontra caminhos aumentantes com menor número of arcos.

FewerAugmentingPath(V, E, s, t)

```
FOREACH e ∈ E
   f(e) ← 0
Gf ← grafo residual

WHILE (houver caminho aumentante)
   encontre um caminho P com o BFS
   f ← aumenta(f, P)
   atualiza Gf
RETURN f
```


Menor caminho aumentante: Descrição da Análise

L1. Durante o algoritmo, o comprimento do menor caminho nunca diminui.

- Prova a seguir.

L2. Depois de no máximo m aumentos de menores caminhos, o comprimento do menor caminho aumentante aumenta estritamente.

- Prova a seguir.

Teorema. O algoritmo menor caminho aumentante rod em tempo $O(m^2n)$.

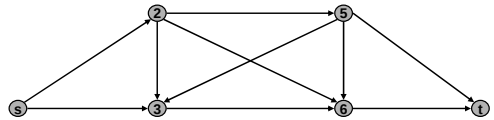
- O(m+n) para encontrar o menor caminho aumentante via BFS.
- O(m) aumentos para caminhos de k arcos exatos.
- Se há um caminho aumentante, ele é único.
  - ⇒ 1 ≤ k < n
  - ⇒ O(mn) aumentos.

Menor caminho aumentante: Análise

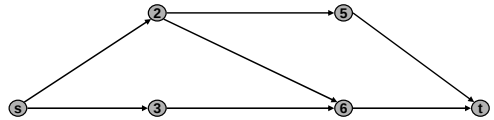
Grafo Nível de (V, E, s).

- para cada vértice v, defina λ(v) como o comprimento (número de arcos) do menor caminho de s a v.
- $L_G = (V, E_G)$ é subgrafo de G que contêm apenas os arcos (v,w) ∈ E com λ(w) = λ(v) + 1.

G:



L_G:



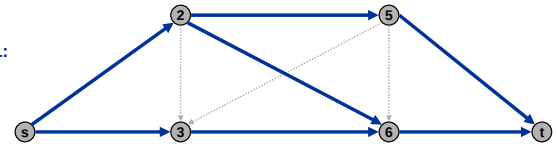
λ = 0 λ = 1 λ = 2 λ = 3

Menor caminho aumentante: Análise

Grafo Nível de (V, E, s).

- para cada vértice v, defina λ(v) como o comprimento (número de arcos) do menor caminho de s a v.
- L = (V, F) é subgrafo de G que contêm apenas os arcos (v,w) ∈ E com λ(w) = λ(v) + 1.
- Compute em tempo O(m+n) usando BFS, removendo arcos laterais e de volta.
- P é um menor s-v caminho em G se e só se ele é um s-v caminho em L.

L:



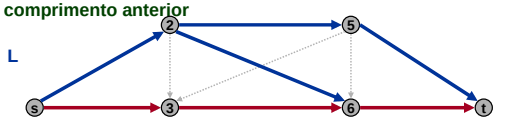
λ = 0 λ = 1 λ = 2 λ = 3

Menor caminho aumentante: Análise

L1. Durante o algoritmo, o comprimento do menor caminho nunca diminui.

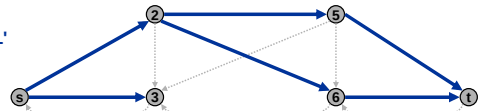
- Sejam f e f' fluxos antes e depois de aumentar o menor.
- Sejam L e L' grafos níveis de G_f e $G_{f'}$.
- Apenas arcos de volta incluidos em $G_{f'}$.
  - Caminhos com arcos voltando tem comprimento maior que o comprimento anterior

L



λ = 0 λ = 1 λ = 2 λ = 3

L'

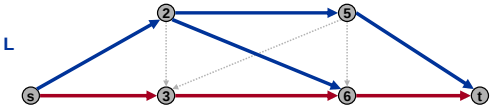


Menor caminho aumentante: Análise

L2. Depois de no máximo m aumentos de menores caminhos, o comprimento do menor caminho aumentante aumenta estritamente.

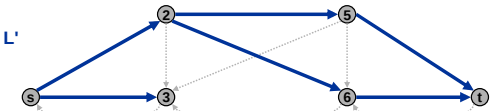
- Pelo menos um arco (o arco gargalo) é removido de L depois de cada aumento.
- Nenhum novo arco é adicionado a L até que o comprimento do menor caminho cresça estritamente.

L



λ = 0 λ = 1 λ = 2 λ = 3

L'



Menor caminho aumentante: Resumo da Análise

L1. Durante o algoritmo, o comprimento do menor caminho nunca diminui.

L2. Depois de no máximo m aumentos de menores caminhos, o comprimento do menor caminho aumentante aumenta estritamente.

Teorema. O algoritmo menor caminho aumentante roda em tempo $O(m^2n)$.

- **O(m+n) para encontrar o menor caminho aumentante via BFS.**
- **O(m) aumentos para caminhos de exatamente k arcos.**
- **O(mn) aumentos.**

Nota: Θ(mn) aumentos são necessários em algumas redes.

- **Tente reduzir o tempo em cada aumento.**
- **Árvore Dinâmicas ⇒ O(mn log n) Sleator-Tarjan, 1983**
- **Idéia Simples ⇒ $O(mn^2)$**

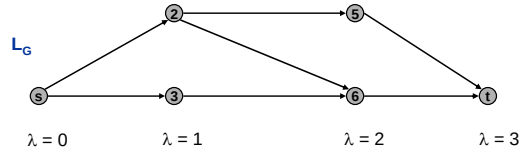
Menor caminho aumentante: Versão Melhorada

Dois tipos de aumentos.

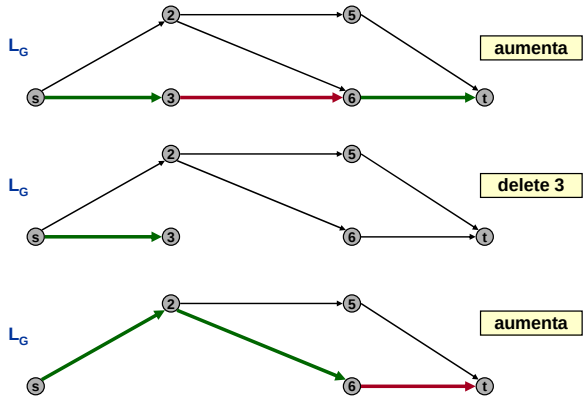
- **Aumento normal: Comprimento do menor caminho não muda.**
- **Aumento especial: comprimento do menor caminho cresce.**

L3. Grupo de aumentos normais tomam tempo O(mn).

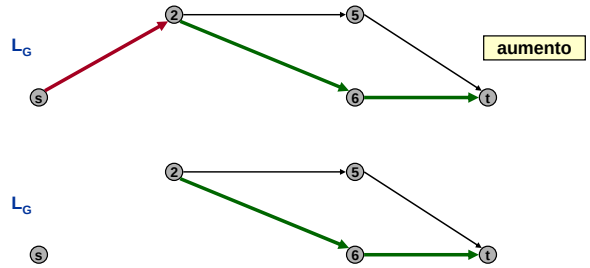
- **Explicitamente mantenha o grafo nível - ele muda de no máximo 2n arcos depois de cada aumento normal.**
- **Inicie em s, avance em um arco em L_G até alcançar t ou não conseguir.**
  - **Se alcaçar t, aumente e delete pelo menos um arco**
  - **Se não conseguir, delete o nó**



Menor caminho aumentante: Versão Melhorada



menor caminho aumentante: Versão Melhorada



PARE: Comprimento do menor caminho deve ter crescido

menor caminho aumentante: Versão Melhorada

```
AdvanceRetreat(V, E, f, s, t)

ARRAY pred[v ∈ V]
L_G ← grafo nível de G_f
v ← s, pred[v] ← nil

REPEAT
   WHILE (houver (v,w) ∈ L_G)
      pred[w] ← v, v ← w                    ← avanço
      IF (v = t)
         P ← caminho def. por pred[]        ← aumenta
         f ← aumenta(f, P)
         atualiza L_G
         v ← s, pred[v] ← nil
   delete v de L_G                          ← retrai
UNTIL (v = s)

RETURN f
```


menor caminho aumentante: Versão Melhorada

Dois tipos de aumentos.

- Aumento normal: Comprimento do menor caminho não muda.
- Aumento especial: comprimento do menor caminho cresce.

L3. Grupo de aumentos normais tomam tempo O(mn).

- Explicitamente mantenha o grafo nível - ele muda de no máximo 2n arcos depois de cada aumento normal.
- Inicie em s, avance em um arco em L_G até alcançar t ou não conseguir.
  - Se alcaçar t, aumente e delete pelo menos um arco
  - Se não conseguir, delete o nó
  - No máximo n avanços antes do evento acima

Teorema. O Algoritmo roda em tempo $O(mn^2)$.

- Tempo O(mn) entre aumentos especiais.
- No máximo n aumentos especiais.

Escolha de bons caminho aumentantes: Resumo

Método	Aumentos	Tempo
Caminho aumentante	nU	mnU
Capacidade Máxima	m log U	m log U (m + n log n)
Capacidade de escala	m log U	m^2 log U
Capacidade de escala melhor	m log U	mn log U
Menor caminho	mn	m^2n
Menor caminho melhorado	mn	mn^2

Os primeiros 4 supõe capacidades entre 0 e U.

Histórico

Ano	Autor	Método	Tempo
1951	Dantzig	Simplex	mn^2U
1955	Ford, Fulkerson	Caminho aumentante	mnU
1970	Edmonds-Karp	Menor caminho	m^2n
1970	Dinitz	Menor caminho	mn^2
1972	Edmonds-Karp, Dinitz	Capacidade de escala	m^2 log U
1973	Dinitz-Gabow	Capacidade de escala	mn log U
1974	Karzanov	Prefluxo-push	n^3
1983	Sleator-Tarjan	Árvore Dinâmica	mn log n
1986	Goldberg-Tarjan	FIFO prefluxo-push	mn log (n^2 / m)
. . .	. . .	. . .	. . .
1997	Goldberg-Rao	Função Comprimento	$m^{3/2}$ log (n^2 / m) log U $mn^{2/3}$ log (n^2 / m) log U