طراحی و تحلیل الگوریتم‌ها

مبحث یازدهم

مطالعه‌ی موضوعی الگوریتم‌ها

شبکه‌های شار

Flow Networks

کاظم فولادی
دانشکده مهندسی برق و کامپیوتر
دانشگاه تهران

http://courses.fouladi.ir/algorithm

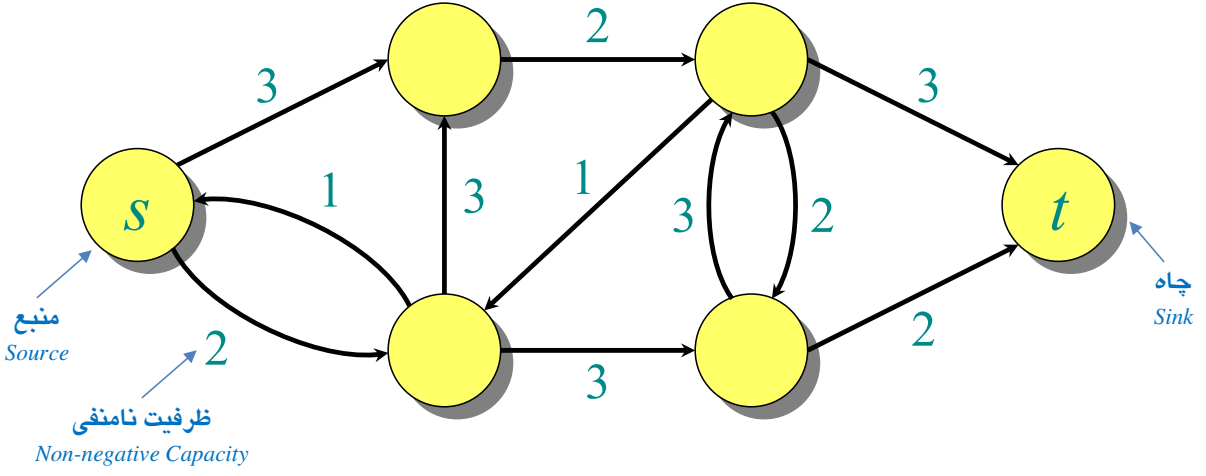
شبکه‌های شار

FLOW NETWORKS

چه میزان سیال می‌توان از منبع s به چاه t جریان پیدا کند؟

فرض می‌کنیم سیال با سرعت یکنواخت از رأس منبع s وارد شبکه می‌شود و از رأس چاه t از شبکه خارج می‌شود.

- **یال‌ها:** مجرای عبور سیال با ظرفیت مشخص
- **رأس‌ها:** محل تقاطع مجراها (با فرض اینکه هیچ سیالی در رأس‌ها ذخیره نمی‌شود.)



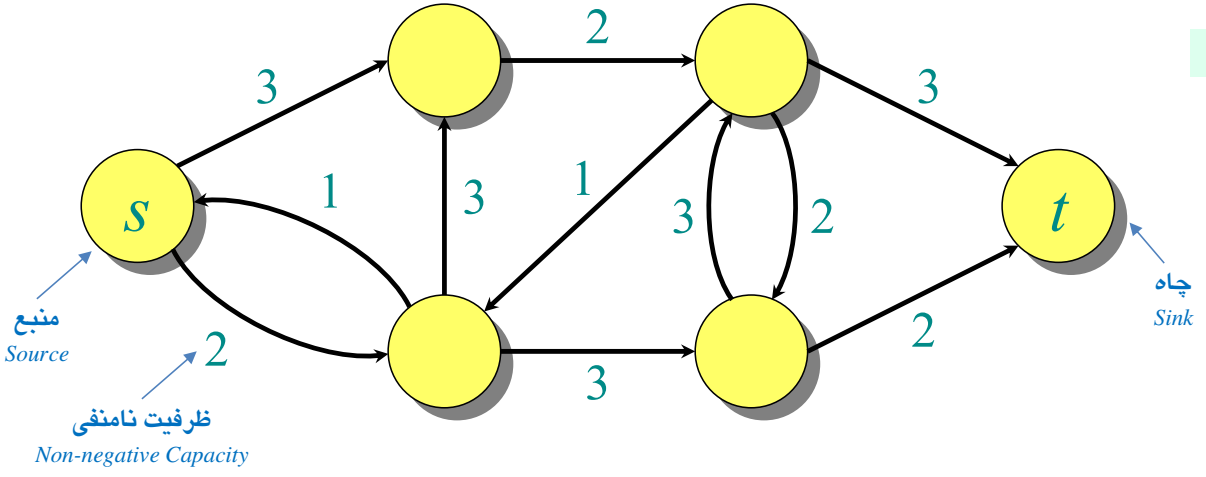
شبکه‌های شار

FLOW NETWORKS

شبکه‌ی شار
Flow Network

یک گراف جهت‌دار $G = (V,E)$
با دو رأس متمایز: **منبع** s (source) و **چاه** t (sink)
هر یال $(u,v) \in E$ دارای یک **ظرفیت** (capacity) نامنفی $c(u,v)$ است.
اگر $(u,v) \notin E$ آن‌گاه $c(u,v) = 0$ است.

مثال



شبکه‌های شار

شار مثبت

POSITIVE FLOW

شار مثبت
Positive Flow

یک شار مثبت روی $G = (V,E)$ تابعی به صورت
$p: V \times V \to \mathbb{R}$
است که قیدهای زیر را ارضا می‌کند:

قید ظرفیت
Capacity Constraint

$\forall u, v \in V \qquad 0 \le p(u, v) \le c(u, v)$

یعنی: شار مثبت هیچ‌گاه از ظرفیت بیشتر نمی‌شود.

بقای شار
Flow Conservation

$\forall u \in V - \{s, t\} \qquad \sum_{v \in V} p(u,v) - \sum_{v \in V} p(v,u) = 0$

یعنی: مجموع شار مثبت ورودی برابر با مجموع شار مثبت خروجی است.

شبکه‌های شار

مقدار یک شار

THE VALUE OF A FLOW

مقدار شار
Flow Value

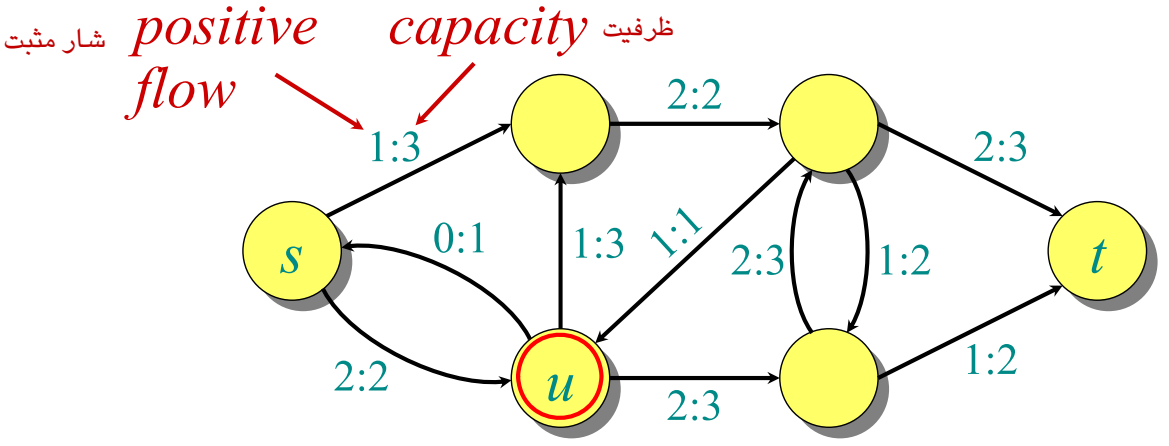
مقدار یک شار، شار خالص خروجی از منبع است:

$$\sum_{v \in V} p(s,v) - \sum_{v \in V} p(v,s)$$

شبکه‌های شار

شار بر روی یک شبکه

A FLOW ON A NETWORK



Flow conservation (like Kirchoff's current law):

- Flow into u is $2 + 1 = 3$.
- Flow out of u is $0 + 1 + 2 = 3$.

The value of this flow is $1 - 0 + 2 = 3$.

شبکه‌های شار

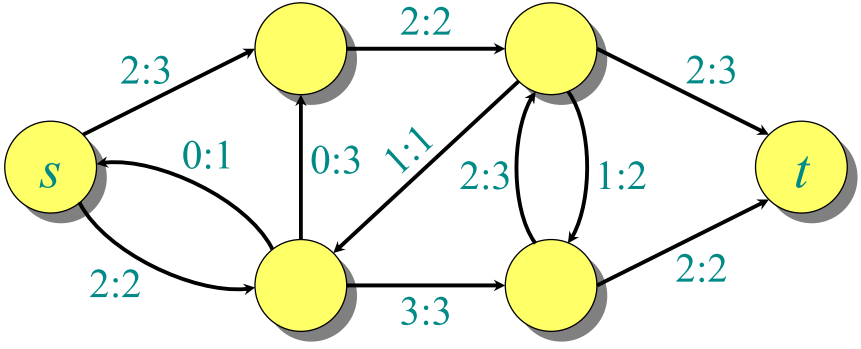
مسئله‌ی ماکزیمم شار

THE MAXIMUM-FLOW PROBLEM

مسئله‌ی ماکزیمم شار

با داشتن یک شبکه‌ی شار G، یک شار با ماکزیمم مقدار روی G بیابید.

مثال



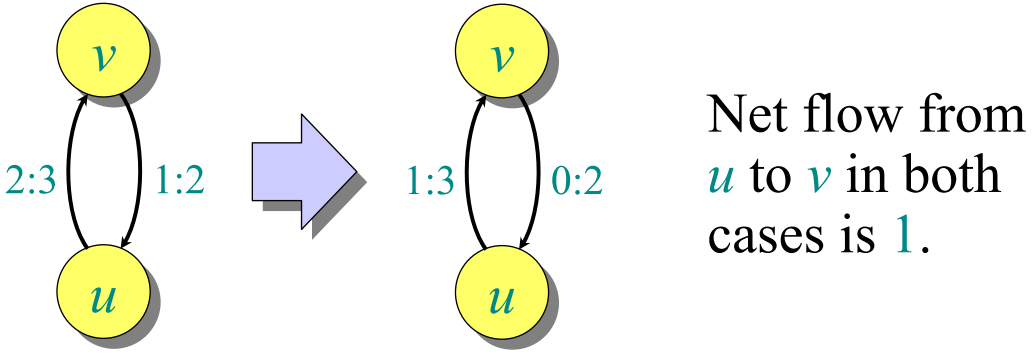
The value of the maximum flow is 4.

شبکه‌های شار

لغو شار

FLOW CANCELLATION

بدون از دست رفتن کلیت، شار مثبت از u به v می‌رود یا از v به u، اما نه هر دو.



قید ظرفیت و بقای شار، با این تبدیل حفظ می‌شود.

نکته‌ی شهودی: به شار باید به‌عنوان یک **نرخ** (rate) نگاه کنیم، نه یک **مقدار**.

شبکه‌های شار

شار خالص

NET FLOW

یک ساده‌سازی در نمادگذاری:
به‌جای کار با شار مثبت، با شار خالص بین دو رأس کار می‌کنیم.

شار (خالص)
(Net) Flow

یک شار خالص روی $G = (V,E)$ تابعی به صورت
$f: V \times V \to \mathbb{R}$
است که قیدهای زیر را ارضا می‌کند:

قید ظرفیت
Capacity Constraint

$\forall u, v \in V \qquad f(u, v) \le c(u, v)$

یعنی: شار خالص هیچ‌گاه از ظرفیت بیشتر نمی‌شود.

بقای شار
Flow Conservation

$\forall u \in V - \{s, t\} \qquad \sum_{v \in V} f(u,v) = 0$

یعنی: مجموع شار خالص وارد شده به یک گره صفر است.

تقارن مورّب
Skew Symmetry

$\forall u, v \in V \qquad f(u, v) = -f(v, u)$

Theorem. The two definitions are equivalent.

Proof. ($\Rightarrow$) Let $f(u, v) = p(u, v) - p(v, u)$.

- ***Capacity constraint:*** Since $p(u, v) \le c(u, v)$ and $p(v, u) \ge 0$, we have $f(u, v) \le c(u, v)$.
- ***Flow conservation:***

$$\sum_{v \in V} f(u,v) = \sum_{v \in V} \big(p(u,v) - p(v,u)\big)$$
$$= \sum_{v \in V} p(u,v) - \sum_{v \in V} p(v,u)$$

- ***Skew symmetry:***

$$f(u, v) = p(u, v) - p(v, u)$$
$$= -\big(p(v, u) - p(u, v)\big)$$
$$= -f(v, u).$$

اثبات معادل بودن دو تعریف «شار مثبت» و «شار خالص» (۲ از ۲)

($\Leftarrow$) Let

$$p(u, v) = \begin{cases} f(u, v) & \text{if } f(u, v) > 0, \\ 0 & \text{if } f(u, v) \le 0. \end{cases}$$

- ***Capacity constraint:*** By definition, $p(u, v) \ge 0$. Since $f(u, v) \le c(u, v)$, it follows that $p(u, v) \le c(u, v)$.
- ***Flow conservation:*** If $f(u, v) > 0$, then $p(u, v) - p(v, u) = f(u, v)$. If $f(u, v) \le 0$, then $p(u, v) - p(v, u) = -f(v, u) = f(u, v)$ by skew symmetry. Therefore,

$$\sum_{v \in V} p(u,v) - \sum_{v \in V} p(v,u) = \sum_{v \in V} f(u,v). \quad \square$$

شبکه‌های شار

مقدار یک شار

مقدار یک شار
Flow Value

مقدار یک شار f که با $|f|$ نشان داده می‌شود، به‌صورت زیر محاسبه می‌شود:

$$|f| = \sum_{v \in V} f(s,v)$$
$$= f(s,V).$$

نمادگذاری مجموع ضمنی

مجموعه در یک فرمول حسابی به معنی
مجموع‌گیری روی عناصر آن مجموعه است.

$\sum_{v \in V} f(u,v) = 0$ ⇨ **Example** — flow conservation: $f(u, V) = 0$ for all $u \in V - \{s, t\}$.

شبکه‌های شار

ماکزیمم شار

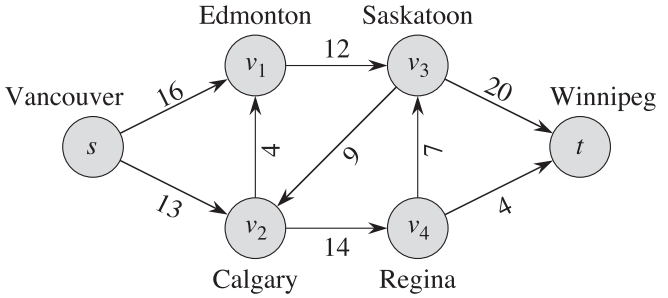
ماکزیمم شار
Maximum Flow

بزرگ‌ترین مقدار شار f ممکن روی شبکه

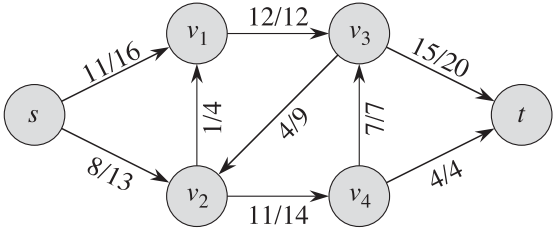
$$\left|f^*\right| = \max\{f : f \text{ is a flow on } G\}$$

شبکه‌های شار

مثال



(a)



(b)

یک شار f روی G با مقدار $|f| = 19$

یک شبکه‌ی شار $G = (V,E)$

کارخانه ای در ونکوور (منبع) و
انباری در وین‌نیپگ (چاه) قرار دارد.
شرکت حمل و نقل باید جعبه‌ها را از طریق شهرهای میانی منتقل کند، اما تعداد جعبه‌ی قابل انتقال بین هر دو شهر در طول یک روز محدود است (عدد روی یال).

شبکه‌های شار

خصوصیات ساده‌ی شار

SIMPLE PROPERTIES OF FLOW

Lemma.

- $f(X, X) = 0$,
- $f(X, Y) = -f(Y, X)$,
- $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$ if $X \cap Y = \varnothing$. □

Theorem. $|f| = f(V, t)$.

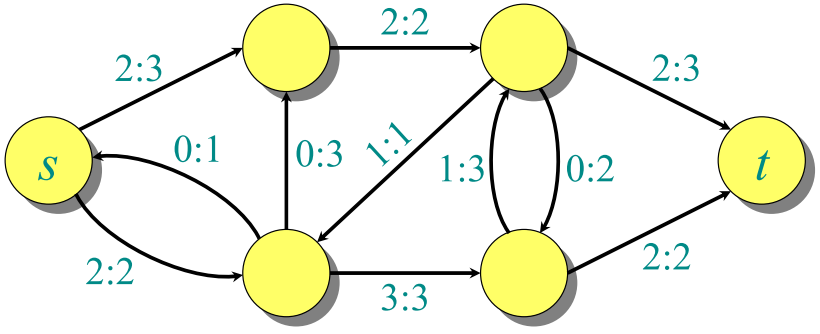
Proof.

$$\begin{aligned} |f| &= f(s, V) \\ &= f(V, V) - f(V-s, V) \qquad \textit{Omit braces.} \\ &= f(V, V-s) \\ &= f(V, t) + f(V, V-s-t) \\ &= f(V, t). \end{aligned}$$ □

شبکه‌های شار

شار وارد به چاه

FLOW INTO THE SINK



$$|f| = f(s, V) = 4 \qquad f(V, t) = 4$$

مقدار یک شار = شار خروجی از منبع = شار ورودی به چاه

شبکه‌های شار

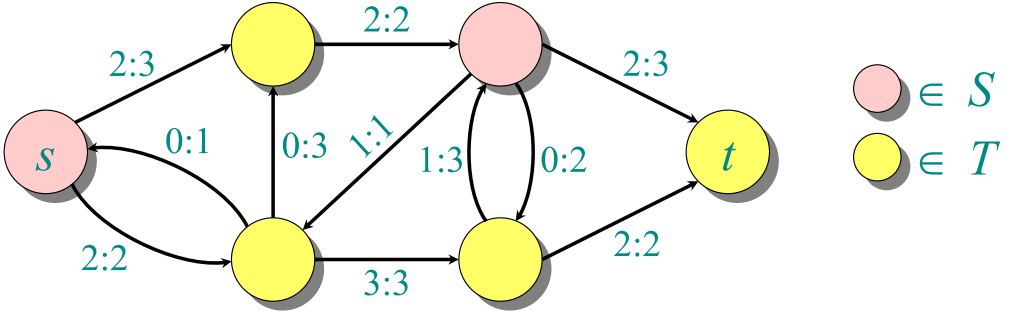
برش‌ها

CUTS

برش
Cut

یک برش (S,T) از یک شبکه‌ی شار $G = (V,E)$،
یک افراز از V است که در آن $s \in S$ و $t \in T$.

اگر f یک شار بر روی G باشد، $f(S,T)$ **شار گذرنده از برش** است.



$$f(S, T) = (2 + 2) + (-2 + 1 - 1 + 2) = 4$$

شبکه‌های شار

مشخصه‌ی دیگر مقدار شار

لم. برای هر شار f و هر برش (S,T) داریم

$$|f| = f(S, T)$$

Proof.

$$\begin{aligned} f(S, T) &= f(S, V) - f(S, S) \\ &= f(S, V) \\ &= f(s, V) + f(S-s, V) \\ &= f(s, V) \\ &= |f|. \end{aligned}$$

□

شبکه‌های شار

ظرفیت یک برش

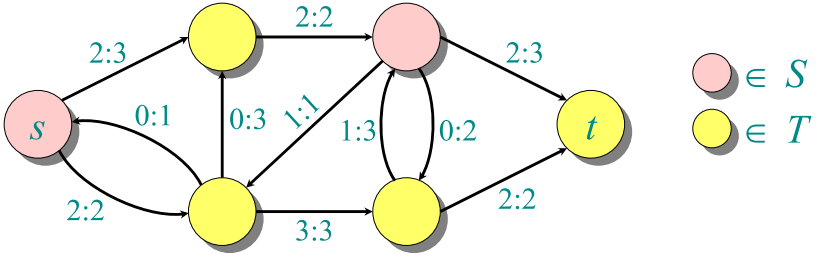
CAPACITY OF A CUT

ظرفیت یک برش
Capacity of a Cut

ظرفیت یک برش (S,T) برابر است با $c(S,T)$:
مجموع ظرفیت یال‌هایی که رأس مبدأ آنها در S و رأس مقصد آنها در T است.

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v)$$

Definition. The ***capacity of a cut*** (S, T) is $c(S, T)$.



$$c(S, T) = (3 + 2) + (1 + 2 + 3) = 11$$

برش می‌نیمم = برشی که دارای کمترین ظرفیت نسبت به سایر برش‌ها باشد.

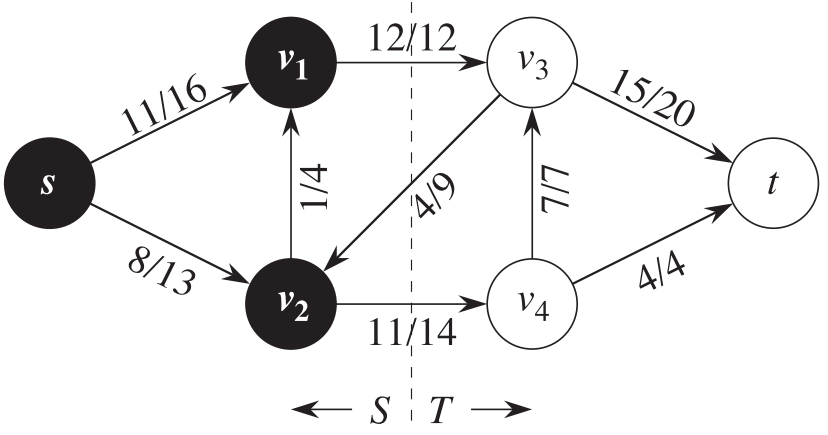
شبکه‌های شار

شار یک برش

CAPACITY OF A CUT

شار یک برش
Flow of a Cut

شار یک برش (S,T) برابر است با $f(S,T)$:
مجموع شار یال‌هایی که رأس مبدأ آنها در S و رأس مقصد آنها در T است.



برای محاسبه‌ی شار یک برش، شار منفی بین رأس‌ها را در نظر می‌گیریم.

$$\begin{aligned} f(S,T) &= f(v_1,v_2) + f(v_2,v_3) + f(v_2,v_4) \\ &= 12 + (-4) + 11 \\ &= 19 \end{aligned}$$

$$f(S,T) \leq c(S,T)$$

شبکه‌های شار

کران بالا بر روی ماکزیمم مقدار شار

UPPER BOUND ON THE MAXIMUM FLOW VALUE

قضیه	مقدار هر شار به ظرفیت هر برش محدود می‌شود.

Proof.

$$\begin{aligned} |f| &= f(S,T) \\ &= \sum_{u \in S} \sum_{v \in T} f(u,v) \\ &\le \sum_{u \in S} \sum_{v \in T} c(u,v) \\ &= c(S,T). \end{aligned}$$

□

شبکه‌های شار

شبکه‌ی مانده

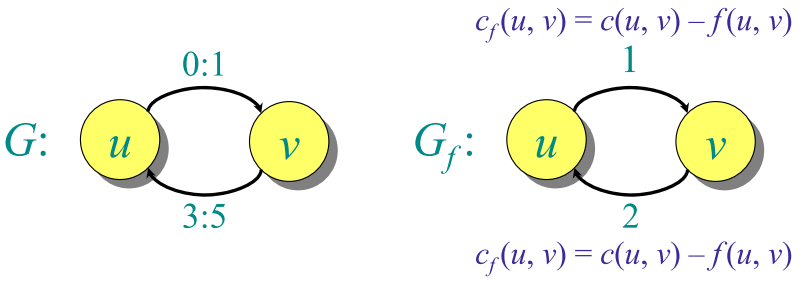
RESIDUAL NETWORK

شبکه‌ی مانده
Residual Network

فرض کنید f یک شار روی $G = (V,E)$ باشد.
در این صورت شبکه‌ی مانده‌ی $G_f = (V,E_f)$ گرافی است با ظرفیت‌های مانده‌ی اکیداً مثبت

$$c_f(u, v) = c(u, v) - f(u, v) > 0$$

یال‌های موجود در E_f شار بیشتری می‌پذیرند.



Lemma. $|E_f| \le 2|E|$.

شبکه‌های شار

مسیرهای افزوده

AUGMENTING PATH

مسیرهای افزوده
Augmenting Path

هر مسیر از s به t در G_f یک مسیر افزوده در G در نسبت با f است.

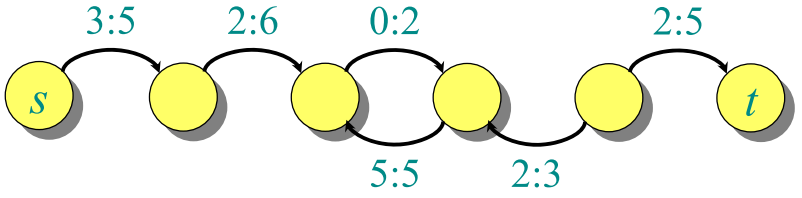
مقدار شار در طول یک مسیر افزوده p می‌تواند به اندازه‌ی $c_f(p)$ افزایش یابد:

$$c_f(p) = \min_{(u,v)\in p} \{c_f(u,v)\}$$

$c_f(p)$ = می‌نیمم ظرفیت یال‌های متعلق به مسیر افزوده p

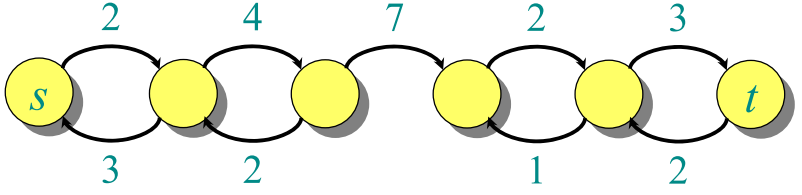
Ex.:

G:



$c_f(p) = 2$

G_f:

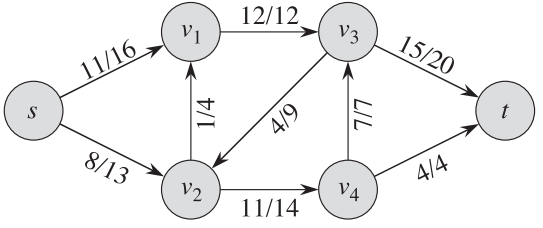


شبکه‌های شار

مسیرهای افزوده: مثال

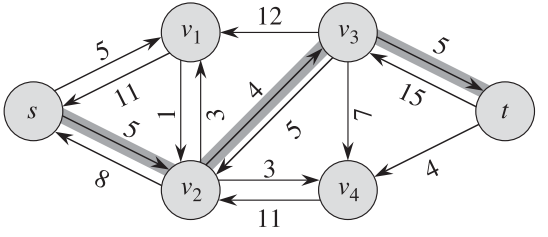
AUGMENTING PATH

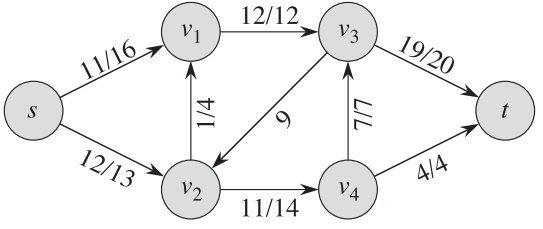
۱) شبکه‌ی شار G با جریان f



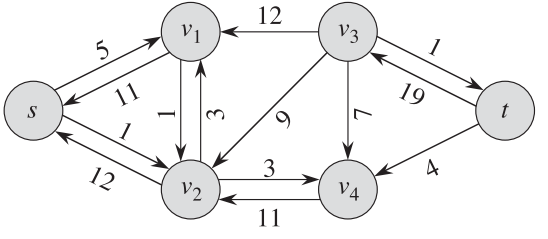
$c_f(p) = \min\{5, 4, 5\} = 4$

۲) شبکه‌ی مانده با مسیر افزوده برجسته شده
(ظرفیت مانده = ۴)





۳) شار درون G که در مسیر افزوده
با مقدار ظرفیت مانده افزایش یافته است.



۴) شبکه‌ی مانده ایجاد شده توسط شار قسمت (۳)

شبکه‌های شار

قضیه: شار ماکزیمم، برش می‌نیمم

MAX-FLOW, MIN-CUT THEOREM

قضیه

گزاره‌های زیر معادل هستند:

۱) f یک شار ماکزیمم است.

۲) f هیچ مسیر افزوده‌ای را نمی‌پذیرد.

۳) f به‌گونه‌ای است که برای حداقل یک برش (S,T) داریم $|f| = c(S,T)$

شبکه‌های شار

قضیه: شار ماکزیمم، برش می‌نیمم: اثبات (۱ از ۲)

MAX-FLOW, MIN-CUT THEOREM

قضیه

گزاره‌های زیر معادل هستند:

۱) f یک شار ماکزیمم است.

۲) f هیچ مسیر افزوده‌ای را نمی‌پذیرد.

۳) f به‌گونه‌ای است که برای حداقل یک برش (S,T) داریم $|f| = c(S,T)$

Proof.

(*1*) $\Rightarrow$ (*2*): Since $|f| \leq c(S, T)$ for any cut (S, T) (by the corollary) the assumption that $|f| = c(S, T)$ implies that f is a maximum flow.

(*2*) $\Rightarrow$ (*3*): If there were an augmenting path, the flow value could be increased, contradicting the maximality of f.

شبکه‌های شار

قضیه: شار ماکزیمم، برش می‌نیمم: اثبات (۲ از ۲)

MAX-FLOW, MIN-CUT THEOREM

قضیه

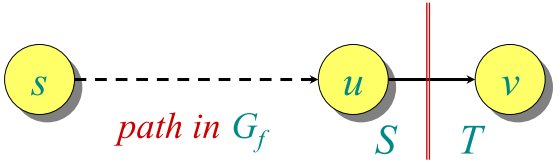
گزاره‌های زیر معادل هستند:

۱) f یک شار ماکزیمم است.

۲) f هیچ مسیر افزوده‌ای را نمی‌پذیرد.

۳) f به‌گونه‌ای است که برای حداقل یک برش (S,T) داریم $|f| = c(S,T)$

$(3) \Rightarrow (1)$: Suppose that f admits no augmenting paths. Define $S = \{v \in V :$ there exists a path in G_f from s to $v\}$, and let $T = V - S$. Observe that $s \in S$ and $t \in T$, and thus (S, T) is a cut. Consider any vertices $u \in S$ and $v \in T$.



We must have $c_f(u, v) = 0$, since if $c_f(u, v) > 0$, then $v \in S$, not $v \in T$ as assumed. Thus, $f(u, v) = c(u, v)$, since $c_f(u, v) = c(u, v) - f(u, v)$. Summing over all $u \in S$ and $v \in T$ yields $f(S, T) = c(S, T)$, and since $|f| = f(S, T)$, the theorem follows. □

شبکه‌های شار

الگوریتم ماکزیمم شار فورد ـ فولکرسون

Algorithm:

$f[u, v] \leftarrow 0$ for all $u, v \in V$

while an augmenting path p in G wrt f exists

do augment f by $c_f(p)$

FORD-FULKERSON-METHOD(G, s, t)

1 initialize flow f to 0
2 **while** there exists an augmenting path p in the residual network G_f
3 augment flow f along p
4 **return** f

- در ابتدا، شار یال‌ها را مساوی صفر قرار می‌دهیم.
- در حلقه‌ی تکرار، مقدار شار را با یافتن مسیرهای افزوده افزایش می‌دهیم.
  (مسیر افزوده: مسیری که در راستای آن می‌توان شار بیشتری انتقال داد.)
- تکرار تا زمانی ادامه می‌یابد که مسیر افزوده‌ی دیگری وجود نداشته باشد.

الگوریتم ماکزیمم شار فورد ـ فولکرسون: مثال (۱ از ۸)

Algorithm:

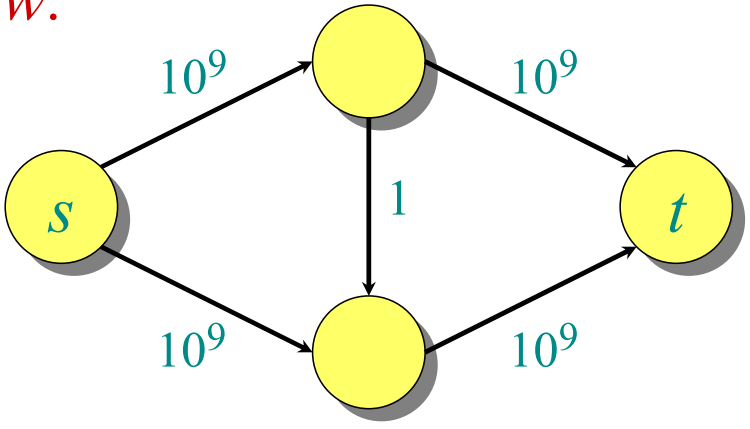
$f[u, v] \leftarrow 0$ for all $u, v \in V$

while an augmenting path p in G wrt f exists

do augment f by $c_f(p)$

Can be slow:

G:



الگوریتم ماکزیمم شار فورد ـ فولکرسون: مثال (۲ از ۸)

Algorithm:

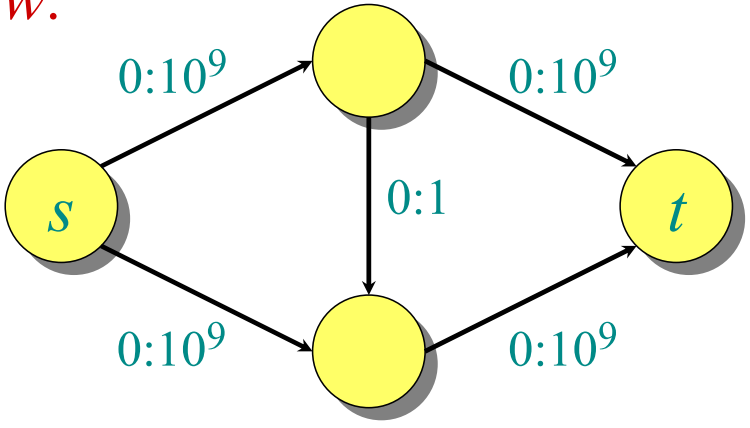
$f[u, v] \leftarrow 0$ for all $u, v \in V$

while an augmenting path p in G wrt f exists

do augment f by $c_f(p)$

Can be slow:

G:



Algorithm:

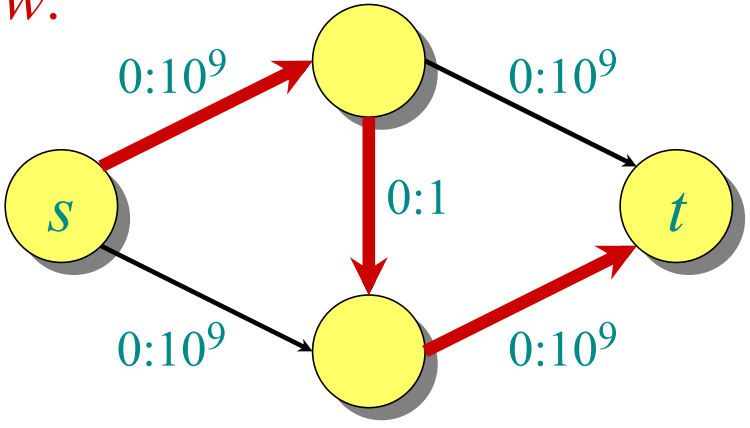
$f[u, v] \leftarrow 0$ for all $u, v \in V$

while an augmenting path p in G wrt f exists

do augment f by $c_f(p)$

Can be slow:

G:



الگوریتم ماکزیمم شار فورد ـ فولکرسون: مثال (۴ از ۸)

Algorithm:

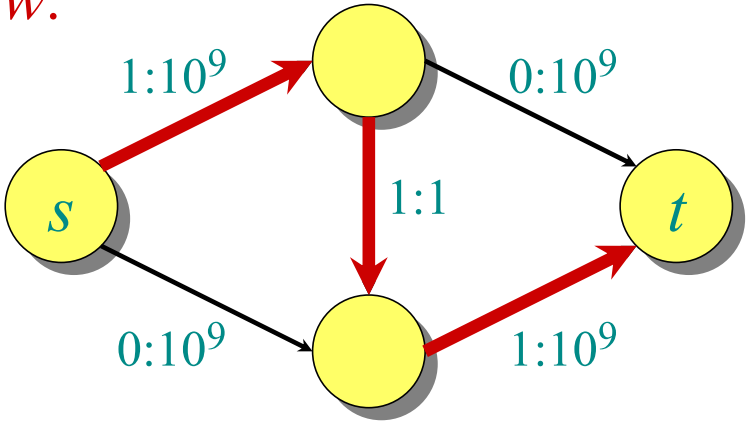
$f[u, v] \leftarrow 0$ for all $u, v \in V$

while an augmenting path p in G wrt f exists

do augment f by $c_f(p)$

Can be slow:

G:



Algorithm:

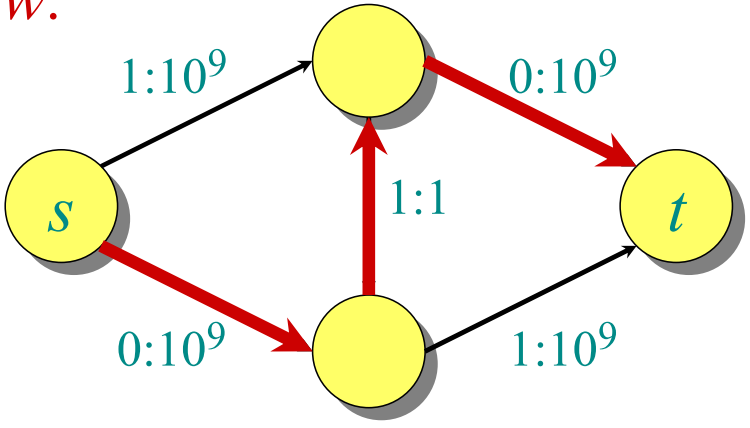
$f[u, v] \leftarrow 0$ for all $u, v \in V$

while an augmenting path p in G wrt f exists

do augment f by $c_f(p)$

Can be slow:

G:



Algorithm:

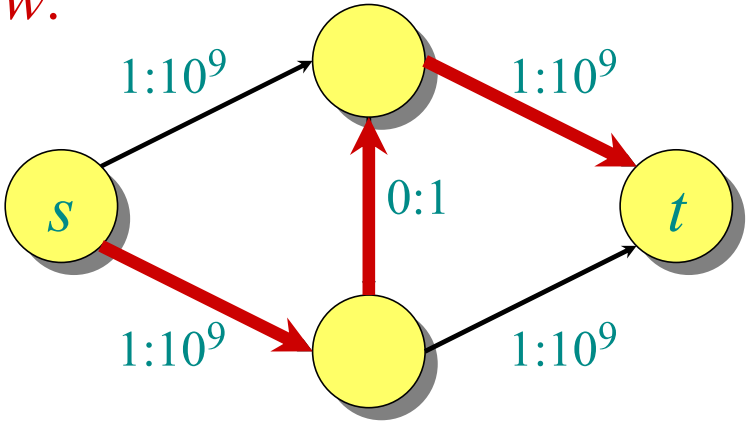
$f[u, v] \leftarrow 0$ for all $u, v \in V$

while an augmenting path p in G wrt f exists

do augment f by $c_f(p)$

Can be slow:

G:



Algorithm:

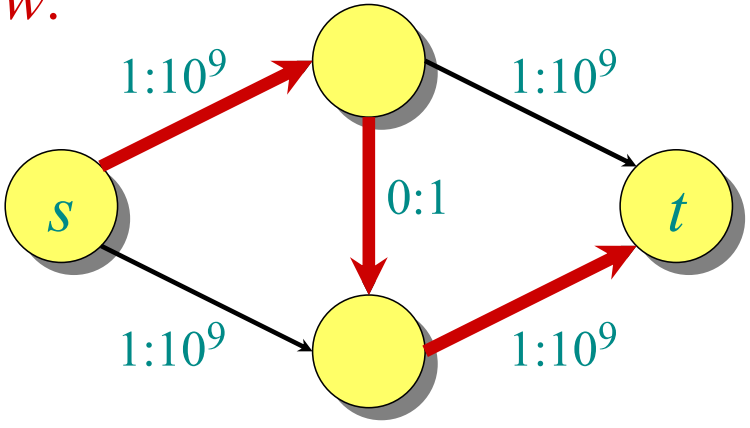
$f[u, v] \leftarrow 0$ for all $u, v \in V$

while an augmenting path p in G wrt f exists

do augment f by $c_f(p)$

Can be slow:

G:



Algorithm:

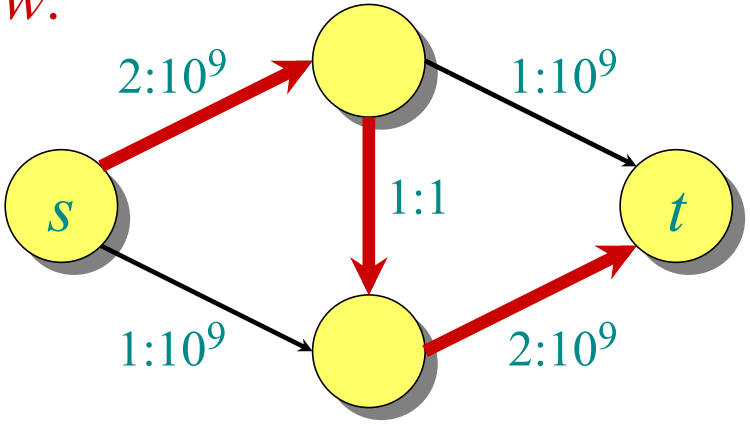
$f[u, v] \leftarrow 0$ for all $u, v \in V$

while an augmenting path p in G wrt f exists

do augment f by $c_f(p)$

Can be slow:

G:



2 billion iterations on a graph with 4 vertices!

شبکه‌های شار

الگوریتم ماکزیمم شار فورد ـ فولکرسون: شبه‌کد

```
FORD-FULKERSON(G, s, t)
1  for each edge (u, v) ∈ E[G]
2       do f[u, v] ← 0
3          f[v, u] ← 0
4  while there exists a path p from s to t in the residual network G_f
5       do c_f(p) ← min {c_f(u, v) : (u, v) is in p}
6          for each edge (u, v) in p
7               do f[u, v] ← f[u, v] + c_f(p)
8                  f[v, u] ← −f[u, v]
```

۱) **شروع:** روی شبکه G شار یال‌ها را مساوی صفر قرار می‌دهیم.
۲) شبکه‌ی مانده‌ی G_f فعلی را محاسبه می‌کنیم.
۳) در شبکه‌ی مانده‌ی G_f فعلی ، به دنبال یک مسیر افزوده می‌گردیم (p)؛ اگر نبود به گام (۶) می‌رویم.
۴) ظرفیت مانده‌ی مسیر p را محاسبه می‌کنیم $(c_f(p))$ و شار روی G را به‌اندازه‌ی آن افزایش می‌دهیم.
۵) وقتی شار بیشتر شد، یک شبکه‌ی جدید داریم؛ به گام (۲) می‌رویم.
۶) **پایان:** شار ماکزیمم به‌دست آمده است.

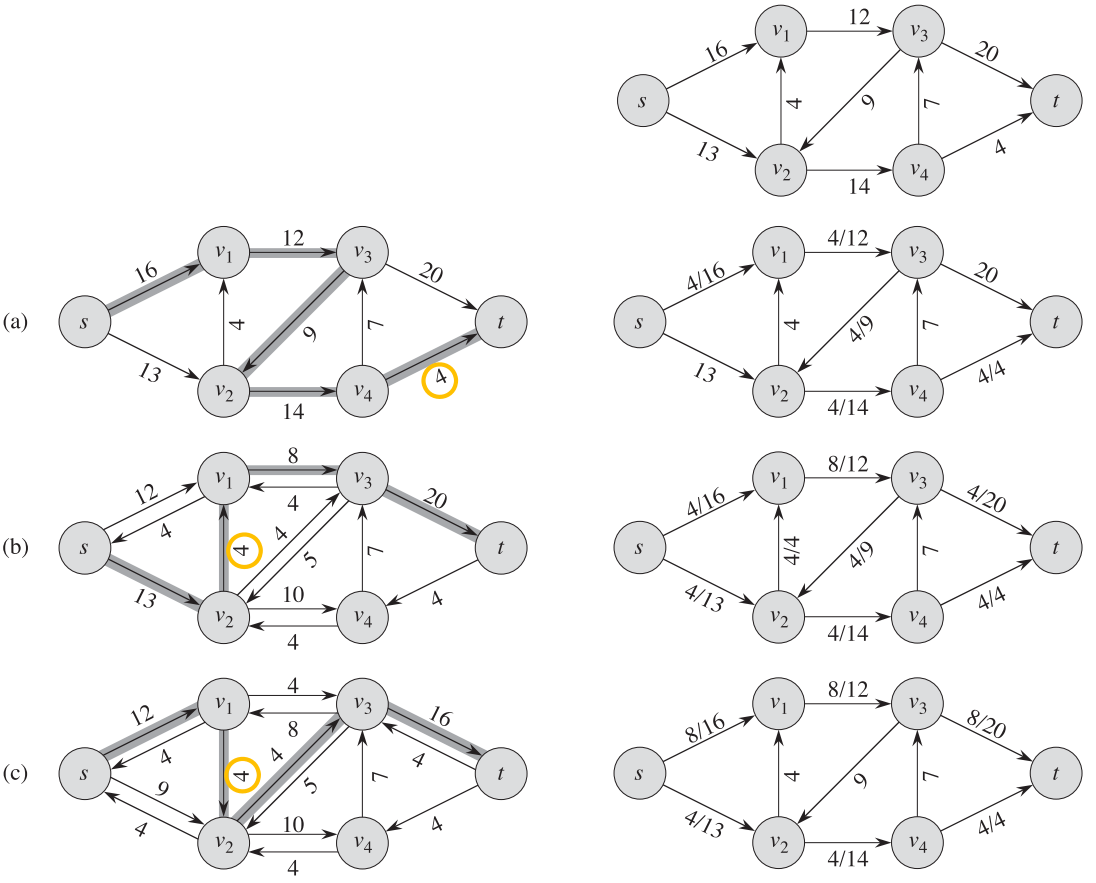
شبکه‌های شار

الگوریتم ماکزیمم شار فورد ـ فولکرسون

```
FORD-FULKERSON(G, s, t)
1  for each edge (u, v) ∈ G.E
2      (u, v).f = 0
3  while there exists a path p from s to t in the residual network G_f
4      c_f(p) = min{c_f(u, v) : (u, v) is in p}
5      for each edge (u, v) in p
6          if (u, v) ∈ E
7              (u, v).f = (u, v).f + c_f(p)
8          else (v, u).f = (v, u).f − c_f(p)
```

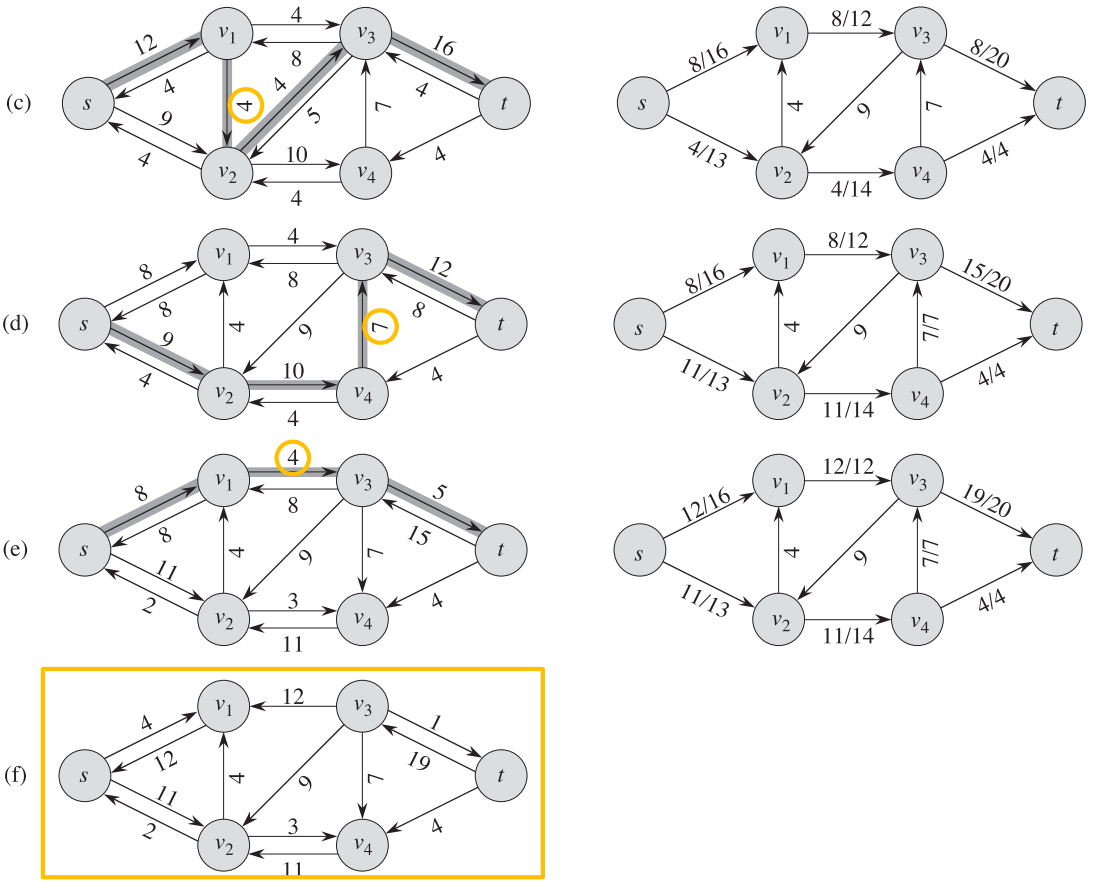
شبکه‌های شار

الگوریتم ماکزیمم شار فورد ـ فولکرسون: مثال (۱ از ۲)



شبکه‌های شار

الگوریتم ماکزیمم شار فورد ـ فولکرسون: مثال (۲ از ۲)



شبکه‌های شار

الگوریتم ماکزیمم شار فورد ـ فولکرسون: زمان اجرا

زمان اجرای این الگوریتم وابسته به روش تعیین مسیرهای افزوده است

```
FORD-FULKERSON(G, s, t)
1  for each edge (u, v) ∈ E[G]
2      do f[u, v] ← 0
3         f[v, u] ← 0
4  while there exists a path p from s to t in the residual network G_f
5      do c_f(p) ← min {c_f(u, v) : (u, v) is in p}
6         for each edge (u, v) in p
7             do f[u, v] ← f[u, v] + c_f(p)
8                f[v, u] ← −f[u, v]
```

$O(E)$

$O\left(E\left|f^*\right|\right)$

حداکثر تعداد تکرار حلقه while: $O(|f^*|)$
(زیرا مقدار شار در هر تکرار حداقل یک واحد افزایش می‌یابد.)

حلقه‌ی داخلی for: حداکثر به تعداد یال‌های شبکه $O(E)$ (اگر از BFS یا DFS برای یافتن یک مسیر در شبکه‌ی مانده استفاده کنیم.)

$$O\left(E + E\left|f^*\right|\right) = O\left(E\left|f^*\right|\right)$$

شبکه‌های شار

الگوریتم ماکزیمم شار ادموندز–کارپ

EDMONDS-KARP ALGORITHM

اگر محاسبه‌ی مسیر افزوده در الگوریتم فورد–فولکرسون را با BFS پیاده‌سازی کنیم،
(یعنی: مسیر افزوده، کوتاه‌ترین مسیر از s به t در شبکه‌ی مانده با یال‌های دارای وزن ۱ است)
روش فورد–فولکرسون که به این صورت پیاده می‌شود، ادموندز–کارپ نام دارد.

زمان اجرای این الگوریتم $O(VE^2)$ است.

(مسیرهای افزوده در زمان $O(VE)$ محاسبه می‌شوند.)

الگوریتم ادموندز–کارپ

EDMONDS-KARP ALGORITHM

Edmonds and Karp noticed that many people's implementations of Ford-Fulkerson augment along a ***breadth-first augmenting path***: a shortest path in G_f from s to t where each edge has weight 1. These implementations would always run relatively fast.

Since a breadth-first augmenting path can be found in $O(E)$ time, their analysis, which provided the first polynomial-time bound on maximum flow, focuses on bounding the number of flow augmentations.

(In independent work, Dinic also gave polynomial-time bounds.)

EDMONDS-KARP ALGORITHM: MONOTONICITY LEMMA

Lemma. Let $\delta(v) = \delta_f(s, v)$ be the breadth-first distance from s to v in G_f. During the Edmonds-Karp algorithm, $\delta(v)$ increases monotonically.

Proof. Suppose that f is a flow on G, and augmentation produces a new flow f'. Let $\delta'(v) = \delta_{f'}(s, v)$. We'll show that $\delta'(v) \geq \delta(v)$ by induction on $\delta(v)$. For the base case, $\delta'(s) = \delta(s) = 0$.

For the inductive case, consider a breadth-first path $s \to \cdots \to u \to v$ in $G_{f'}$. We must have $\delta'(v) = \delta'(u) + 1$, since subpaths of shortest paths are shortest paths. Certainly, $(u, v) \in E_{f'}$, and now consider two cases depending on whether $(u, v) \in E_f$.

EDMONDS-KARP ALGORITHM: MONOTONICITY LEMMA

Case 1

Case: $(u, v) \in E_f$.

We have

$$\begin{aligned} \delta(v) &\le \delta(u) + 1 && \text{(triangle inequality)} \\ &\le \delta'(u) + 1 && \text{(induction)} \\ &= \delta'(v) && \text{(breadth-first path),} \end{aligned}$$

and thus monotonicity of $\delta(v)$ is established.

EDMONDS-KARP ALGORITHM: MONOTONICITY LEMMA

Case 2

Case: $(u, v) \notin E_f$.

Since $(u, v) \in E_{f'}$, the augmenting path p that produced f' from f must have included (v, u). Moreover, p is a breadth-first path in G_f:

$$p = s \to \cdots \to v \to u \to \cdots \to t\ .$$

Thus, we have

$$\begin{aligned} \delta(v) &= \delta(u) - 1 && \text{(breadth-first path)} \\ &\le \delta'(u) - 1 && \text{(induction)} \\ &\le \delta'(v) - 2 && \text{(breadth-first path)} \\ &< \delta'(v)\ , \end{aligned}$$

thereby establishing monotonicity for this case, too. □

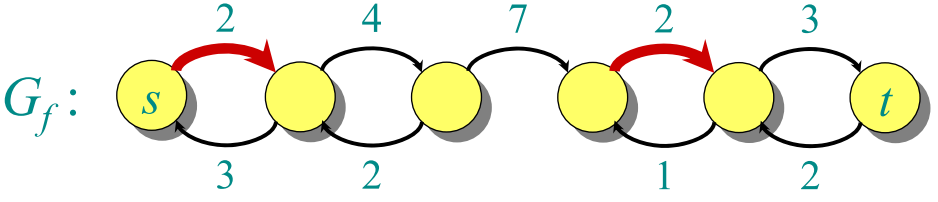
الگوریتم ادموندز–کارپ: شمارش تعداد شارهای افزوده (۱ از ۸)

EDMONDS-KARP ALGORITHM: COUNTING FLOW AUGMENTATIONS

Theorem. The number of flow augmentations in the Edmonds-Karp algorithm (Ford-Fulkerson with breadth-first augmenting paths) is $O(VE)$.

Proof. Let p be an augmenting path, and suppose that we have $c_f(u, v) = c_f(p)$ for edge $(u, v) \in p$. Then, we say that (u, v) is ***critical***, and it disappears from the residual graph after flow augmentation.

Example: $c_f(p) = 2$



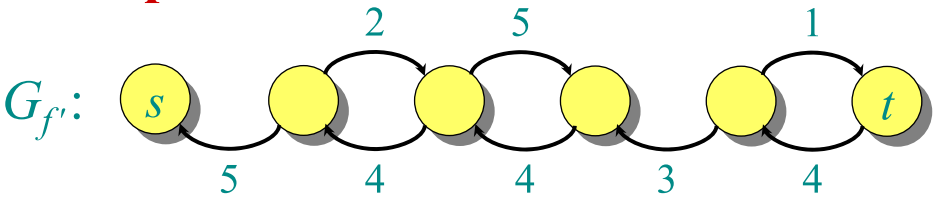
الگوریتم ادموندز–کارپ: شمارش تعداد شارهای افزوده (۲ از ۸)

EDMONDS-KARP ALGORITHM: COUNTING FLOW AUGMENTATIONS

Theorem. The number of flow augmentations in the Edmonds-Karp algorithm (Ford-Fulkerson with breadth-first augmenting paths) is $O(VE)$.

Proof. Let p be an augmenting path, and suppose that the residual capacity of edge $(u, v) \in p$ is $c_f(u, v) = c_f(p)$. Then, we say (u, v) is ***critical***, and it disappears from the residual graph after flow augmentation.

Example:



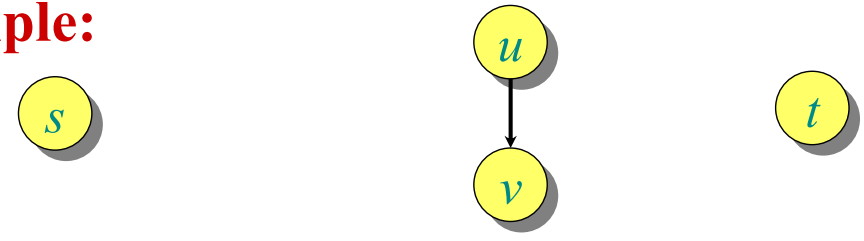
الگوریتم ادموندز–کارپ: شمارش تعداد شارهای افزوده (۳ از ۸)

EDMONDS-KARP ALGORITHM: COUNTING FLOW AUGMENTATIONS

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

Example:



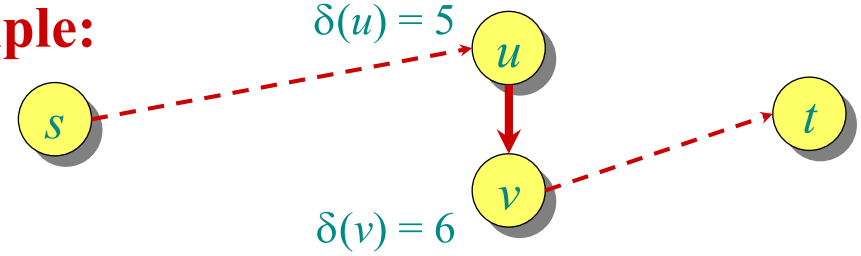
الگوریتم ادموندز–کارپ: شمارش تعداد شارهای افزوده (۴ از ۸)

EDMONDS-KARP ALGORITHM: COUNTING FLOW AUGMENTATIONS

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

Example:



شبکه‌های شار

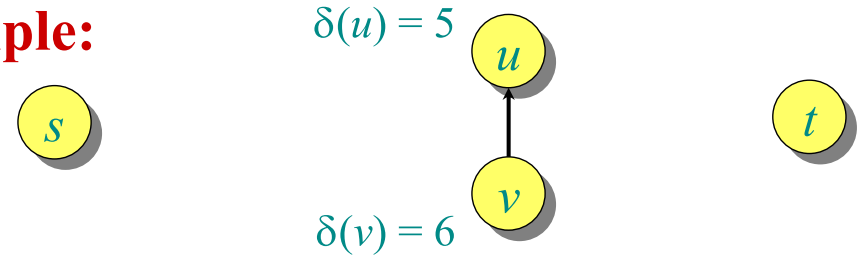
الگوریتم ادموندز–کارپ: شمارش تعداد شارهای افزوده (۵ از ۸)

EDMONDS-KARP ALGORITHM: COUNTING FLOW AUGMENTATIONS

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

Example:



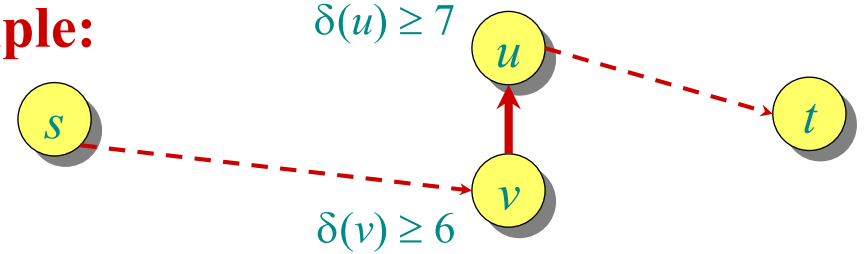
الگوریتم ادموندز–کارپ: شمارش تعداد شارهای افزوده (۶ از ۸)

EDMONDS-KARP ALGORITHM: COUNTING FLOW AUGMENTATIONS

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

Example:



شبکه‌های شار

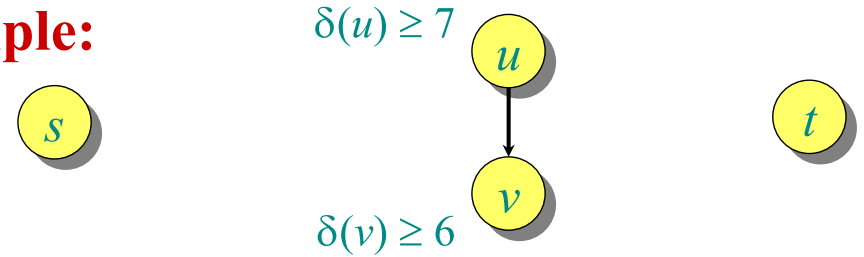
الگوریتم ادموندز–کارپ: شمارش تعداد شارهای افزوده (۷ از ۸)

EDMONDS-KARP ALGORITHM: COUNTING FLOW AUGMENTATIONS

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

Example:



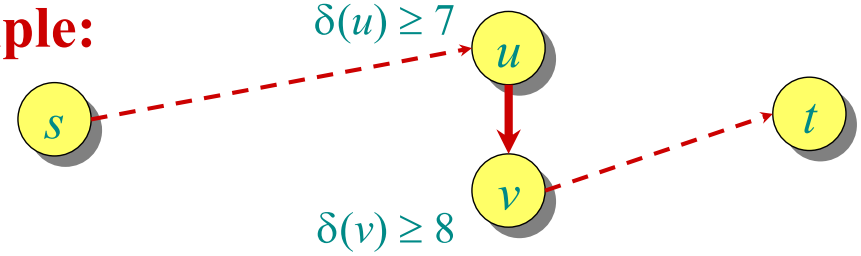
الگوریتم ادموندز–کارپ: شمارش تعداد شارهای افزوده (۸ از ۸)

EDMONDS-KARP ALGORITHM: COUNTING FLOW AUGMENTATIONS

The first time an edge (u, v) is critical, we have $\delta(v) = \delta(u) + 1$, since p is a breadth-first path. We must wait until (v, u) is on an augmenting path before (u, v) can be critical again. Let δ' be the distance function when (v, u) is on an augmenting path. Then, we have

$$\begin{aligned} \delta'(u) &= \delta'(v) + 1 && \text{(breadth-first path)} \\ &\geq \delta(v) + 1 && \text{(monotonicity)} \\ &= \delta(u) + 2 && \text{(breadth-first path).} \end{aligned}$$

Example:



EDMONDS-KARP ALGORITHM: RUNNING TIME

Distances start out nonnegative, never decrease, and are at most $|V| - 1$ until the vertex becomes unreachable. Thus, (u, v) occurs as a critical edge $O(V)$ times, because $\delta(v)$ increases by at least 2 between occurrences. Since the residual graph contains $O(E)$ edges, the number of flow augmentations is $O(VE)$. □

Corollary. The Edmonds-Karp maximum-flow algorithm runs in $O(VE^2)$ time.

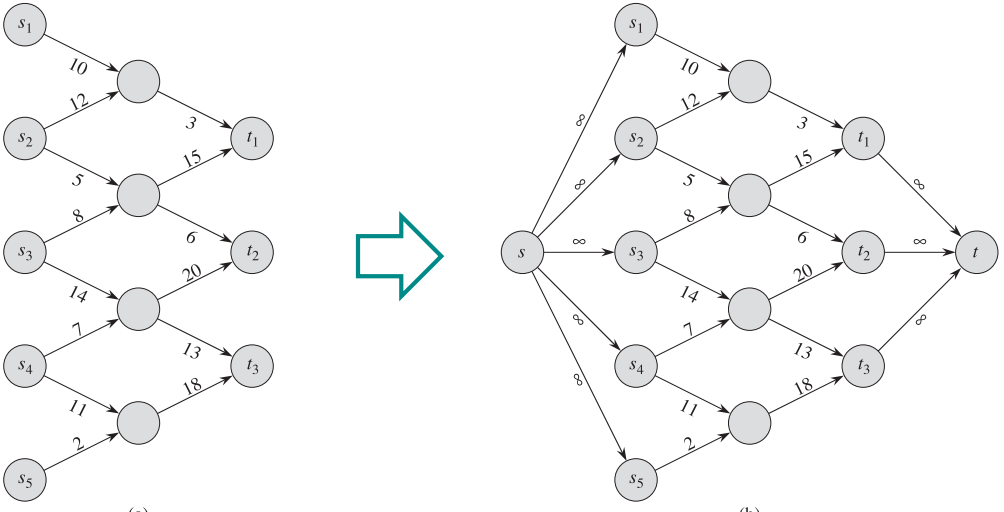
Proof. Breadth-first search runs in $O(E)$ time, and all other bookkeeping is $O(V)$ per augmentation. □

EDMONDS-KARP ALGORITHM: RUNNING TIME

- The asymptotically fastest algorithm to date for maximum flow, due to King, Rao, and Tarjan, runs in $O(VE \log_{E/(V \lg V)} V)$ time.
- If we allow running times as a function of edge weights, the fastest algorithm for maximum flow, due to Goldberg and Rao, runs in time $O(\min\{V^{2/3}, E^{1/2}\} \cdot E \lg (V^2/E + 2) \cdot \lg C)$, where C is the maximum capacity of any edge in the graph.

شبکه‌های شار

شبکه‌های دارای چند منبع و چند چاه



چند منبع – چند چاه
Multiple Source – Multiple Sink

تک منبع – تک چاه
Single Source – Single Sink

Converting a multiple-source, multiple-sink maximum-flow problem into a problem with a single source and a single sink. **(a)** A flow network with five sources $S = \{s_1, s_2, s_3, s_4, s_5\}$ and three sinks $T = \{t_1, t_2, t_3\}$. **(b)** An equivalent single-source, single-sink flow network. We add a supersource s and an edge with infinite capacity from s to each of the multiple sources. We also add a supersink t and an edge with infinite capacity from each of the multiple sinks to t.

تطابق دوبخشی ماکزیمم

MAXIMUM BIPARTITE MATCHING

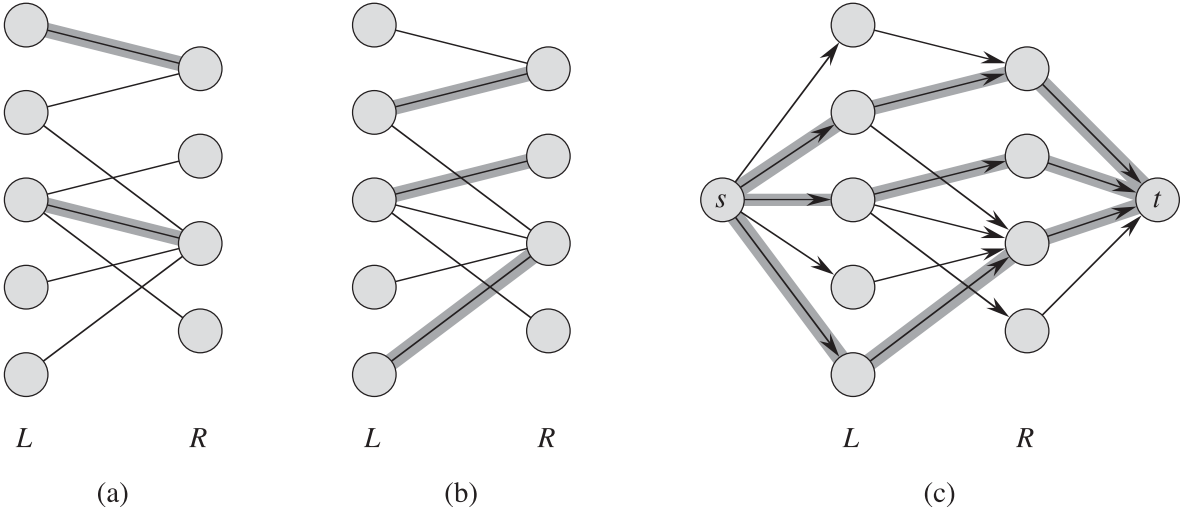


Figure 26.8 A bipartite graph $G = (V, E)$ with vertex partition $V = L \cup R$. **(a)** A matching with cardinality 2, indicated by shaded edges. **(b)** A maximum matching with cardinality 3. **(c)** The corresponding flow network G' with a maximum flow shown. Each edge has unit capacity. Shaded edges have a flow of 1, and all other edges carry no flow. The shaded edges from L to R correspond to those in the maximum matching from (b).