Fluxos em redes: o Problema do Fluxo máximo

Instância: uma rede (G, c) e vértices s e t

- $G = (V, E)$ um grafo dirigido
- s e $t \in V$ são dois vértices distinguidos
- $c\colon E \to \mathbb{R}_{\geq 0}$ é uma *função capacidade* nas arestas

Queremos: um (s,t)*-fluxo* $f\colon E \to \mathbb{R}_{\geq 0}$ em (G, c) de *valor* $\mathsf{val}(f)$ máximo

O método de Ford e Fulkerson

Ford–Fulkerson(G, c, s, t):

1. $f \equiv 0$
2. **enquanto** existe caminho aumentador P **faça**
   2.1 aumente f ao longo de P
3. **Devolva** f

O teorema do fluxo máximo e corte mínimo (maxflow-mincut)

- (s,t)-*corte*: corte $(S, V \setminus S)$ que separa s de t, isto é, $s \in S$ e $t \in V \setminus S$
- *capacidade de um corte*: $c(S, V \setminus S) = \sum_a c(a)$, onde a soma é sobre todo arco a que sai de S e entra em $V \setminus S$
- *corte mínimo*: corte que separa s e t e tem capacidade mínima
- *fluxo através de um corte*: $f(S, V \setminus S) = \sum_a f(a) - \sum_b f(b)$, onde a soma é sobre $a \in E(S, V \setminus S)$ e sobre $b \in E(V \setminus S, S)$

O teorema do fluxo máximo e corte mínimo

Teorema 1. *Sejam G, c, s e t como acima. Então*

$$\max_f \mathrm{val}(f) = \min_S c(S, V \setminus S),$$

onde o máximo é tomado sobre todos os (s,t)-fluxos e o mínimo é tomado sobre todos os cortes que separam s e t.

O teorema do fluxo máximo e corte mínimo

- Prova do teorema do fluxo máximo/corte mínimo: o método de Ford e Fulkerson
- *Caminho aumentador*: (s, t)-caminho (não necessariamente dirigido) em que os arcos que avançam no caminho não estão saturados e os arcos que retrocedem no caminho tem fluxo positivo

Implementação do método de Ford-Fulkerson

▷ Como encontrar um caminho aumentador?

Convenção

Dados: $G = (V, E)$ e s, $t \in V$. Até agora: (s,t)-fluxo $f : E \to \mathbb{R}_{\geq 0}$

Mais conveniente:

- $F : V \times V \to \mathbb{R}$
- Se $(x,y) \in E$ e $(y,x) \notin E$, então $F(x,y) = -F(y,x) = f(x,y)$.
- Se $(x,y) \in E$ e $(y,x) \in E$, então $F(x,y) = f(x,y) - f(y,x)$.
- Se $(x,y) \notin E$ e $(y,x) \notin E$, então $F(x,y) = 0$.

Escrevemos simplesmente f em vez de F.

Para capacidades: estendemos para $c : V \times V \to \mathbb{R}_{\geq 0}$, pondo $c(x,y) = 0$ se $(x,y) \notin E$.

Rede residual

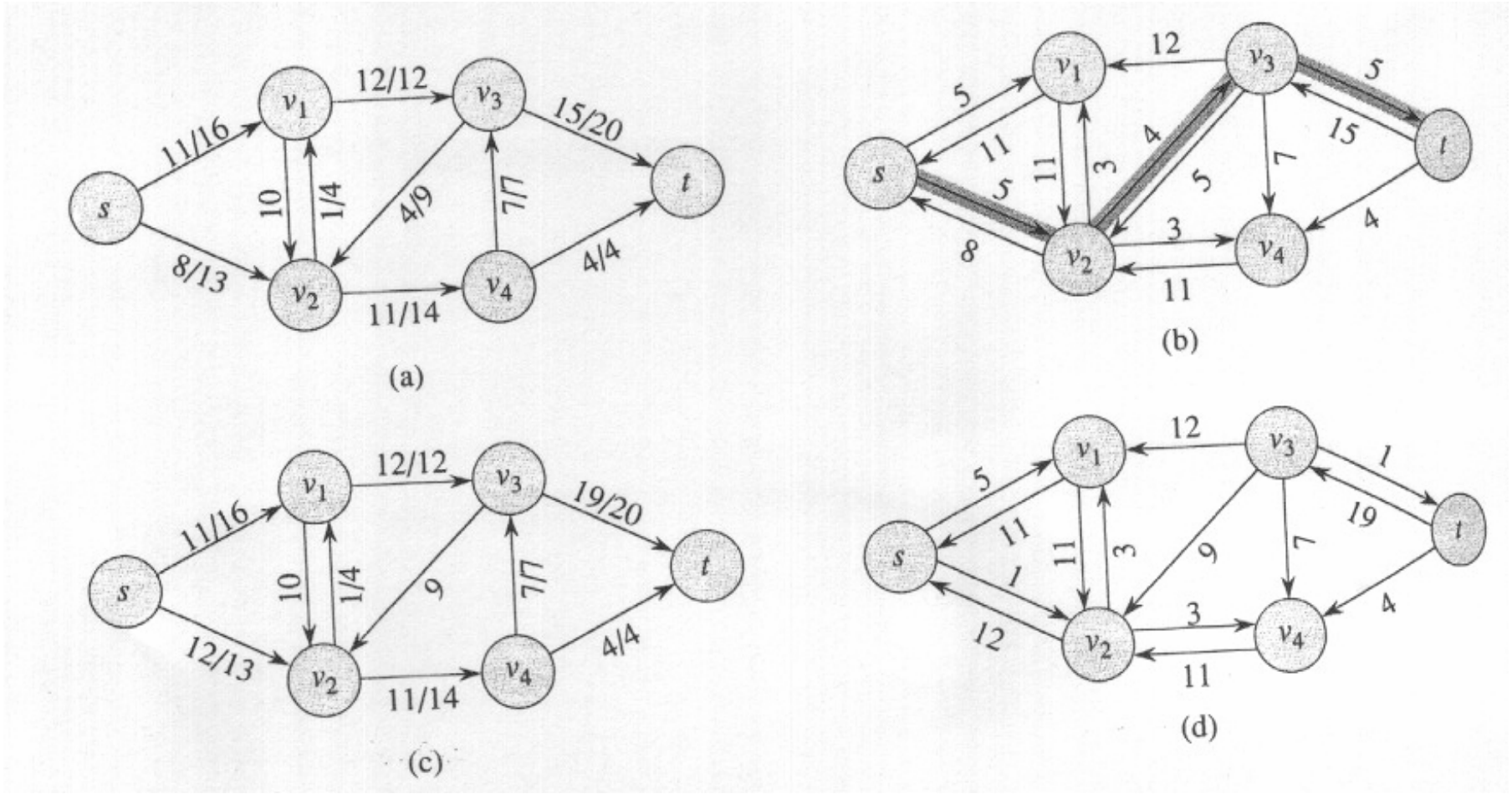
Dados $G = (V, E)$, s, $t \in V$ e um fluxo $f : V \times V \to \mathbb{R}$, pomos

$$c_f(x, y) = c(x, y) - f(x, y)$$

para todo $(x, y) \in V \times V$.

▷ Rede residual: rede $G_f = (V, E_f)$, onde

$$E_f = \{e \in E : c_f(e) > 0\}$$

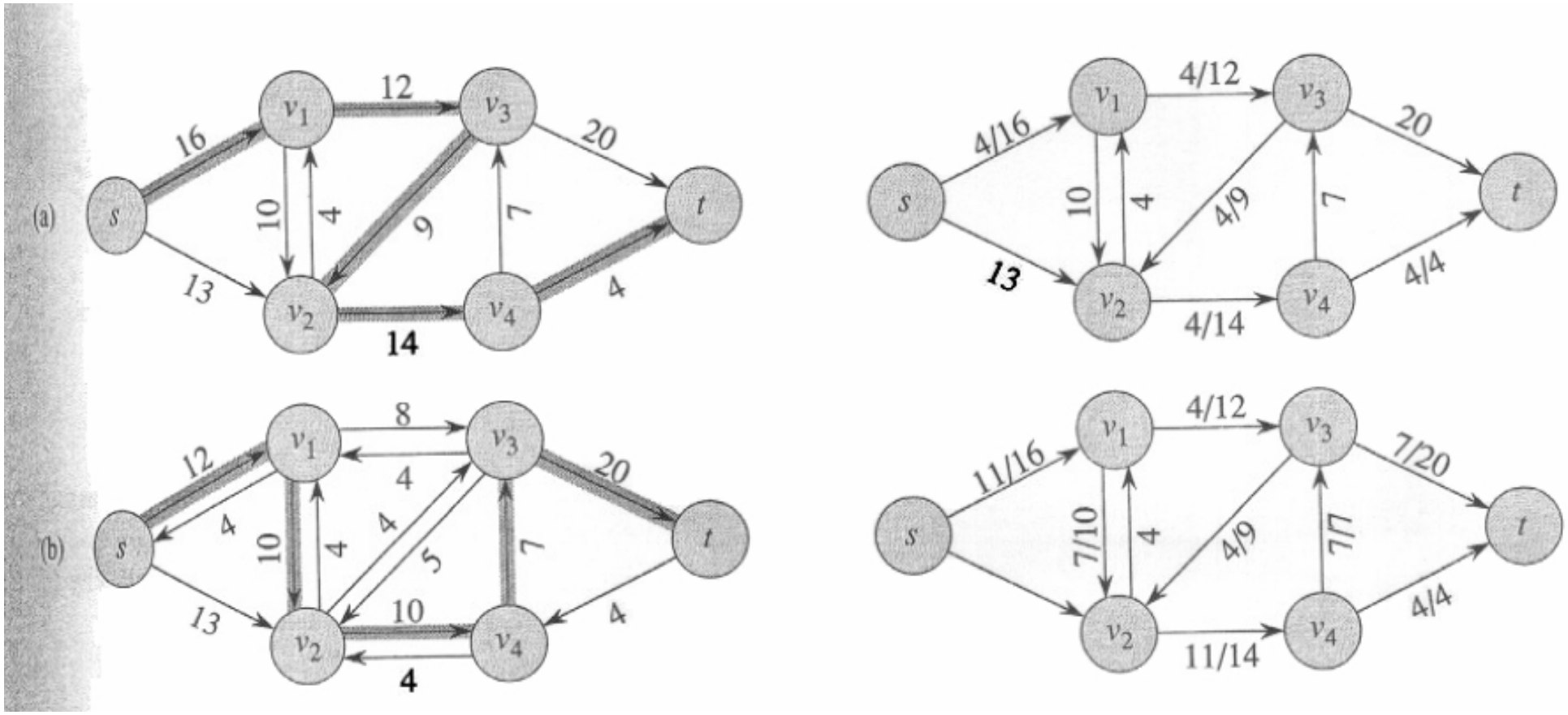


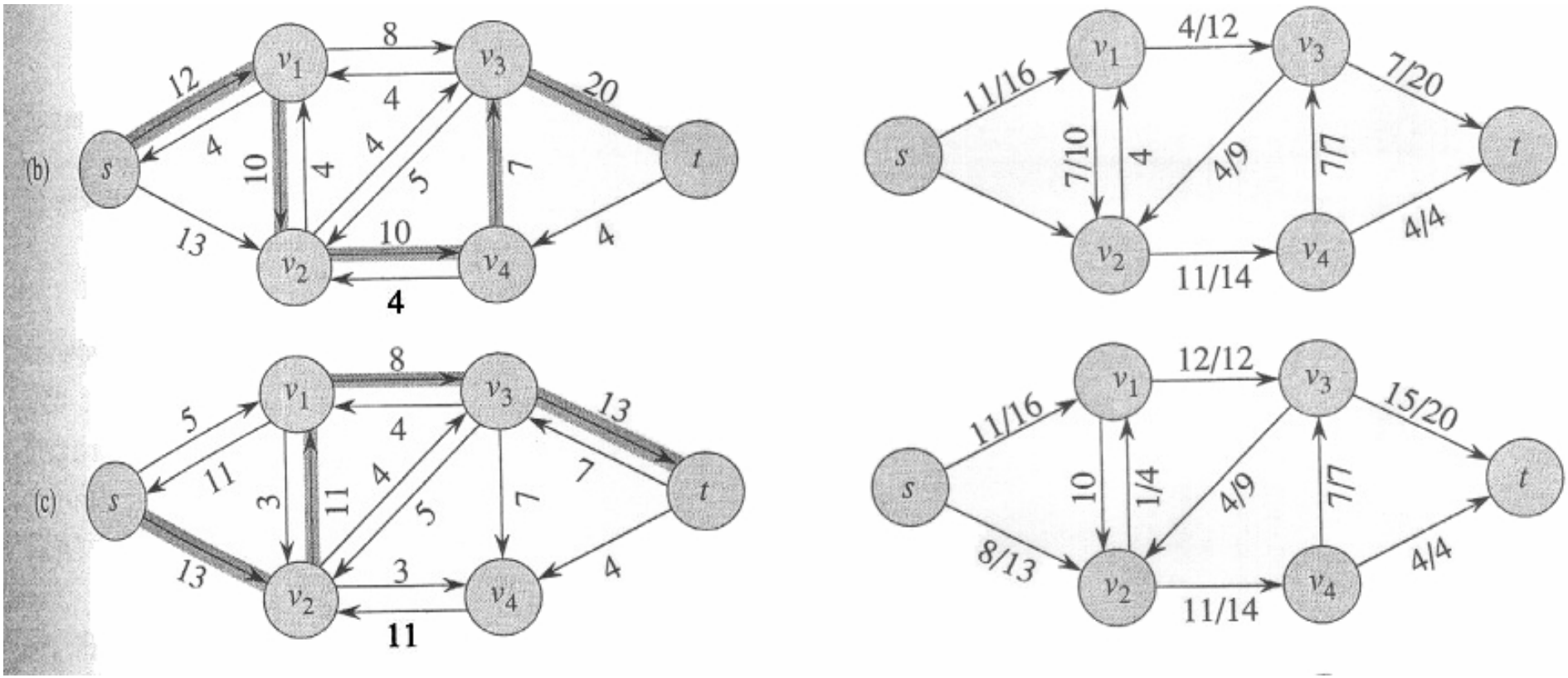
(a) fluxo, (b) rede residual e um caminho aumentador, (c) fluxo aumentado,

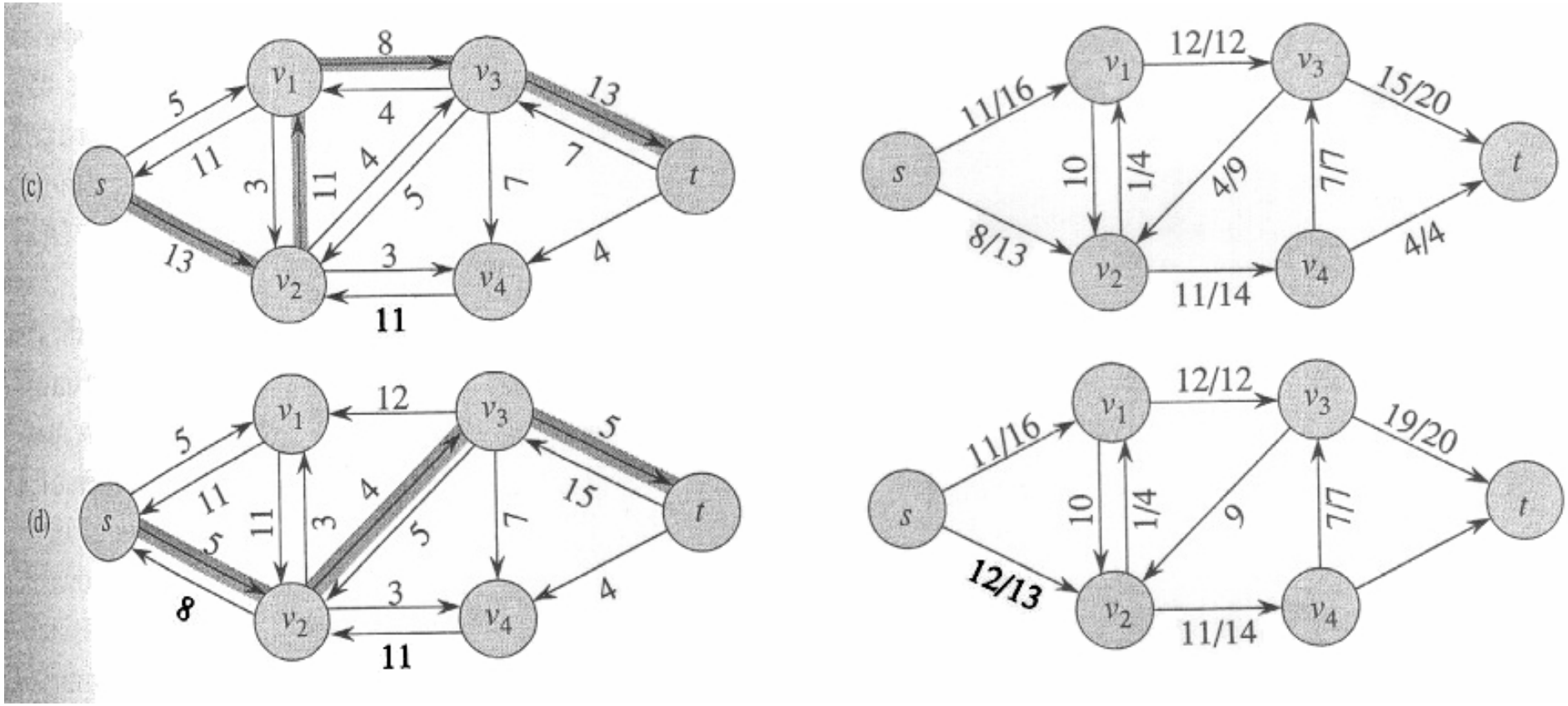
(d) nova rede residual

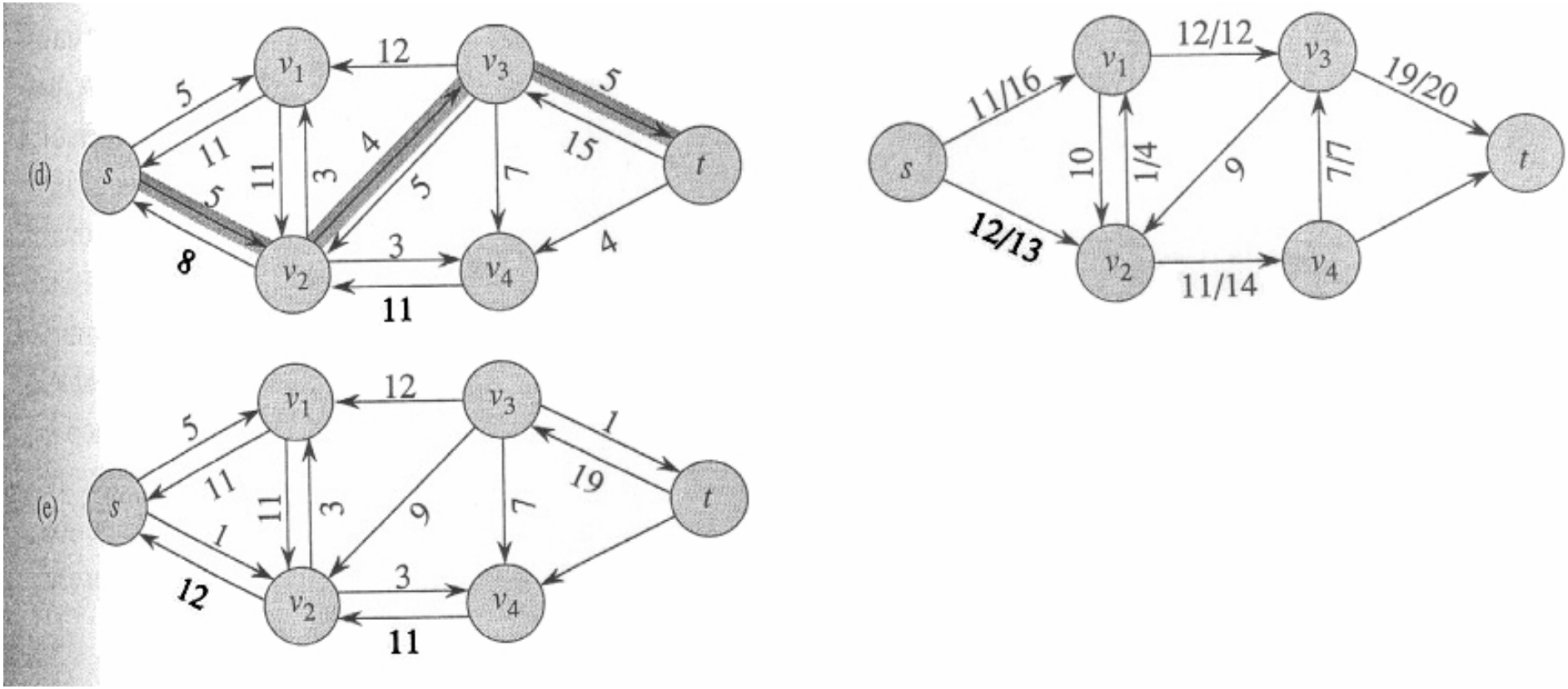
Ford–Fulkerson

▷ Exemplo


(a)
s
v_1
v_2
v_3
v_4
t
16
13
12
10
4
9
7
20
4
14
4/16
4/12
20
4/9
7
4/4
4/14
(b)
12
8
4
4
10
4
4
5
7
20
13
10
4
4/12
7/20
11/16
7/10
4/9
7/7
4/4
11/14


(b)
s
v_1
v_2
v_3
v_4
t
12
4
10
13
8
4
4
4
5
10
7
20
4
4
(c)
s
v_1
v_2
v_3
v_4
t
5
11
3
13
8
4
11
4
5
3
7
13
7
4
11
11/16
4/12
7/20
7/10
4
4/9
7/7
4/4
11/14
11/16
12/12
15/20
10
1/4
4/9
7/7
8/13
4/4
11/14


(c)
s
v_1
v_2
v_3
v_4
t
5
11
13
8
4
3
11
4
5
7
7
3
4
11
11/16
8/13
12/12
11/14
10
1/4
4/9
7/7
15/20
4/4
(d)
12
5
11
5
8
3
4
5
7
15
3
11
12/13
12/12
11/14
10
1/4
9
7/7
19/20



Em (d), após o aumento do fluxo, temos um fluxo máximo (basta considerar o corte dado por $\{s, v_1, v_2, v_4\}$). O grafo residual em (e) não tem caminhos aumentadores.

Ford–Fulkerson

▷ Política de escolha do caminha aumentador é importante.

Edmonds– Karp

▷ **Natural:** encontrar o caminho aumentador fazendo uma BeL na rede residual (Edmonds–Karp).

Teorema 2. *Com a política de Edmonds–Karp, o método de Ford–Fulkerson encontra um fluxo máximo em tempo* $O(nm^2)$.

▷ Goldberg/Goldberg–Tarjan: $O(n^2m)$, $O(n^3)$

Exemplo

	cap	fluxo	fluxo
0-1	2	2	2
0-2	3	1	2
1-3	3	2	1
1-4	1	0	1
2-3	1	0	1
2-4	1	1	1
3-5	2	2	2
4-5	3	1	2

Exemplo

	0	1	2	3	4	5
0	0	2	3	*	*	*
1	-2	0	*	3	1	*
2	-3	*	0	1	1	*
3	*	-3	-1	0	*	2
4	*	-1	-1	*	0	3
5	*	*	*	-2	-3	0

capacidades

	0	1	2	3	4	5
0	0	2	1	*	*	*
1	-2	0	*	2	0	*
2	-1	*	0	0	1	*
3	*	-2	0	0	*	2
4	*	0	-1	*	0	1
5	*	*	*	-2	-1	0

fluxo

Exemplo

```
typedef struct node *link;
struct node
  { int v; int cap; int flow; link dup; link next;};
struct graph
  { int V; int E; link *adj; };
```

```
0 ->  0, 0, 0, * ->  1, 2, 2, * ->  2, 3, 1, *
1 ->  0,-2,-2, * ->  1, 0, 0, * ->  3, 3, 2, * ->  4, 1, 0, *
2 ->  0,-3,-1, * ->  2, 0, 0, * ->  3, 1, 0, * ->  4, 1, 1, *
3 ->  1,-3,-2, * ->  2,-1, 0, * ->  3, 0, 0, * ->  5, 2, 2, *
4 ->  1,-1, 0, * ->  2,-1,-1, * ->  4, 0, 0, * ->  5, 3, 1, *
5 ->  3,-2,-2, * ->  4,-3,-1, * ->  5, 0, 0, *
```

Programa 22.1: TAD para fluxos

```
#include <stdlib.h>
#include "GRAPH.h"
typedef struct node *link;
struct node
  { int v; int cap; int flow; link dup; link next;};
struct graph
  { int V; int E; link *adj; };
link NEW(int v, int cap, int flow, link next)
  { link x = malloc(sizeof *x);
    x->v = v; x->cap = cap; x->flow = flow;
    x->next = next;
    return x;
  }
```

Programa 22.1: TAD para fluxos

```
Graph GRAPHinit(int V)
  { int i;
    Graph G = malloc(sizeof *G);
    G->adj = malloc(V*sizeof(link));
    G->V = V; G->E = 0;
    for (i = 0; i < V; i++) G->adj[i] = NULL;
    return G;
  }
```

Programa 22.1: TAD para fluxos

```
void GRAPHinsertE(Graph G, Edge e)
  { int v = e.v, w = e.w;
    G->adj[v] = NEW(w, e.cap, e.flow, G->adj[v]);
    G->adj[w] = NEW(v, -e.cap, -e.flow, G->adj[w]);
    G->adj[v]->dup = G->adj[w];
    G->adj[w]->dup = G->adj[v];
    G->E++;
  }
```

Programa 22.2: Kirchhoff e valor de fluxo

```
static int flowV(Graph G, int v)
  { link t; int x = 0;
    for (t = G->adj[v]; t != NULL; t = t->next)
      x += t->flow;
    return x;
  }
int GRAPHflow(Graph G, int s, int t)
  { int v, val = flowV(G, s);
    for (v = 0; v < G->V; v++)
      if ((v != s) && (v != t))
        if (flowV(G, v) != 0) return 0;
    if (val + flowV(G, t) != 0) return 0;
    if (val <= 0) return 0;
    return val;
  }
```

Ford–Fulkerson, Programa 22.3

```
static int wt[maxV];
#define Q (u->cap < 0 ? -u->flow : u->cap - u->flow)
```

Ford–Fulkerson, Programa 22.3

```
int GRAPHpfs(Graph G, int s, int t, link st[])
  { int v, w, d = M; link u;
    PQinit(); priority = wt;
    for (v = 0; v < G->V; v++)
      { st[v] = NULL; wt[v] = 0; PQinsert(v); }
    wt[s] = M; PQinc(s);
    while (!PQempty()) { ...
```

Ford–Fulkerson, Programa 22.3

```
  while (!PQempty()) {
      v = PQdelmax();
      if ((wt[v] == 0) || (v == t)) break;
      for (u = G->adj[v]; u != NULL; u = u->next)
        if (Q > 0)
          if (P > wt[w = u->v])
            { wt[w] = P; PQinc(w); st[w] = u; }
      wt[v] = M;
    }
  if (wt[t] == 0) return 0;
  for (w = t; w != s; w = st[w]->dup->v)
    { u = st[w]; d = ( Q > d ? d : Q ); }
  return d;
}
```

Ford–Fulkerson, Programa 22.3

```
void GRAPHmaxflow(Graph G, int s, int t)
  { int x, d;
    link st[maxV];
    while ((d = GRAPHpfs(G, s, t, st)) != 0)
      for (x = t; x != s; x = st[x]->dup->v)
        { st[x]->flow += d; st[x]->dup->flow -= d; }
  }
```