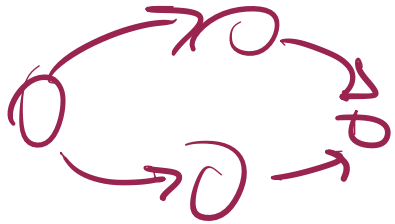
FLUXOS EM REDES

Network

Network flow

Marcus Ritt

Outline



1. Introducão

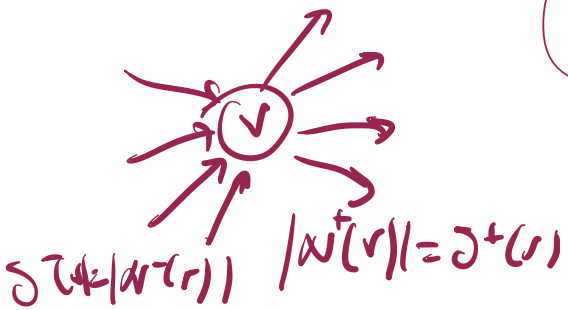
2 Solução com Programação Linear

3. O algoritmo de Ford-Fulkerson

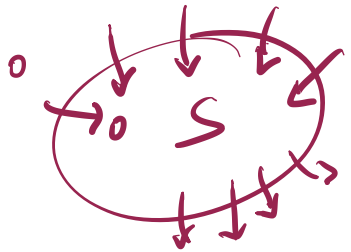
- Contexto: grafo direcionado $G = (V, A)$.
- Dado um arco $a = (u, v)$ podemos definir um fluxo $f_a \in [0, c_a]$ no arco.
- A quantidade c_a é a capacidade do arco.
- Fluxos tem aplicações obvias, por exemplo em
  - redes de transporte, e
  - redes elétricas.

- Para um vértice $v \in V$ temos uma balança de fluxo

$$f(v) := \sum_{a \in \delta^+(v)} f_a - \sum_{a \in \delta^-(v)} f_a \qquad (1)$$



Extensão a conjuntos $S \subseteq V$:



- Escreve

$$\delta^-(S) = (\bigcup_{v \in S} \delta^-(v)) \setminus S^2 \tag{2}$$

$$\delta^+(S) = (\bigcup_{v \in S} \delta^+(v)) \setminus S^2 \tag{3}$$

- $f(S) := \sum_{v \in S} f(v)$
- Logo:

$$f(S) = \sum_{a \in \delta^+(S)} f_a - \sum_{a \in \delta^-(S)} f_a \tag{4}$$

$(u, v) \in S^2$

$$f(S) = \sum_{v \in S} f(S)$$

$$= \sum_{v \in S} \left(\sum_{a \in N^+(S)} f_a - \sum_{a \in N^-(S)} f_a \right)$$

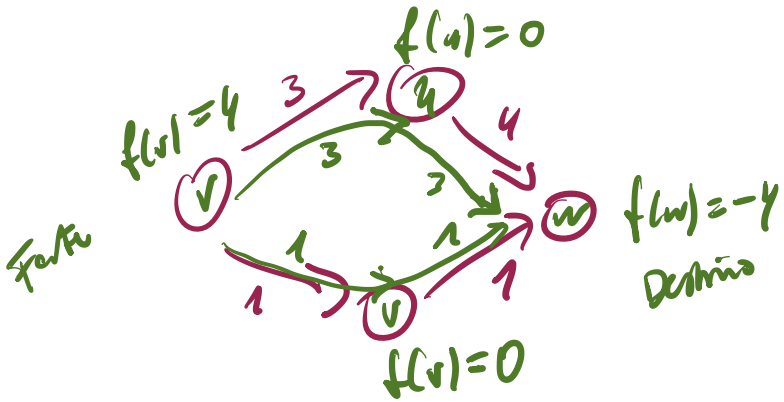
$$= \sum_{\substack{(u,v) \in A \\ u \in S,\, v \in S}} f_a - \sum_{\substack{(u,v) \in A \\ u \in S,\, v \in S}} f_a$$

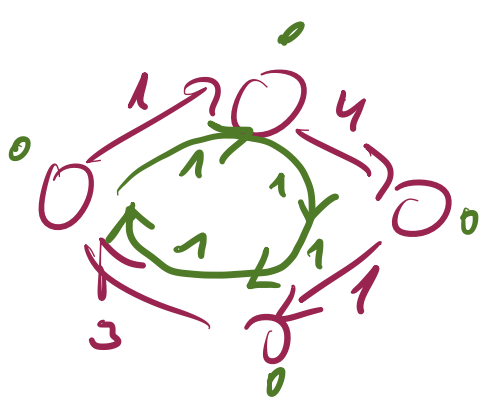
$$= \sum_{a \in N^+(S)} f_a - \sum_{a \in N^-(S)} f_a .$$

- Um fluxo é uma função $f : A \to \mathbb{R}_+$ com $0 \leq f_a \leq c_a$.
- Cada fluxo satisfaz

$$f(V) = \sum_{v \in V} f(v) = 0 \tag{5}$$

- Caso $f(v) = 0$ para todo $v \in V$, f é uma circulação.





1. Para demandas $d_v \in \mathbb{R}$, $v \in V$, encontra uma fluxo tal que $f(v) = d_v$.
2. Caso particular: $d_v = 0, \forall v \in V \setminus \{s, t\}$, $d_s \geq 0$, $d_t \leq 0$ é um fluxo $s - t$.
   - Lema 1: $f(s) + f(t) = 0$.
   - Problema correspondente: fluxo máximo $s - t$: maximiza $f(s)$.
3. Generalização: $V = S \dot\cup \bar{V} \dot\cup T$, com $b_v = 0, \forall v \in \bar{V}$, $b_s \geq 0$, $s \in S$, $b_t \leq 0$, $t \in T$
   - Lema 2: $f(S) + f(T) = 0$
   - Fluxo máximo $S - T$: maximiza $f(S)$
4. Limites inferiores: $b_a \leq f_a \leq c_a$.

- Para fluxo $s - t$ máximo:

$$\begin{aligned} & \textbf{maximiza} && f(s) && && (6) \\ & \textbf{sujeito a} && f(v) = 0, && \forall v \in V^\dagger, \\ & && 0 \leq f_a \leq c_a, && \forall a \in A. \end{aligned}$$

com $V^\dagger = V \setminus \{s, t\}$.

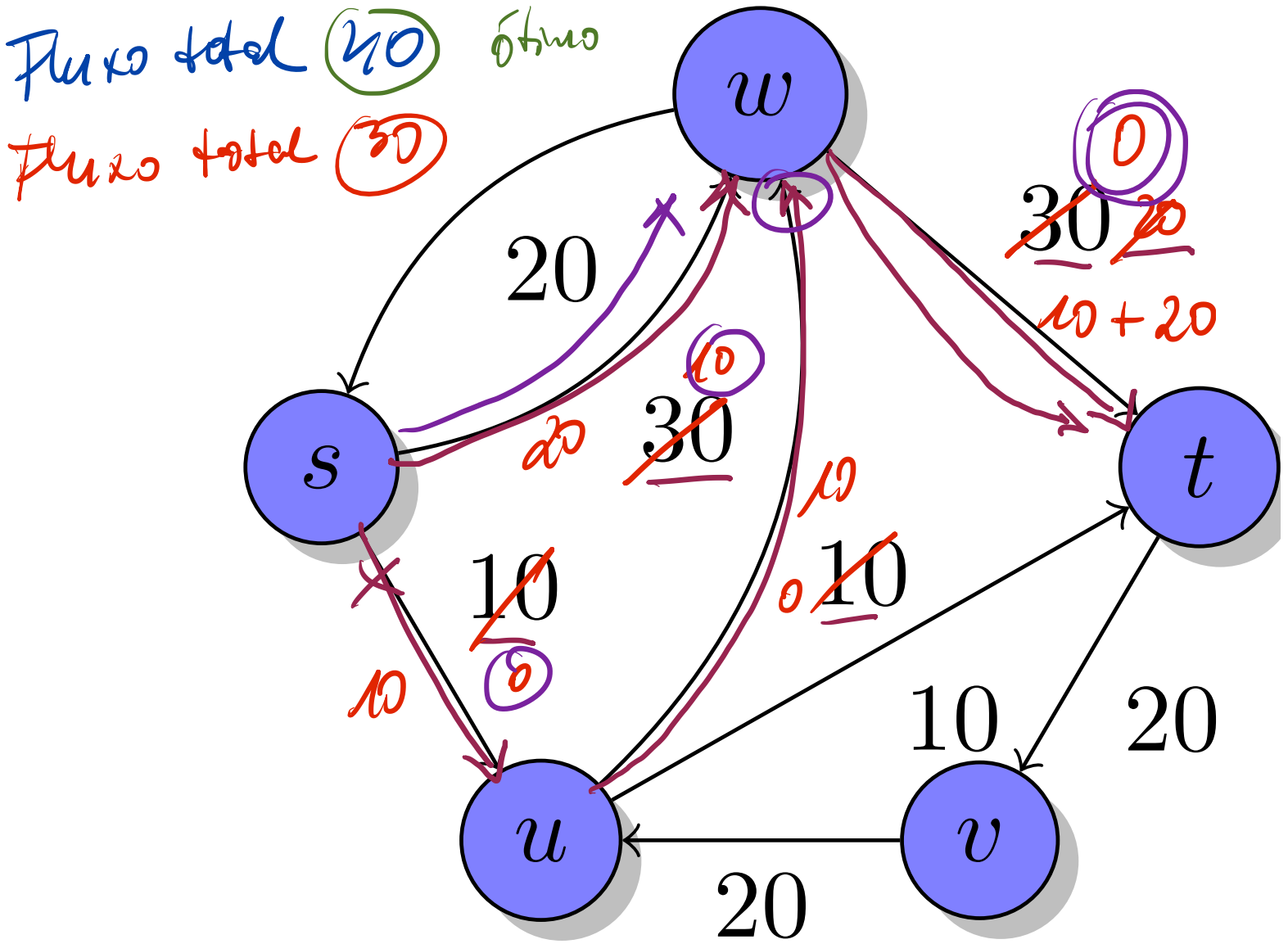
- O PL possui uma solução (e.g. a solução trivial $f \equiv 0$) e é limitado ($f \leq c$), logo possui solução ótima
- Logo: solução polinomial via PL.

- Objetivo: um algoritmo combinatorial mais eficiente.
- Idéia: Começa com fluxo viável $f_a = 0$ e aumenta gradualmente.
- Observação:
  - Considera caminho $s - t$ $P = (v_0 = s, v_1, \ldots, v_{n-1}, v_n = t)$
  - Define o "gargalo"

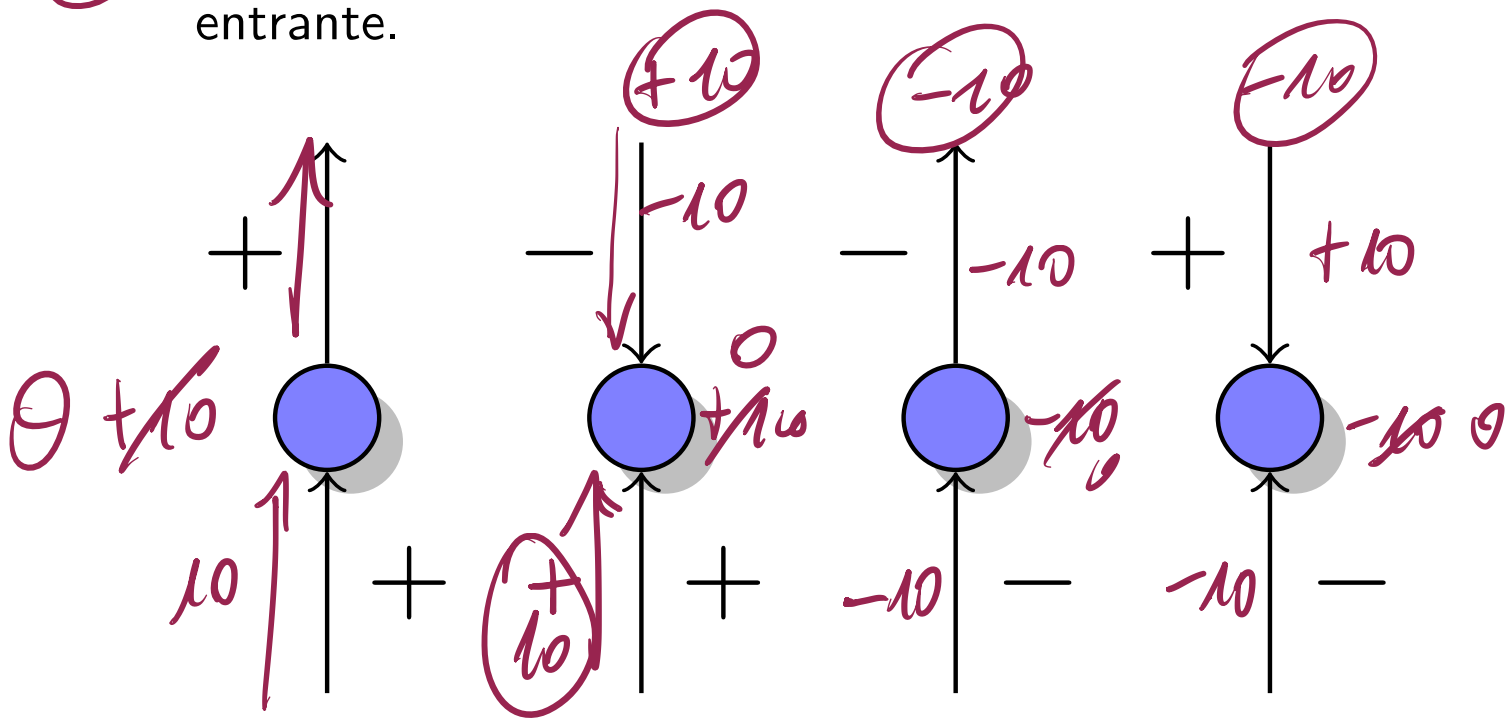
$$g(f, P) := \min_{\substack{a=(v_i, v_{i+1}) \\ 0 \le i < n}} c_a - f_a.$$

  - Podemos aumentar o fluxo atual f por $g(f, P)$.
- Logo: busca um caminho com gargalo positivo, aumente o fluxo, repete. Certo?

Exemplo

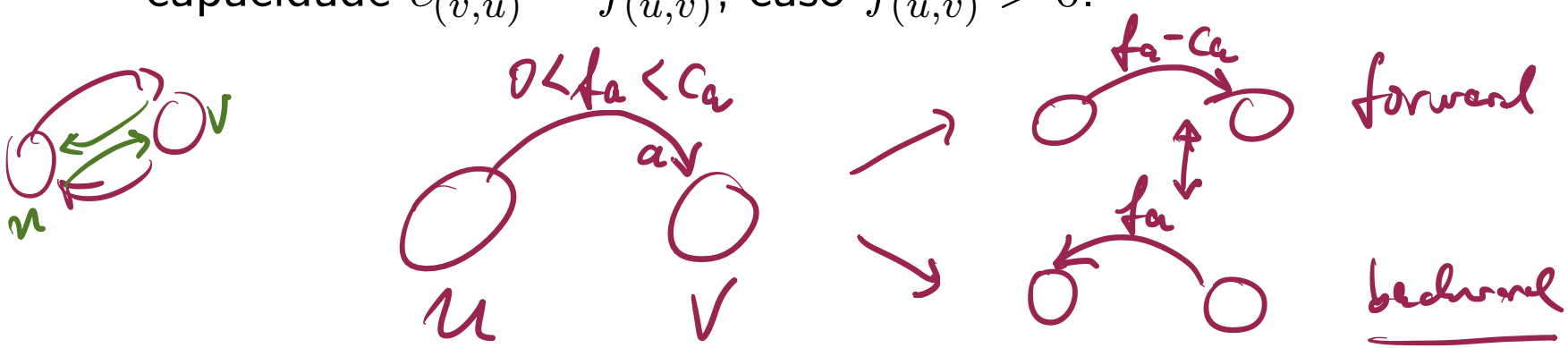

Fluxo total 40
ótimo
Fluxo total 30
w
s
t
u
v
20
30
10
10
20
30
10
10
10
20
20
0
10 + 20

- Para aumentar o fluxo e manter a conservação do fluxo em v podemos
  1. aumentar o fluxo num arco entrante e sainte
  2. aumentar o fluxo num arco entrante, e diminuir num outro arco entrante
  3. diminuir o fluxo num arco entrante e diminuir num arco sainte e
  4. diminuir o fluxo num arco entrante e aumentar num arco entrante.

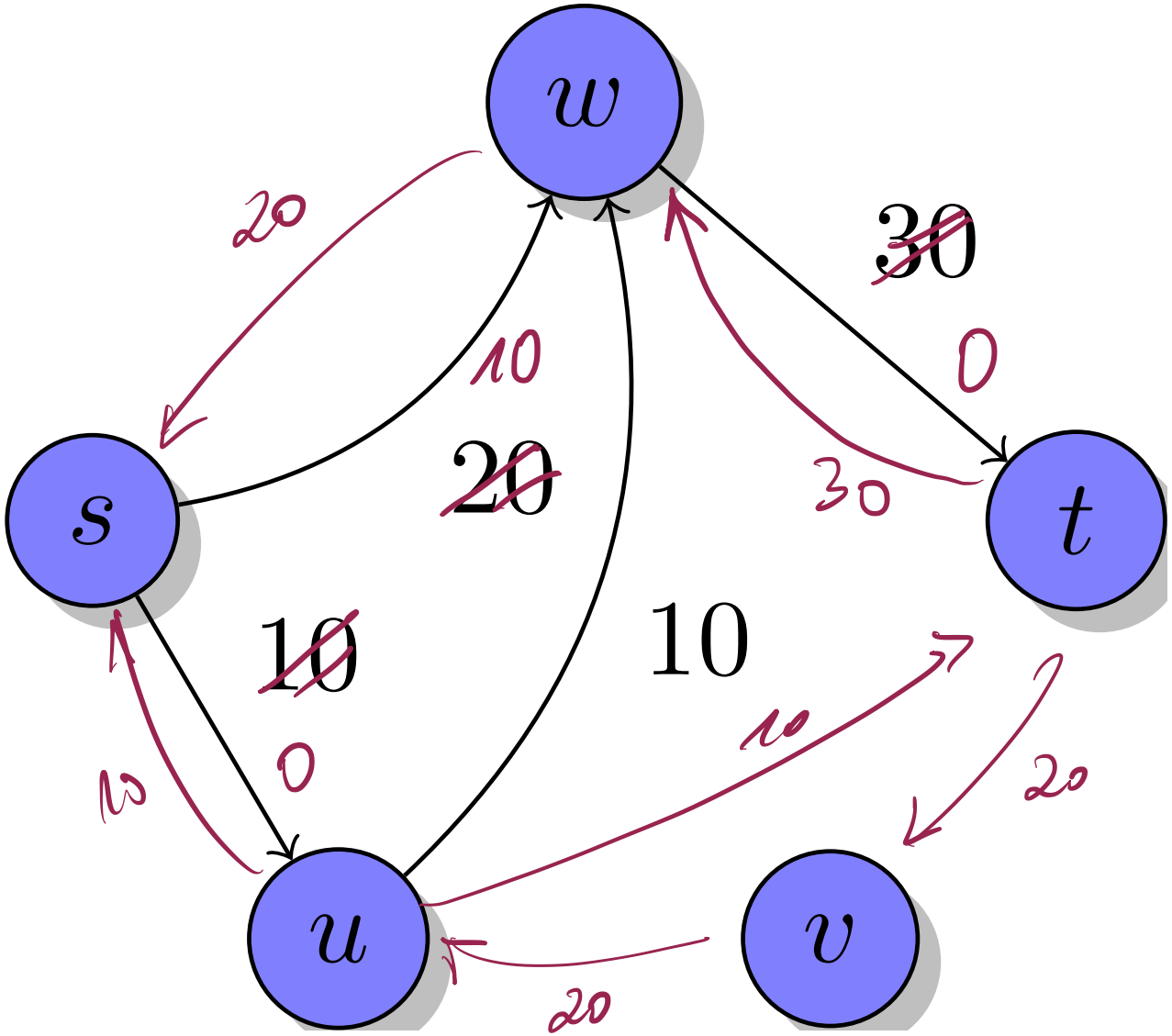


Isso motiva definir para dado fluxo f o grafo residual G_f:

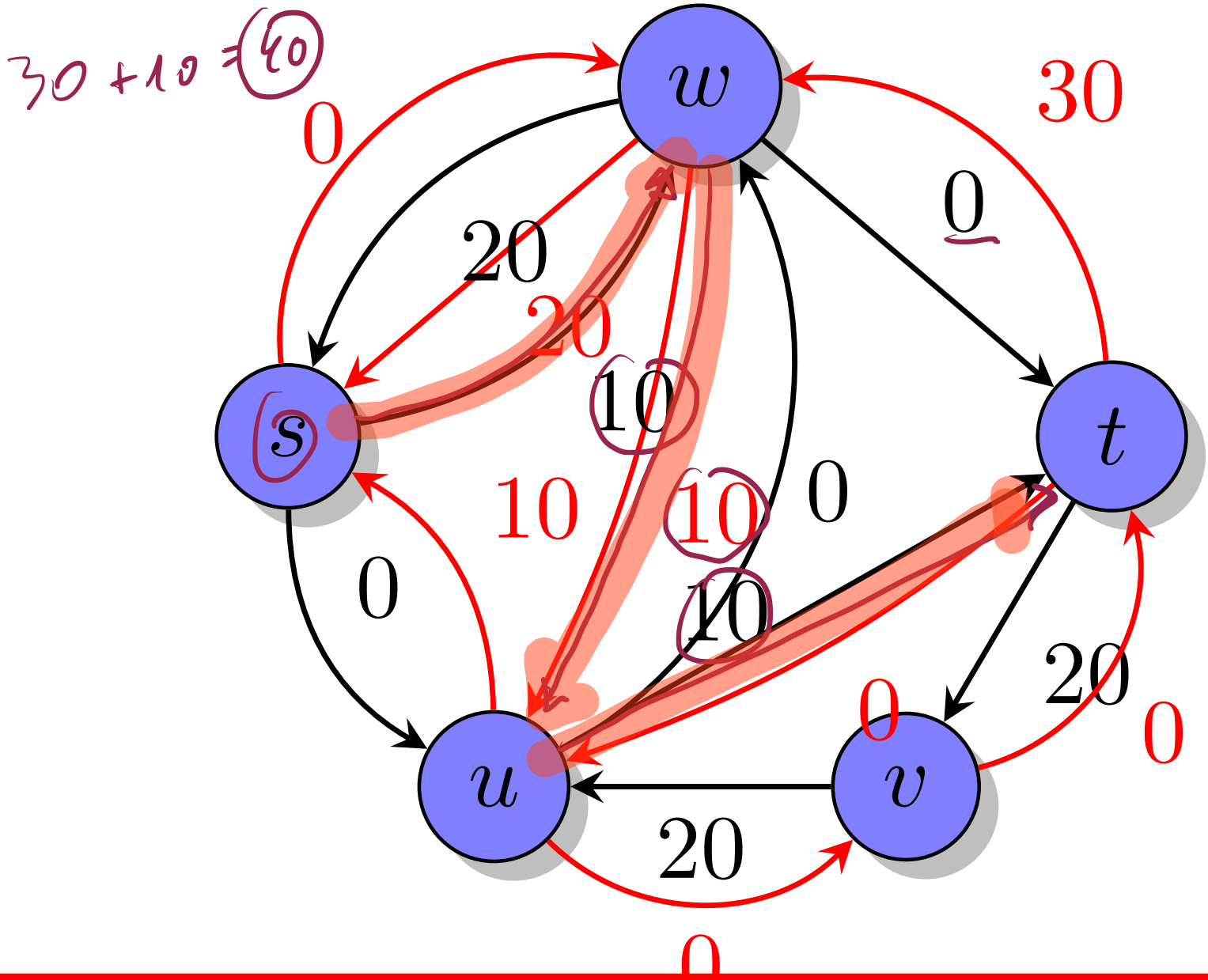
- Vértices V
- Arcos para frente ("forward") A com capacidade $c_a - f_a$, caso $f_a < c_a$.
- Arcos para atras ("backward") $A' = \{(v, u) \mid (u, v) \in E\}$ com capacidade $c_{(v,u)} = f_{(u,v)}$, caso $f_{(u,v)} > 0$.



Exemplo



Exemplo

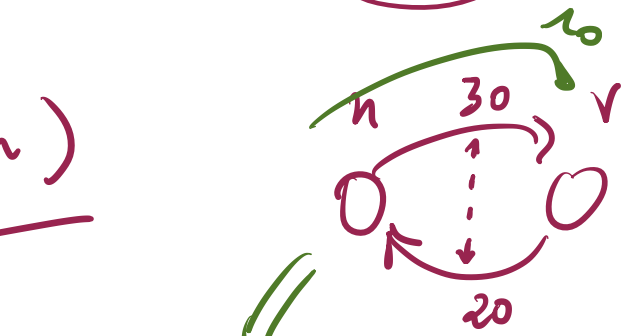


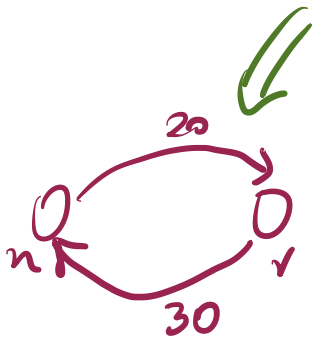
for all $a \in A$: $f_a := 0$
while existe um caminho $s - t$ *em* G_f *do*
    Seja P *um caminho* $s - t$ *simples*
    Aumenta o fluxo f *um valor* $g(f, P)$
end while
return f

Busca em tempo $O(n+m)$

$O(n)$

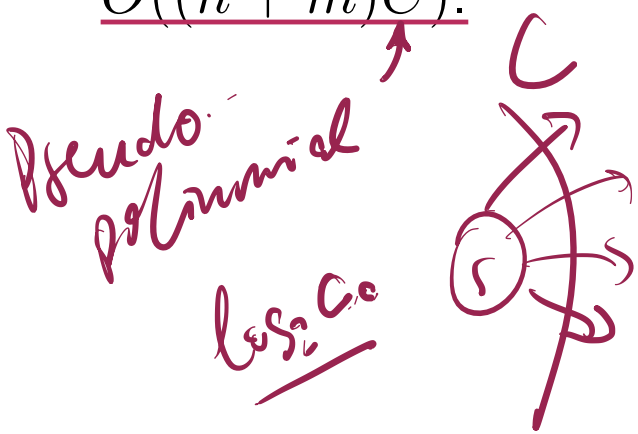
iterações · $O(n+m)$

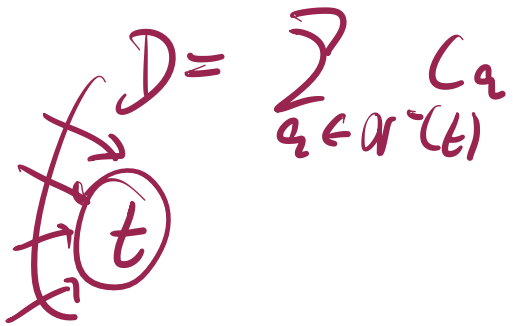




Hipótese: capacidades em $\mathbb{Q}$ ou $\mathbb{N}$.

- Lema 3: Para capacidades inteiras, todo fluxo intermediário e as capacidades residuais são inteiros.
- Lema 4: Em cada iteração, o fluxo aumenta por pelo menos 1.
- Lema 5: O número de iterações do algoritmo Ford-Fulkerson é limitado por $C = \sum_{a \in \delta^+(s)} c_a$. Portanto FF tem complexidade $O((n+m)C)$.





Lema 3: Fluxo inicial $f_0 = 0$, e f é int.

Indução nas iterações:

$\mathbb{Z}$

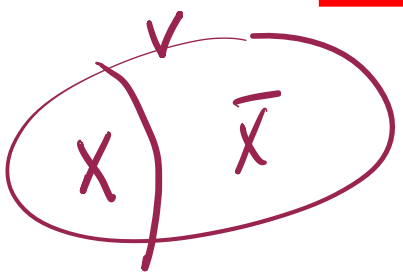
$\mathbb{Q} \rightarrow \ldots$

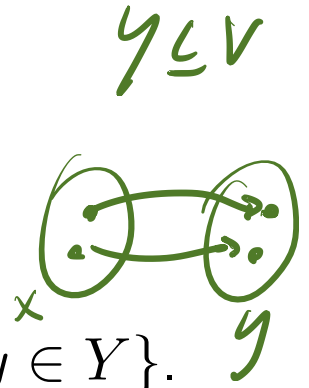
- → Cap. residuais inteiros
- → ganho é inteiro
- → fluxo aumenta p/ uma quant. inteira

Lema 4: ganho é $g(P, f) > 0 \overset{\mathbb{N}}{\Rightarrow} \geq 1$.

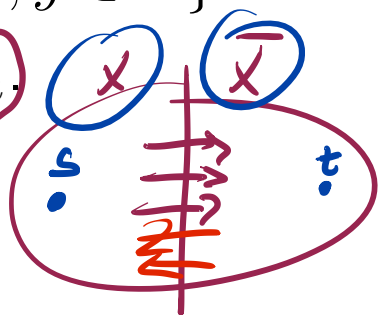
Lema 5: Fluxo Máximo de $C \in \mathbb{Z}$, aumenta pelo menos 1 por iteração.

Corretude: notação

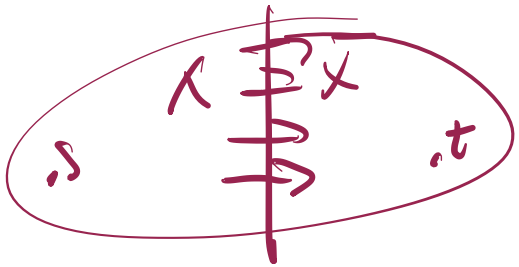




- Complemento $\bar{X} := V \setminus X$.
- Arcos de X para Y: $F(X,Y) := \{(x,y) \mid x \in X, y \in Y\}$.
- Fluxo correspondente: $f(X,Y) := \sum_{a \in F(X,Y)} f_a$.
- Logo: $f(X) = f(X,\bar{X}) - f(\bar{X},X)$.
- Capacidades $c(X,Y) := \sum_{a \in F(X,Y)} c_a$.
- Partição $(X,\bar{X})$ com $s \in X$, $t \in \bar{X}$: se chama um corte $s-t$.
- Um arco a é saturado para um fluxo f, caso $f_a = c_a$.



cap. res. é 0.



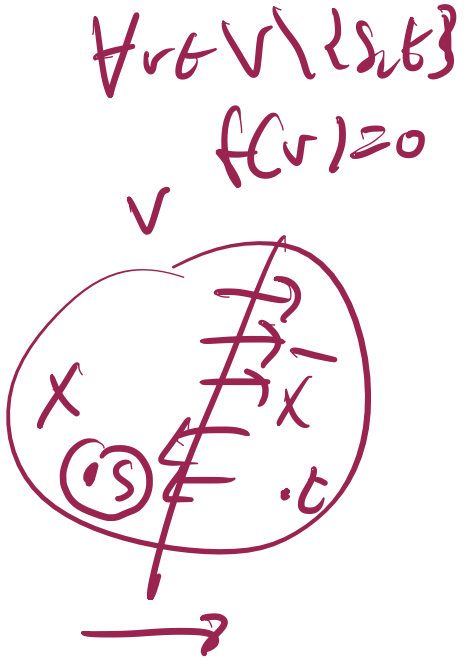
- Lema 6: Para qualquer corte $(X, \bar{X})$ temos $f(X) = f(s)$.
- Lema 7: O valor $c(X, \bar{X})$ de um corte $s - t$ é um limite superior para um fluxo $s - t$.

Consequência:

O fluxo máximo é menor ou igual a o corte mínimo.

Lema 6:

$$f(X) = f(X, \bar{X}) - f(\bar{X}, X)$$
$$= \sum_{v \in X} f(v) = f(s).$$



Lema 7: Fluxo sat f

$$f(G) = f(X) = f(X, \bar{X}) - f(\bar{X}, X)$$
$$\leq f(X, \bar{X}) \leq c(X, \bar{X}).$$

Teorema: Fluxo máximo – corte mínimo

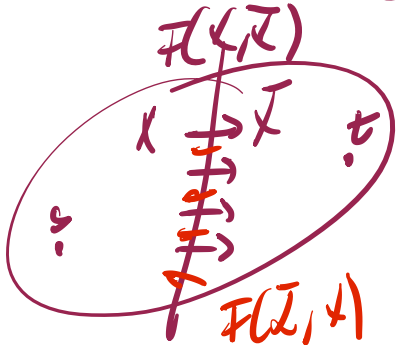
O valor do fluxo máximo entre dois vértices s e t é igual ao valor do corte mínimo.

Lema: corretude

Quando o algoritmo de Ford-Fulkerson termina, o valor do fluxo é máximo.

Prova: FF termina com fluxo f. $\Rightarrow$ Não existe um caminho s-t em G_f. Seja X o conjunto alcançável por s em G_f.

$\Rightarrow (X, \bar{X})$ é um corte. Se $x, t \notin X \Rightarrow t \in \bar{X}$.

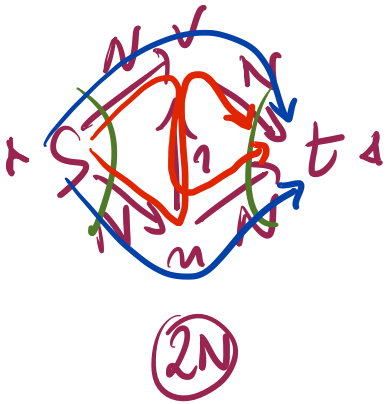


$a = (u, v) \in F(X, \bar{X})$: $f_a = c_a$ (arco: forward com cap. > 0)

$a = (u, v) \in F(\bar{X}, X)$: $f_a = 0$ (arco: backward cap. > 0)

$f(s) = f(X) = f(X, \bar{X}) - f(\bar{X}, X) = f(X, \bar{X}) = c(X, \bar{X})$ ((v, a))

- O número de iterações C pode ser alto, e existem grafos em que C iterações são necessárias.
- É possível que o algoritmo FF não termina para capacidades reais.
- Usando uma busca por profundidade para achar caminhos $s - t$ FF termina, mas é ineficiente.



1+1 = 2 it. / fluxo 2
⇒ 2N
2.