Graphical Models
Discrete Inference and Learning
Lecture 5

MVA

2021 – 2022

http://thoth.inrialpes.fr/~alahari/disinflearn

Slides based on material from M. Pawan Kumar

Practical matters - Projects

- 15 projects – 30 students
- Responded to all proposals
- If you haven't heard, we have not received it
  - Email: karteek.alahari@inria.fr

Quiz

- What inference algorithms have we seen?
- When would you (not) recommend BP?
- What are better alternatives to BP?
- Is “learning” necessary?
- How and what would you “learn” in graphical models?

Outline

- Recap: Preliminaries
  - Functions and Excess Functions
  - s-t Flow
  - s-t Cut
  - Flows vs. Cuts
- Maximum Flow
- Algorithms
- Energy minimization with max flow/min cut

Context

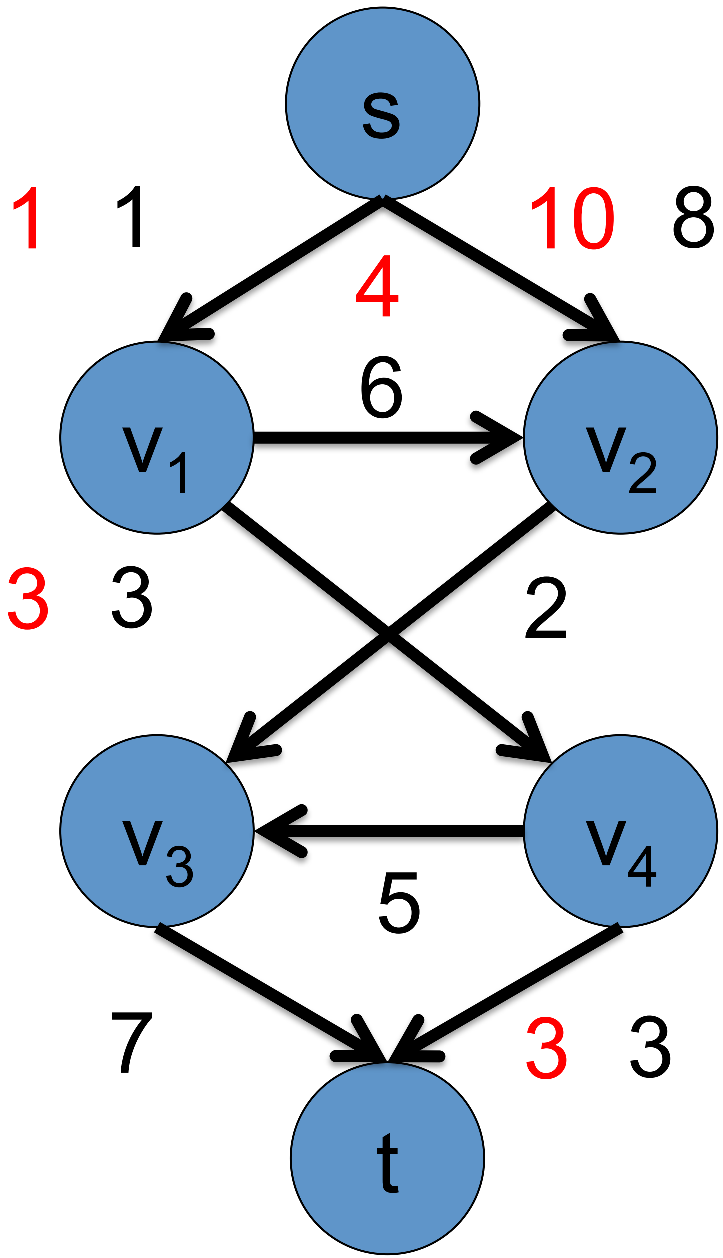
- Example: network optimization problems

Nodes	Arcs	Flow
Intersections	Roads	Vehicles
Airports	Air lanes	Aircraft
Switching points	Wires, channels	Messages
Pumping stations	Pipes	Fluids
Work centers	Materials-handling routes	Jobs

Maximum flow problem

- Applications
  - Maximize the flow through a company's distribution network from factories to customers
  - Maximize the flow of oil through a system of pipelines
  - Maximize the flow of vehicles through a transportation network

Functions on Arcs



D = (V, A)

Arc capacities c(a)

Function f: A → Reals

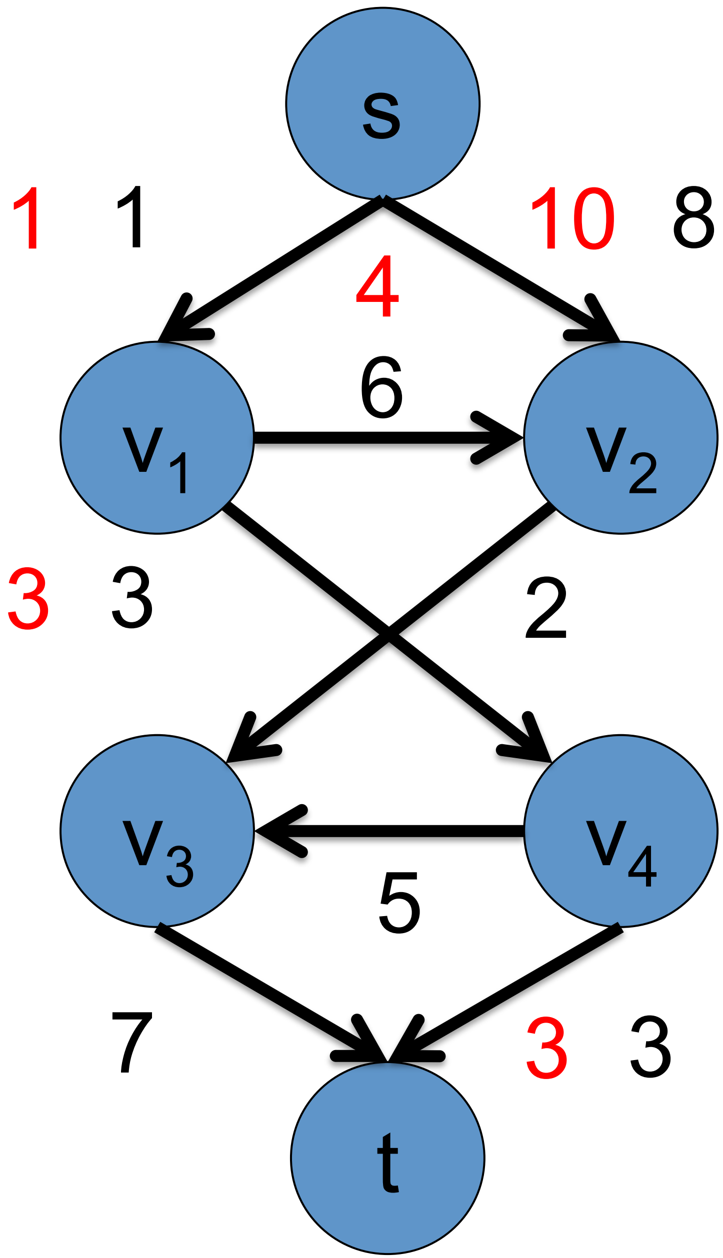
Excess function $E_f(v)$

Incoming value

-

Outgoing value

Functions on Arcs



D = (V, A)

Arc capacities c(a)

Function f: A ➔ Reals

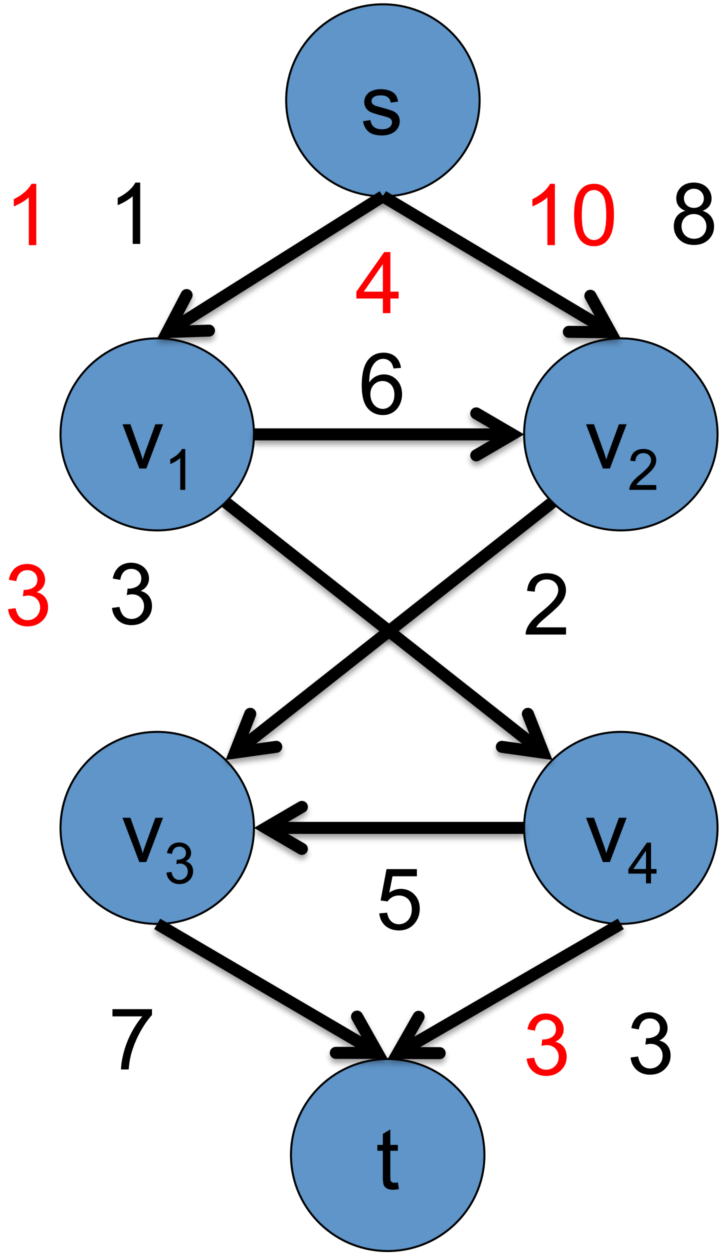
Excess function $E_f(v)$

$$\Sigma_{a \in \text{in-arcs}(v)} f(a)$$

$$-$$

$$\Sigma_{a \in \text{out-arcs}(v)} f(a)$$

Functions on Arcs



D = (V, A)

Arc capacities c(a)

Function f: A ➔ Reals

Excess function $E_f(v)$

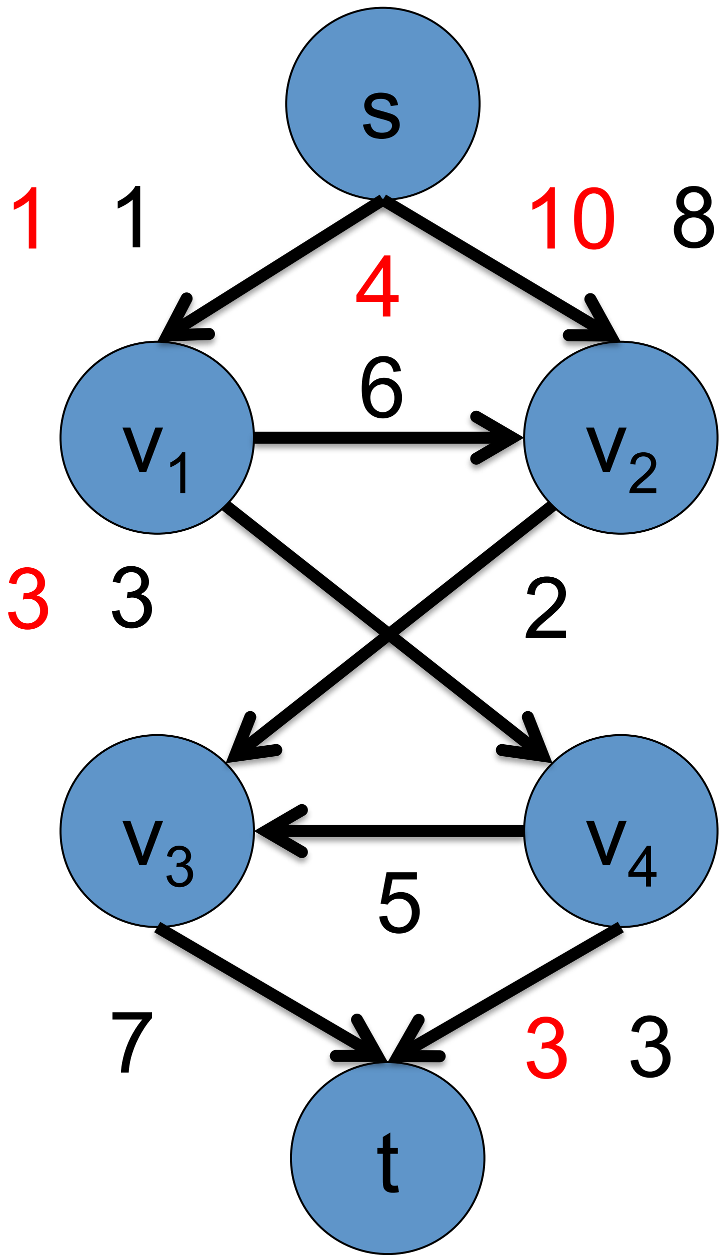
f(in-arcs(v))

-

f(out-arcs(v))

$E_f(v_1)$ -6

Functions on Arcs



D = (V, A)

Arc capacities c(a)

Function f: A → Reals

Excess function $E_f(v)$

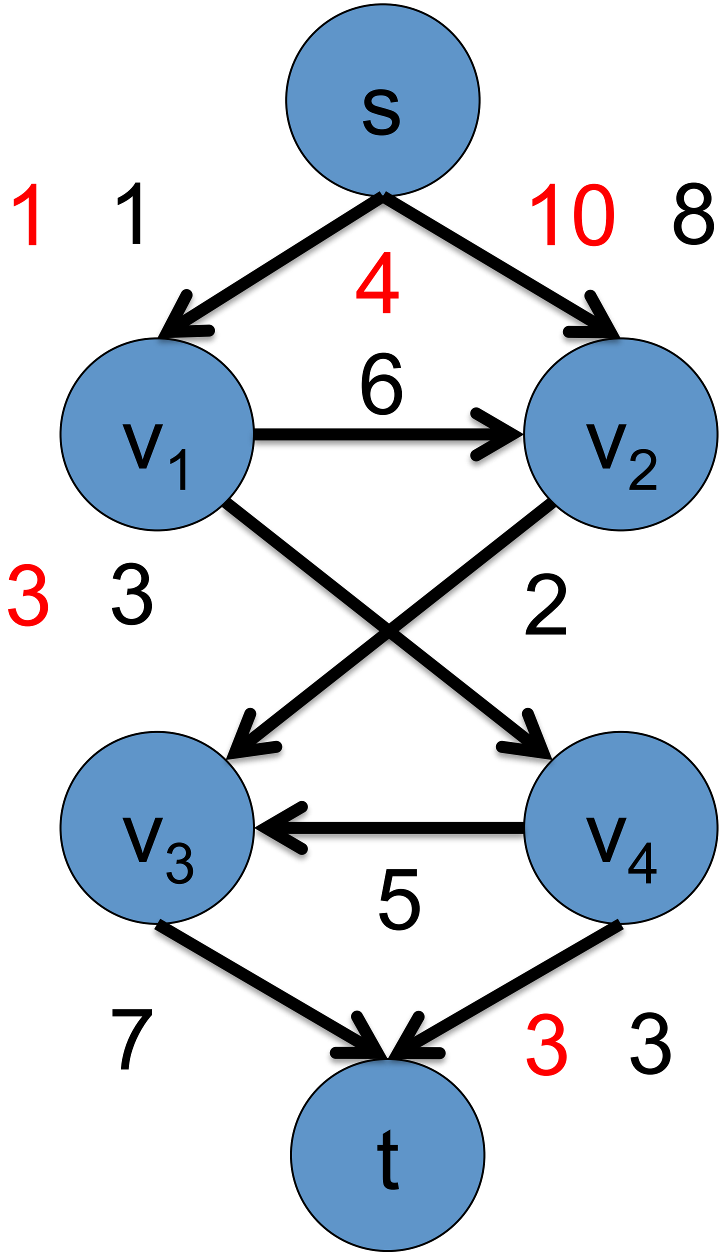
f(in-arcs(v))

-

f(out-arcs(v))

$E_f(v_2)$ 14

Excess Functions of Vertex Subsets



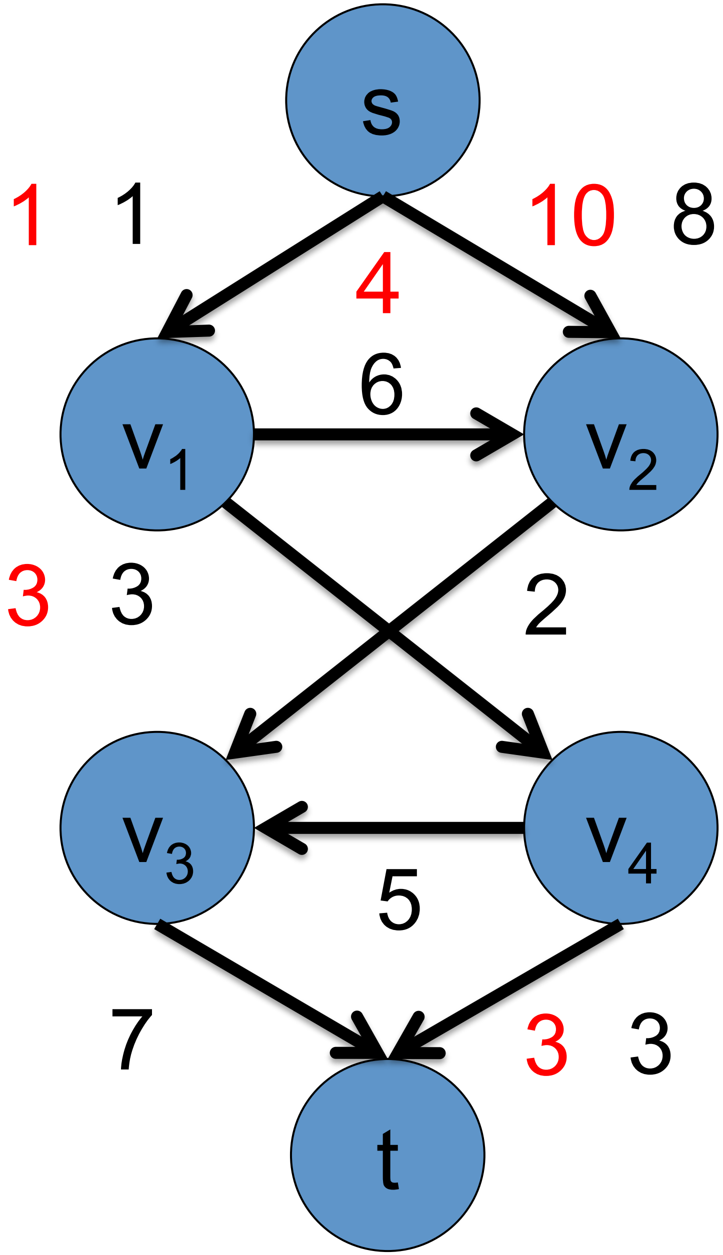
Excess function $E_f(U)$

Incoming Value

-

Outgoing Value

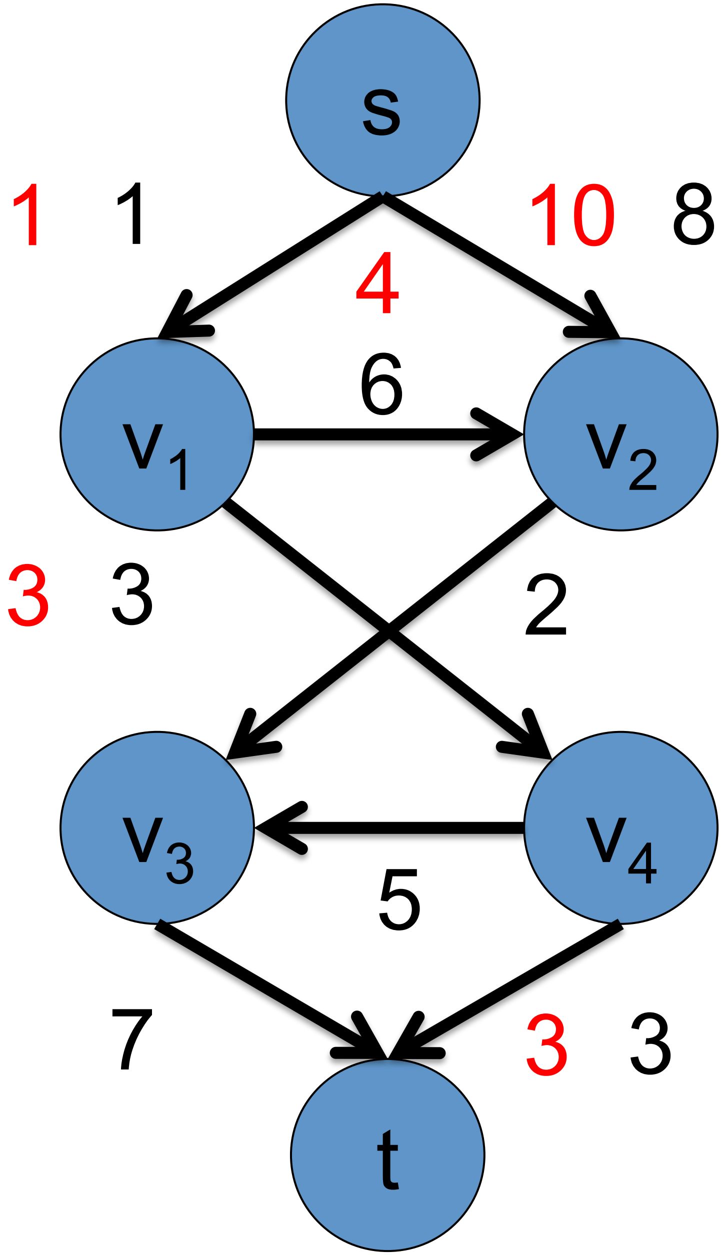
Excess Functions of Vertex Subsets



Excess function $E_f(U)$

$$\Sigma_{a \in \text{in-arcs}(U)} f(a) - \Sigma_{a \in \text{out-arcs}(U)} f(a)$$

Excess Functions of Vertex Subsets



Excess function $E_f(U)$

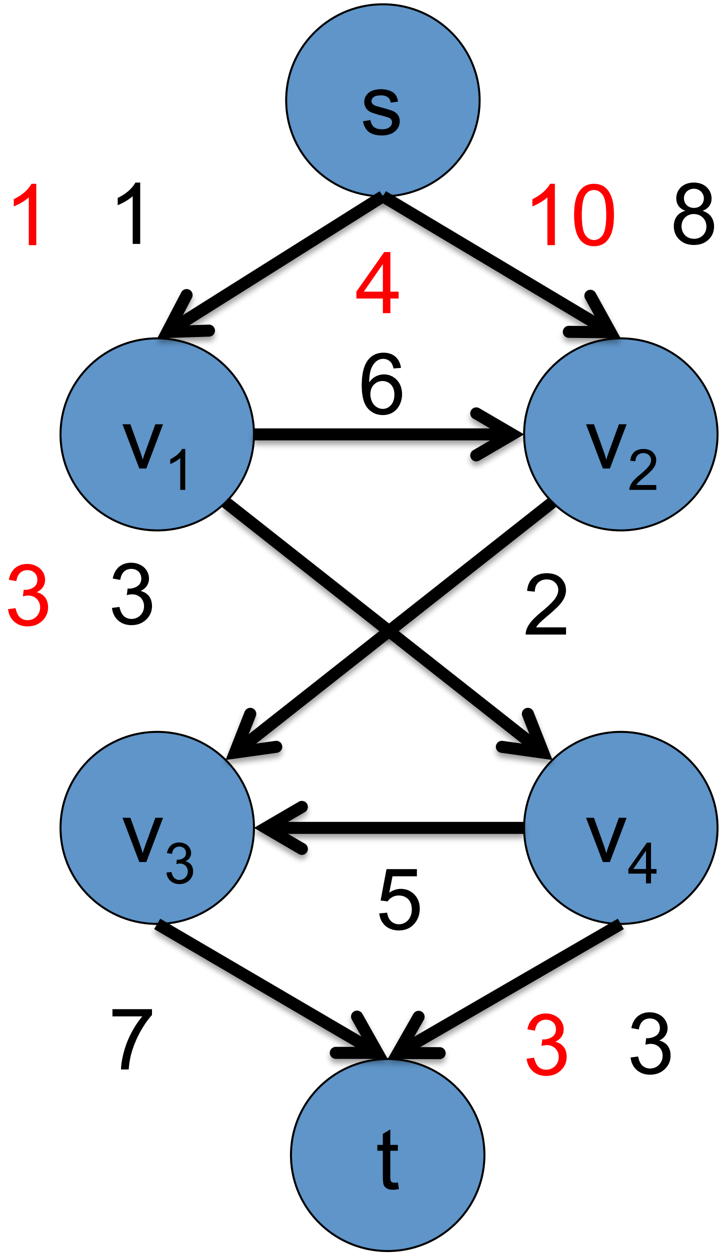
f(in-arcs(U))

-

f(out-arcs(U))

$E_f(\{v_1,v_2\})$ 8

Excess Functions of Vertex Subsets



Excess function $E_f(U)$

f(in-arcs(U))

-

f(out-arcs(U))

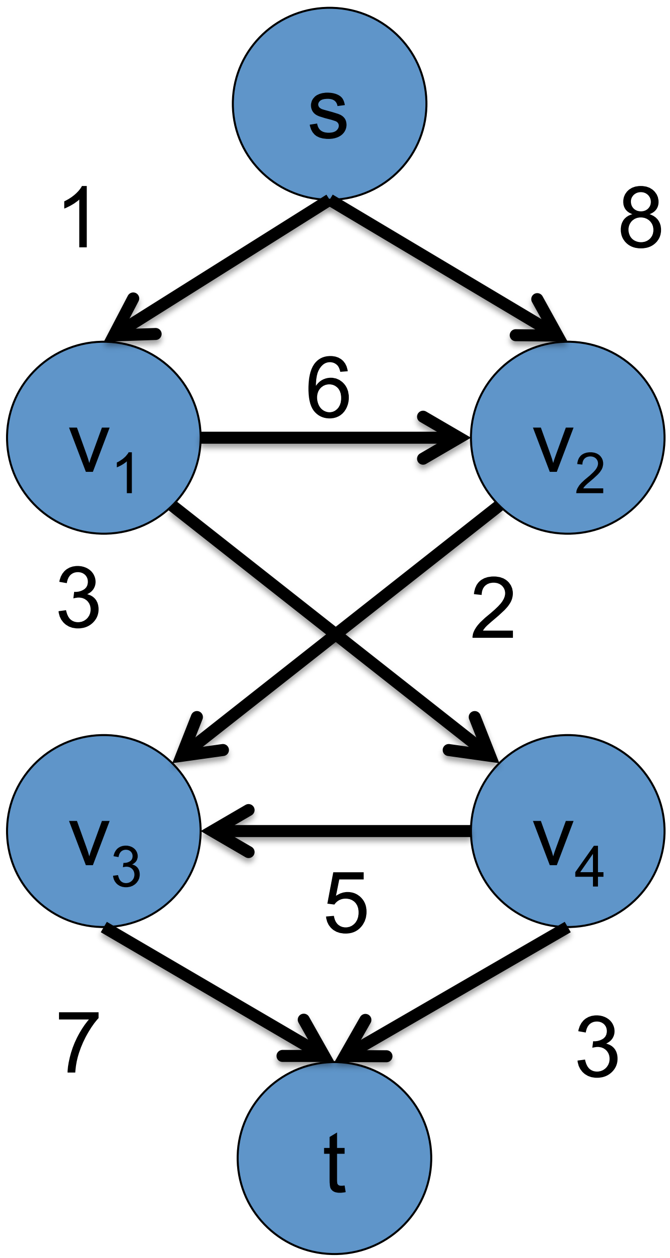
$E_f(\{v_1,v_2\})$ -6 + 14

$E_f(U) = \Sigma_{v \in U} E_f(v)$

Outline

- Preliminaries
  - Functions and Excess Functions
  - **s-t Flow**
  - s-t Cut
  - Flows vs. Cuts
- Maximum Flow
- Algorithms
- Energy minimization with max flow/min cut

s-t Flow



Function flow: A ➔ R

Flow of arc ≤ arc capacity

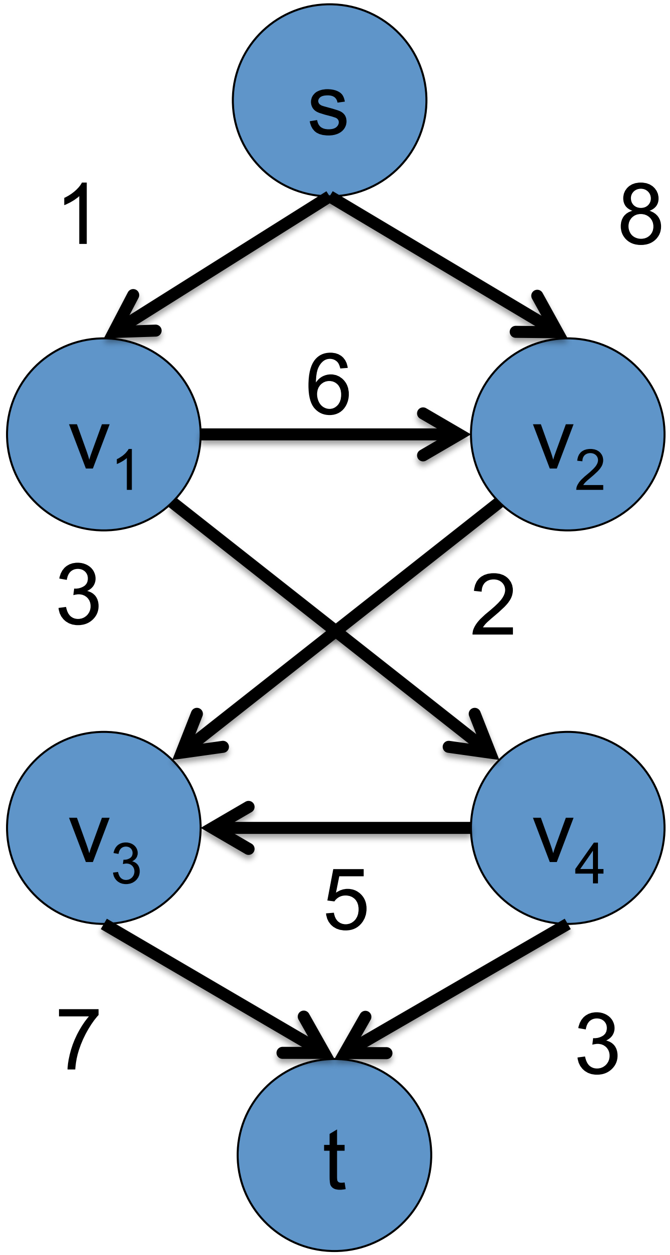
Flow is non-negative

For all vertex except s,t

Incoming flow

= Outgoing flow

s-t Flow



Function flow: A ➔ R

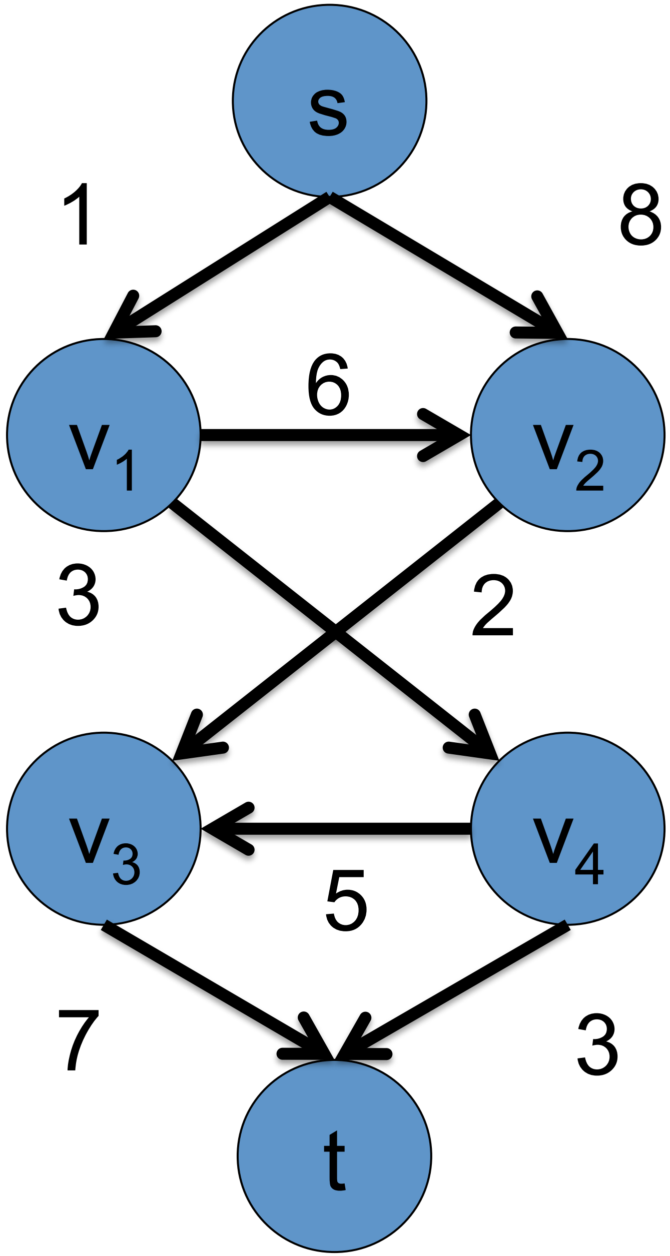
flow(a) ≤ c(a)

flow(a) ≥ 0

For all $v \in V \setminus \{s,t\}$

$$\Sigma_{(u,v)\in A}\ \text{flow}((u,v)) = \Sigma_{(v,u)\in A}\ \text{flow}((v,u))$$

s-t Flow



Function flow: A → R

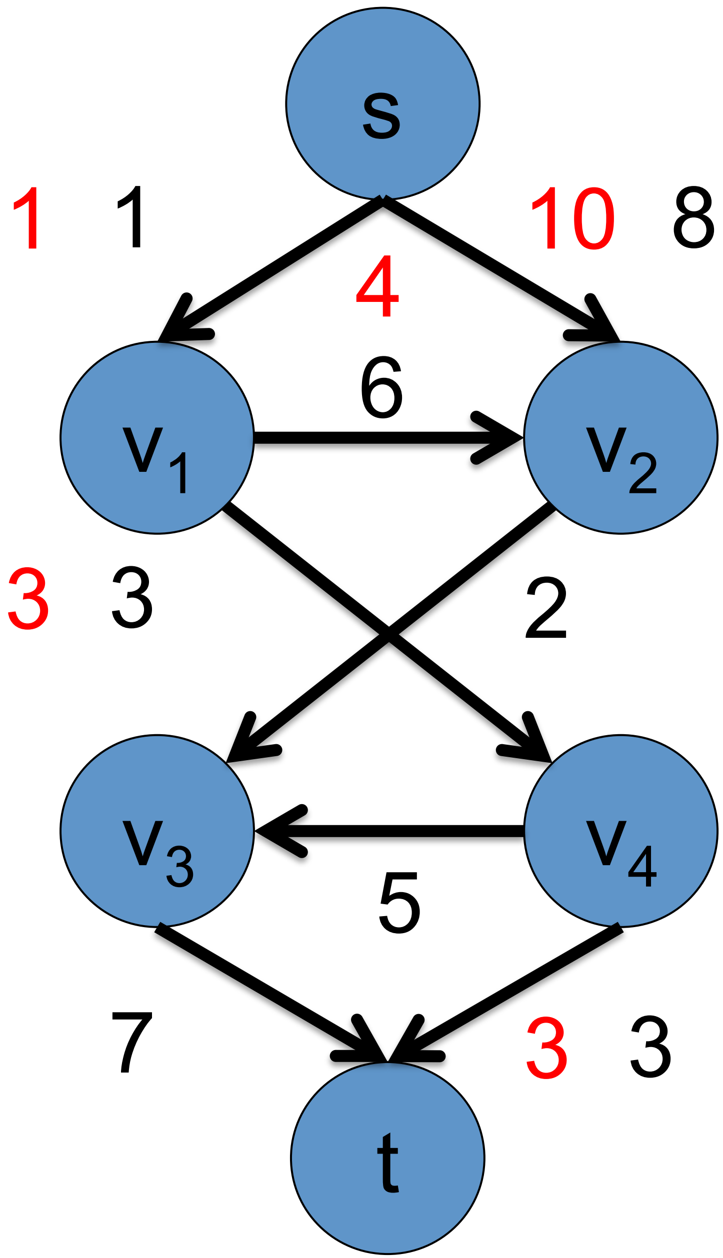
flow(a) ≤ c(a)

flow(a) ≥ 0

For all $v \in V \setminus \{s,t\}$

$E_{flow}(v) = 0$

s-t Flow



Function flow: A → R

flow(a) ≤ c(a)

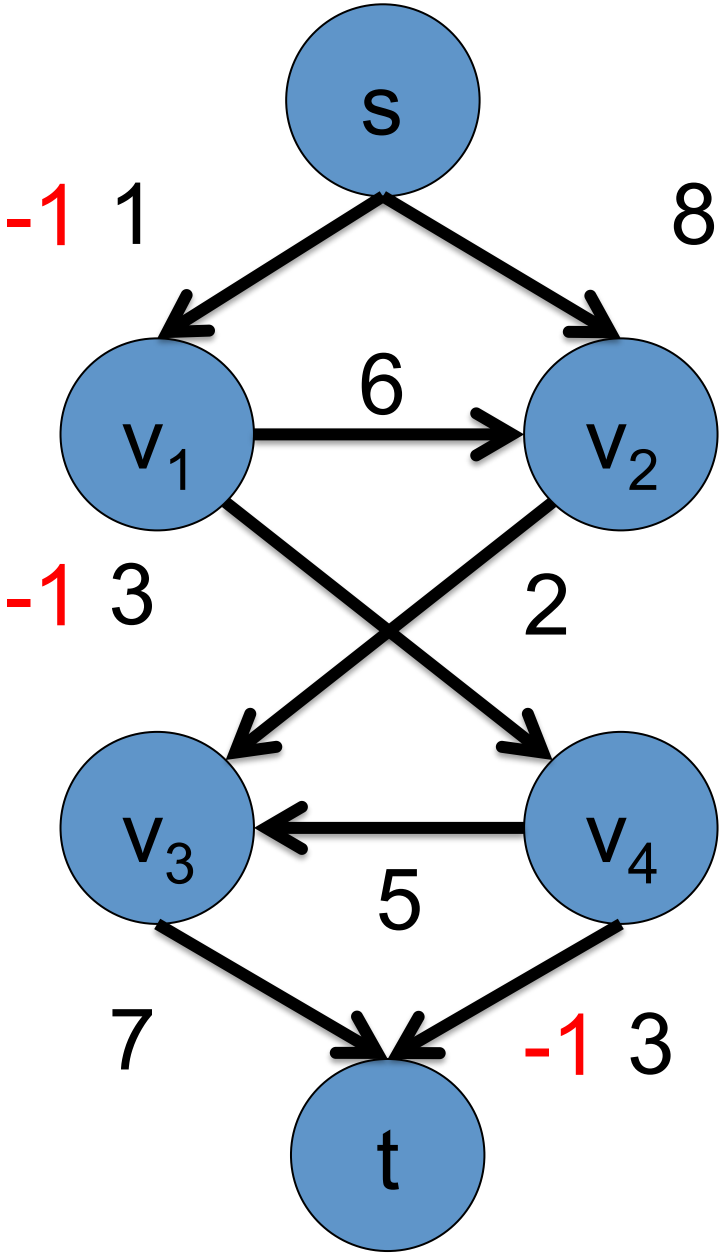
flow(a) ≥ 0

For all v ∈ V \ {s,t}

$E_{flow}(v) = 0$

✗

s-t Flow



Function flow: A ➔ R

flow(a) ≤ c(a)

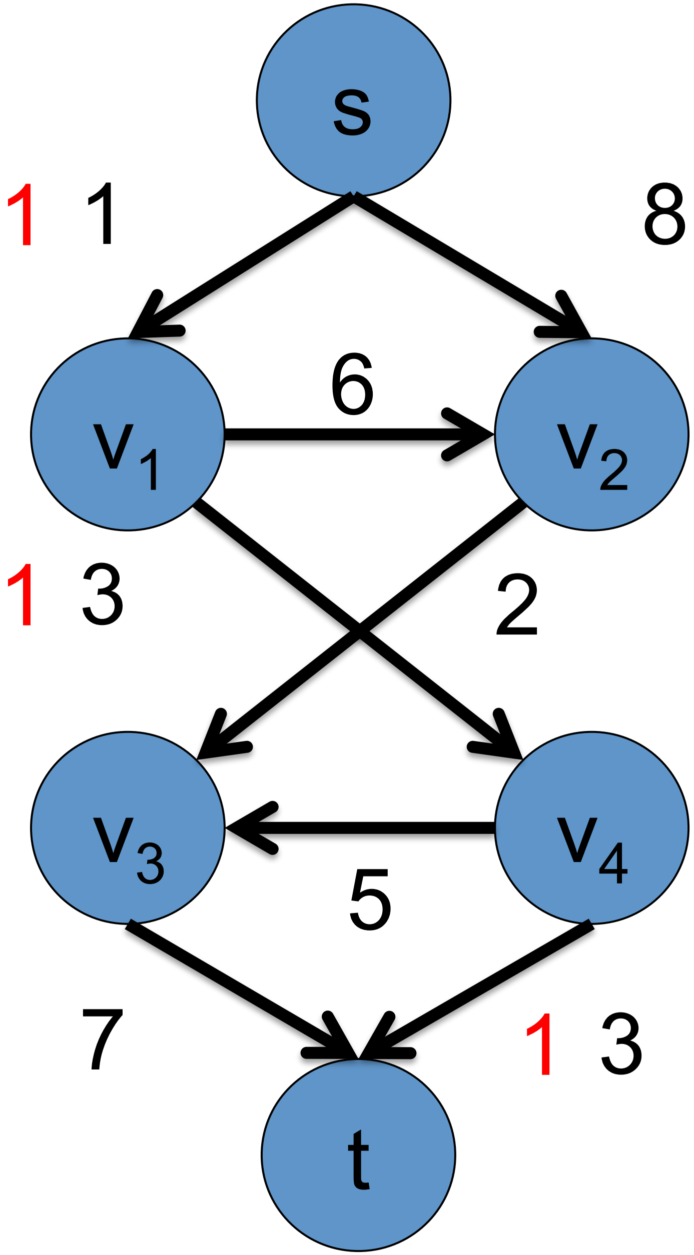
flow(a) ≥ 0

For all v ∈ V \ {s,t}

$E_{flow}(v) = 0$

✗

s-t Flow



Function flow: A ➔ R

flow(a) ≤ c(a)

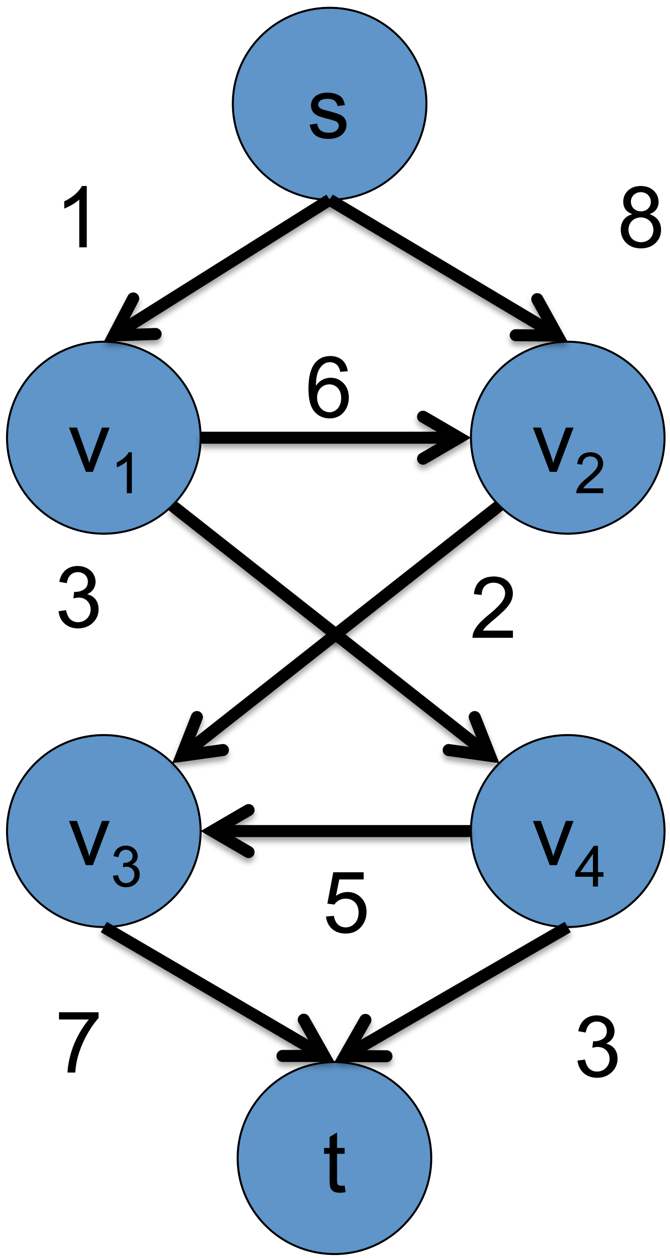
flow(a) ≥ 0

For all $v \in V \setminus \{s,t\}$

$E_{flow}(v) = 0$

✓

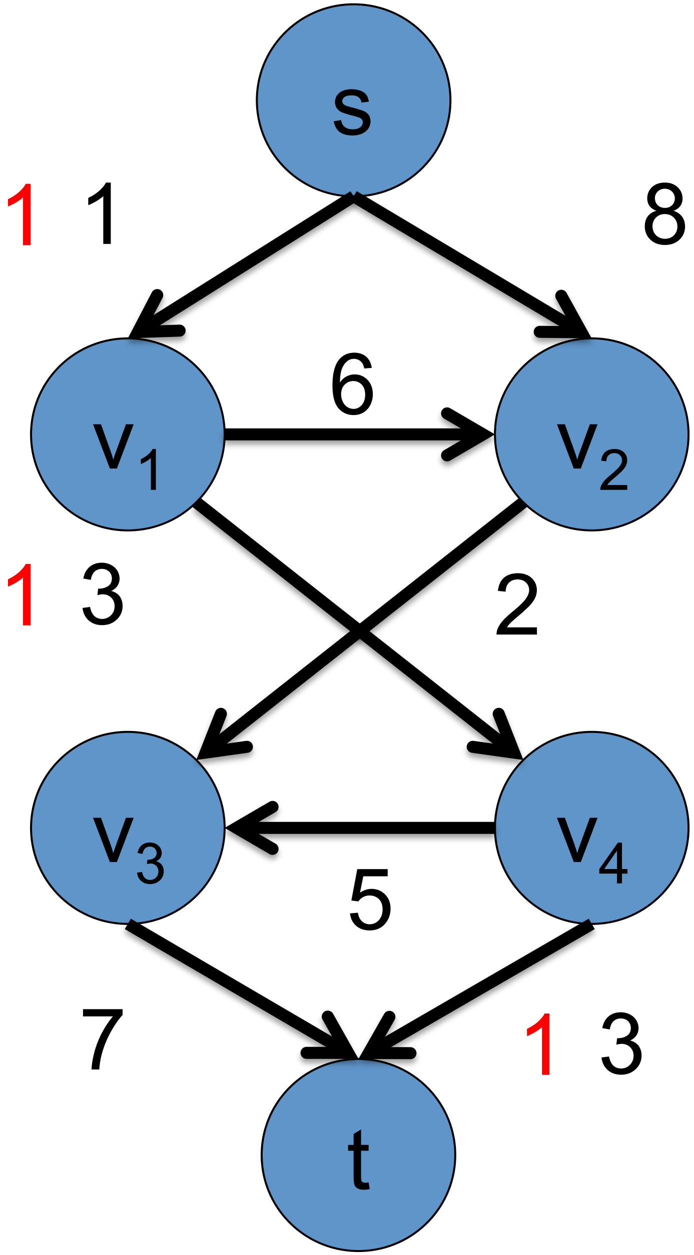
Value of s-t Flow



Outgoing flow of s

- Incoming flow of s

Value of s-t Flow



$-E_{flow}(s)$ $\quad$ $E_{flow}(t)$

$$\Sigma_{(s,v)\in A} \text{flow}((s,v))$$
$$- \Sigma_{(u,s)\in A} \text{flow}((u,s))$$

Value = 1

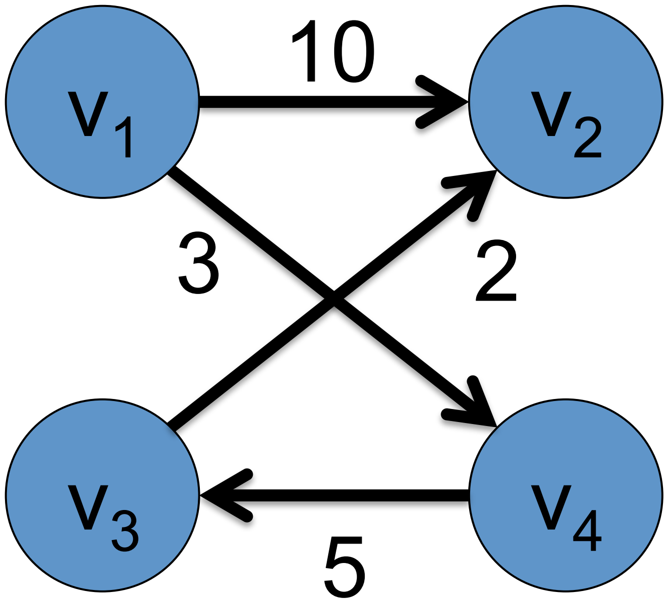
Outline

- Preliminaries
  - Functions and Excess Functions
  - s-t Flow
  - **s-t Cut**
  - Flows vs. Cuts

- Maximum Flow
- Algorithms
- Energy minimization with max flow/min cut

Cut

D = (V, A)



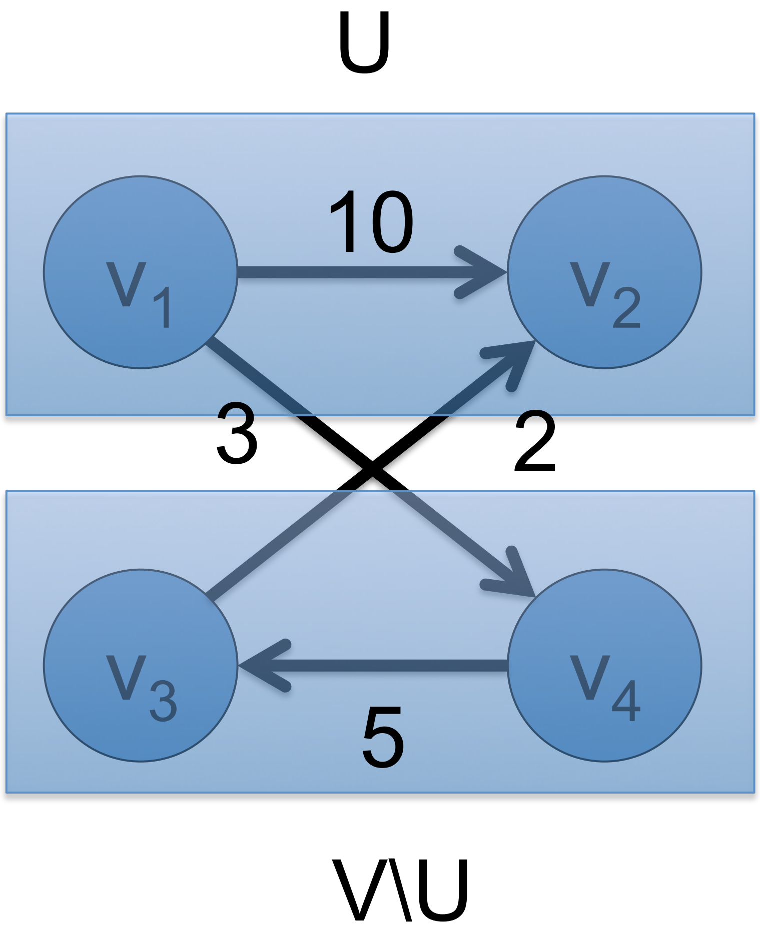
Let U be a subset of V

C is a set of arcs such that

- $(u,v) \in A$
- $u \in U$
- $v \in V \setminus U$

C is a cut in the digraph D

Cut



D = (V, A)

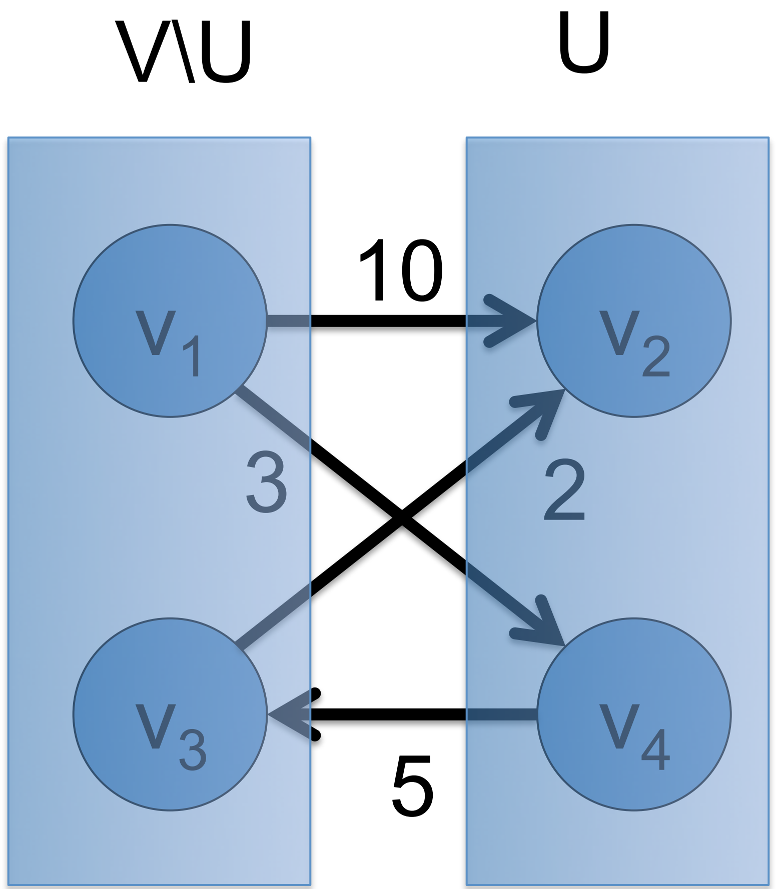
What is C?

$\{(v_1,v_2),(v_1,v_4)\}$?

$\{(v_1,v_4),(v_3,v_2)\}$?

✓ $\{(v_1,v_4)\}$?

Cut



D = (V, A)

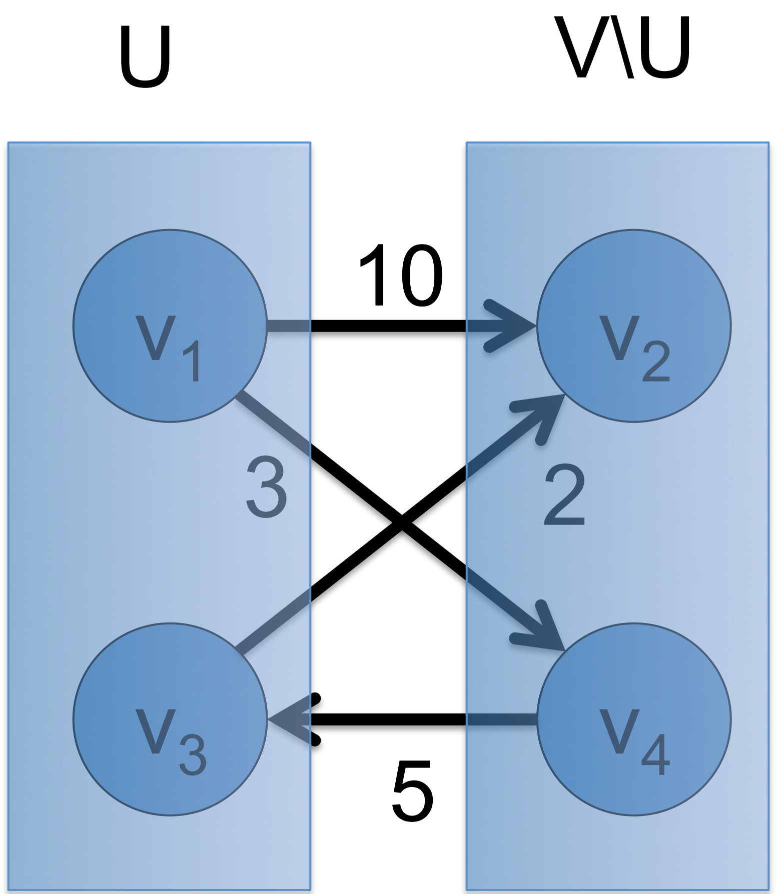
What is C?

$\{(v_1,v_2),(v_1,v_4),(v_3,v_2)\}$?

✓ $\{(v_4,v_3)\}$?

$\{(v_1,v_4),(v_3,v_2)\}$?

Cut



D = (V, A)

What is C?

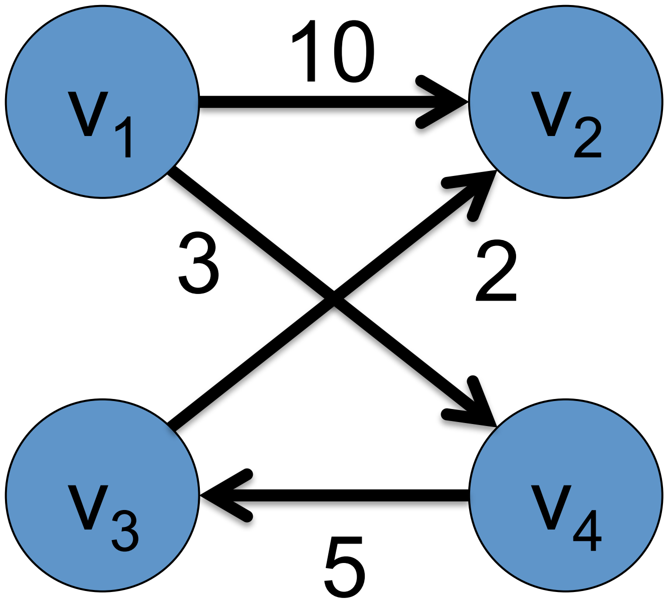
✓ $\{(v_1,v_2),(v_1,v_4),(v_3,v_2)\}$?

$\{(v_3,v_2)\}$?

$\{(v_1,v_4),(v_3,v_2)\}$?

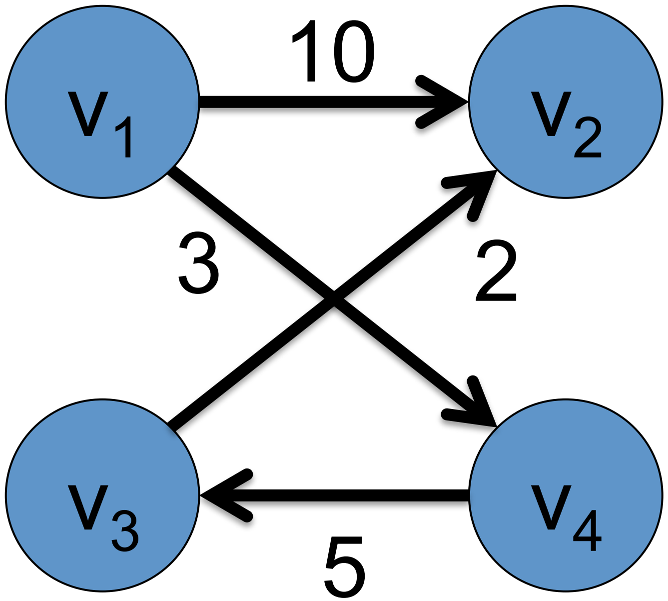
Cut

D = (V, A)



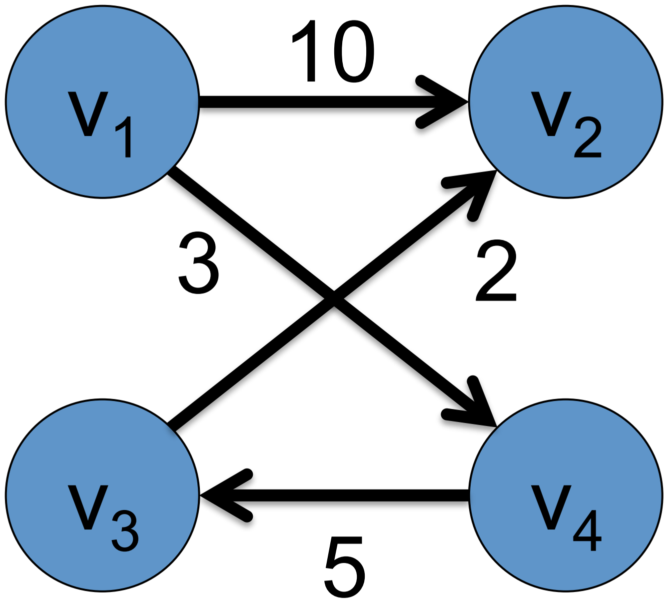
C = out-arcs(U)

Capacity of Cut



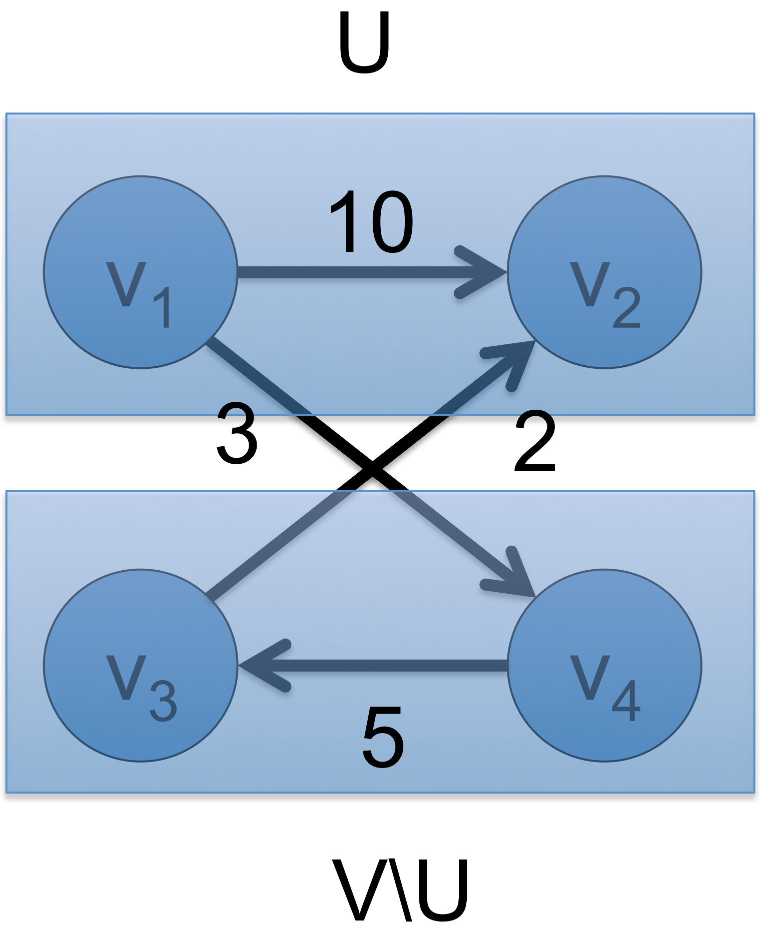
Sum of capacity of all arcs in C

Capacity of Cut



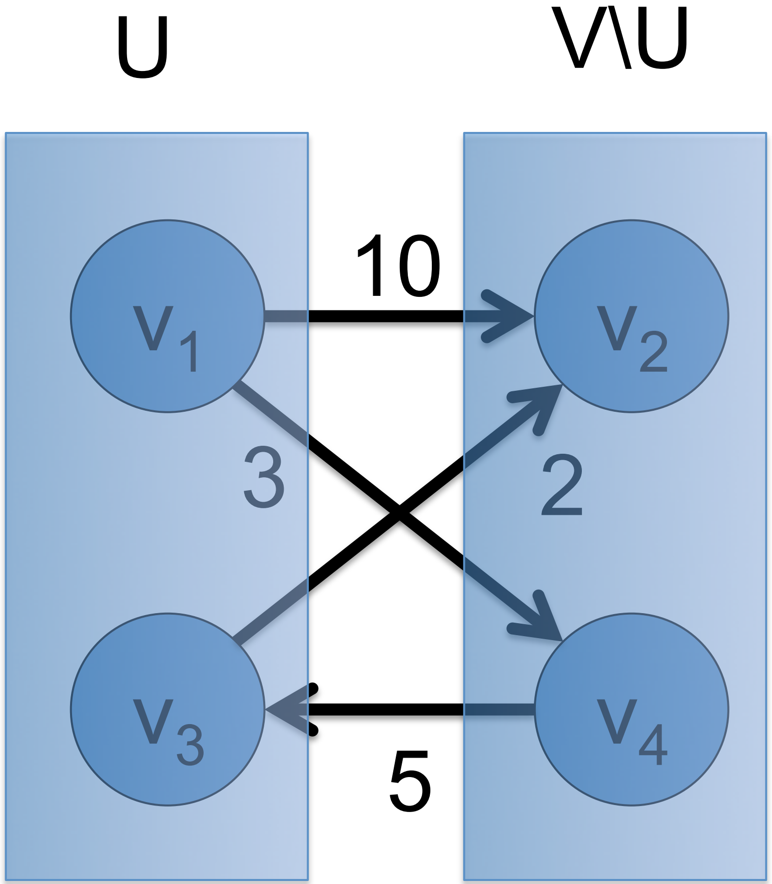
$\Sigma_{a \in C}\ c(a)$

Capacity of Cut



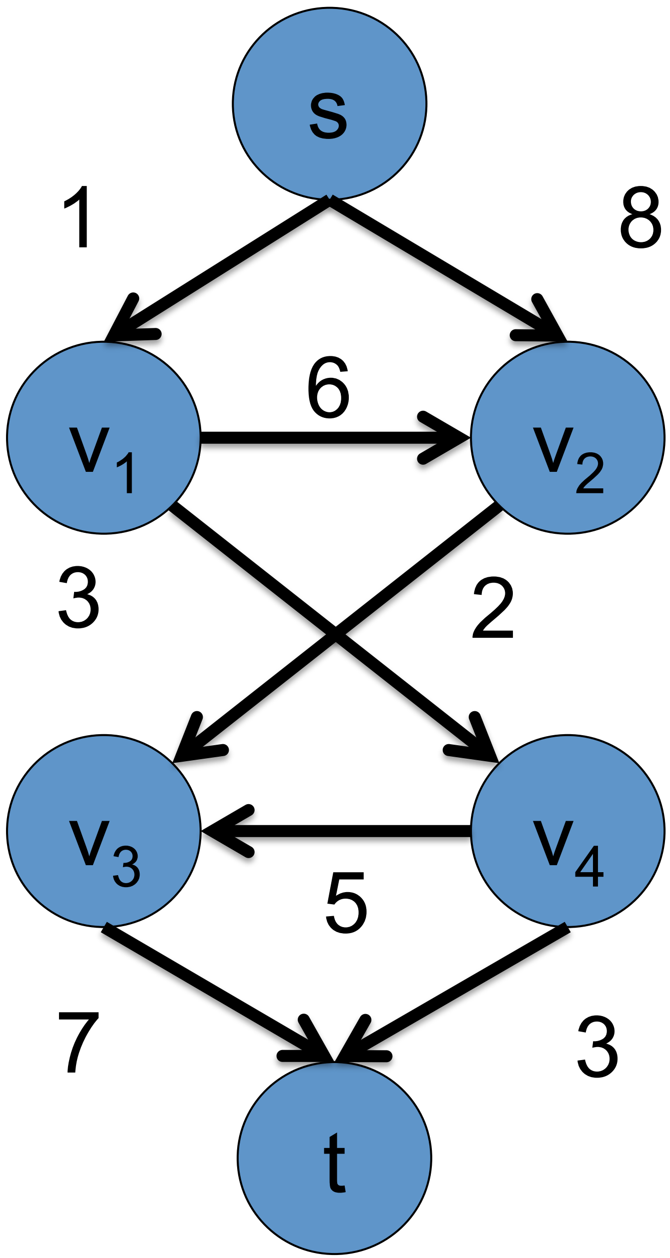
3

Capacity of Cut



15

s-t Cut



D = (V, A)

A source vertex "s"

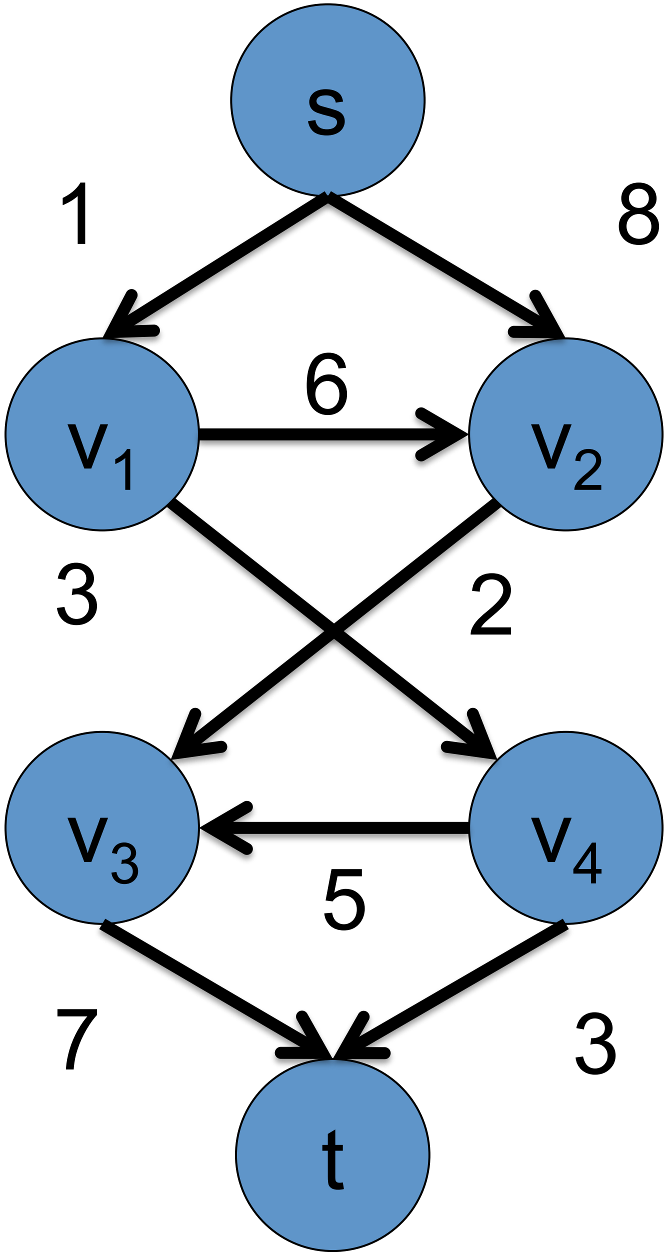
A sink vertex "t"

C is a cut such that

- $s \in U$
- $t \in V \setminus U$

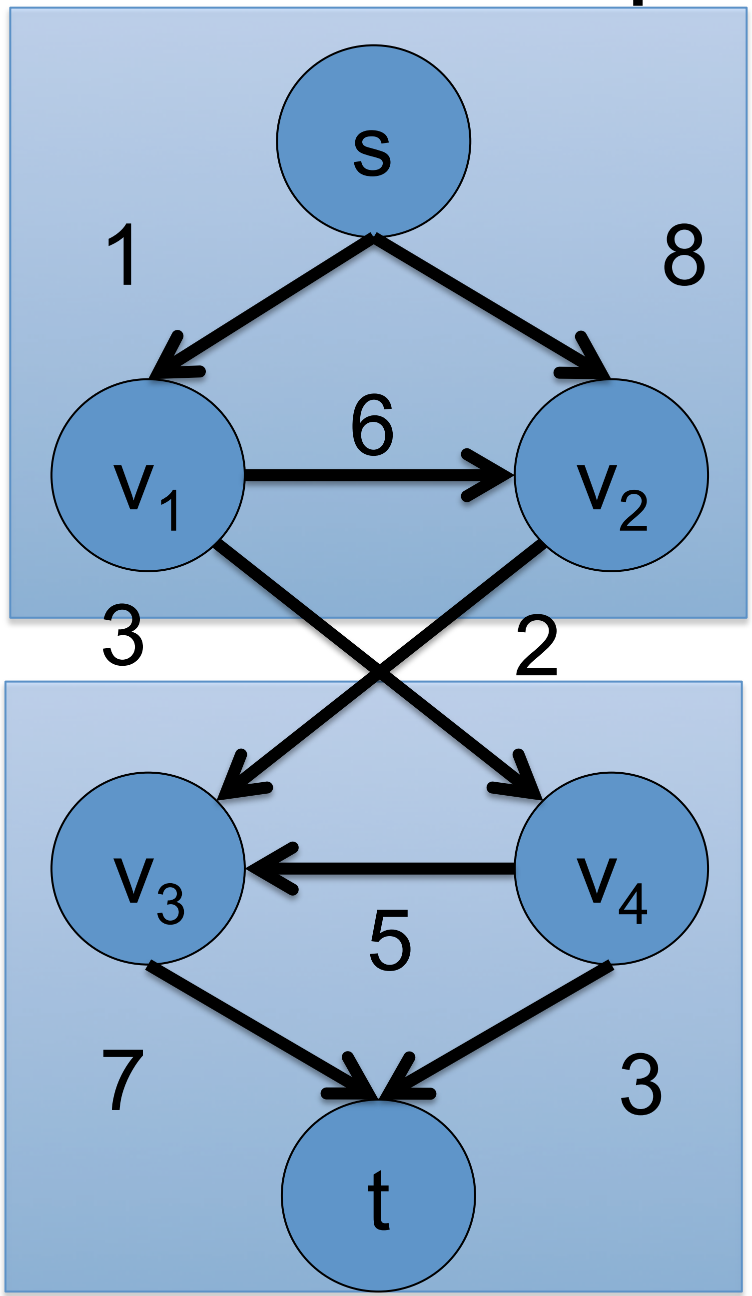
C is an s-t cut

Capacity of s-t Cut



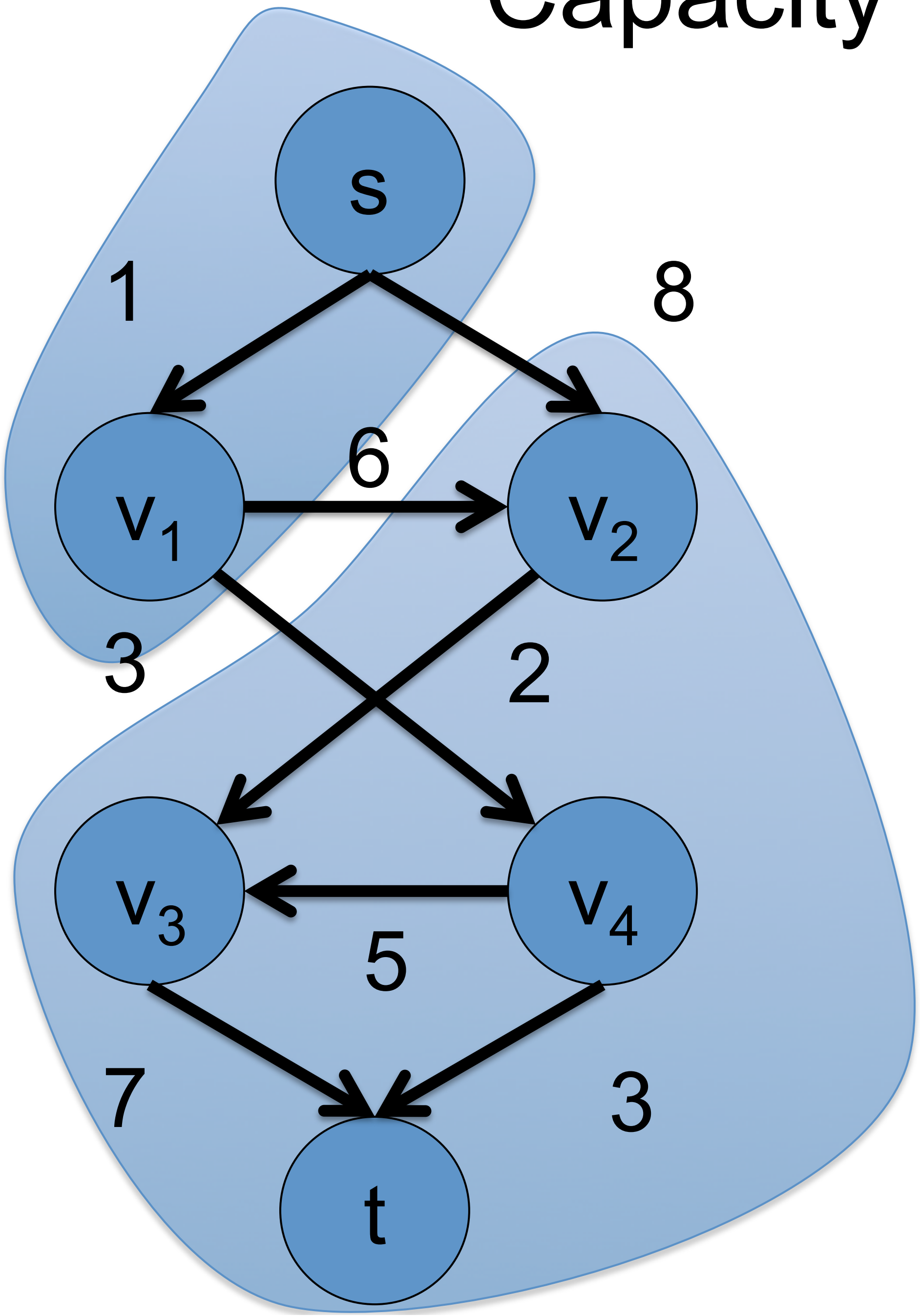
$\Sigma_{a \in C} c(a)$

Capacity of s-t Cut



5

Capacity of s-t Cut



17

Outline

- Preliminaries
  - Functions and Excess Functions
  - s-t Flow
  - s-t Cut
  - **Flows vs. Cuts**

- Maximum Flow
- Algorithms
- Energy minimization with max flow/min cut

Flows vs. Cuts

An s-t flow function : A ➔ Reals

An s-t cut C such that $s \in U$, $t \in V \setminus U$

Value of flow ≤ Capacity of C

Flows vs. Cuts

Value of flow $= -E_{flow}(s)$

$= -E_{flow}(s) - \Sigma_{v \in U \setminus \{s\}} E_{flow}(v)$

$= -E_{flow}(U)$

= flow(out-arcs(U))
- flow(in-arcs(U))

≤ Capacity of C
- flow(in-arcs(U))

Flows vs. Cuts

Value of flow $= -E_{flow}(s)$

$= -E_{flow}(s) - \Sigma_{v \in U \setminus \{s\}} E_{flow}(v)$

$= -E_{flow}(U)$

= flow(out-arcs(U))
- flow(in-arcs(U))

≤ Capacity of C

When does equality hold?

Flows vs. Cuts

Value of flow $= -E_{flow}(s)$

$= -E_{flow}(s) - \Sigma_{v \in U \setminus \{s\}} E_{flow}(v)$

$= -E_{flow}(U)$

$=$ flow(out-arcs(U))

$-$ flow(in-arcs(U))

$\leq$ Capacity of C

flow(a) = c(a), $a \in$ out-arcs(U)   flow(a) = 0, $a \in$ in-arcs(U)

Flows vs. Cuts

Value of flow $= -E_{flow}(s)$

$= -E_{flow}(s) - \Sigma_{v \in U \setminus \{s\}} E_{flow}(v)$

$= -E_{flow}(U)$

$=$ flow(out-arcs(U)) - flow(in-arcs(U))

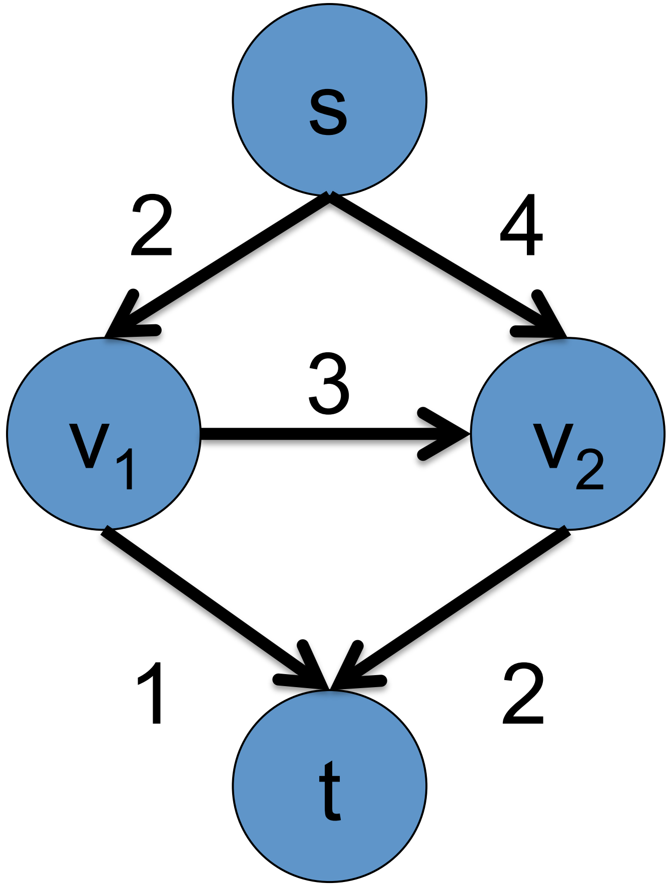
$=$ Capacity of C

flow(a) = c(a), $a \in$ out-arcs(U) flow(a) = 0, $a \in$ in-arcs(U)

Outline

- Preliminaries
- **Maximum Flow**
  - Residual Graph
  - Max-Flow Min-Cut Theorem
- Algorithms
- Energy minimization with max flow/min cut

Maximum Flow Problem



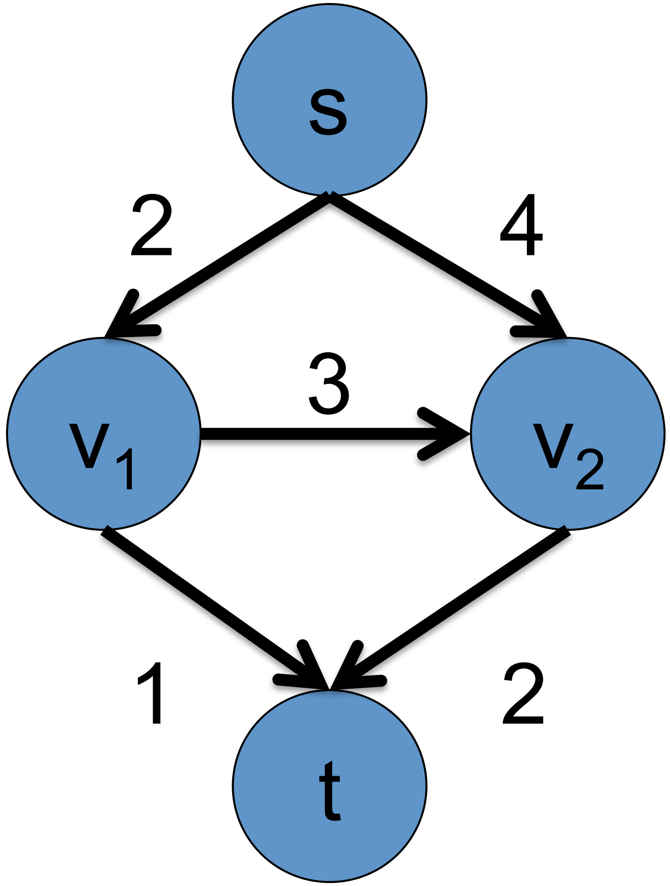
Find the flow with the maximum value !!

$\Sigma_{(s,v)\in A}$ flow((s,v))

- $\Sigma_{(u,s)\in A}$ flow((u,s))

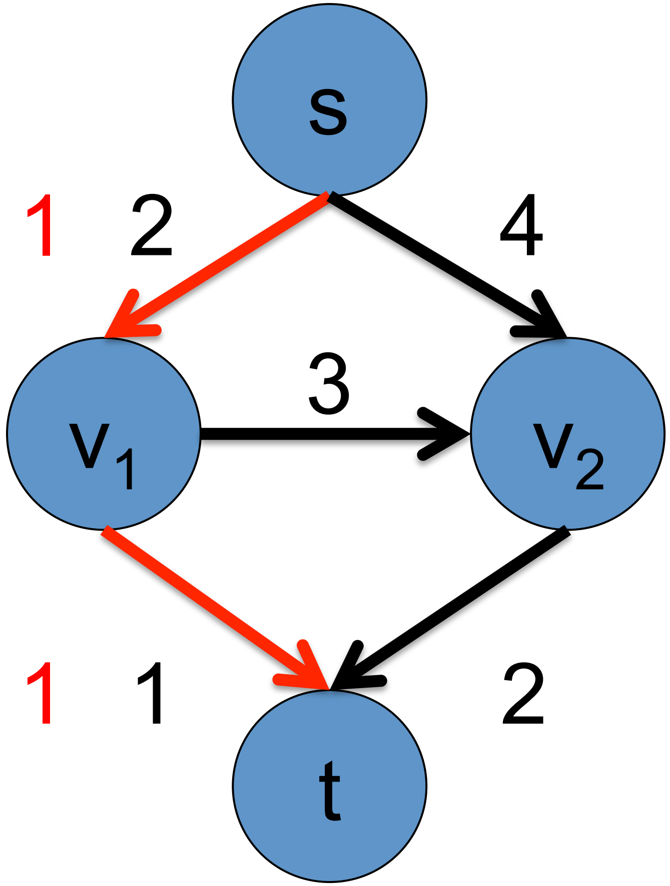
First suggestion to solve this problem !!

Passing Flow through s-t Paths



Find an s-t path where flow(a) < c(a) for all arcs

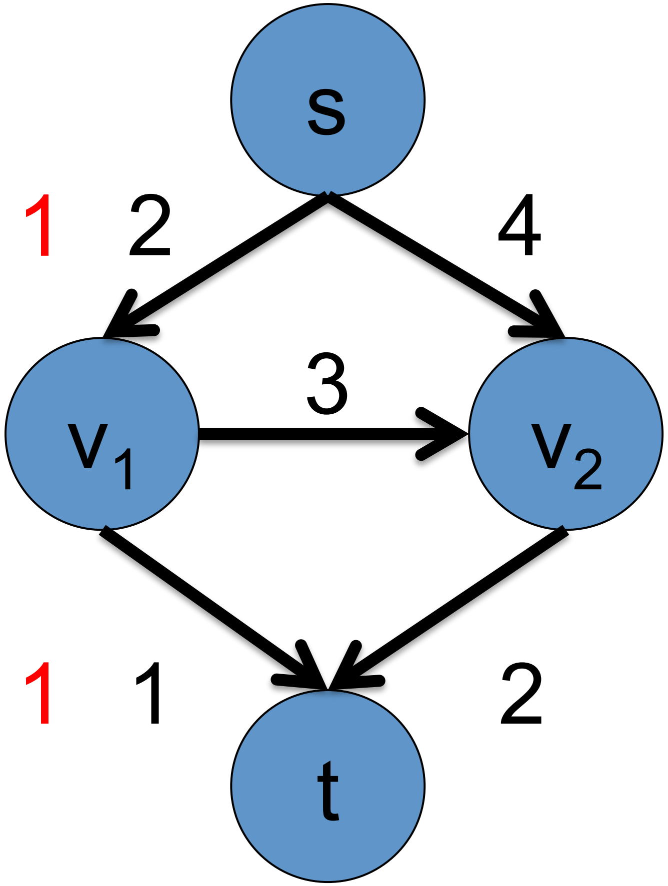
Passing Flow through s-t Paths



Find an s-t path where flow(a) < c(a) for all arcs

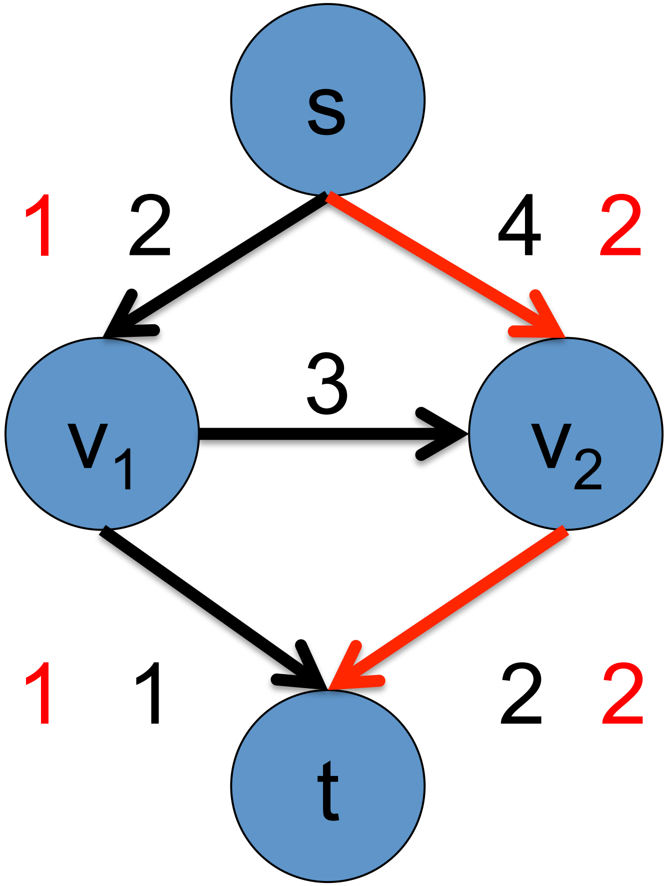
Pass maximum allowable flow through the arcs

Passing Flow through s-t Paths



Find an s-t path where flow(a) < c(a) for all arcs

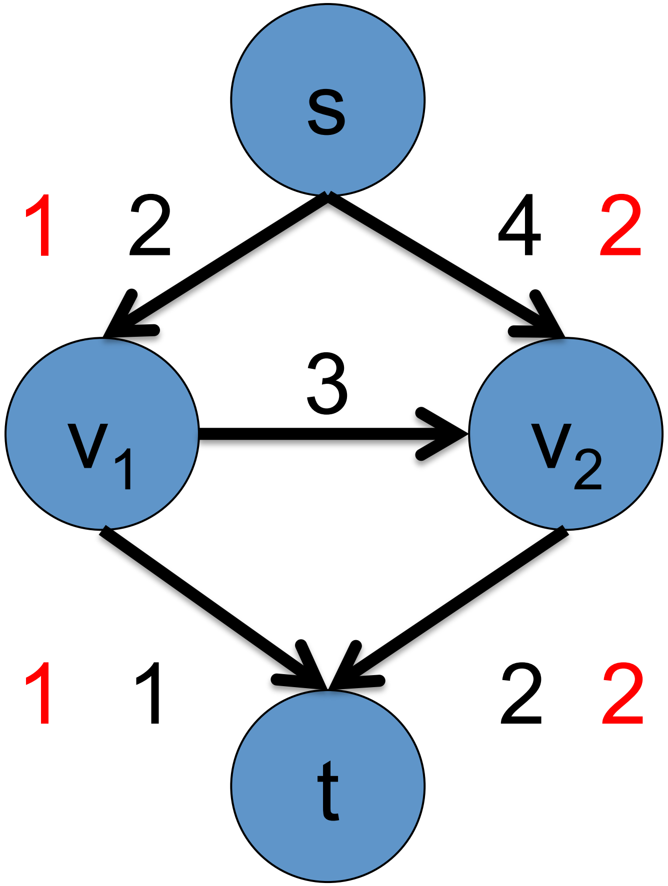
Passing Flow through s-t Paths



Find an s-t path where
flow(a) < c(a) for all arcs

Pass maximum allowable
flow through the arcs

Passing Flow through s-t Paths

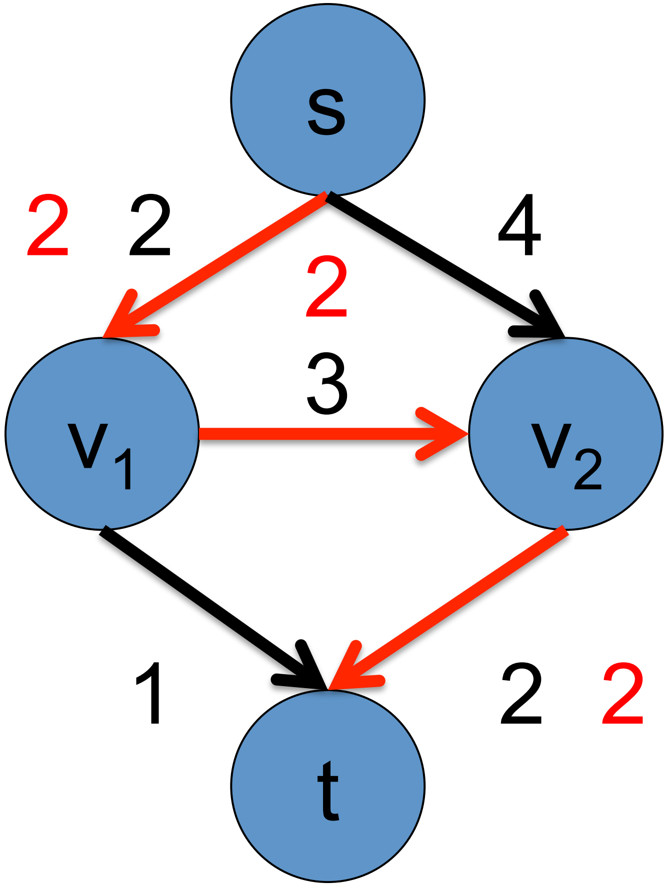


Find an s-t path where flow(a) < c(a) for all arcs

No more paths. Stop.

Will this give us maximum flow? **NO !!!**

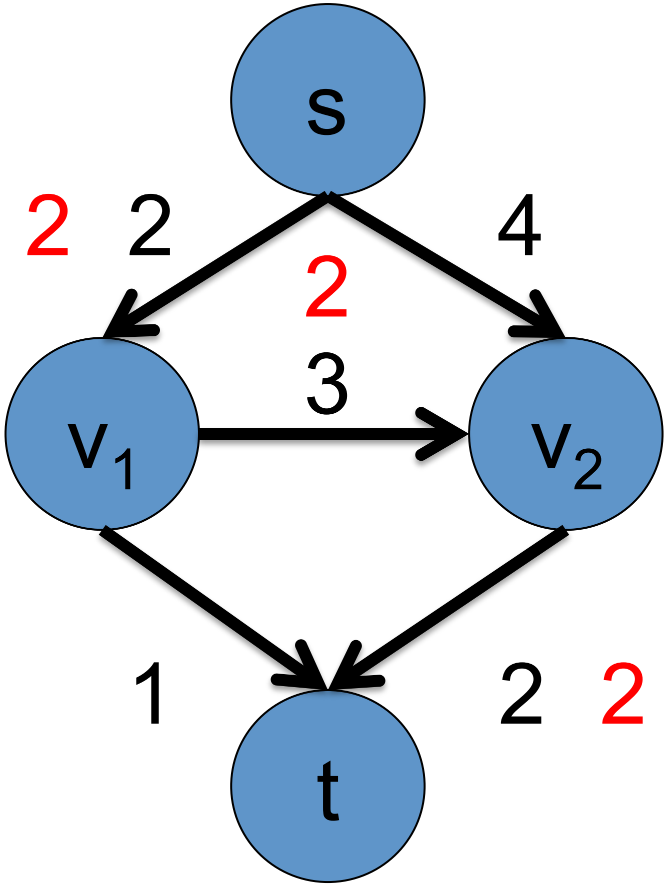
Passing Flow through s-t Paths



Find an s-t path where flow(a) < c(a) for all arcs

Pass maximum allowable flow through the arcs

Passing Flow through s-t Paths



Find an s-t path where flow(a) < c(a) for all arcs

No more paths. Stop.

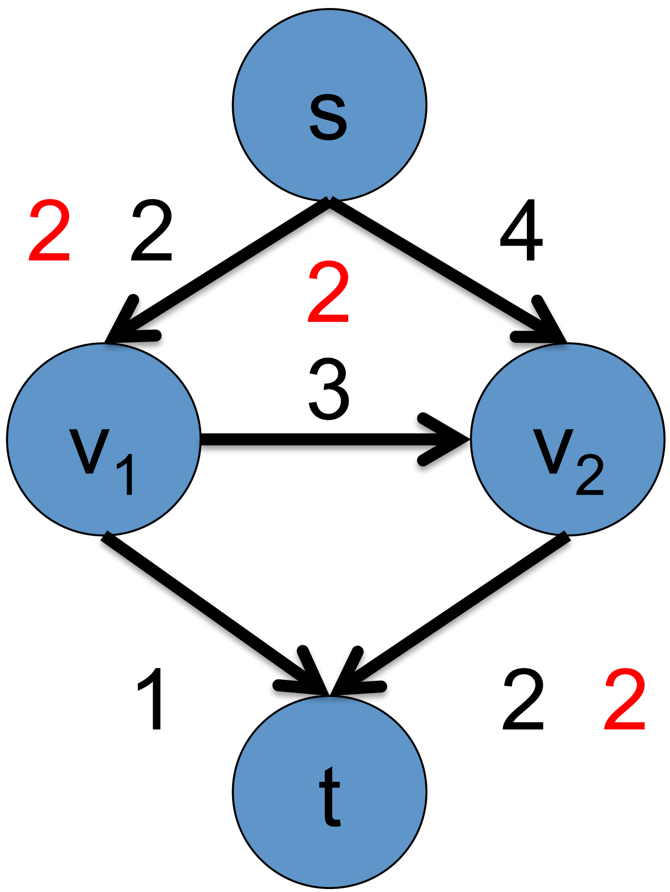
Another method?

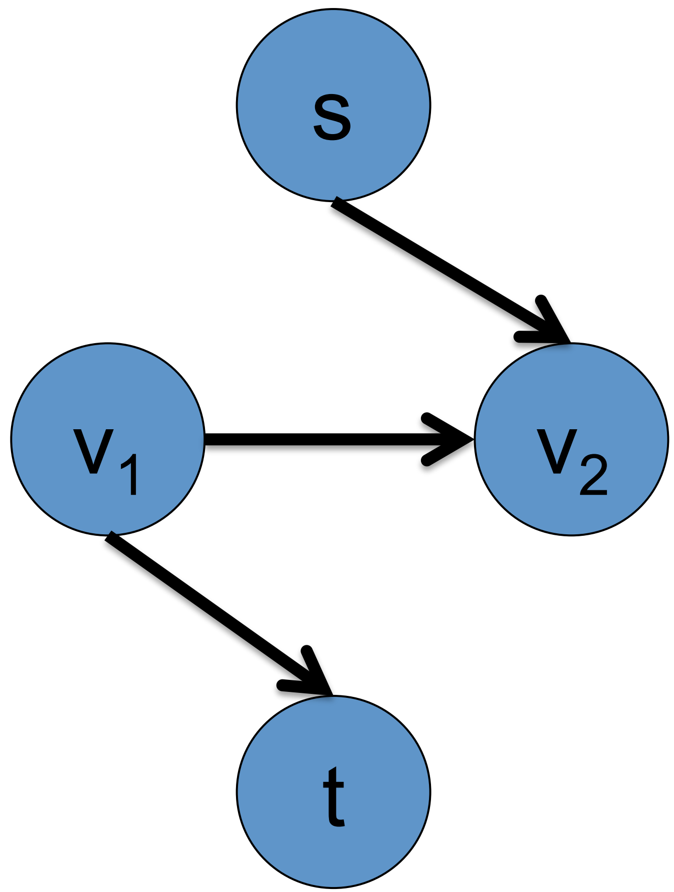
Incorrect Answer !!

Outline

- Preliminaries
- Maximum Flow
  - **Residual Graph**
  - Max-Flow Min-Cut Theorem
- Algorithms
- Energy minimization with max flow/min cut

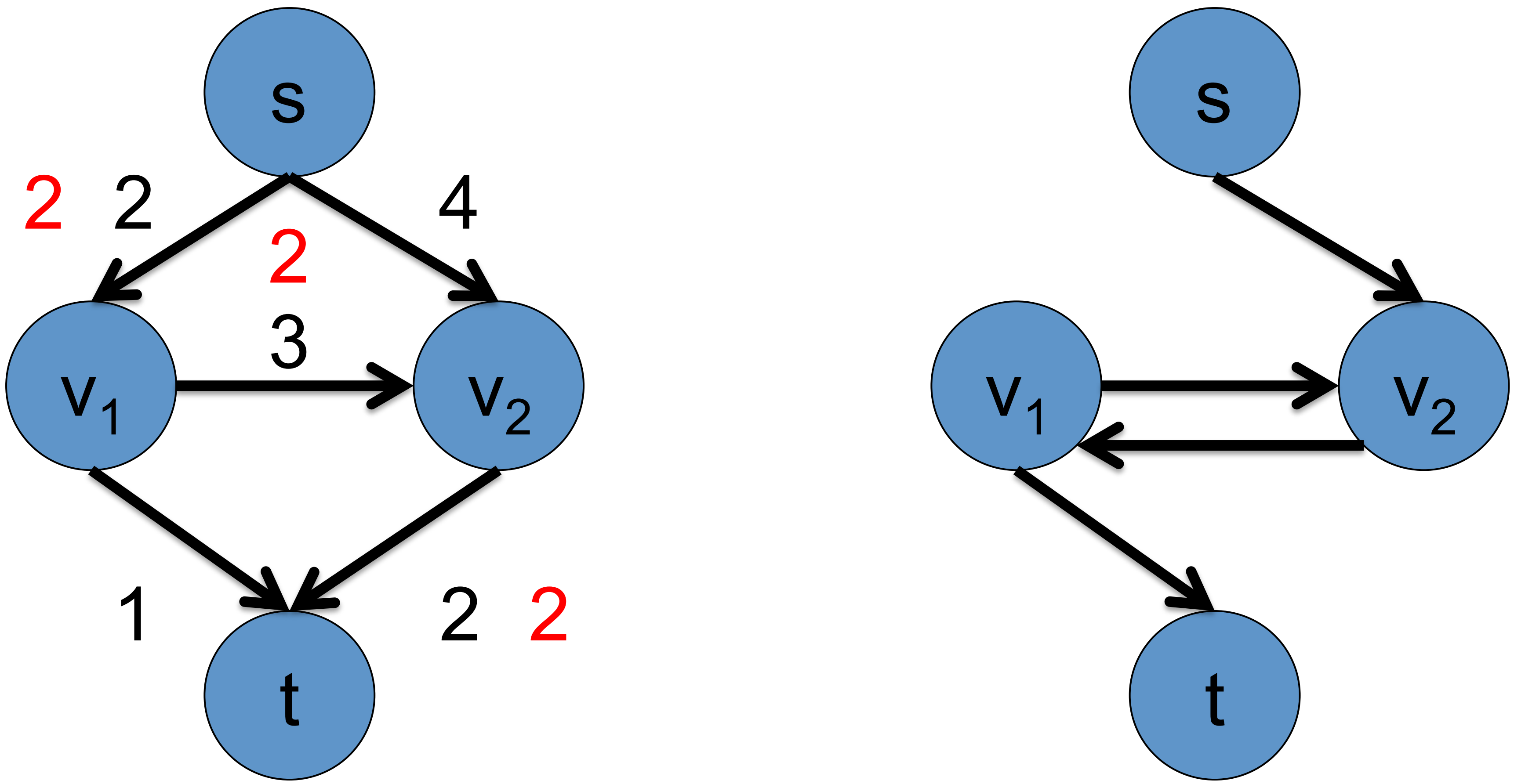
Residual Graph





Arcs where flow(a) < c(a)

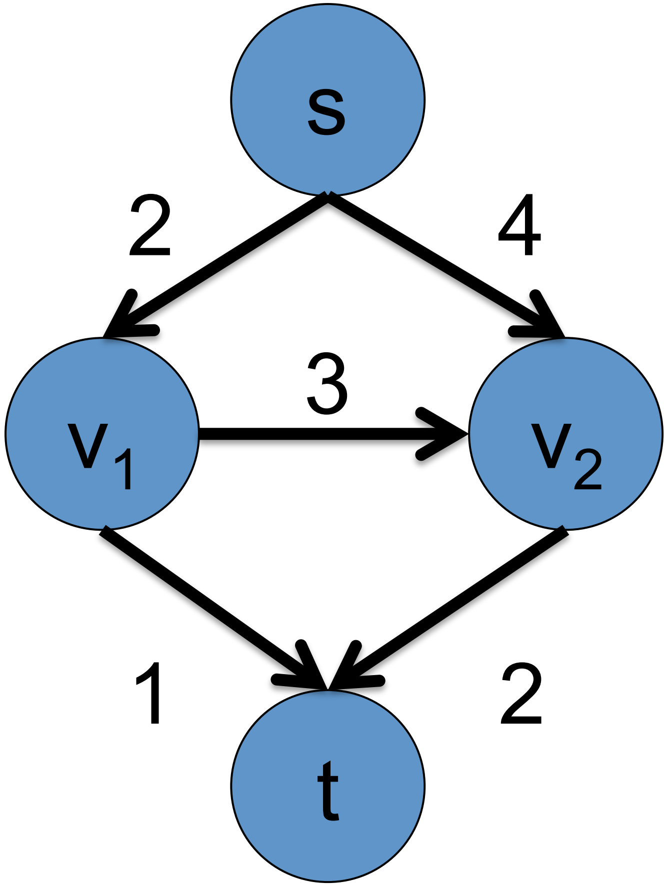
Residual Graph

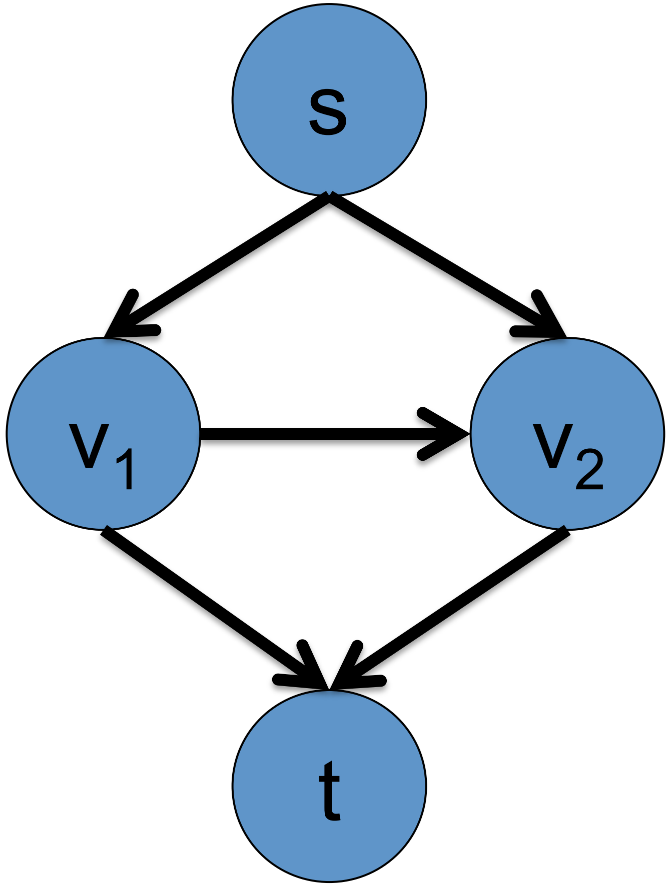


Including arcs to s and from t is not necessary

Inverse of arcs where flow(a) > 0

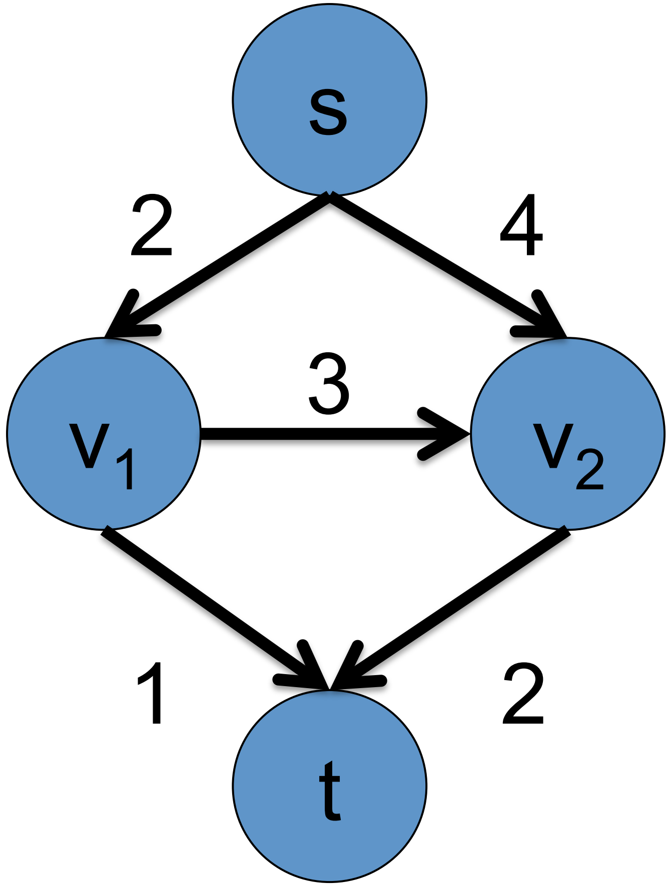
Maximum Flow using Residual Graphs

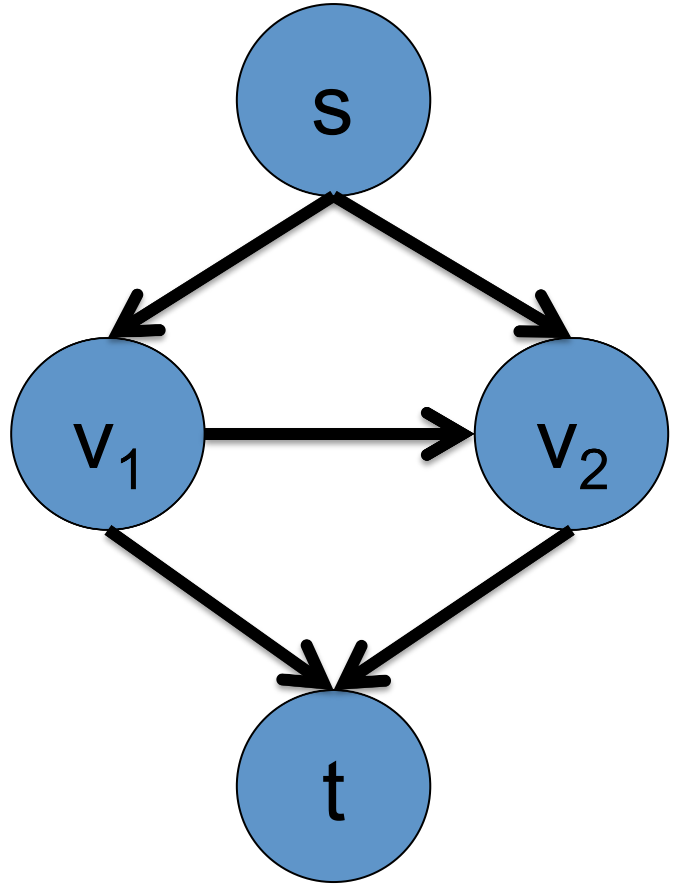




Start with zero flow.

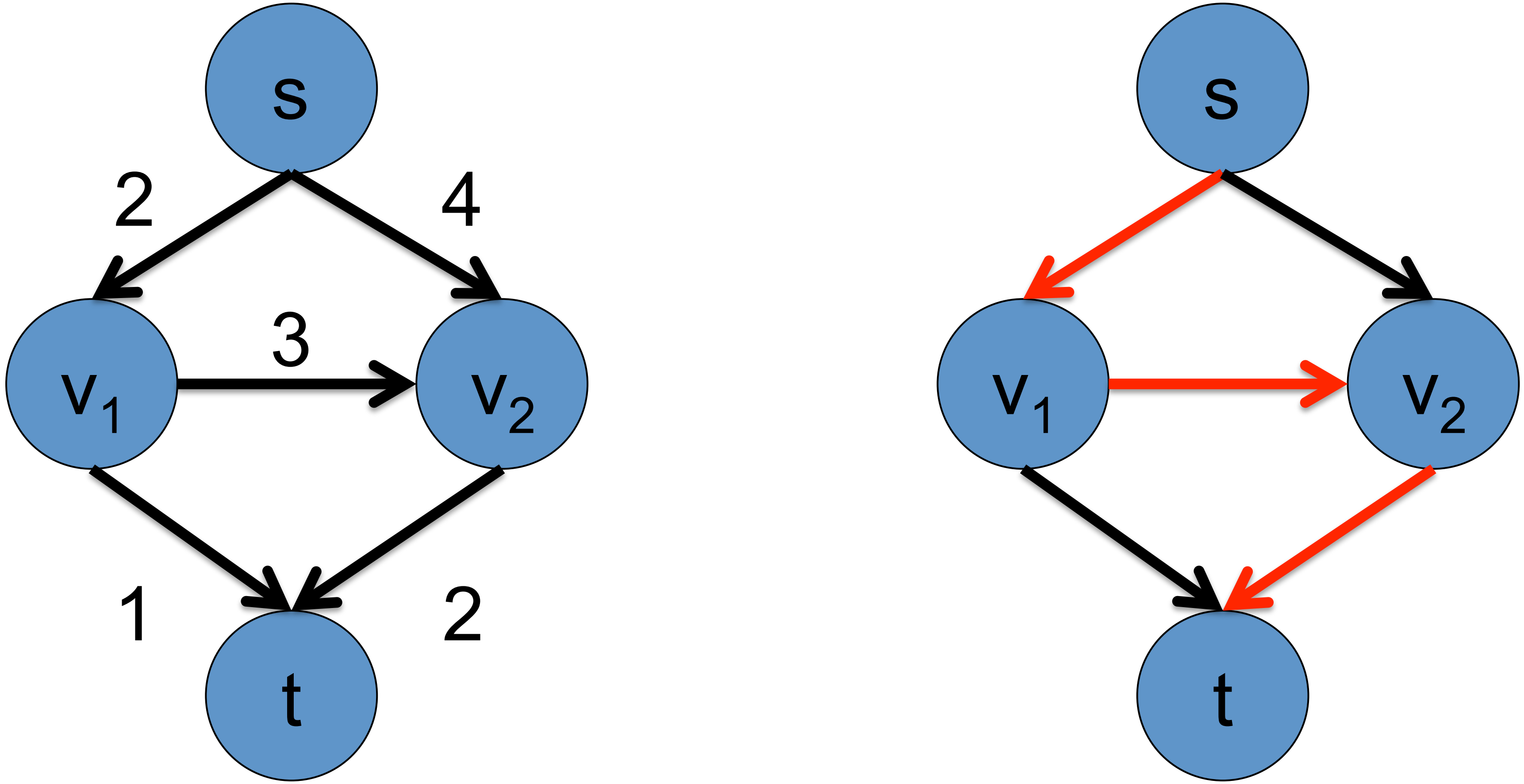
Maximum Flow using Residual Graphs





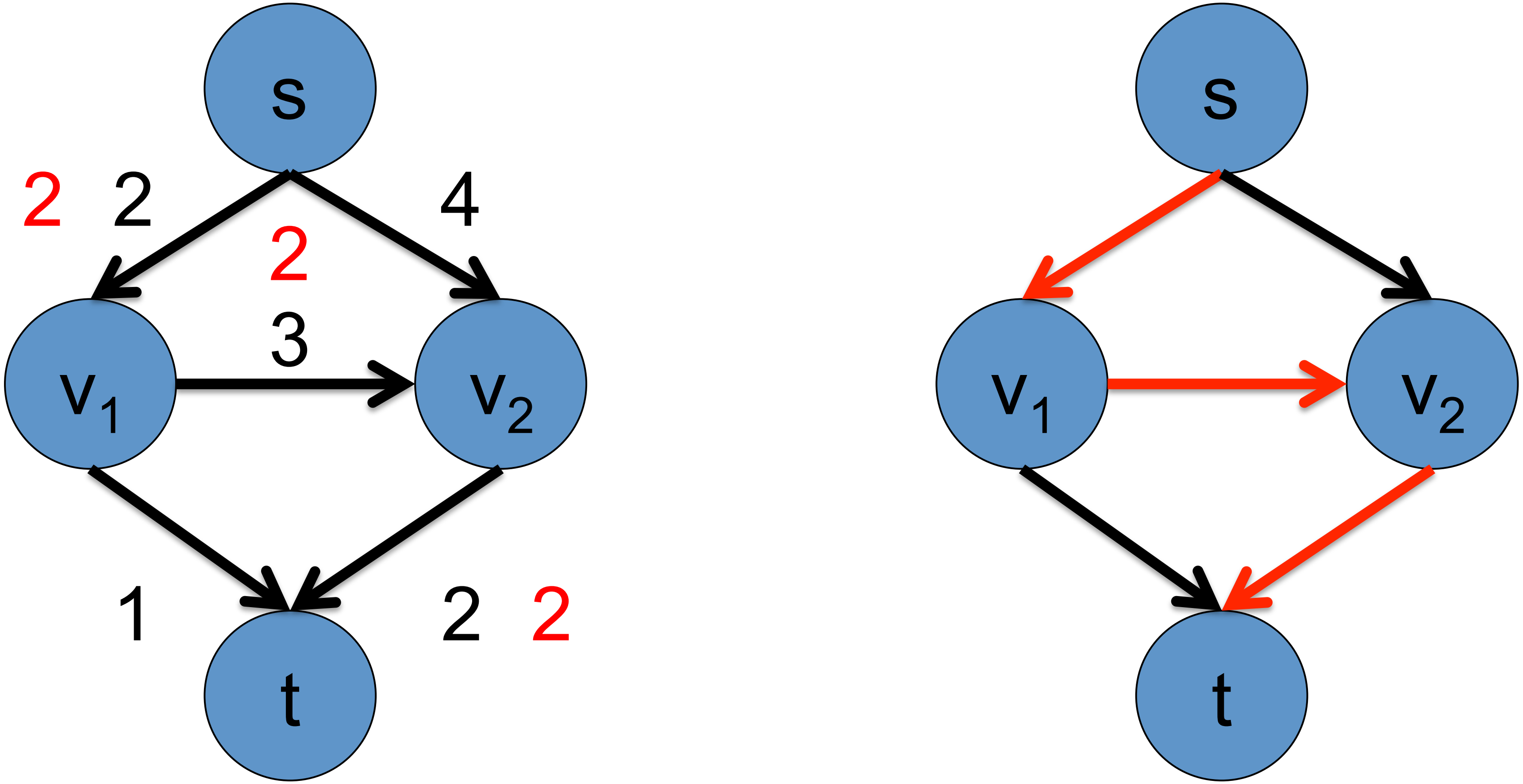
Find an s-t path in the residual graph.

Maximum Flow using Residual Graphs



For inverse arcs in path, subtract flow K.

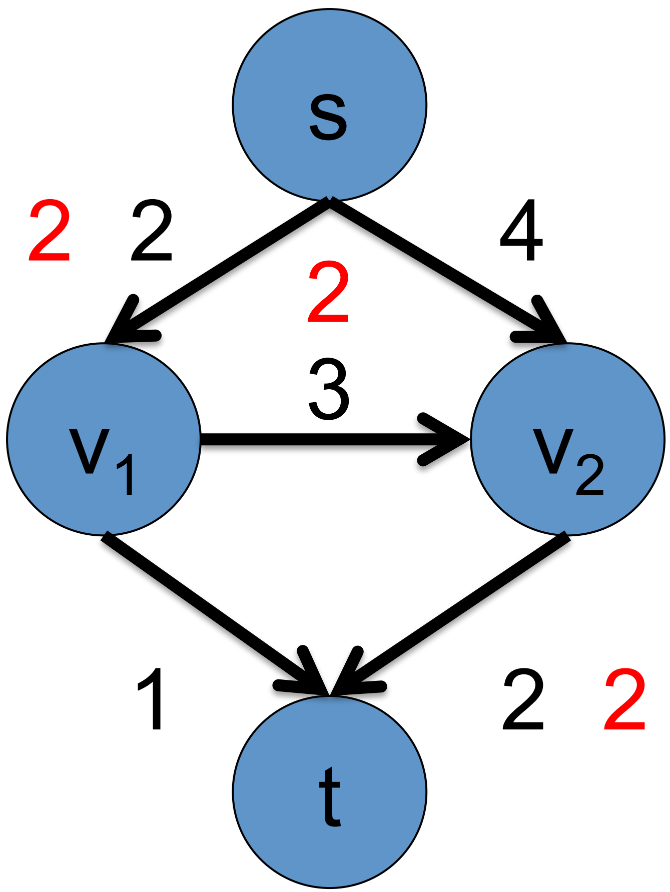
Maximum Flow using Residual Graphs

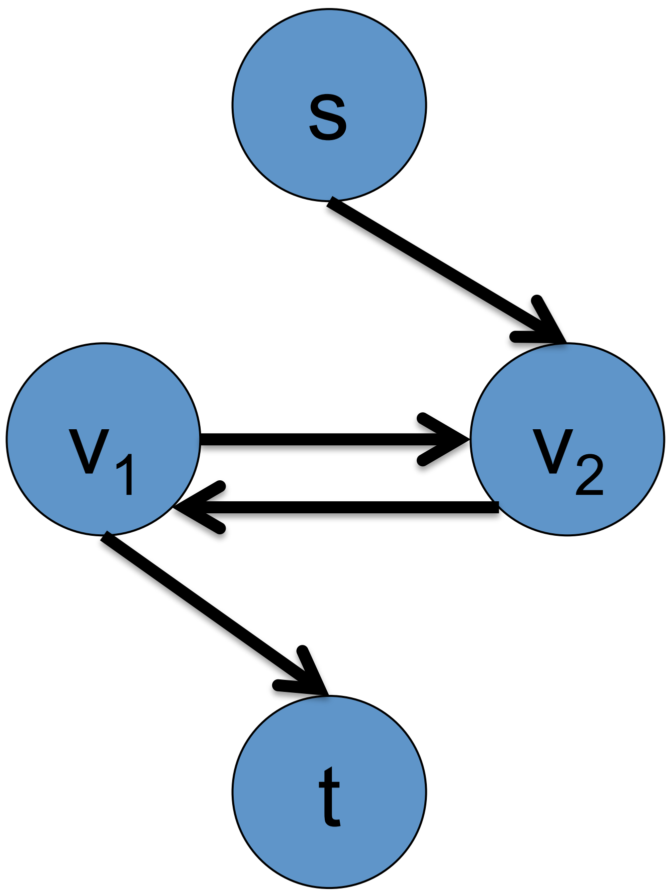


Choose maximum allowable value of K.

For forward arcs in path, add flow K.

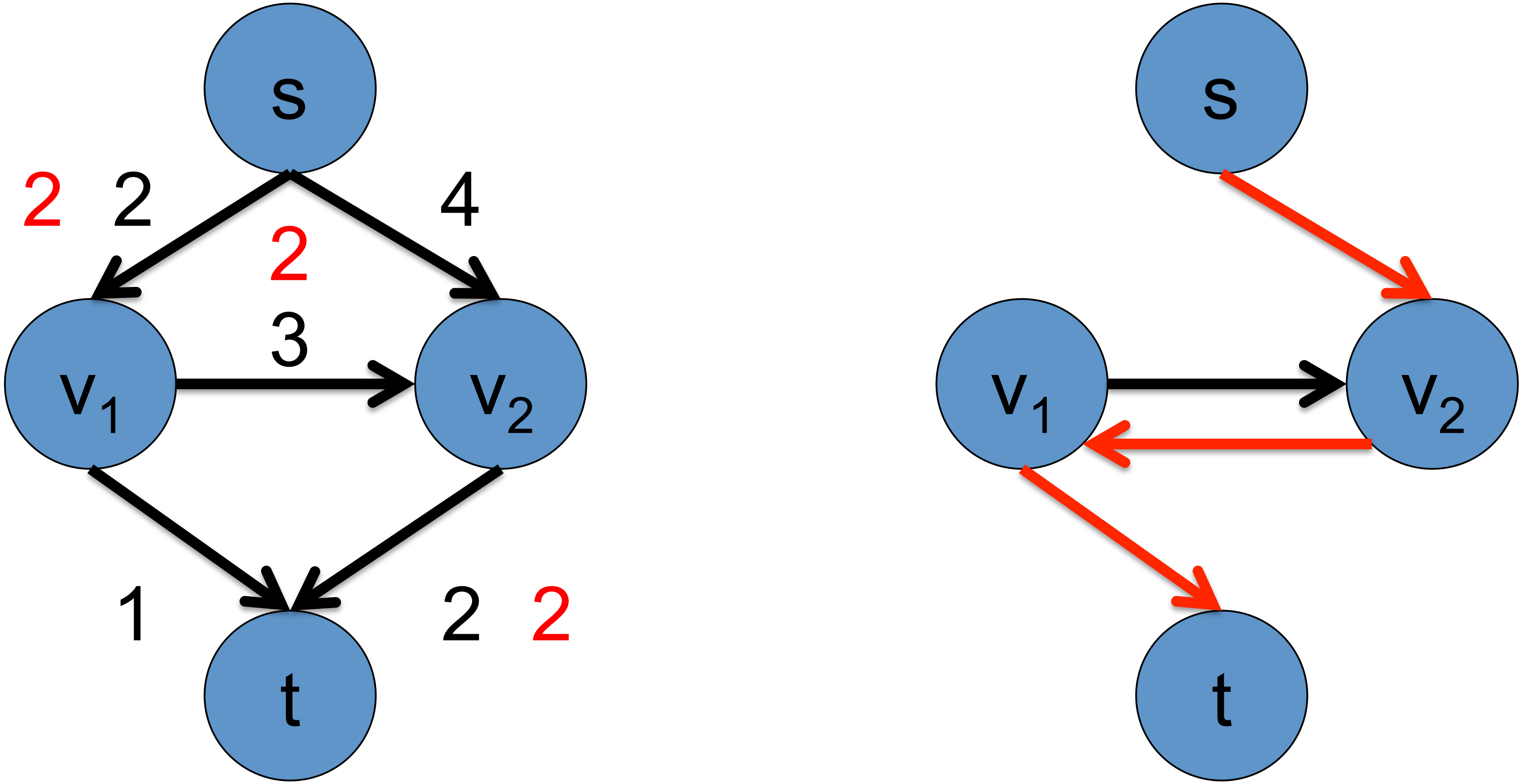
Maximum Flow using Residual Graphs





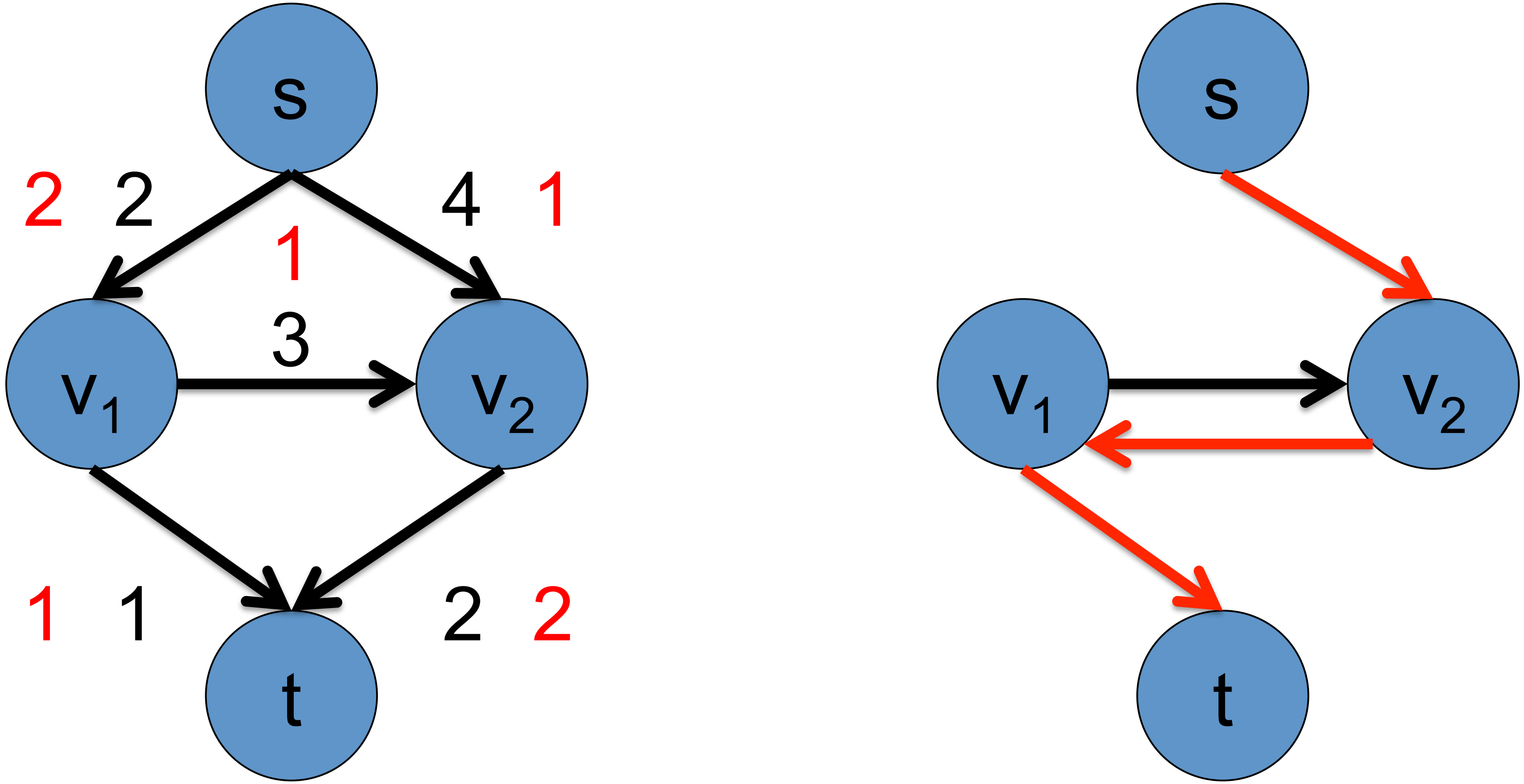
Update the residual graph.

Maximum Flow using Residual Graphs



Find an s-t path in the residual graph.

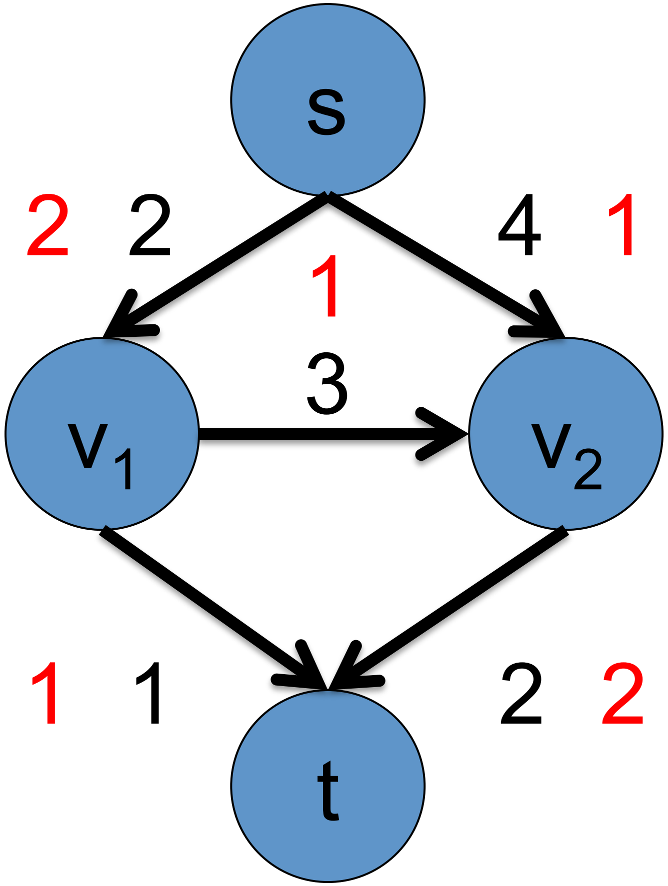
Maximum Flow using Residual Graphs

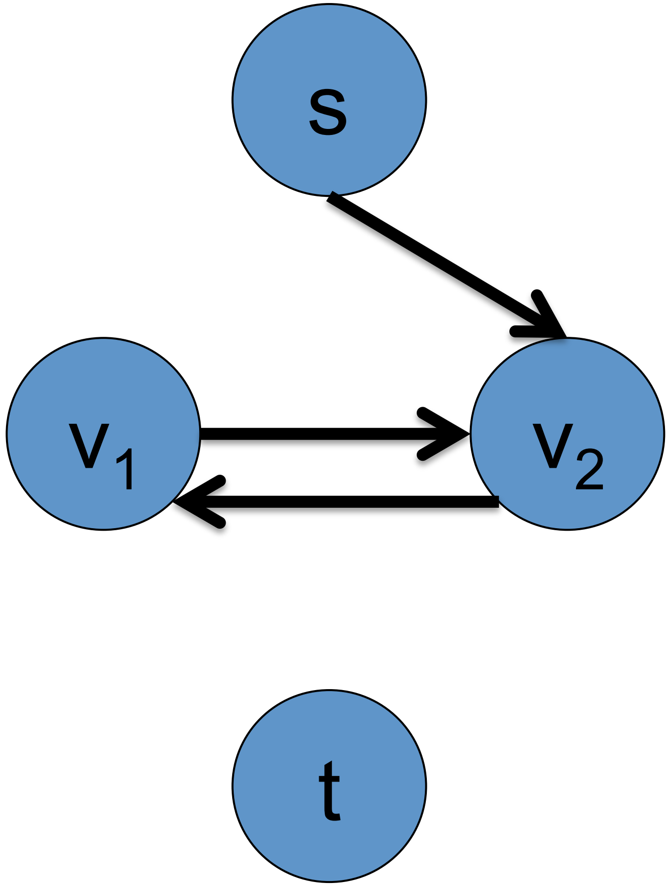


Choose maximum allowable value of K.

Add K to (s,v_2) and (v_1,t). Subtract K from (v_1,v_2).

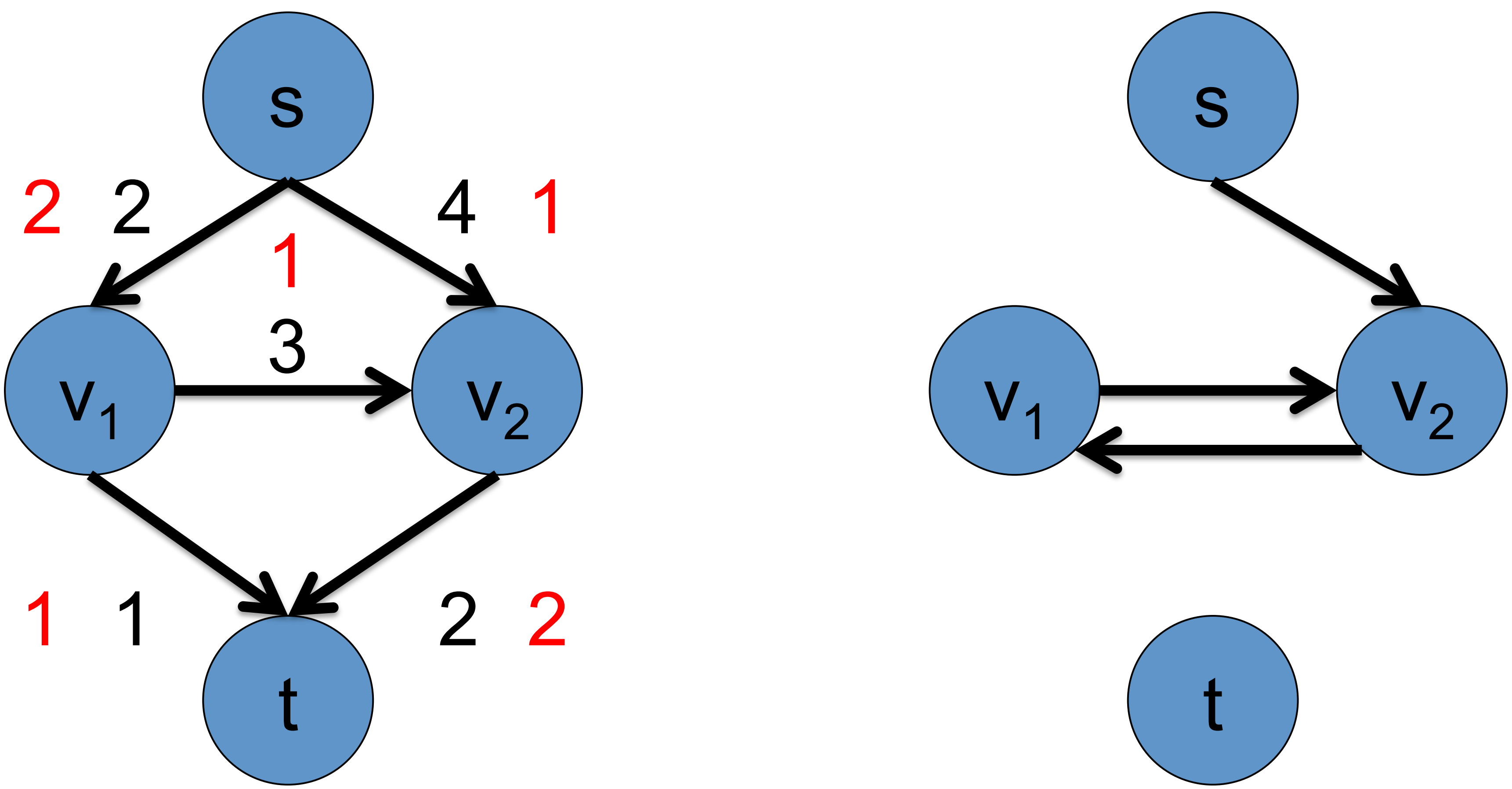
Maximum Flow using Residual Graphs





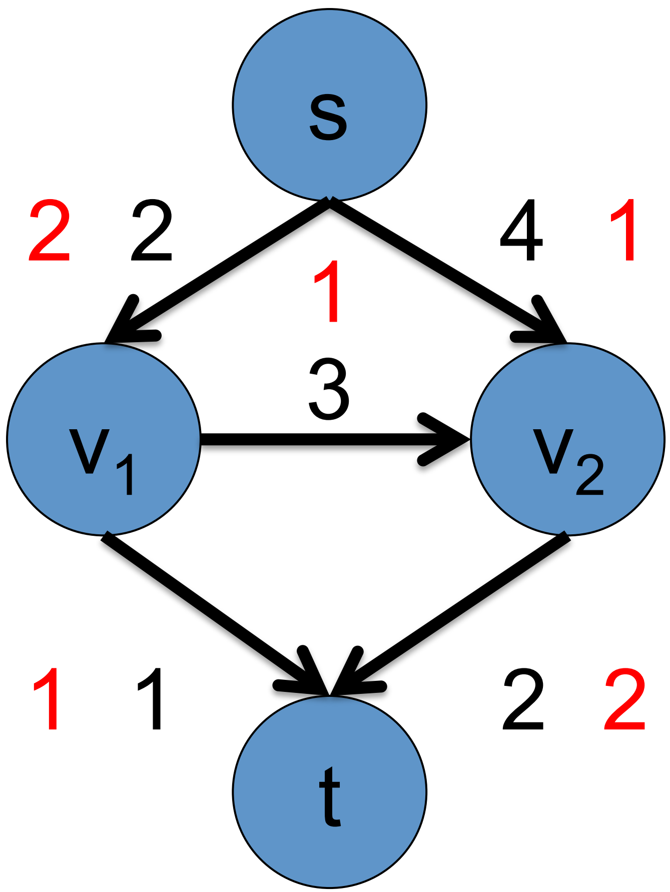
Update the residual graph.

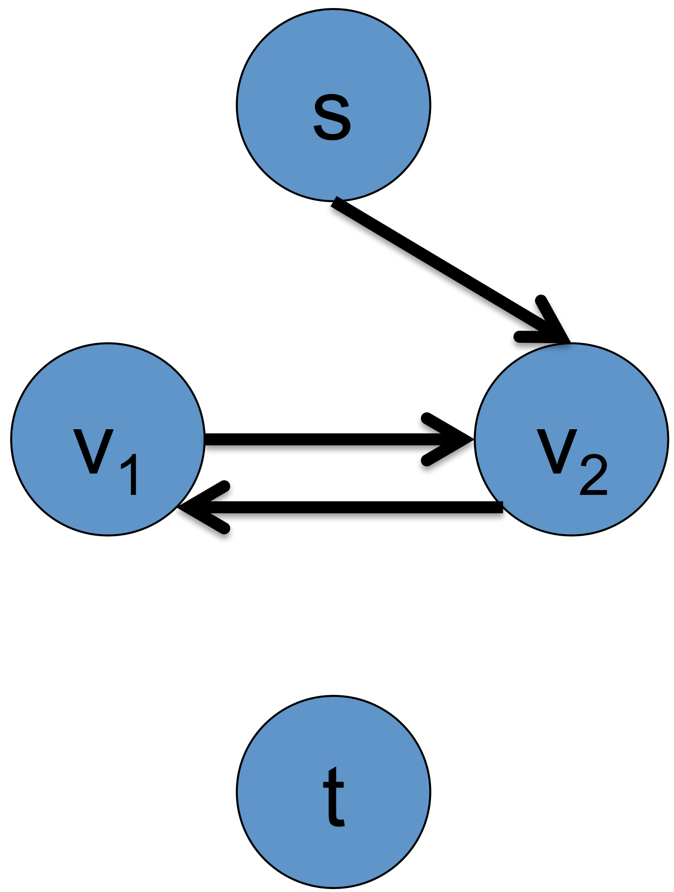
Maximum Flow using Residual Graphs



Find an s-t path in the residual graph.

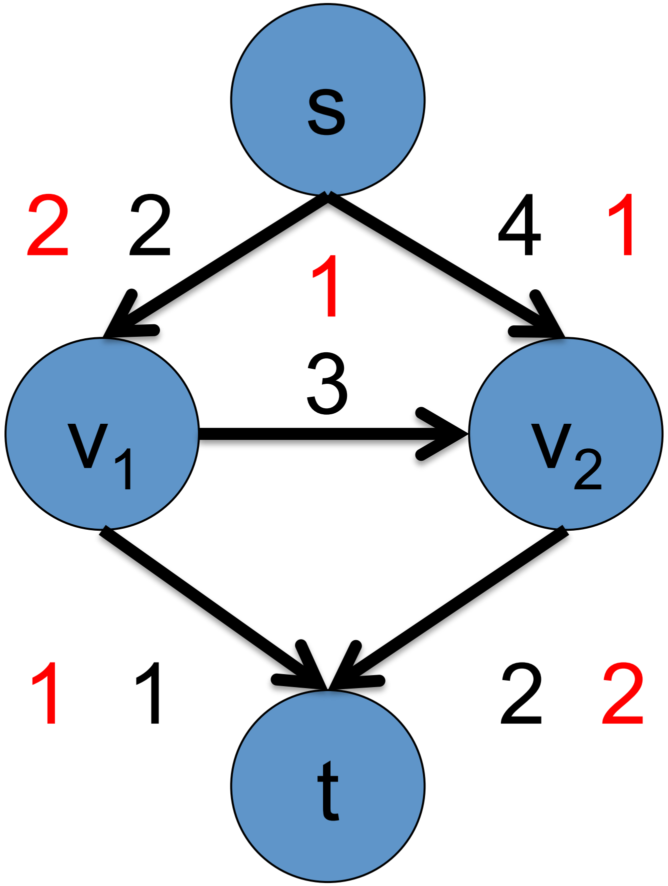
Maximum Flow using Residual Graphs

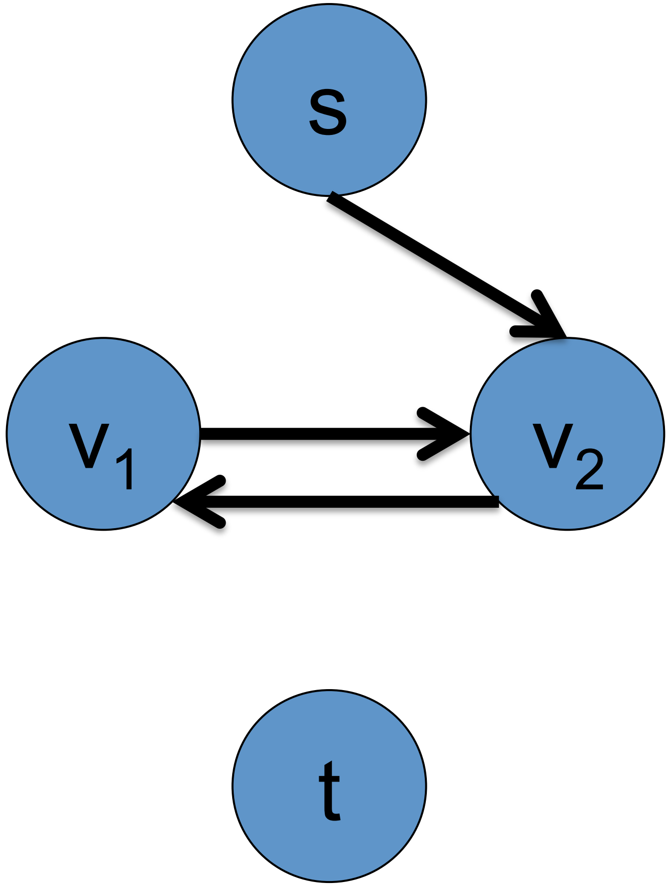




No more s-t paths. Stop.

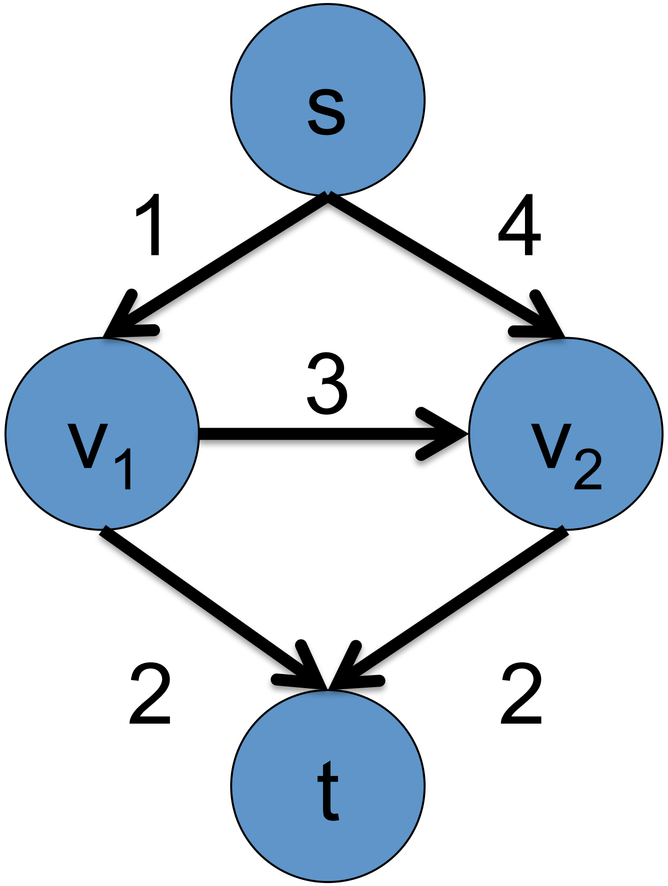
Maximum Flow using Residual Graphs

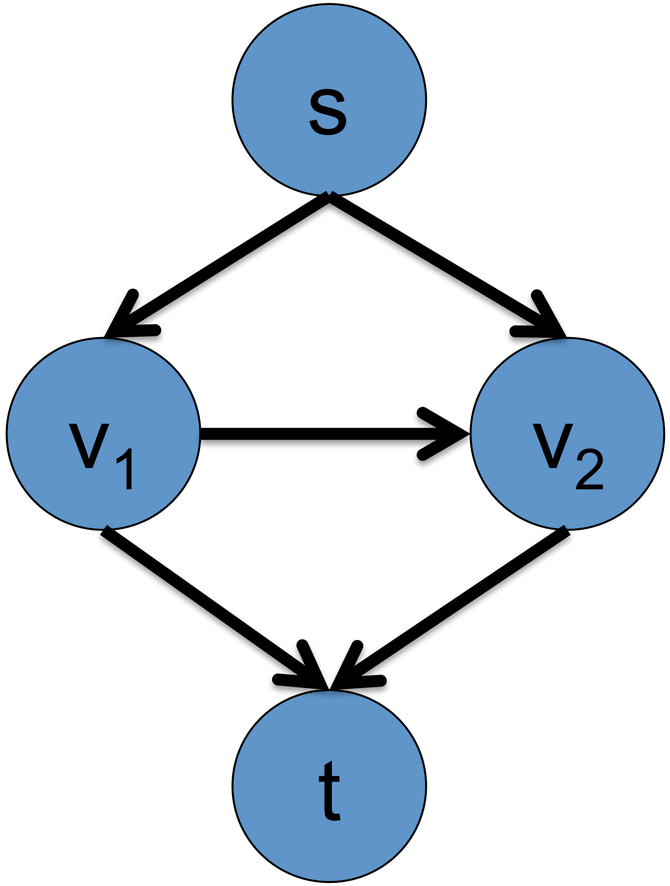




Correct Answer.

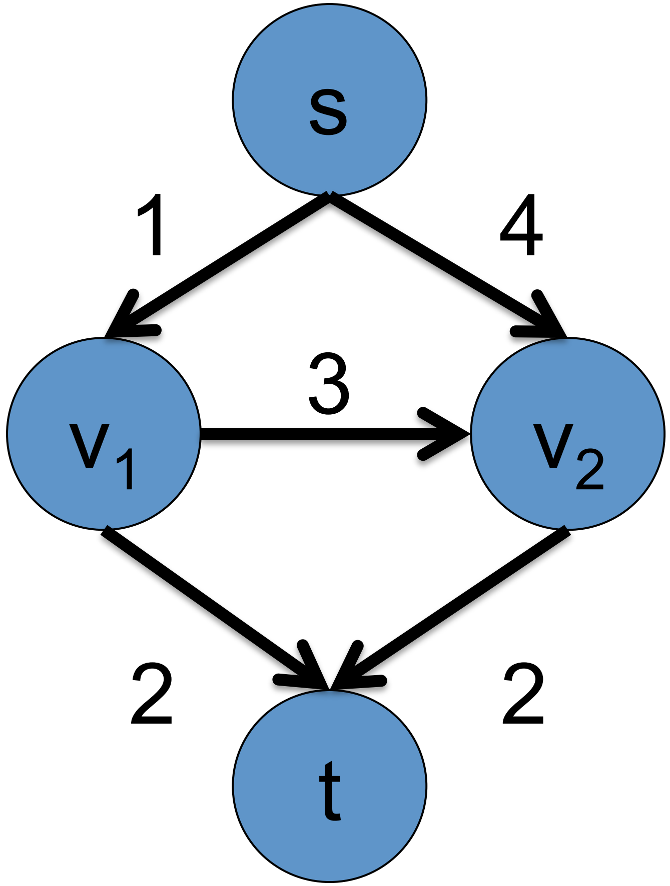
Maximum Flow using Residual Graphs

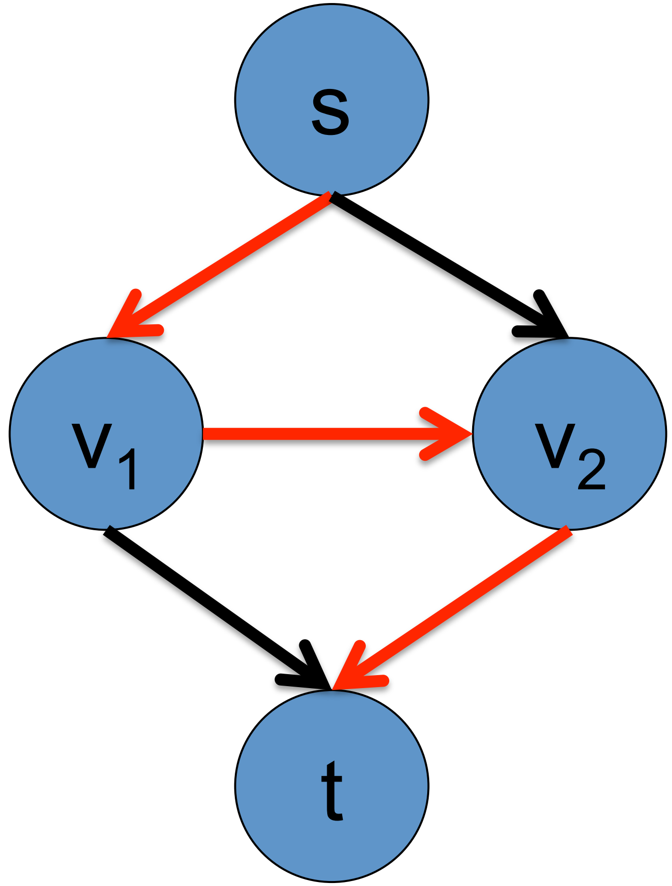




Start with zero flow

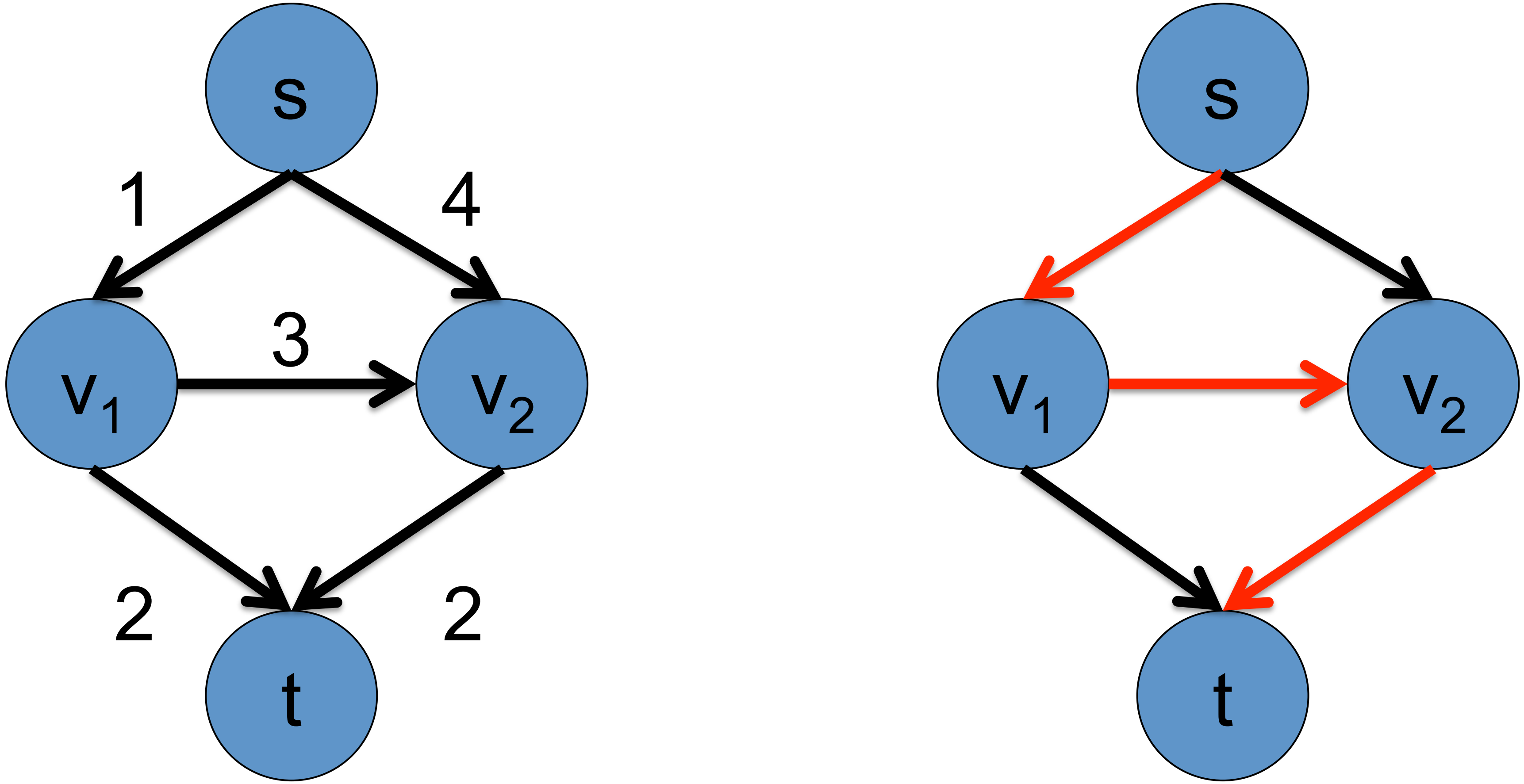
Maximum Flow using Residual Graphs





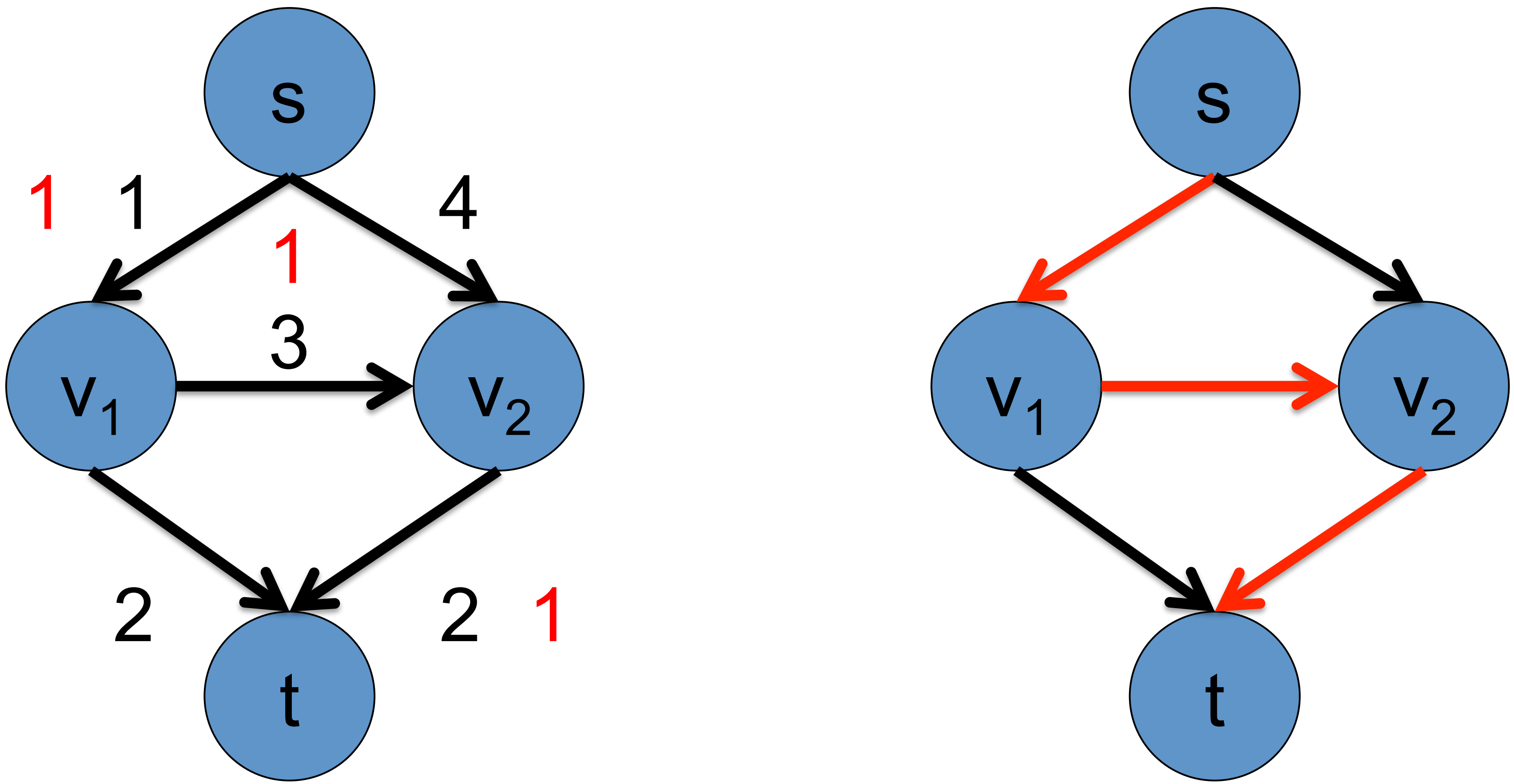
Find an s-t path in the residual graph.

Maximum Flow using Residual Graphs



For inverse arcs in path, subtract flow K.

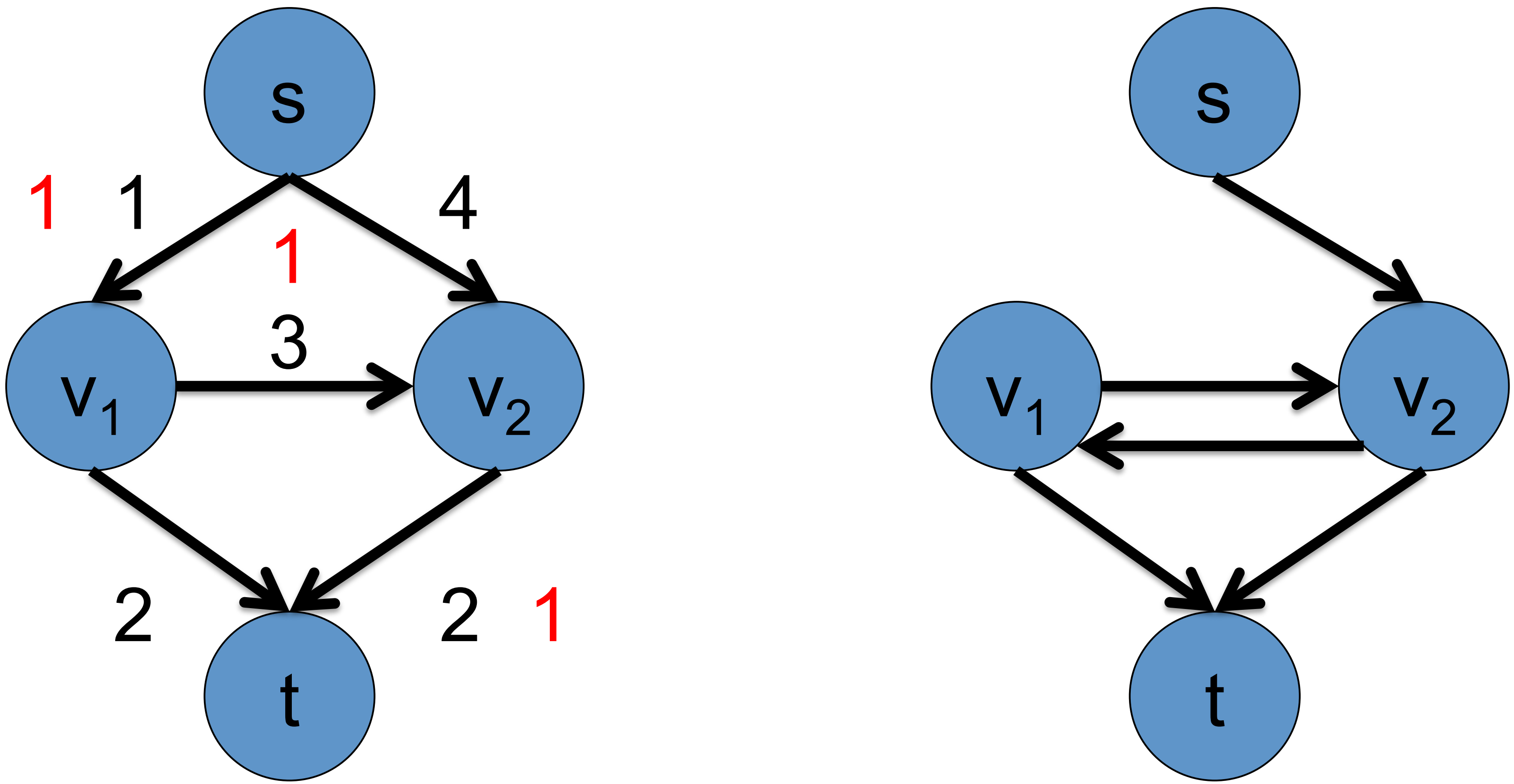
Maximum Flow using Residual Graphs



Choose maximum allowable value of K.

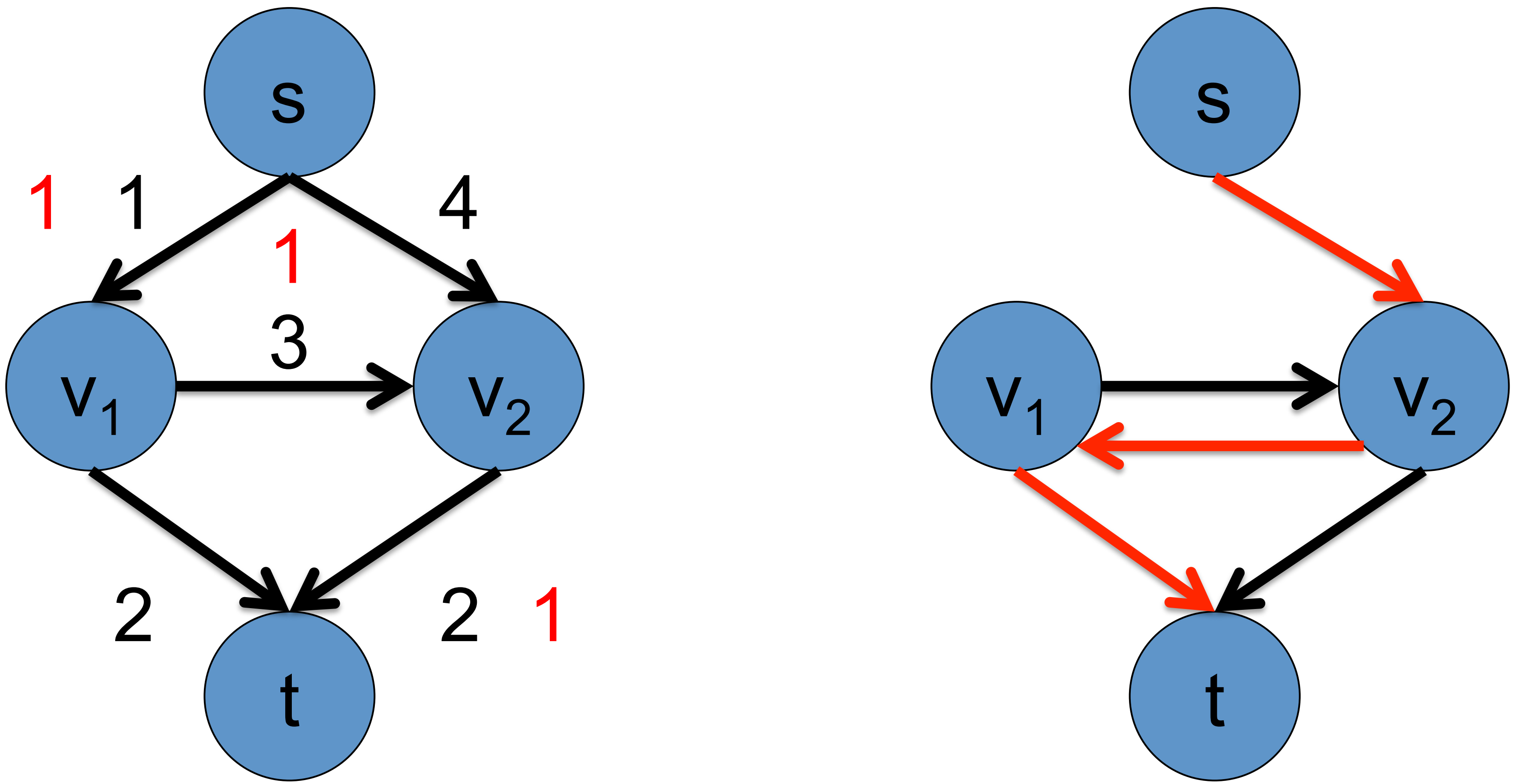
For forward arcs in path, add flow K.

Maximum Flow using Residual Graphs



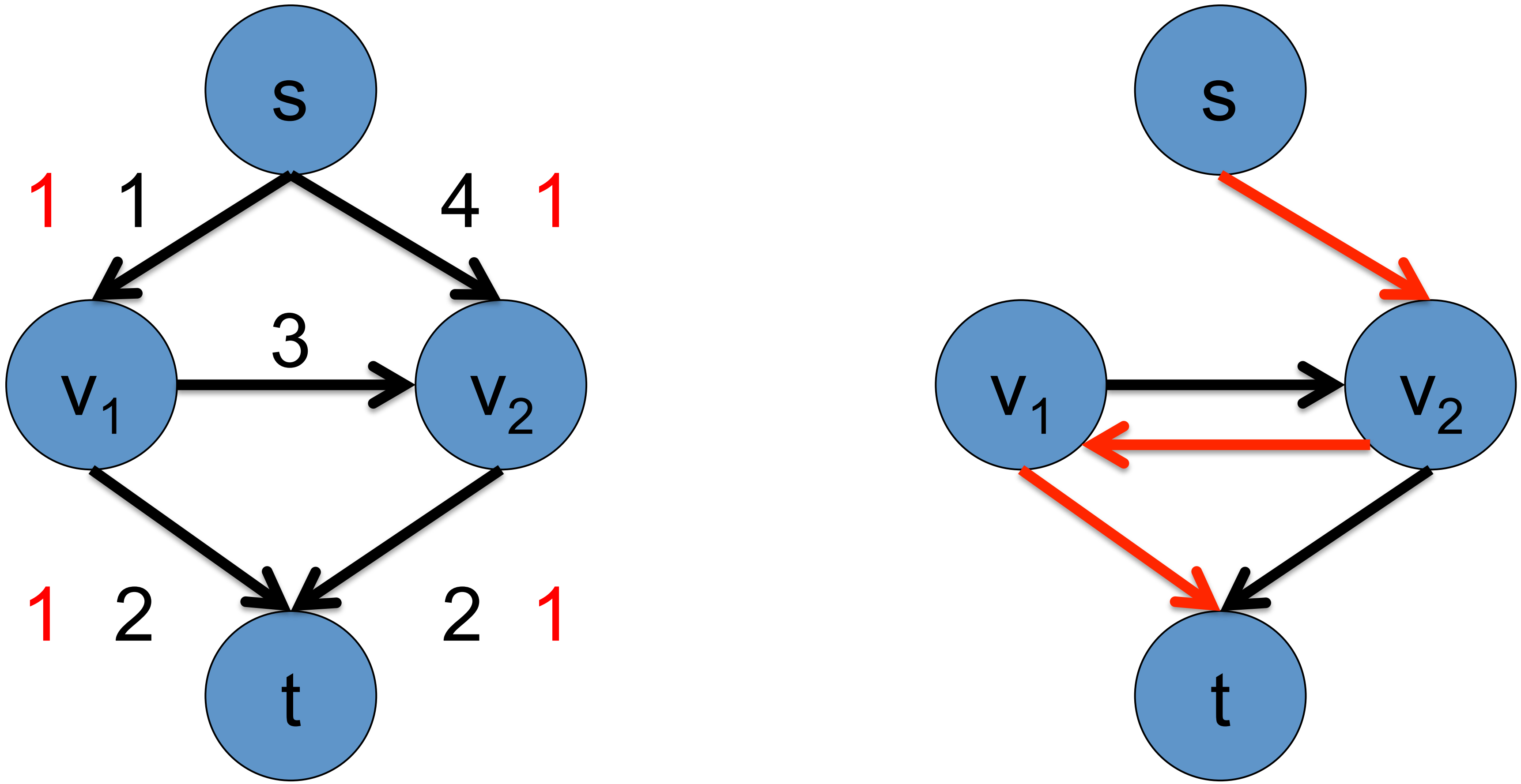
Update the residual graph.

Maximum Flow using Residual Graphs



Find an s-t path in the residual graph.

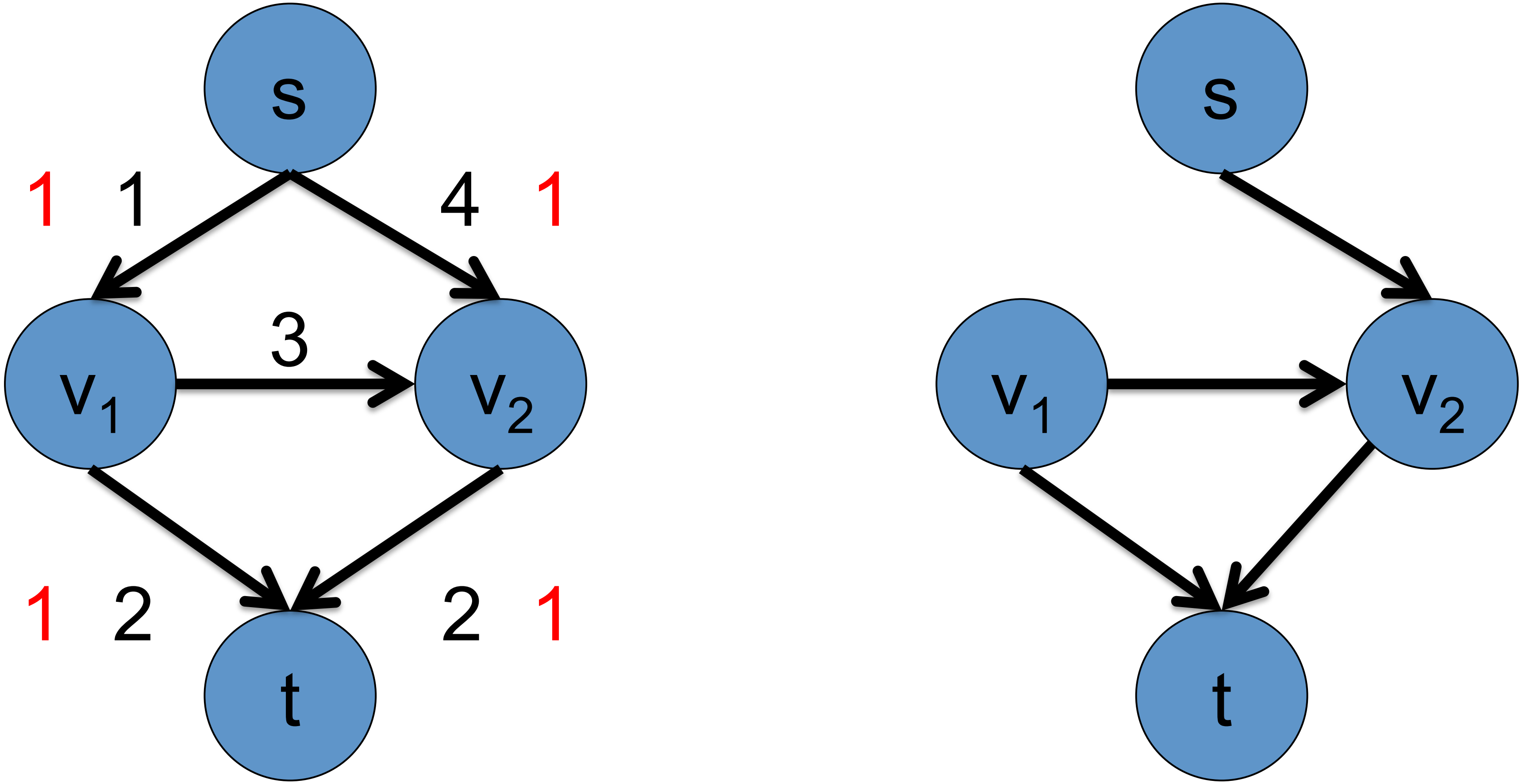
Maximum Flow using Residual Graphs



Choose maximum allowable value of K.

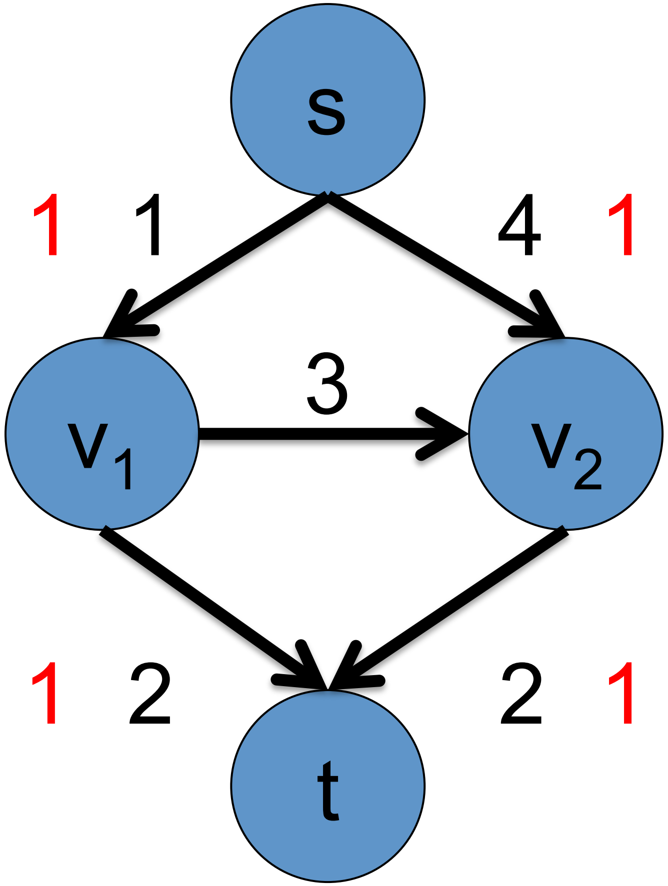
Add K to (s,v_2) and (v_1,t). Subtract K from (v_1,v_2).

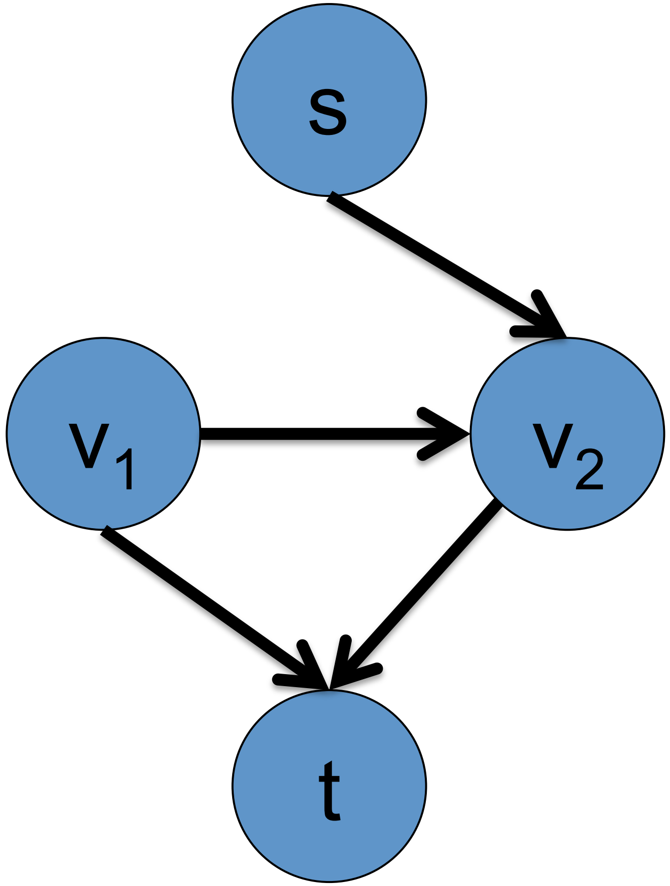
Maximum Flow using Residual Graphs



Update the residual graph.

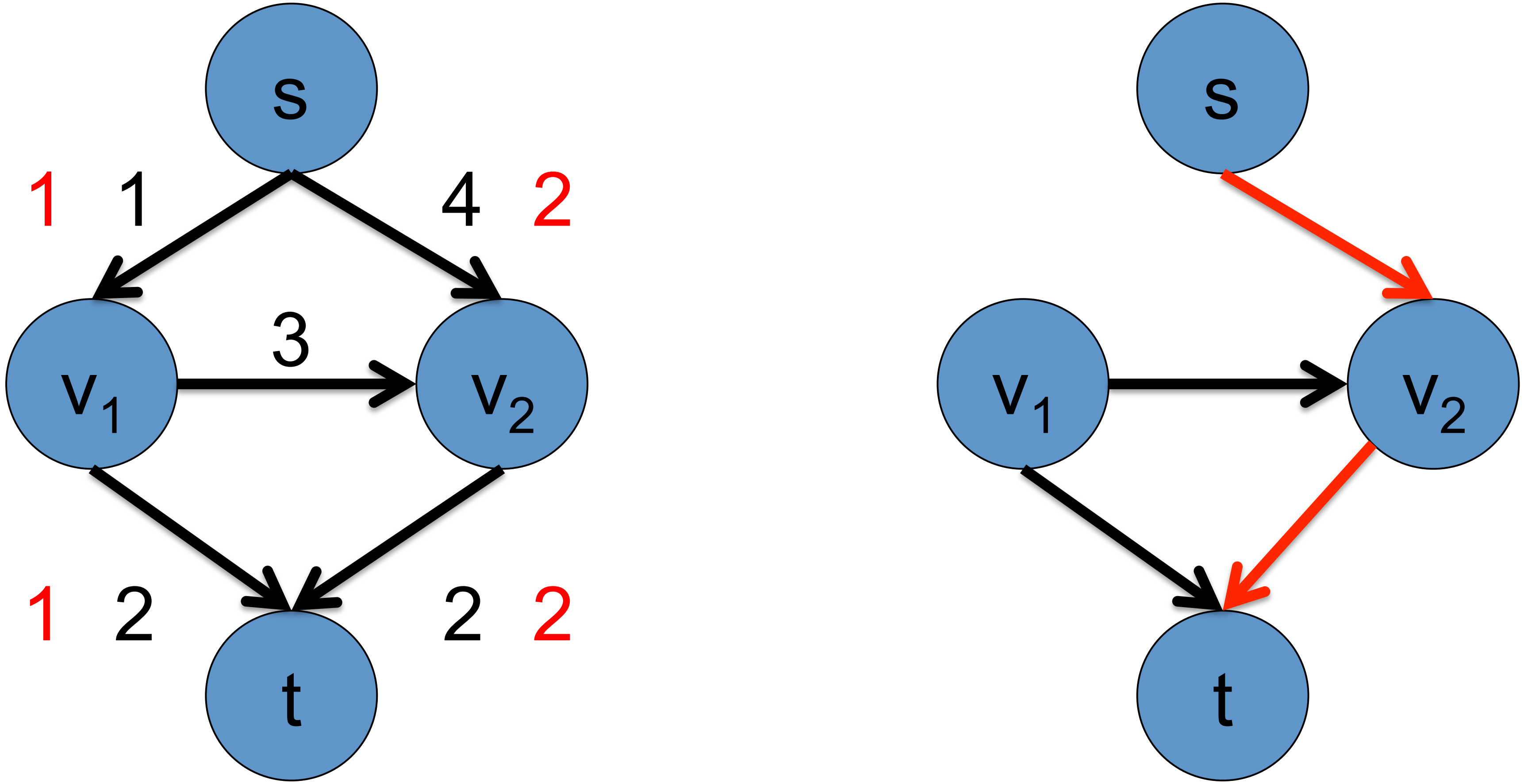
Maximum Flow using Residual Graphs





Find an s-t path in the residual graph.

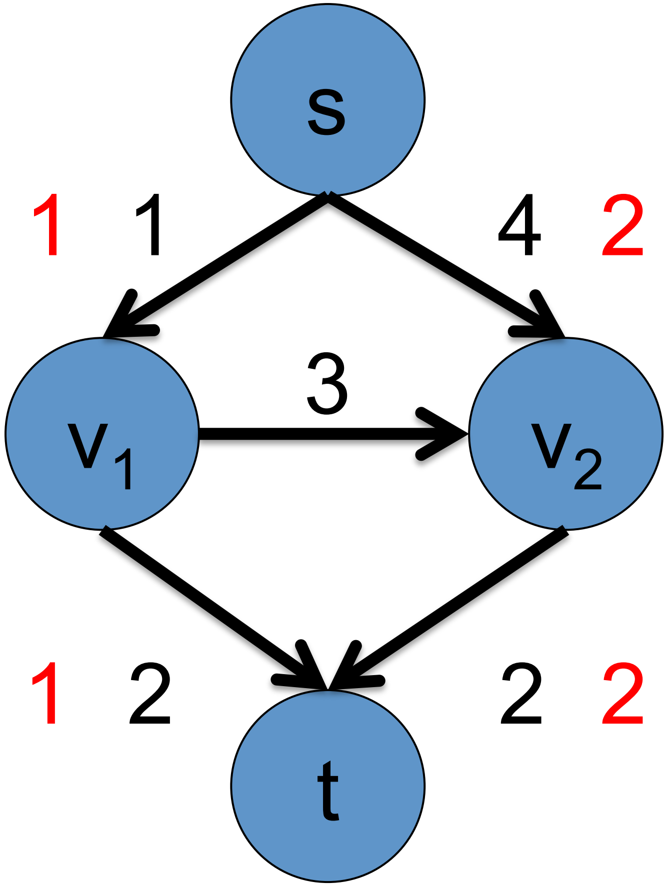
Maximum Flow using Residual Graphs

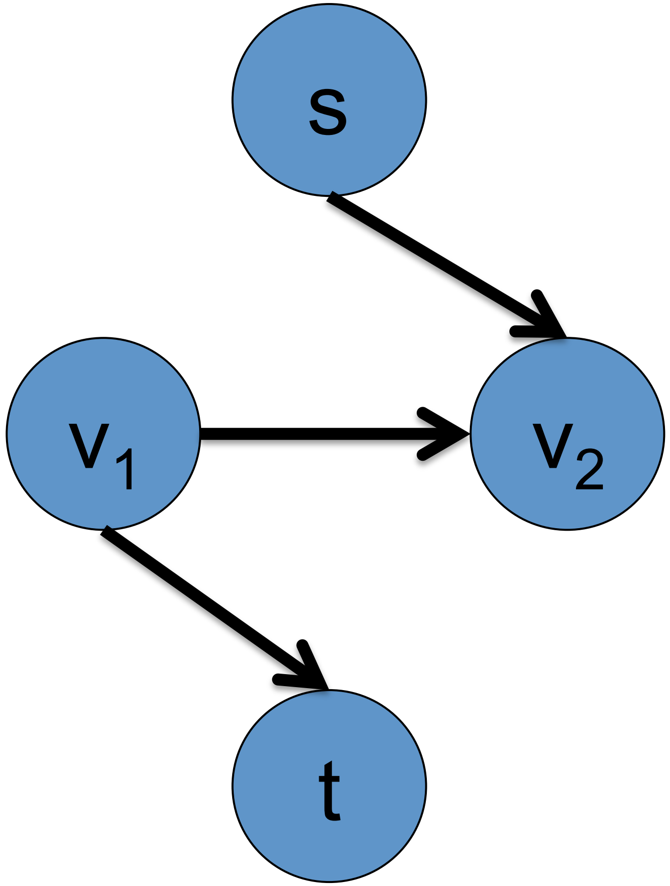


Choose maximum allowable value of K.

Add K to (s,v_2) and (v_2,t).

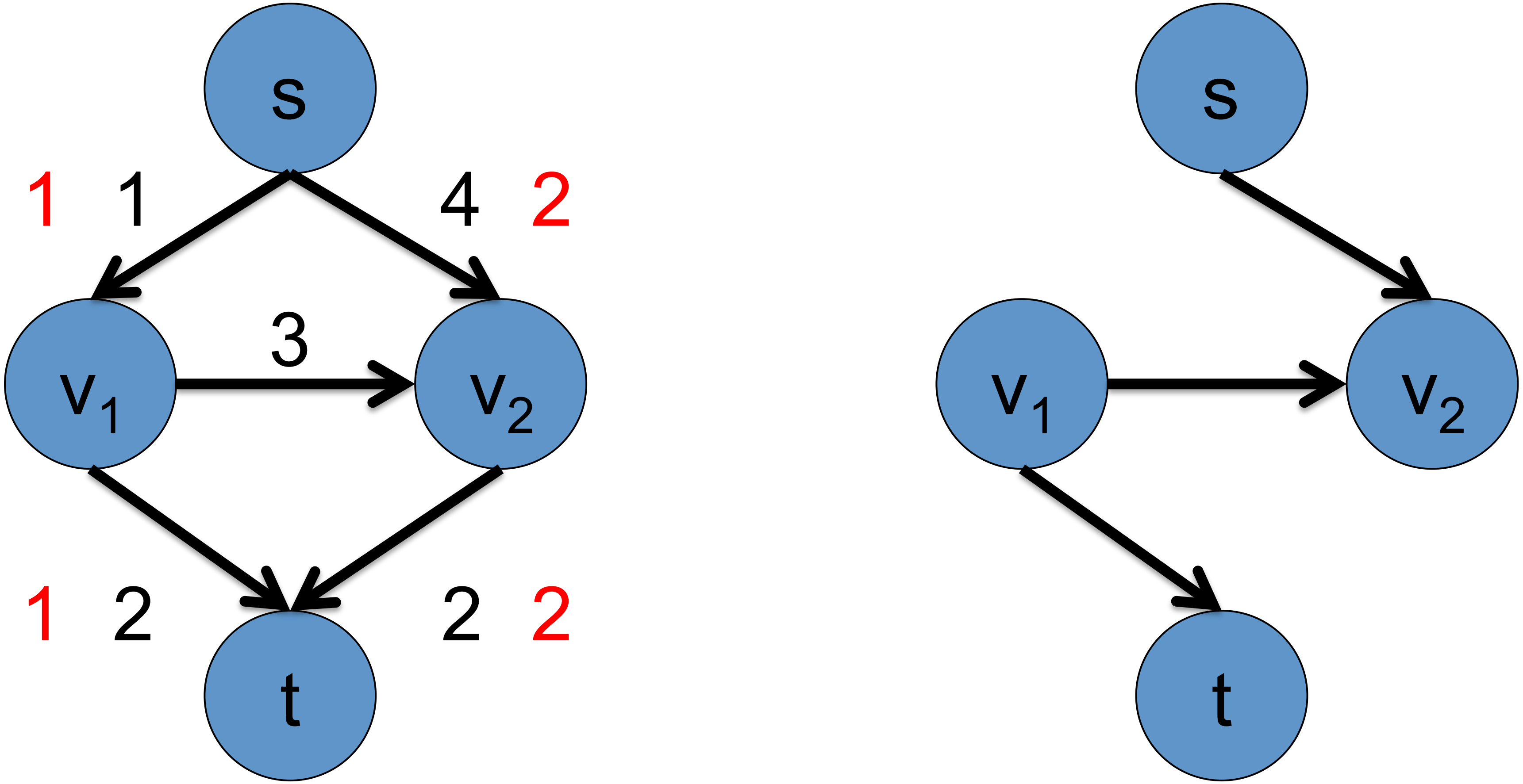
Maximum Flow using Residual Graphs





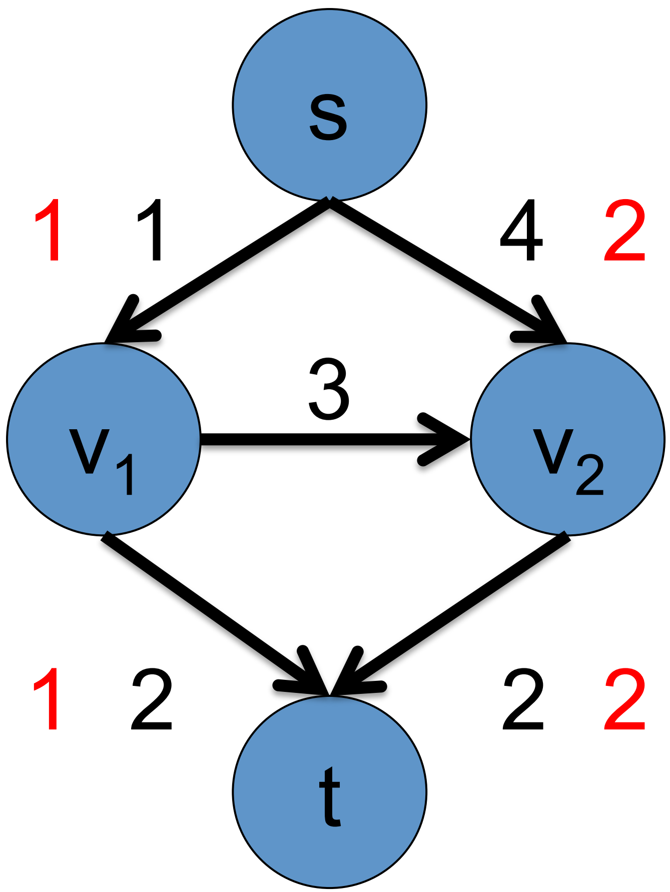
Update residual graph.

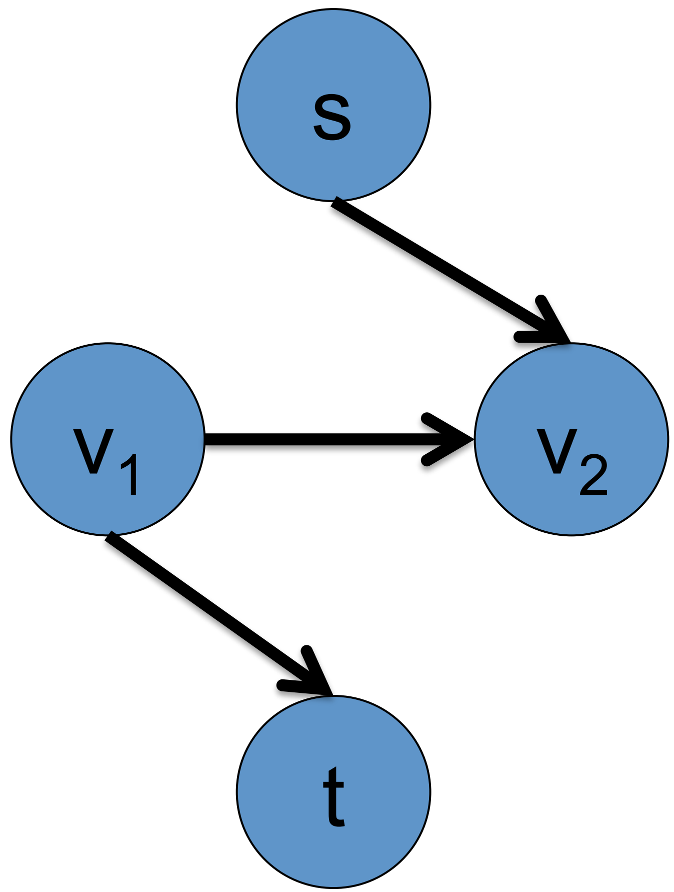
Maximum Flow using Residual Graphs



No more s-t paths. Stop.

Maximum Flow using Residual Graphs



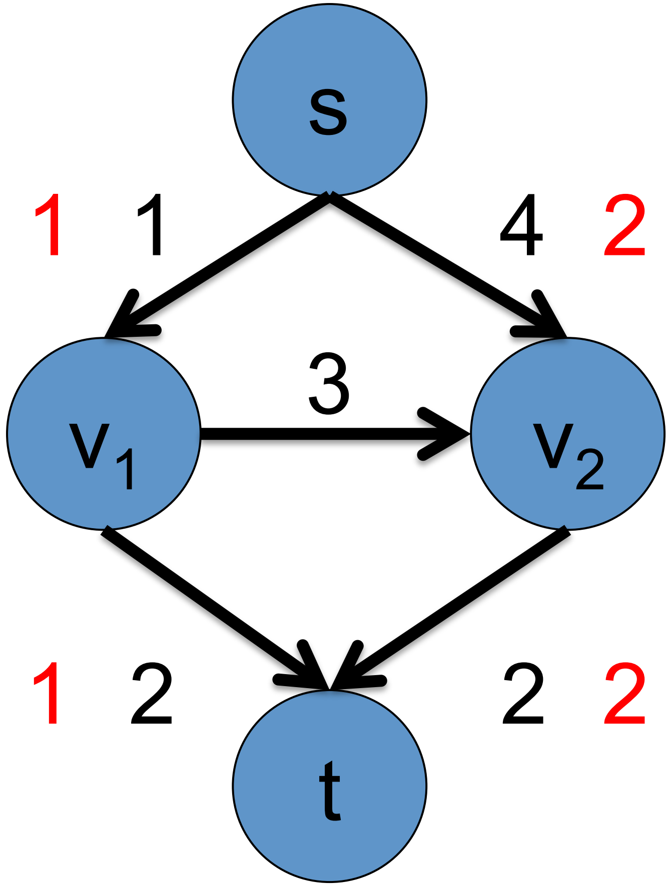


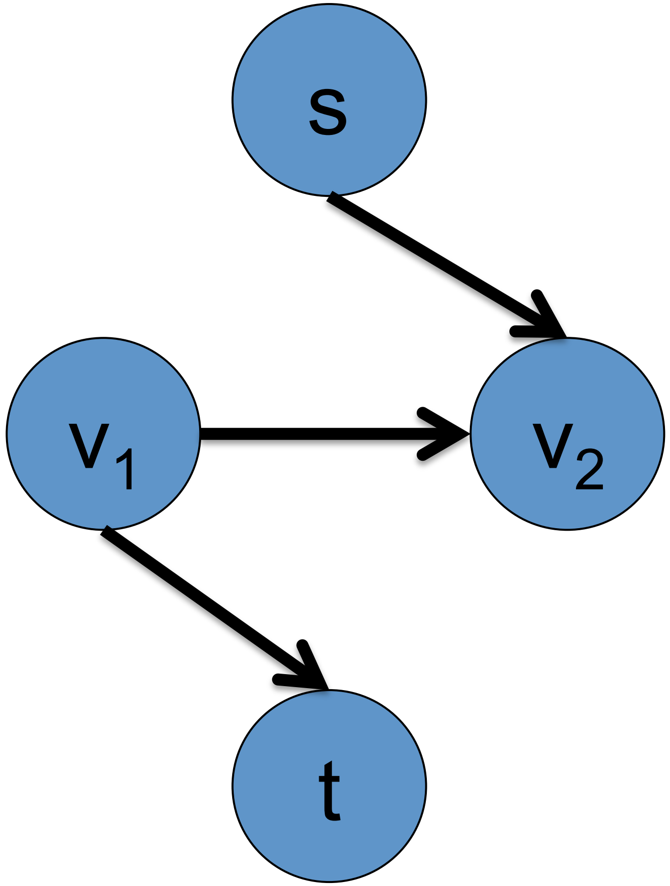
How can I be sure this will always work?

Outline

- Preliminaries
- Maximum Flow
  - Residual Graph
  - **Max-Flow Min-Cut Theorem**
- Algorithms
- Energy minimization with max flow/min cut

Max-Flow Min-Cut

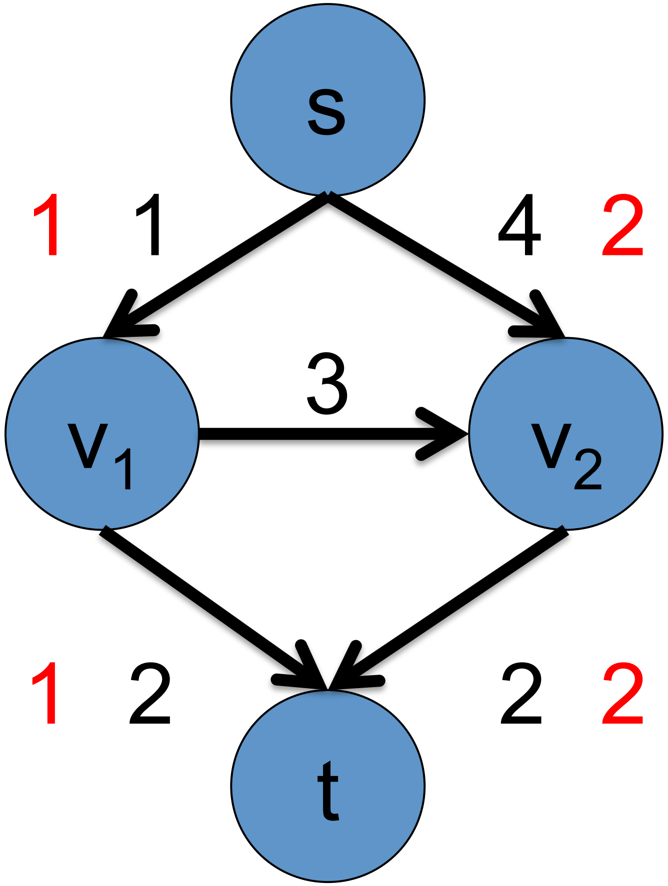


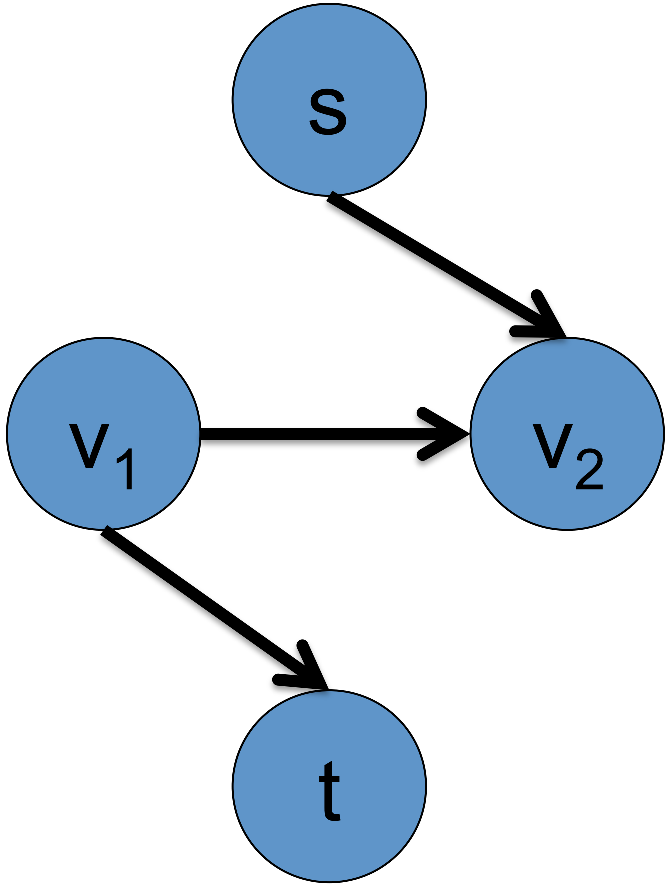


t is not in U.

Let the subset of vertices U be reachable from s.

Max-Flow Min-Cut

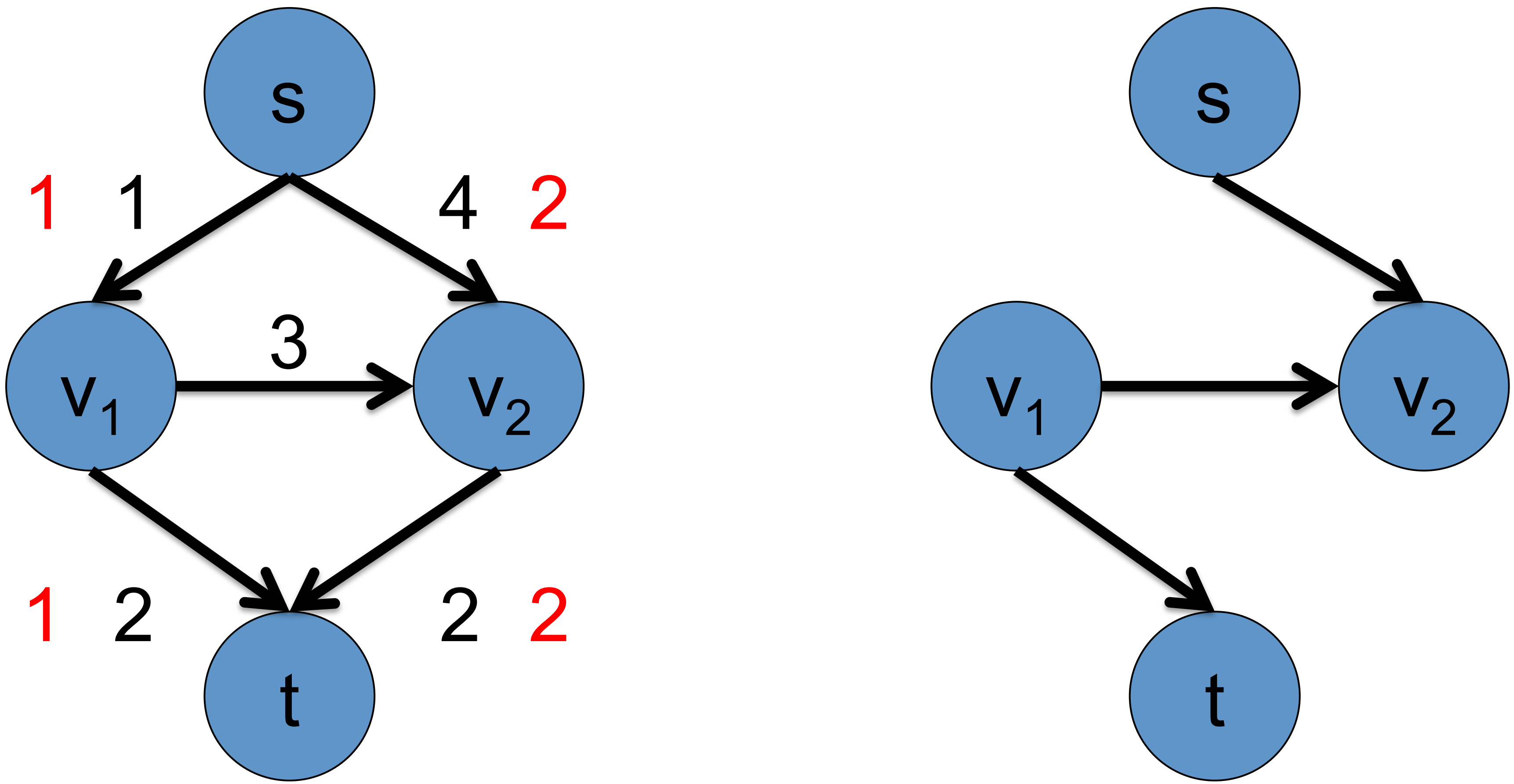




Or else a will be in the residual graph.

For all a ∈ out-arcs(U), flow(a) = c(a).

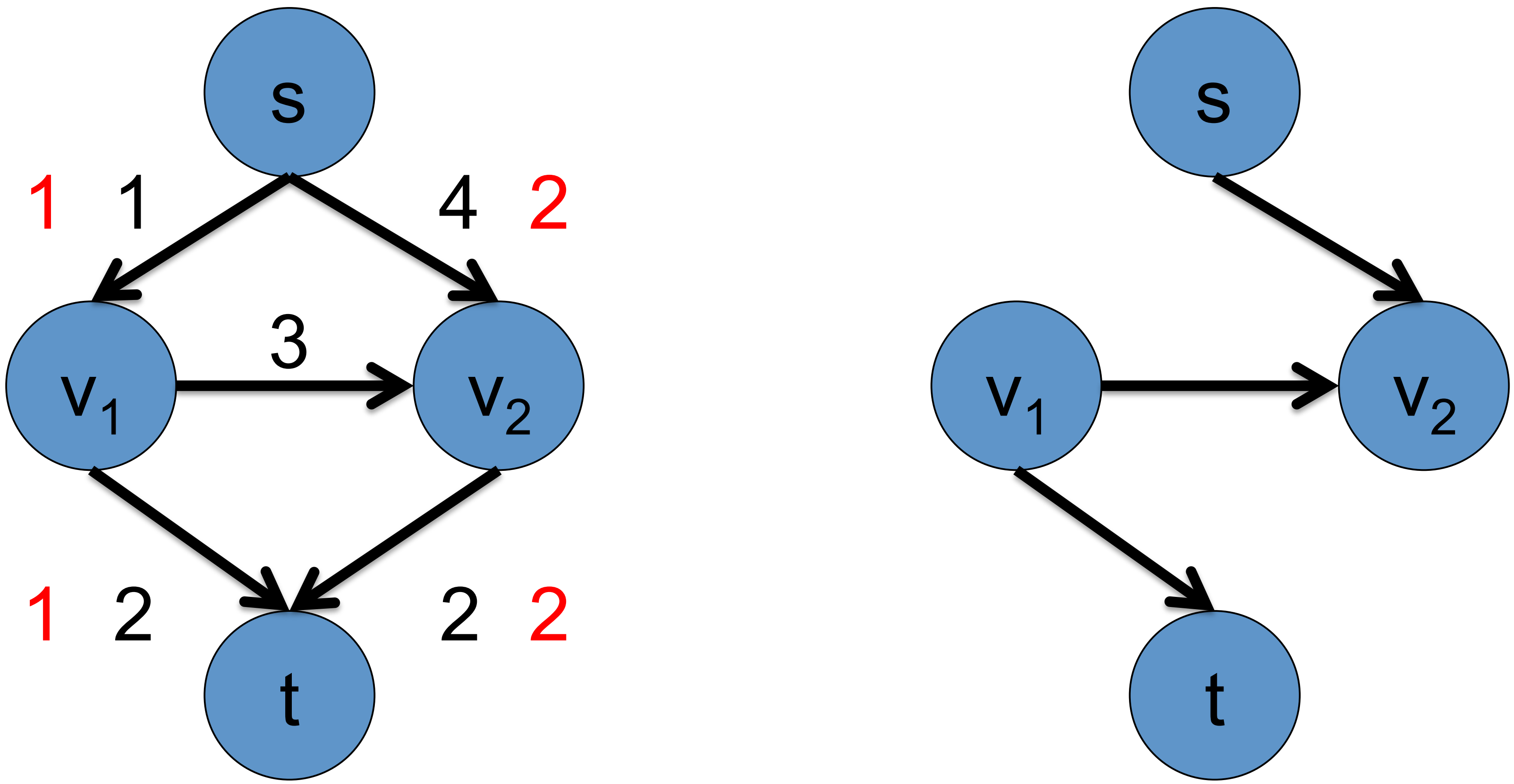
Max-Flow Min-Cut



Or else inverse of a will be in the residual graph.

For all a ∈ in-arcs(U), flow(a) = 0.

Max-Flow Min-Cut



For all a ∈ out-arcs(U), flow(a) = c(a).

For all a ∈ in-arcs(U), flow(a) = 0.

Flows vs. Cuts

Value of flow $= -E_{flow}(s)$

$= -E_{flow}(s) - \Sigma_{v \in U \setminus \{s\}} E_{flow}(v)$

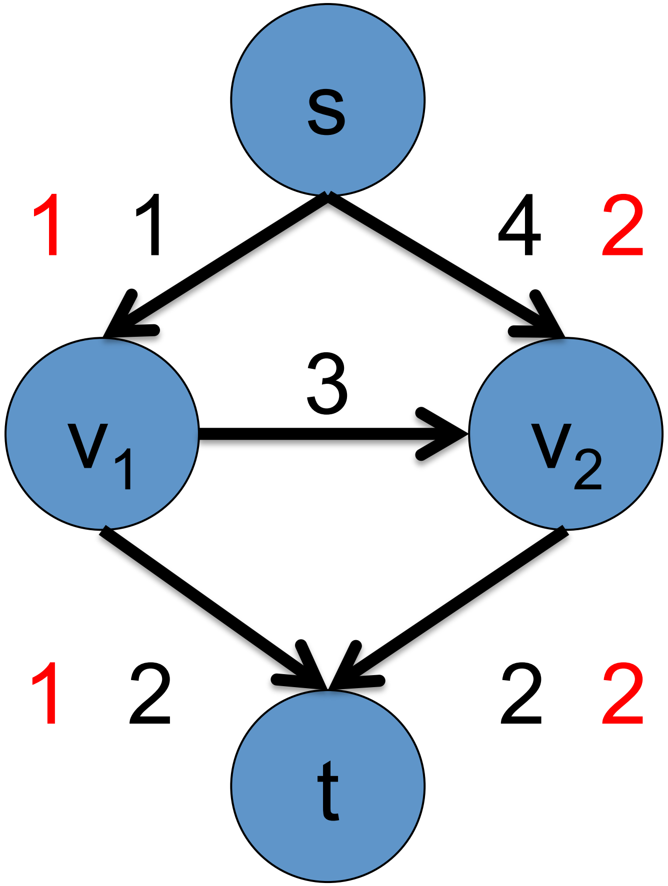
$= -E_{flow}(U)$

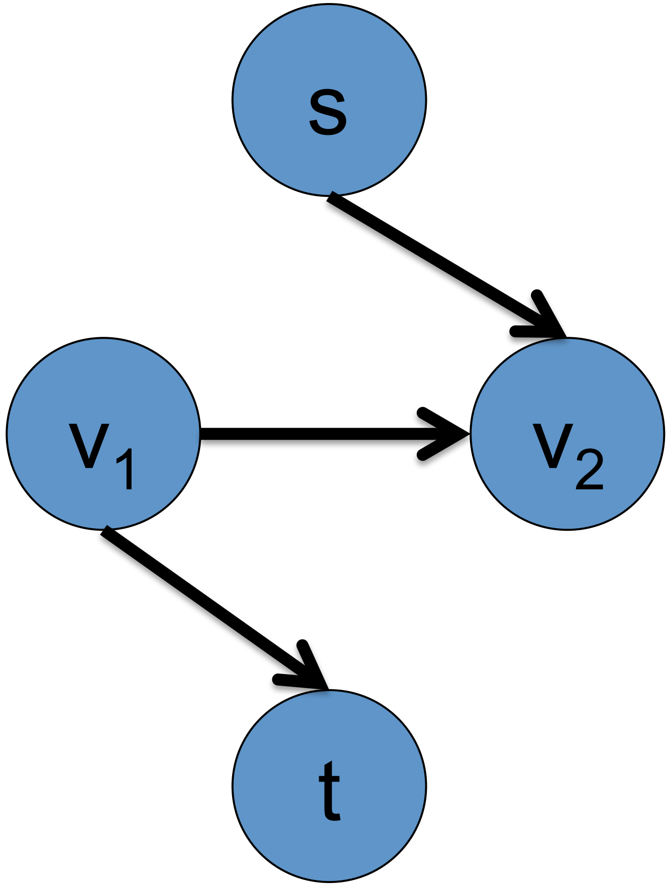
$=$ flow(out-arcs(U))
 - flow(in-arcs(U))

$=$ Capacity of C

flow(a) = c(a), $a \in$ out-arcs(U)   flow(a) = 0, $a \in$ in-arcs(U)

Max-Flow Min-Cut





Minimum Cut

Maximum Flow

Capacity(C) = Value(flow)

Outline

- Preliminaries
- Maximum Flow
- **Algorithms**
  - **Ford-Fulkerson Algorithm**
  - Dinits Algorithm
- Energy minimization with max flow/min cut

Ford-Fulkerson Algorithm

Start with flow = 0 for all arcs.

Find an s-t path in the residual graph.

Pass maximum allowable flow.

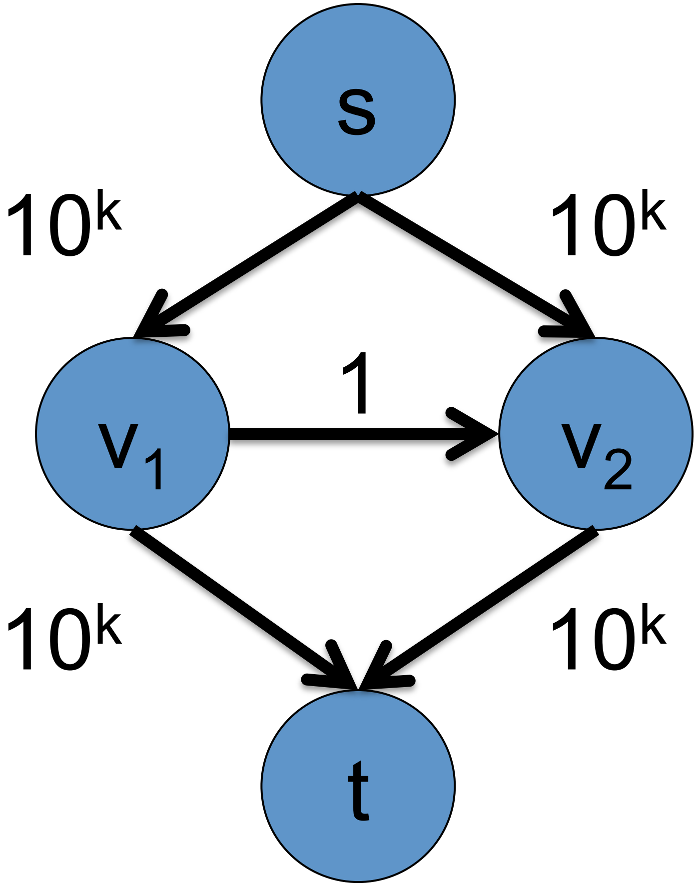
Subtract from inverse arcs.

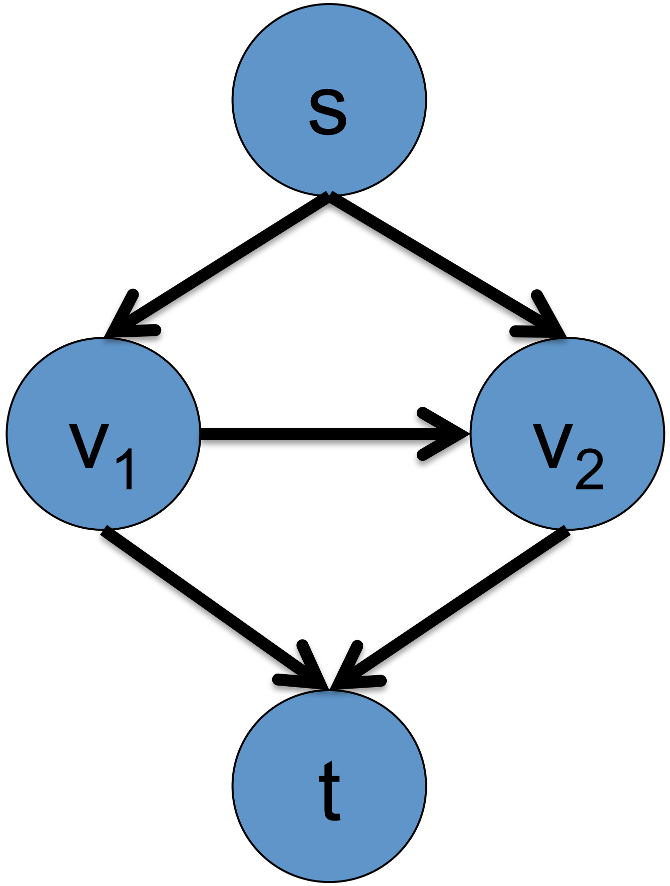
Add to forward arcs.

REPEAT

Until s and t are disjoint in the residual graph.

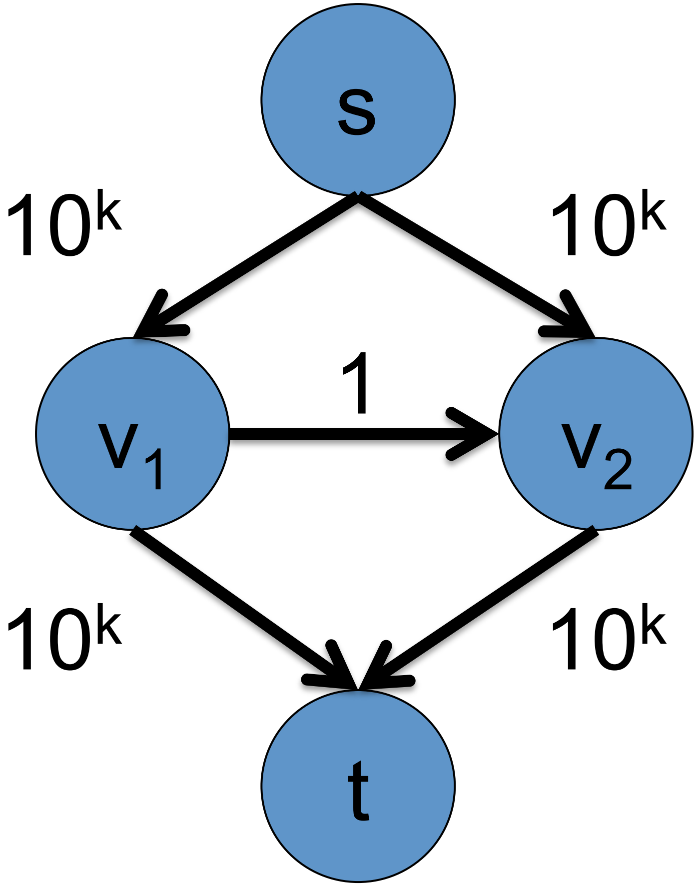
Ford-Fulkerson Algorithm

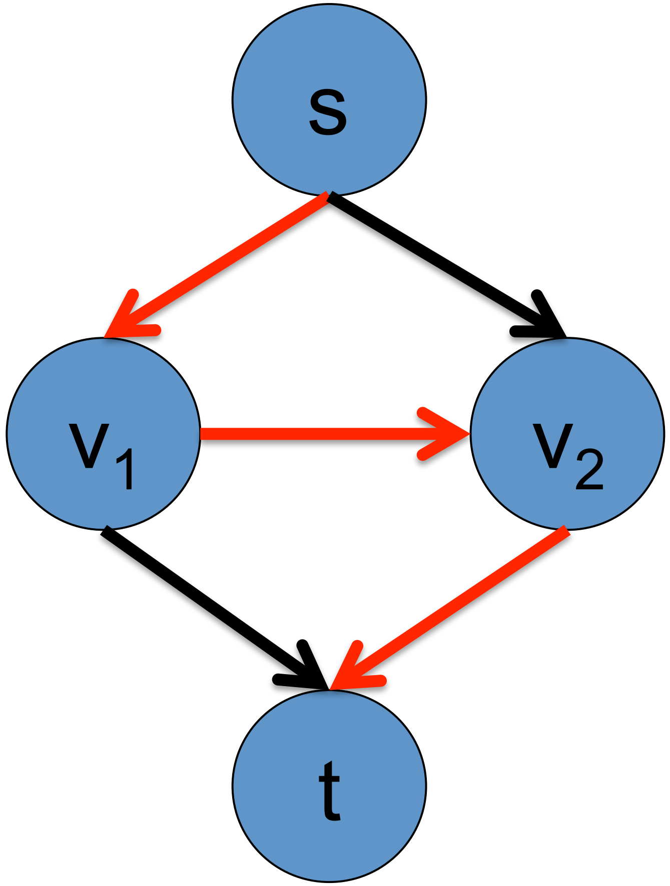




Start with zero flow

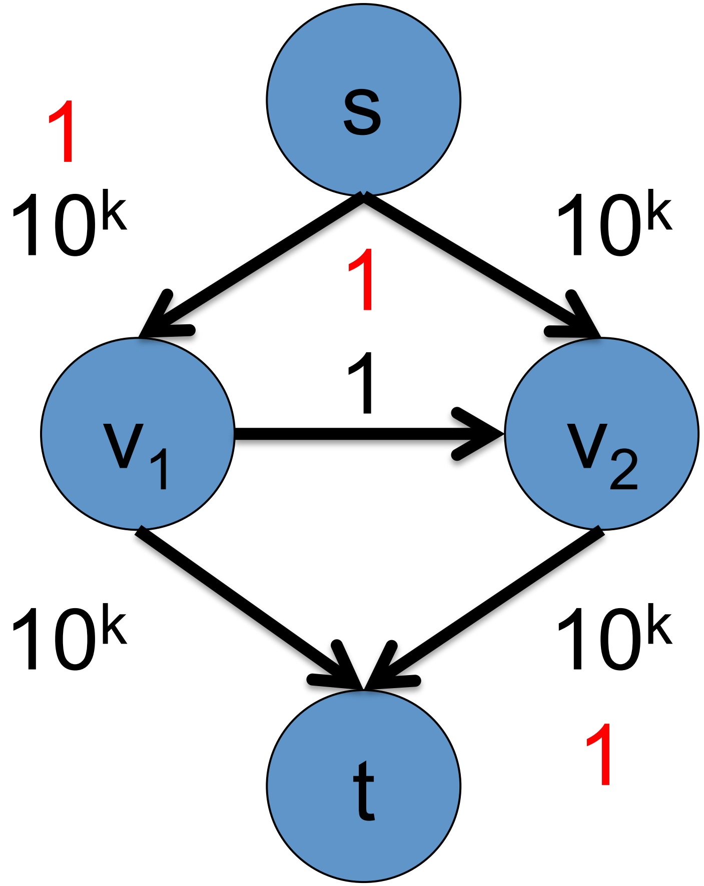
Ford-Fulkerson Algorithm

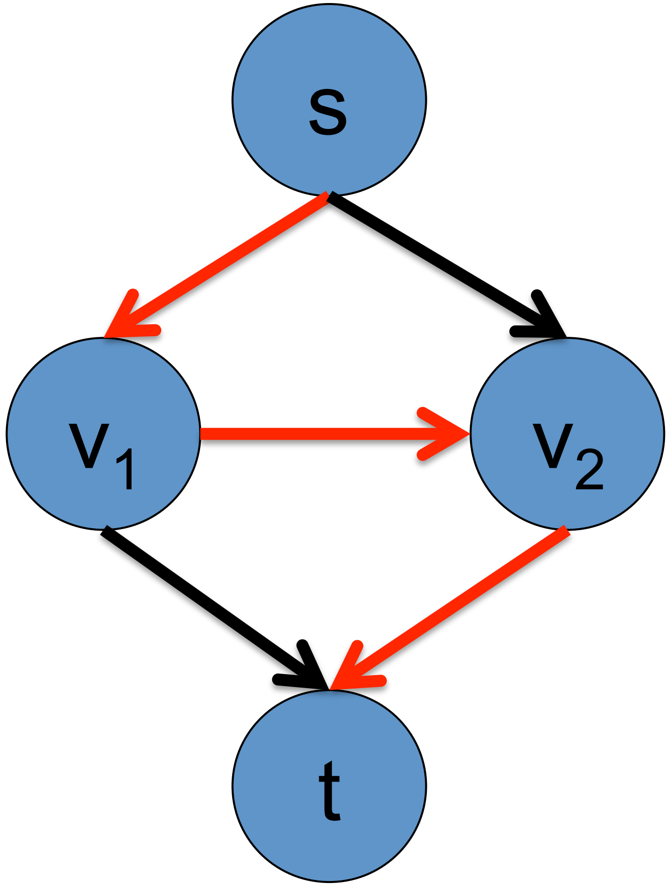




Find an s-t path in the residual graph.

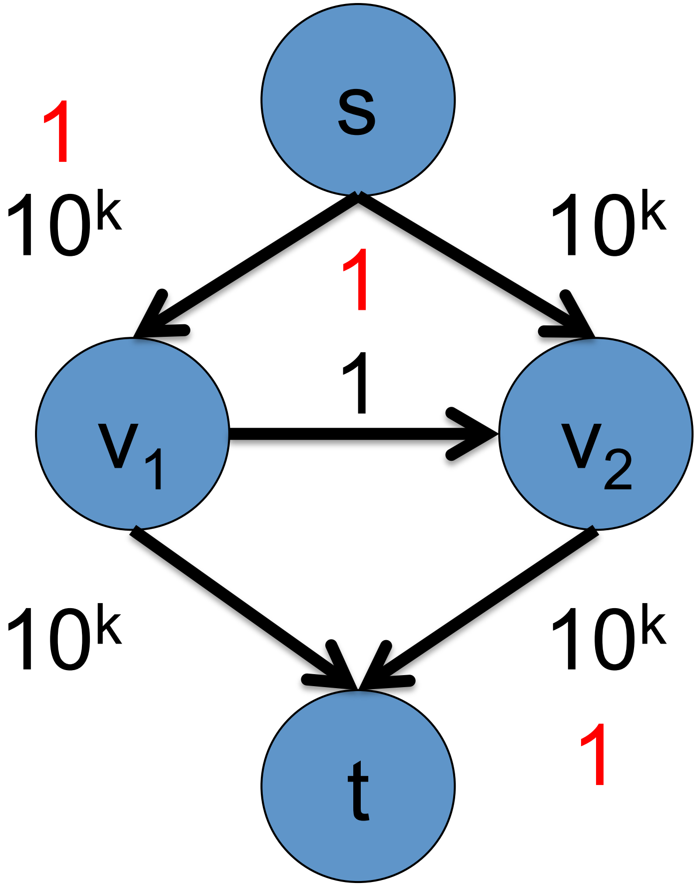
Ford-Fulkerson Algorithm

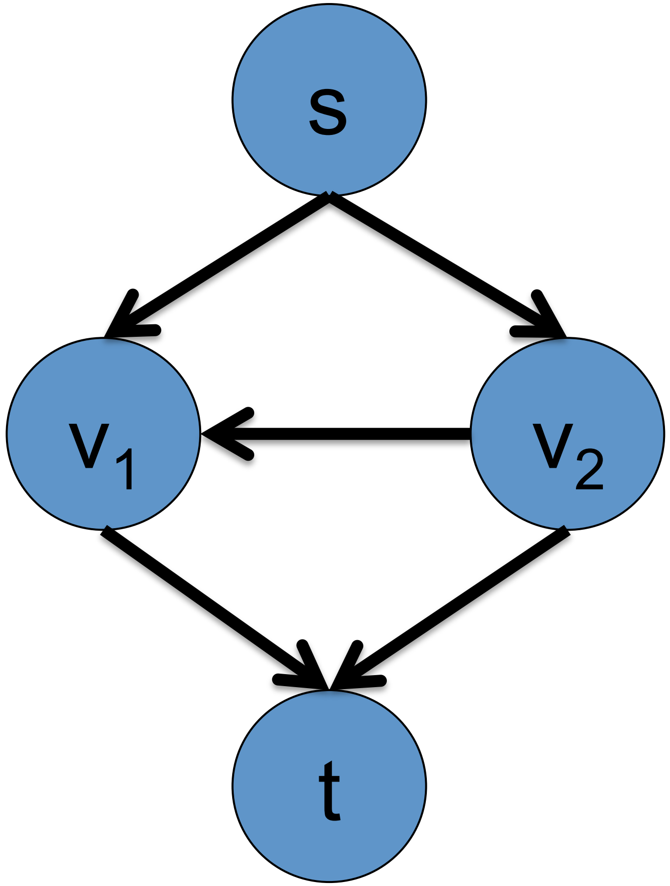




Pass the maximum allowable flow.

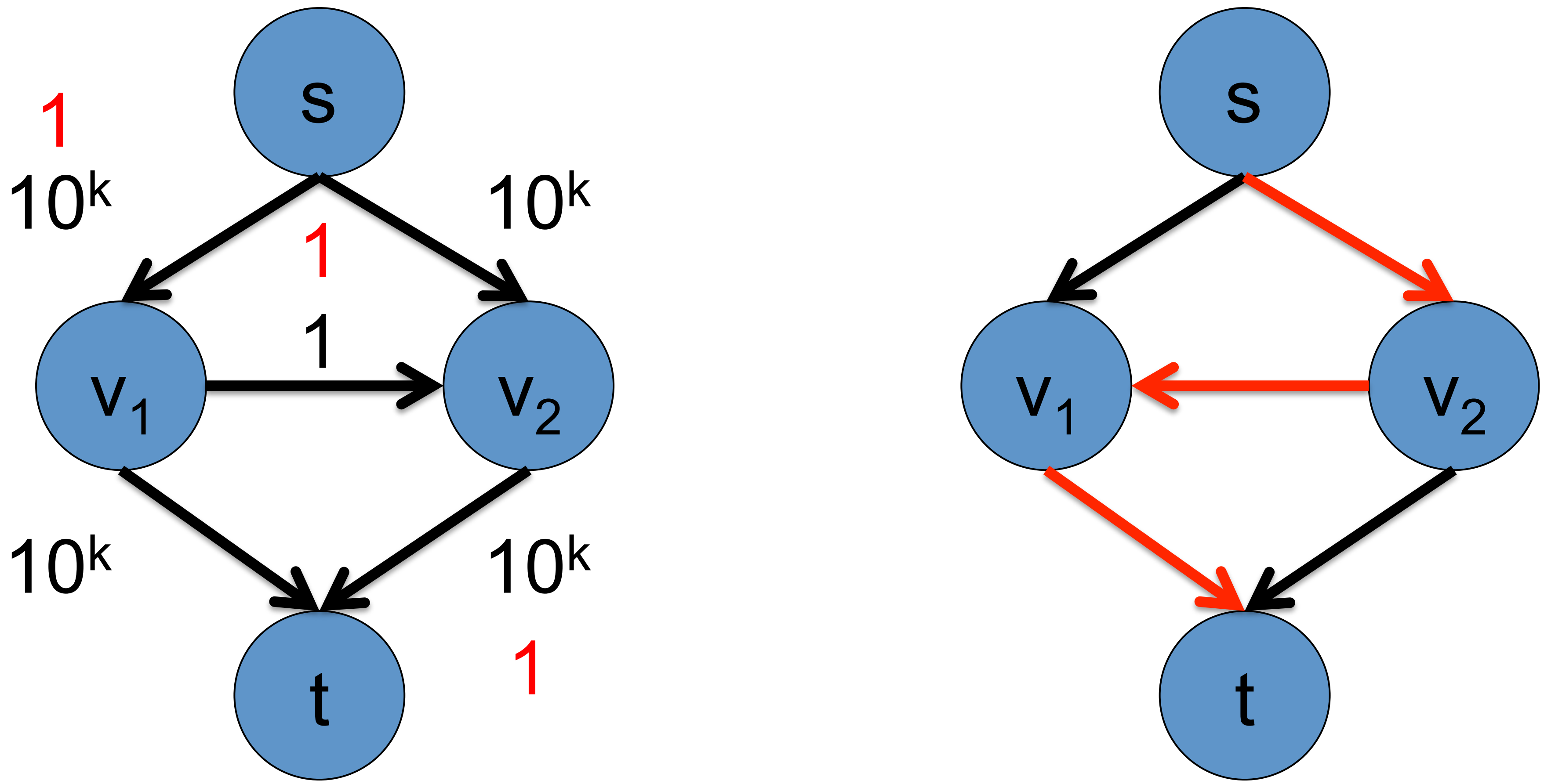
Ford-Fulkerson Algorithm





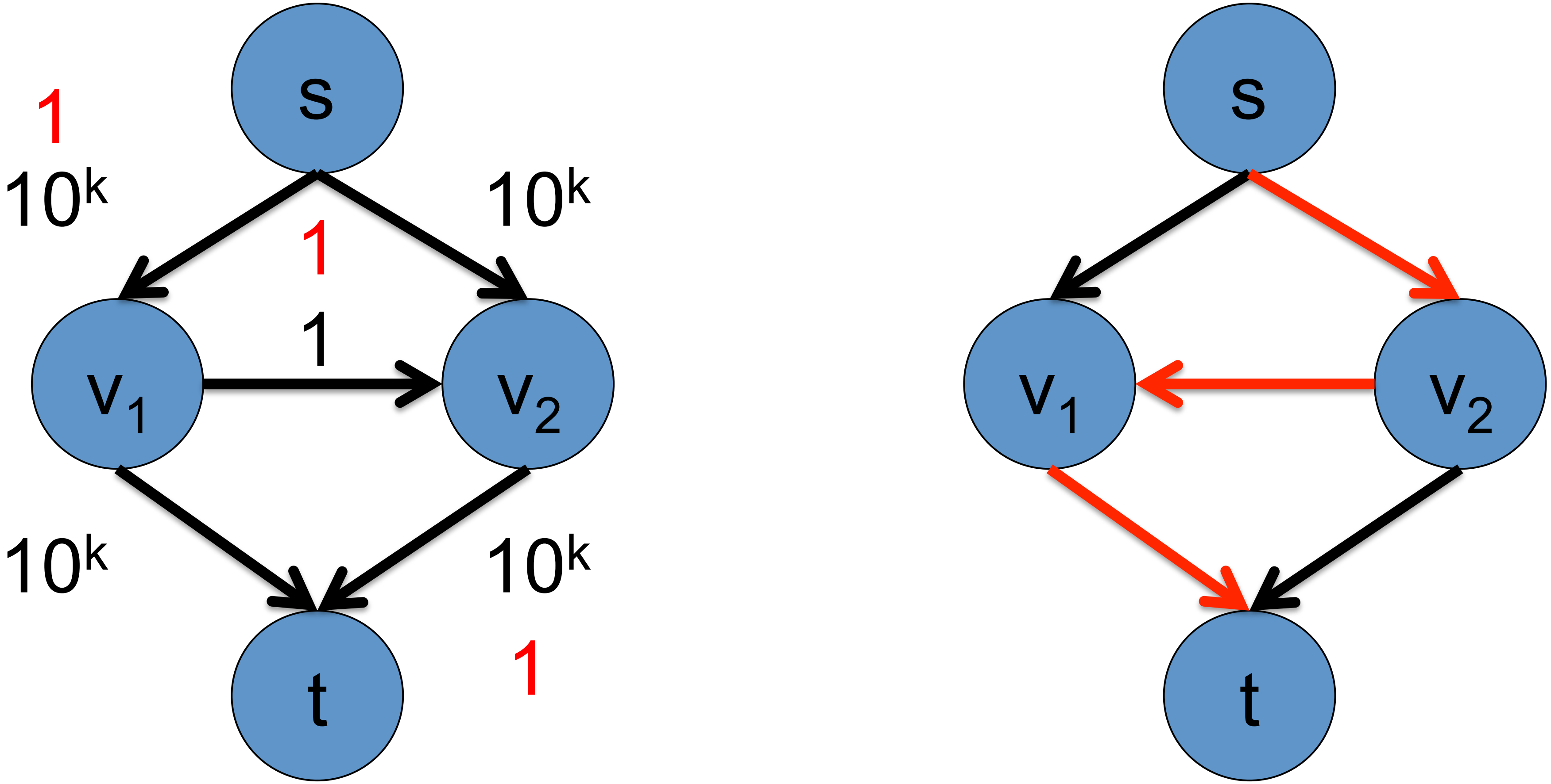
Update the residual graph.

Ford-Fulkerson Algorithm



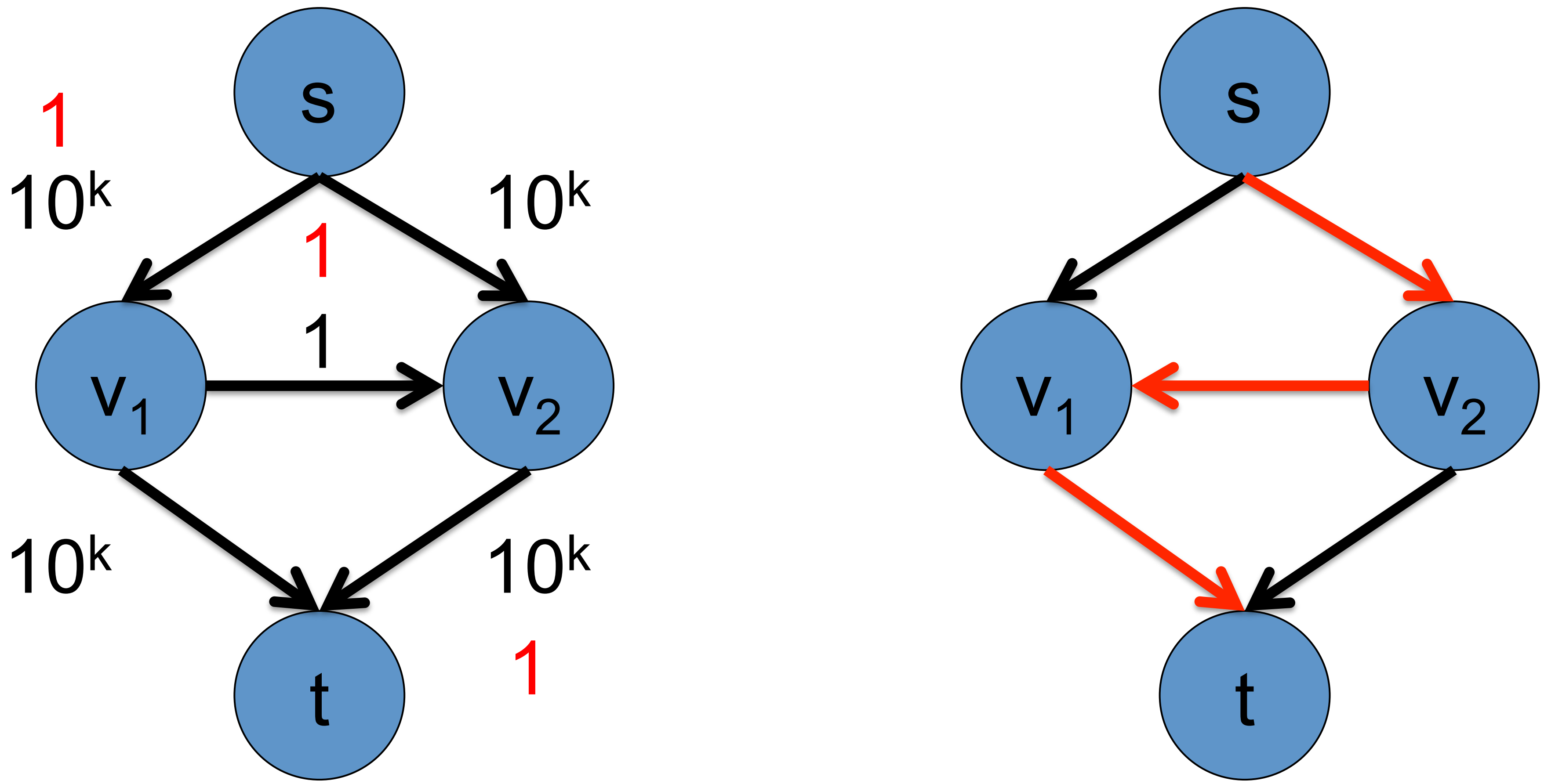
Find an s-t path in the residual graph.

Ford-Fulkerson Algorithm



Complexity is exponential in k.

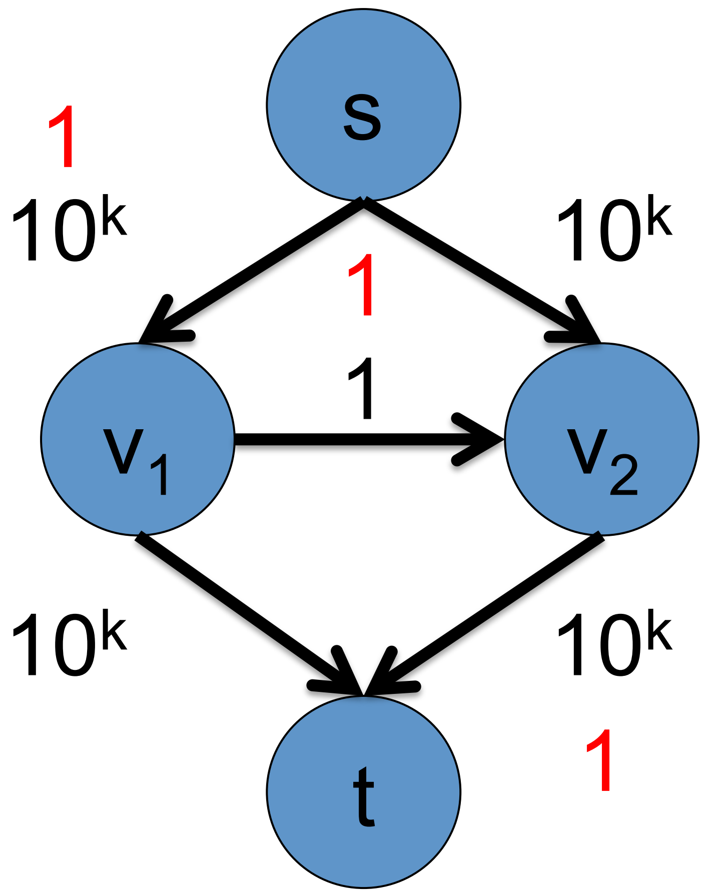
Ford-Fulkerson Algorithm

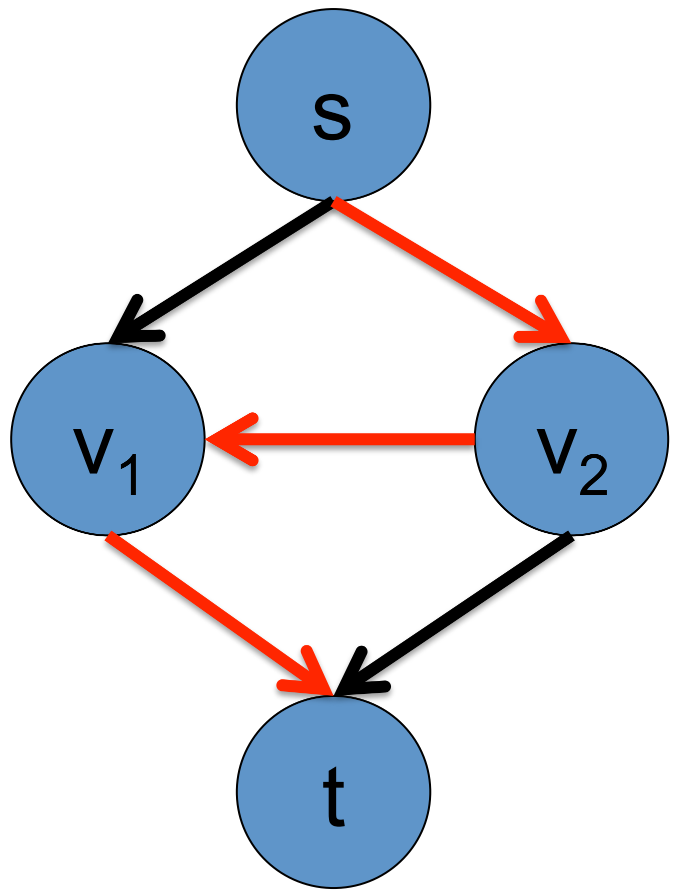


For examples, see Uri Zwick, 1993

Irrational arc lengths can lead to infinite iterations.

Ford-Fulkerson Algorithm





Choose wisely.

There are good paths and bad paths.

Outline

- Preliminaries
- Maximum Flow
- **Algorithms**
  - Ford-Fulkerson Algorithm
  - **Dinits Algorithm**
- Energy minimization with max flow/min cut

Dinits Algorithm

Start with flow = 0 for all arcs.

Find the **minimum s-t path**
in the residual graph.

Pass maximum allowable flow.

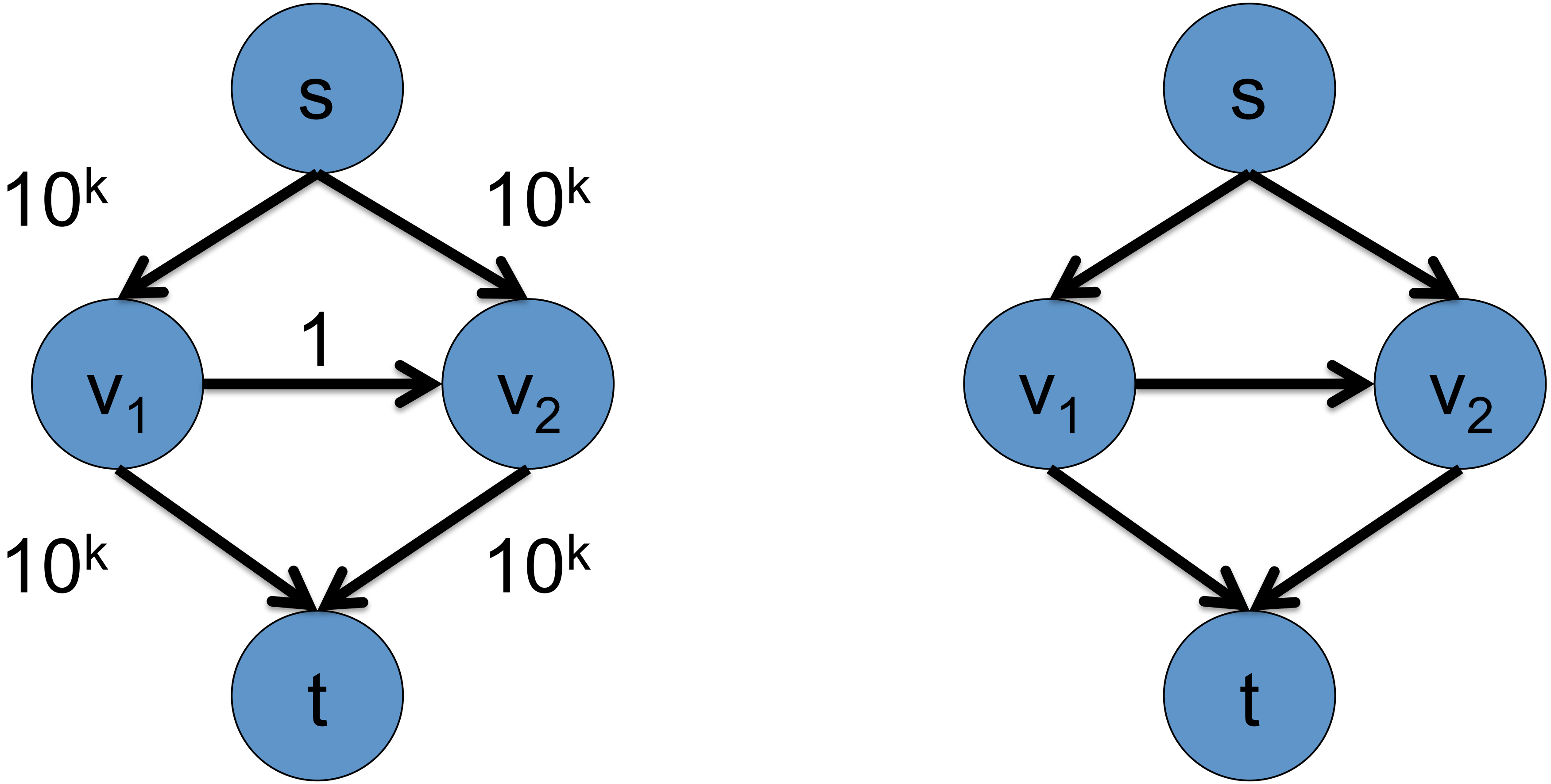
Subtract from inverse arcs.

Add to forward arcs.

REPEAT

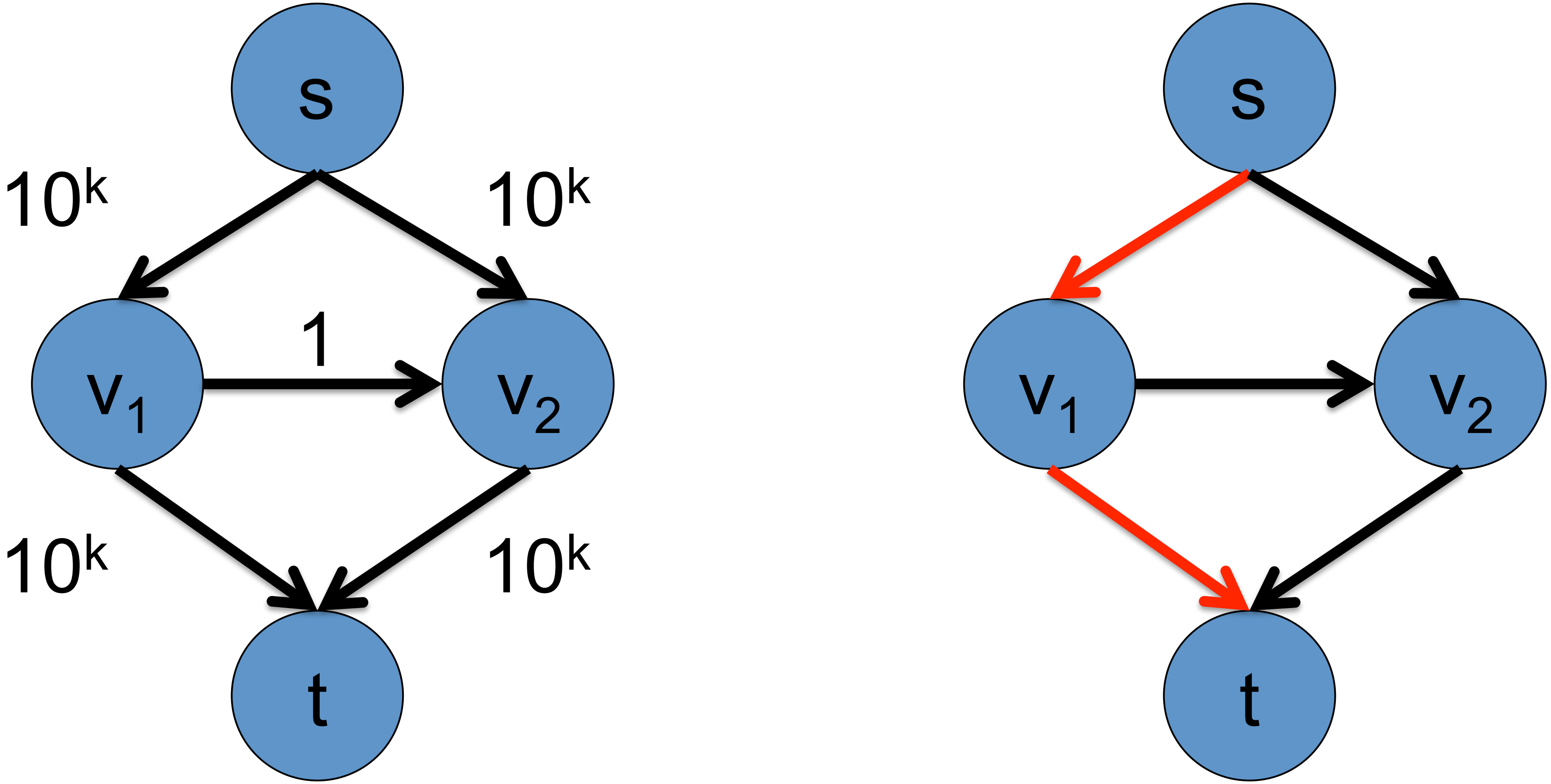
Until s and t are disjoint in the residual graph.

Dinits Algorithm



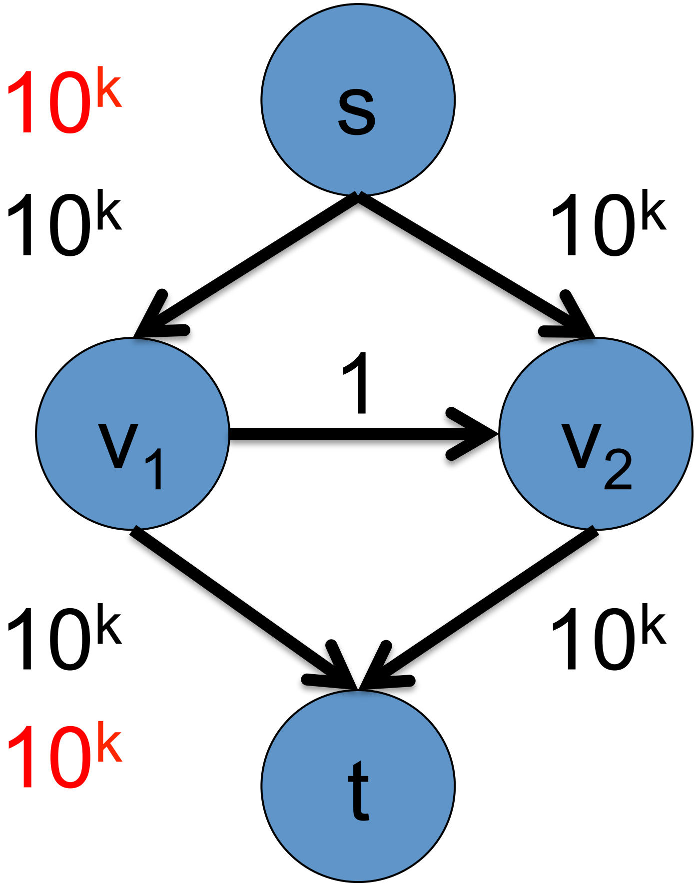
Start with zero flow

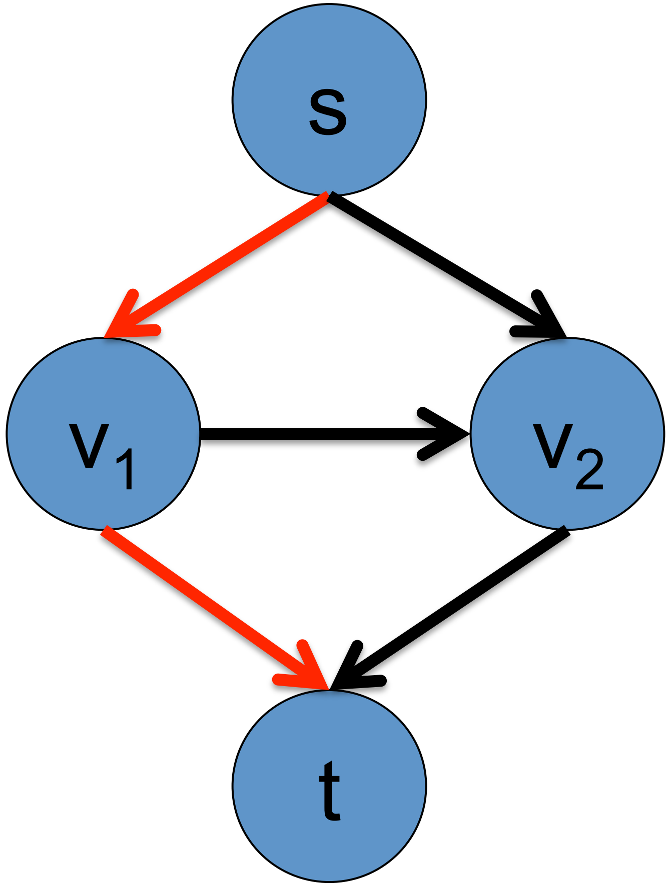
Dinits Algorithm



Find the minimum s-t path in the residual graph.

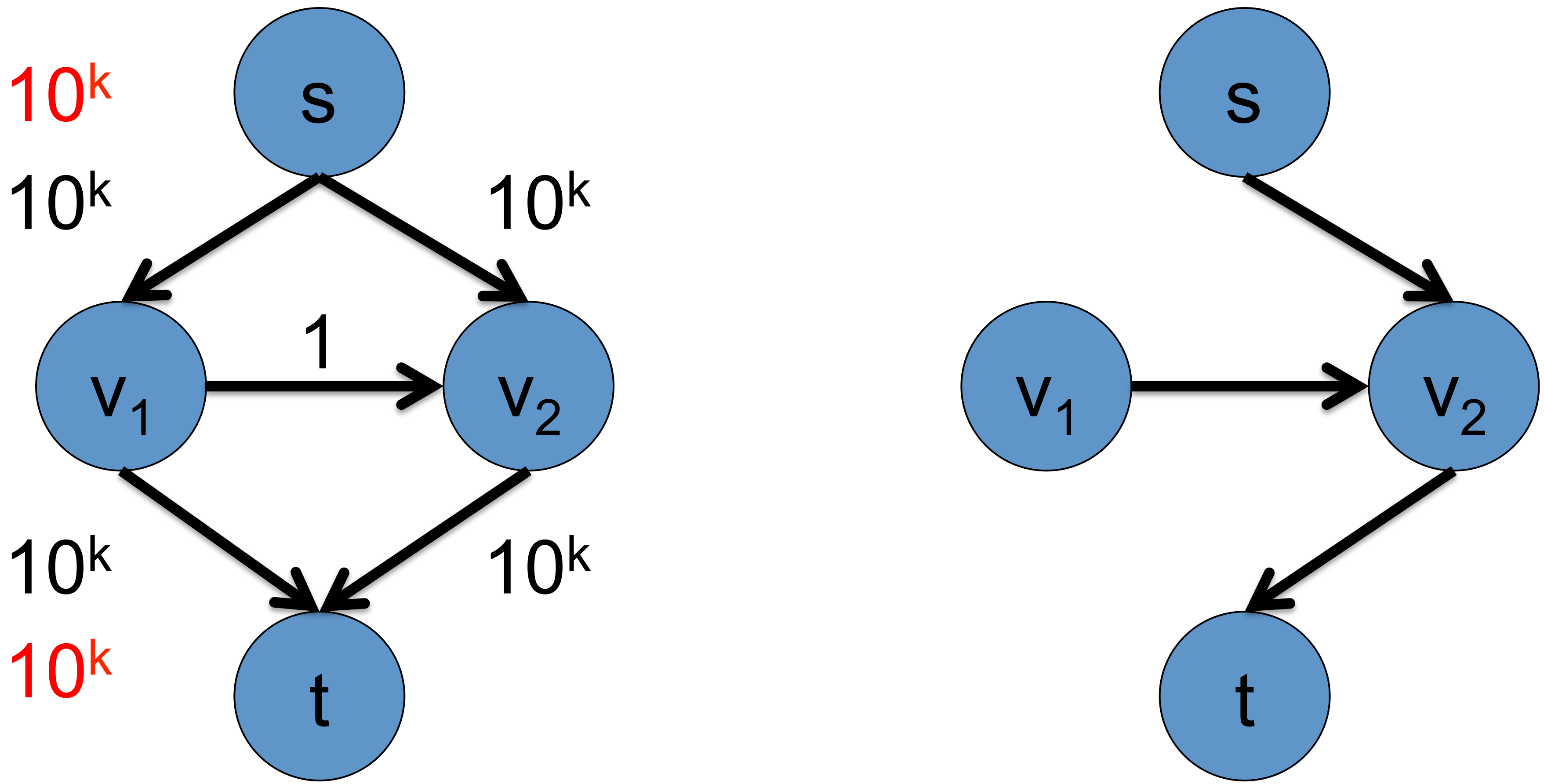
Dinits Algorithm





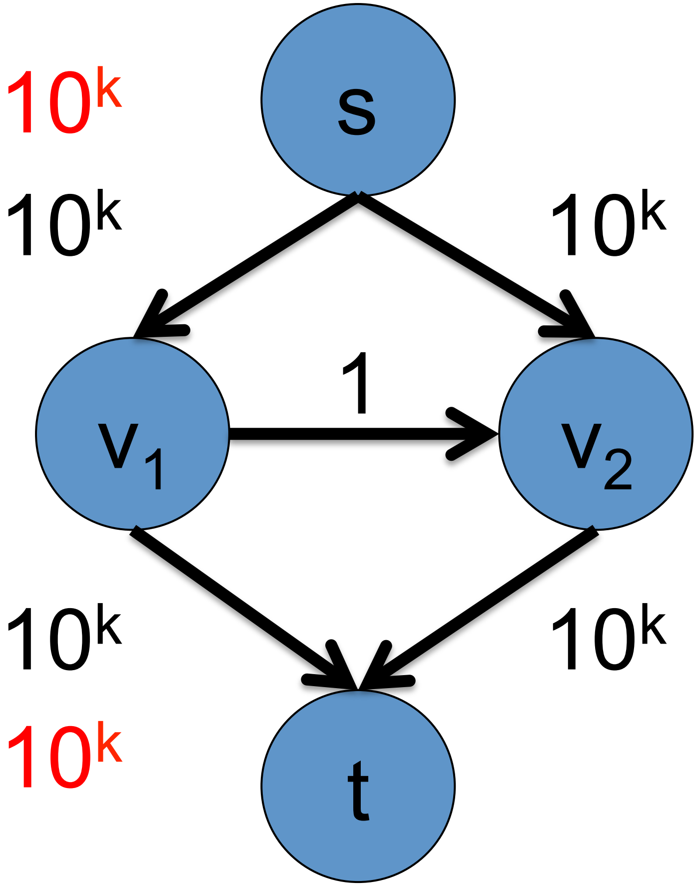
Pass the maximum allowable flow.

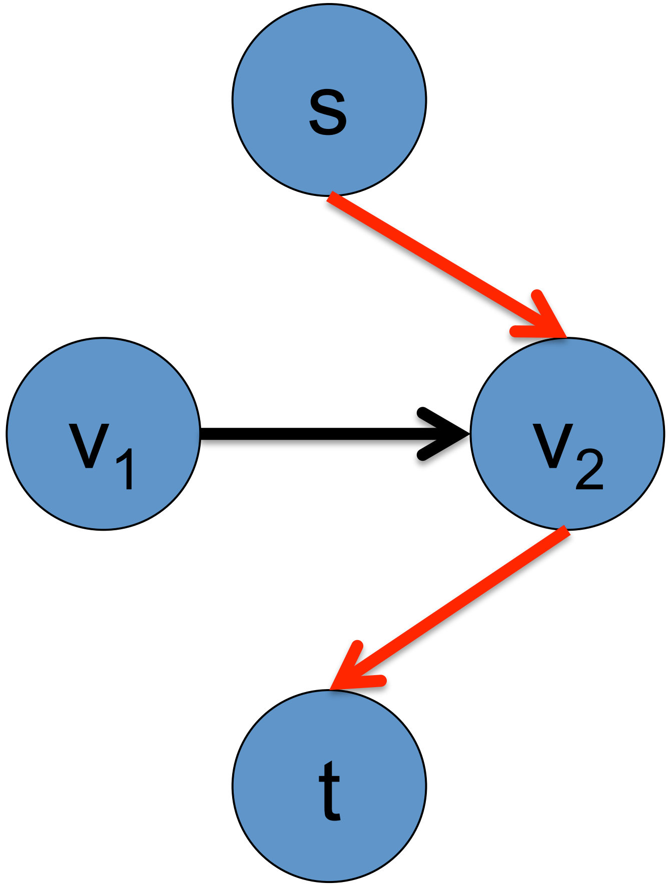
Dinits Algorithm



Update the residual graph.

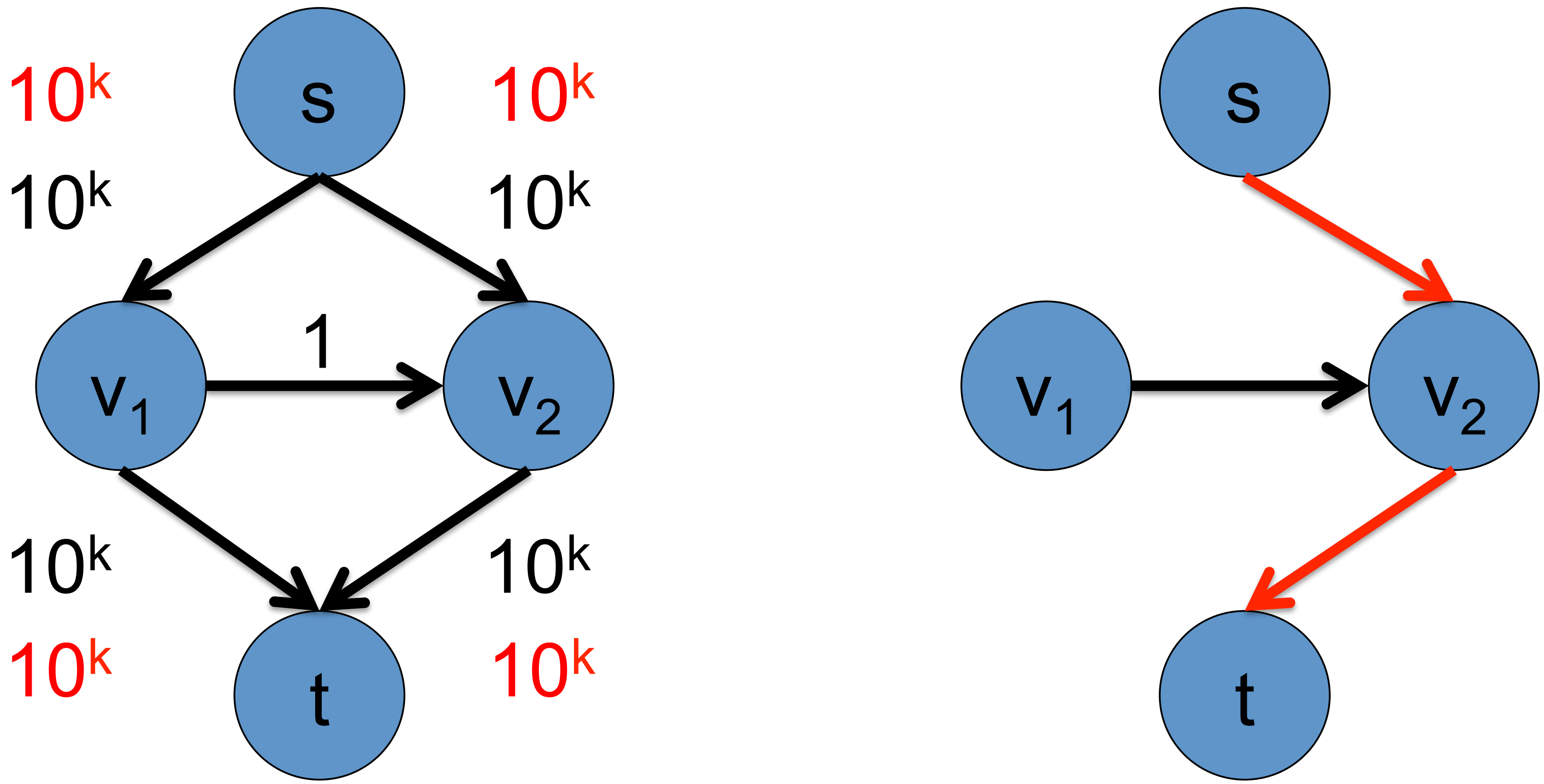
Dinits Algorithm





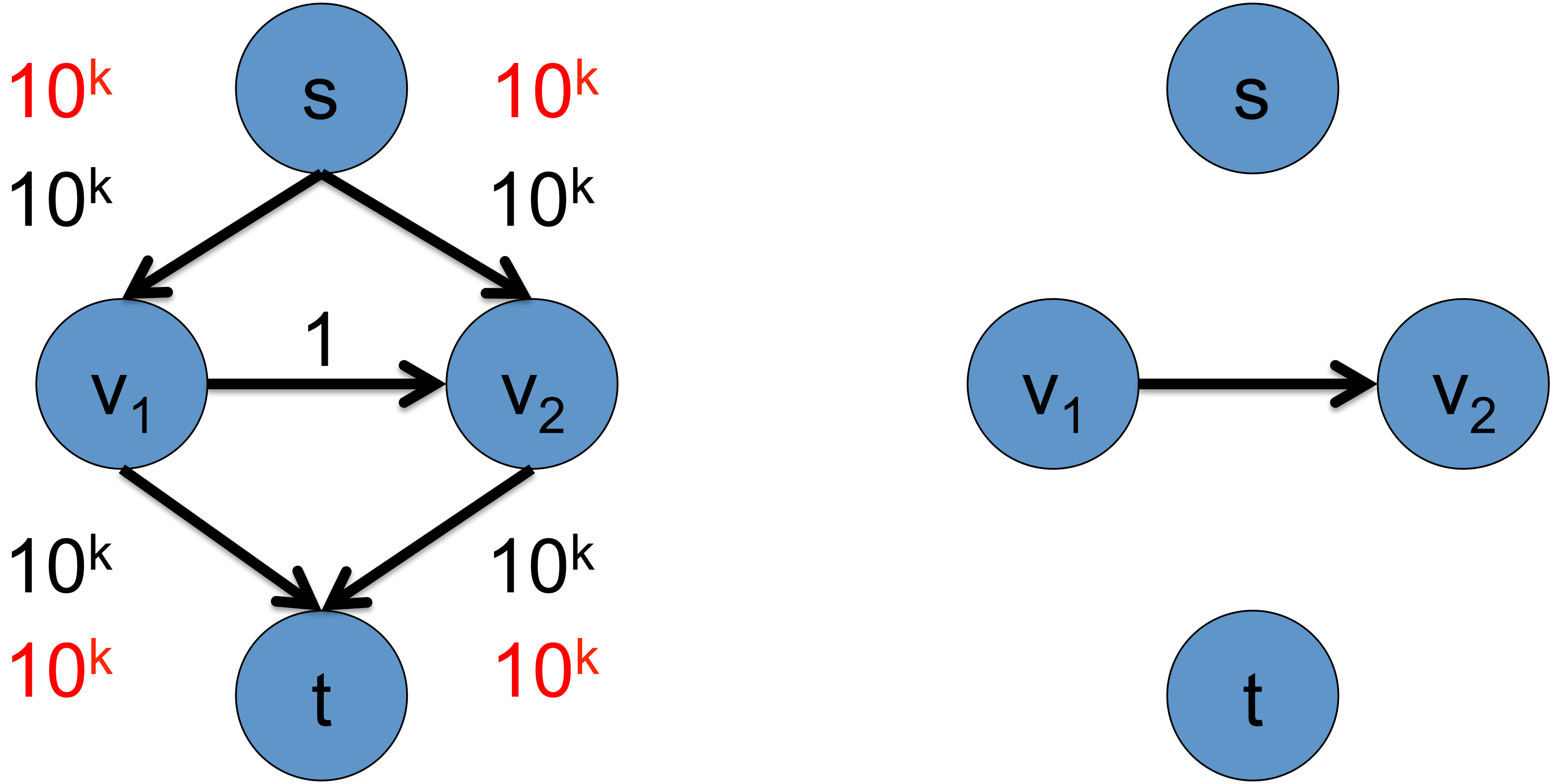
Find the minimum s-t path in the residual graph.

Dinits Algorithm



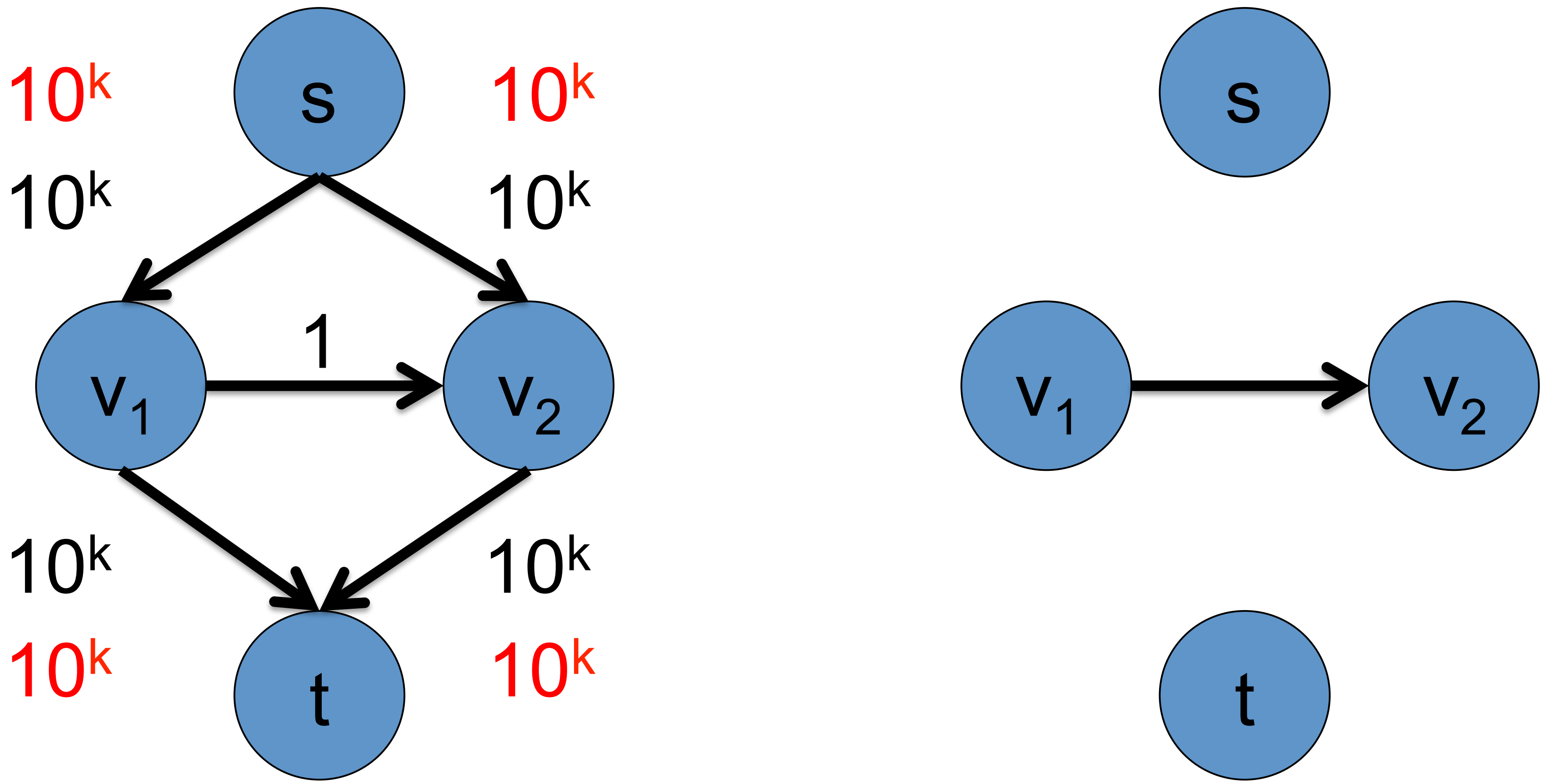
Pass the maximum allowable flow.

Dinits Algorithm



Update the residual graph.

Dinits Algorithm



No more s-t paths. Stop.

Solvers for the Minimum-Cut Problem

Augmenting Path and Push-Relabel

year	discoverer(s)	bound
1951	Dantzig	$O(n^2mU)$
1955	Ford & Fulkerson	$O(m^2U)$
1970	Dinitz	$O(n^2m)$
1972	Edmonds & Karp	$O(m^2 \log U)$
1973	Dinitz	$O(nm \log U)$
1974	Karzanov	$O(n^3)$
1977	Cherkassky	$O(n^2m^{1/2})$
1980	Galil & Naamad	$O(nm \log^2 n)$
1983	Sleator & Tarjan	$O(nm \log n)$
1986	Goldberg & Tarjan	$O(nm \log(n^2/m))$
1987	Ahuja & Orlin	$O(nm + n^2 \log U)$
1987	Ahuja et al.	$O(nm \log(n\sqrt{\log U}/m))$
1989	Cheriyan & Hagerup	$E(nm + n^2 \log^2 n)$
1990	Cheriyan et al.	$O(n^3/\log n)$
1990	Alon	$O(nm + n^{8/3} \log n)$
1992	King et al.	$O(nm + n^{2+\epsilon})$
1993	Phillips & Westbrook	$O(nm(\log_{m/n} n + \log^{2+\epsilon} n))$
1994	King et al.	$O(nm \log_{m/(n \log n)} n)$
1997	Goldberg & Rao	$O(m^{3/2} \log(n^2/m) \log U)$ $O(n^{2/3}m \log(n^2/m) \log U)$

n: #nodes

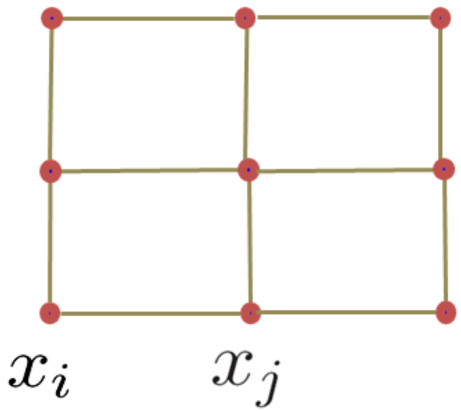
m: #arcs

U: maximum arc length

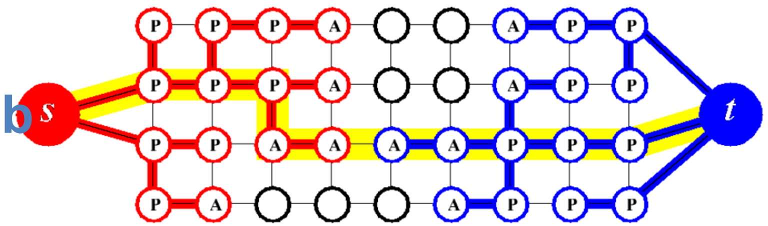
[Slide credit: Andrew Goldberg]

Max-Flow in Computer Vision

- **Specialized algorithms for vision problems**
  - Grid graphs
  - Low connectivity (m ~ O(n))

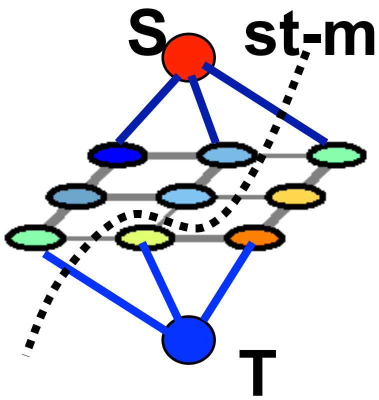


- **Dual search tree augmenting path algorithm**
  [Boykov and Kolmogorov PAMI 2004]
  - Finds approximate shortest augmenting paths efficiently
  - High worst-case time complexity
  - Empirically outperforms other algorithms on vision problems
  - Efficient code available on the web



http://www.adastral.ucl.ac.uk/~vladkolm/software.html

St-mincut and Energy Minimization



⟺

Minimizing a Qudratic Pseudoboolean function E(x)

Functions of boolean variables

Pseudoboolean?

$$E: \{0,1\}^n \rightarrow R$$

$$E(y) = \sum_i c_i y_i + \sum_{i,j} c_{ij} y_i(1-y_j)$$

$c_{ij} \geq 0$

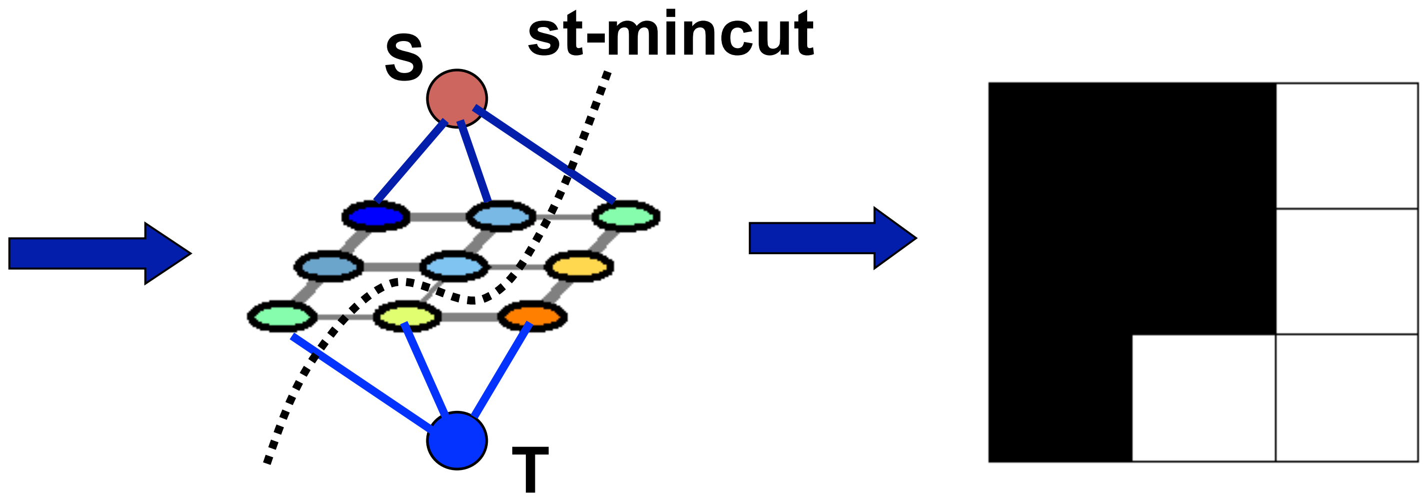
Polynomial time st-mincut algorithms require non-negative edge weights

So how does this work?

Construct a graph such that:

1. Any st-cut corresponds to an assignment of x
2. The cost of the cut is equal to the energy of x : E(x)

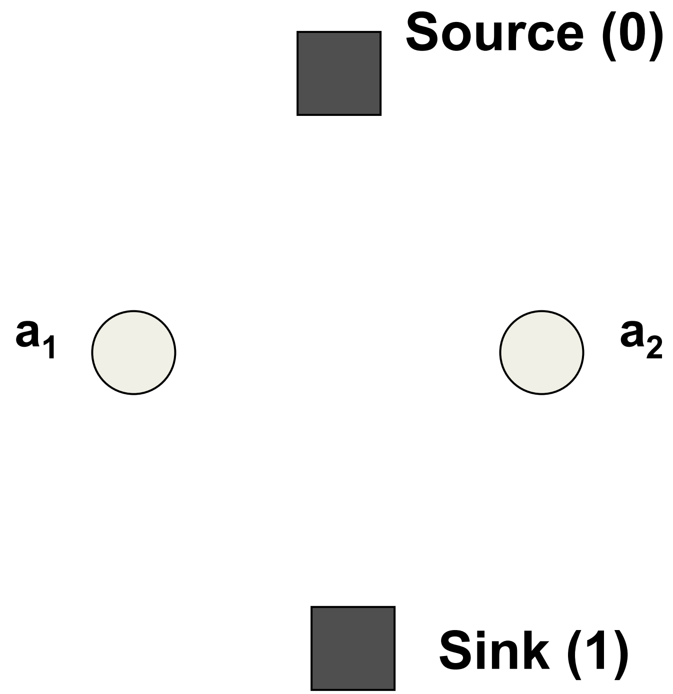
E(y)



Solution

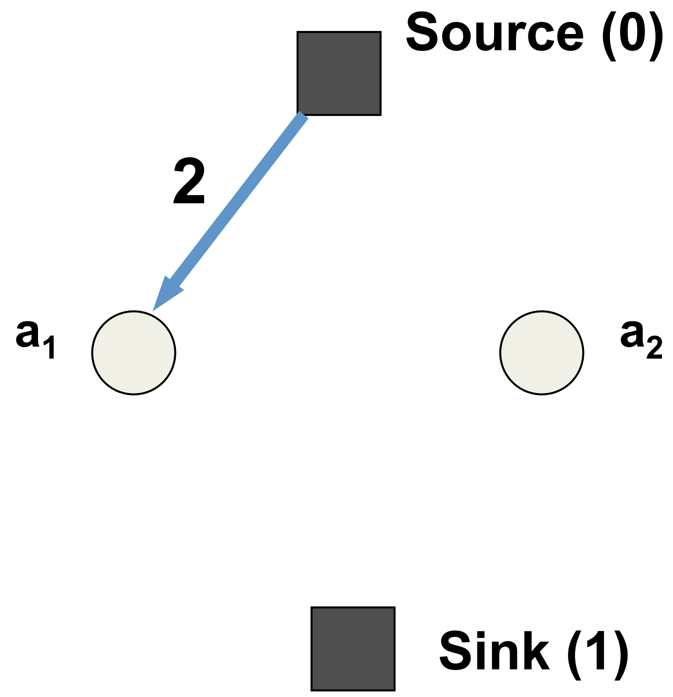
Graph Construction

$E(a_1,a_2)$



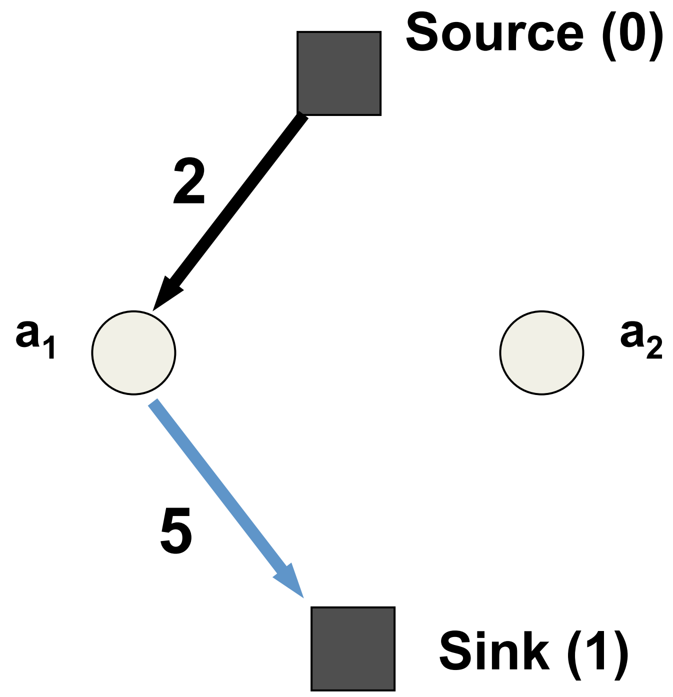
Graph Construction

$E(a_1,a_2) = 2a_1$



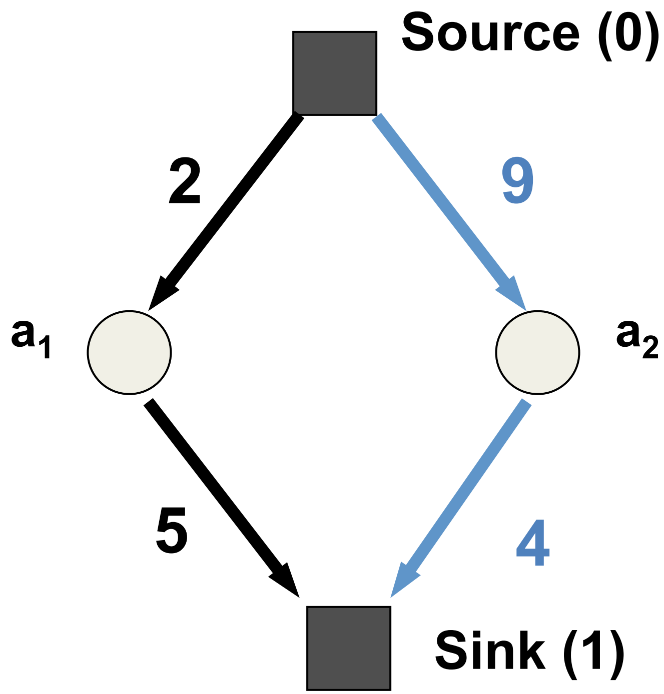
Graph Construction

$E(a_1,a_2) = 2a_1 + 5\bar{a}_1$



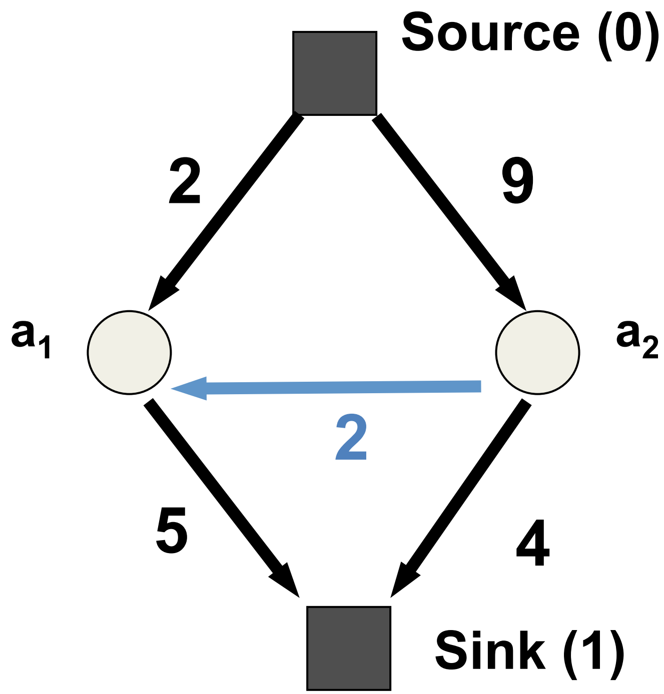
Graph Construction

$E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2$



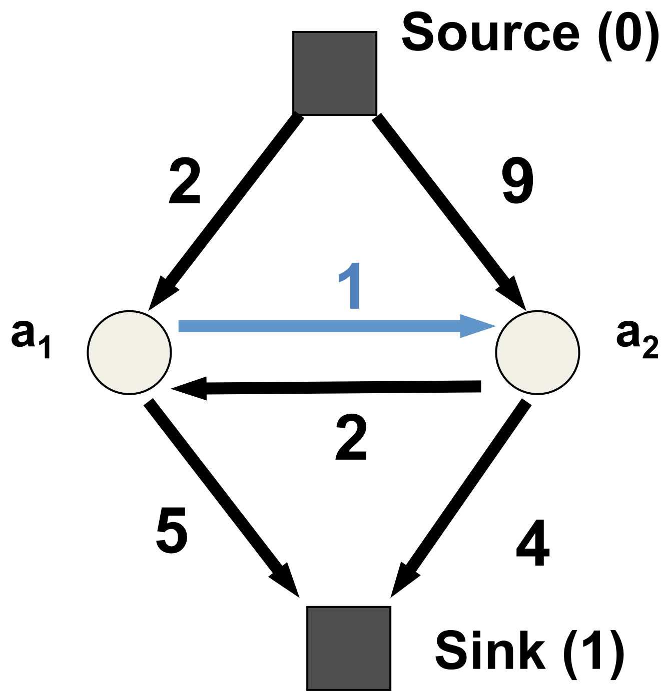
Graph Construction

$$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2$$



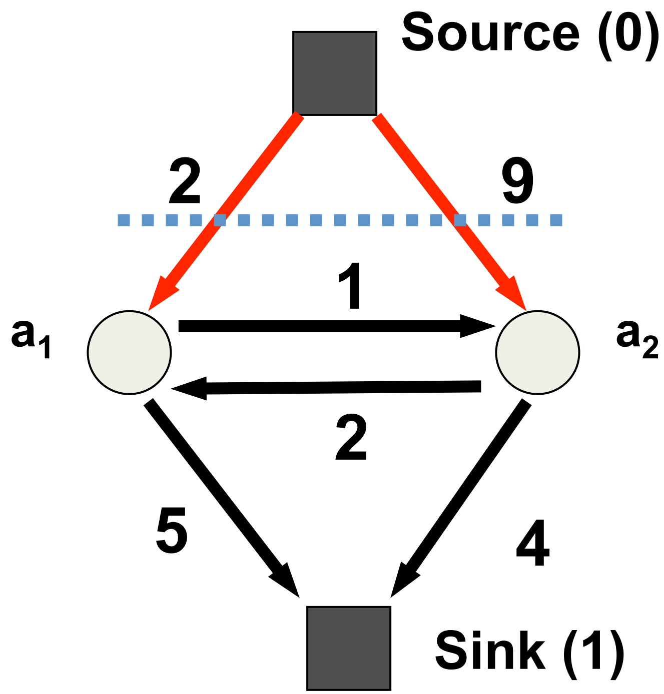
Graph Construction

$$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$$



Graph Construction

$$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$$



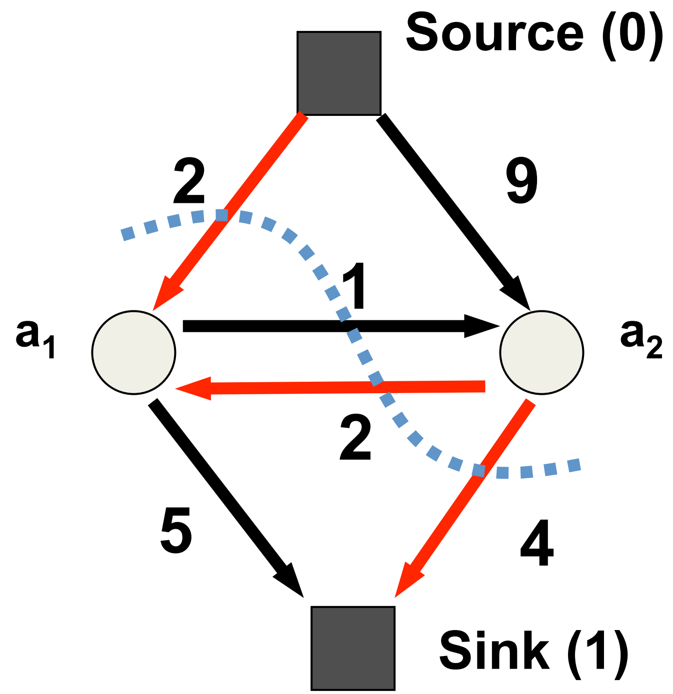
Cost of cut = 11

$a_1 = 1 \quad a_2 = 1$

$E(1,1) = 11$

Graph Construction

$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$



st-mincut cost = 8

$a_1 = 1 \quad a_2 = 0$

$E(1,0) = 8$

Energy Function Reparameterization

Two functions E_1 and E_2 are reparameterizations if

$$E_1(\mathbf{x}) = E_2(\mathbf{x}) \quad \text{for all } \mathbf{x}$$

For instance:

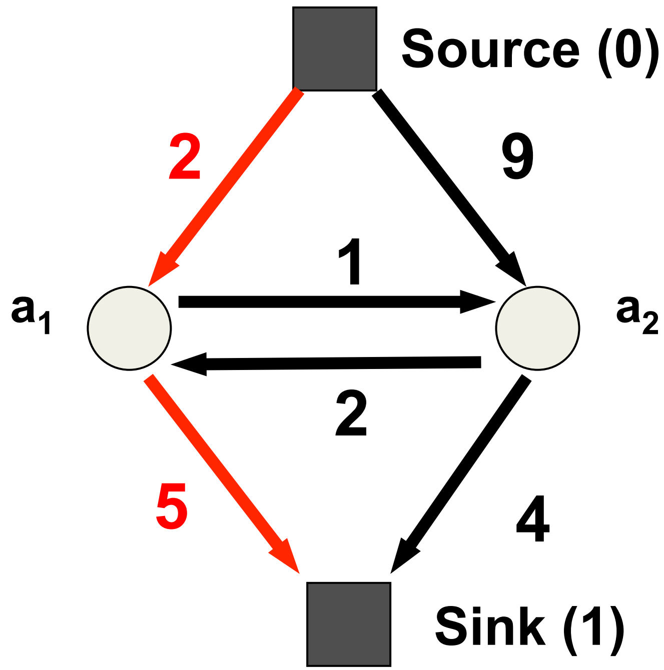
$$E_1(a_1) = 1 + 2a_1 + 3\bar{a}_1$$

$$E_2(a_1) = 3 + \bar{a}_1$$

a_1	$\bar{a}_1$	$1 + 2a_1 + 3\bar{a}_1$	$3 + \bar{a}_1$
0	1	4	4
1	0	3	3

Flow and Reparametrization

$E(a_1,a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$



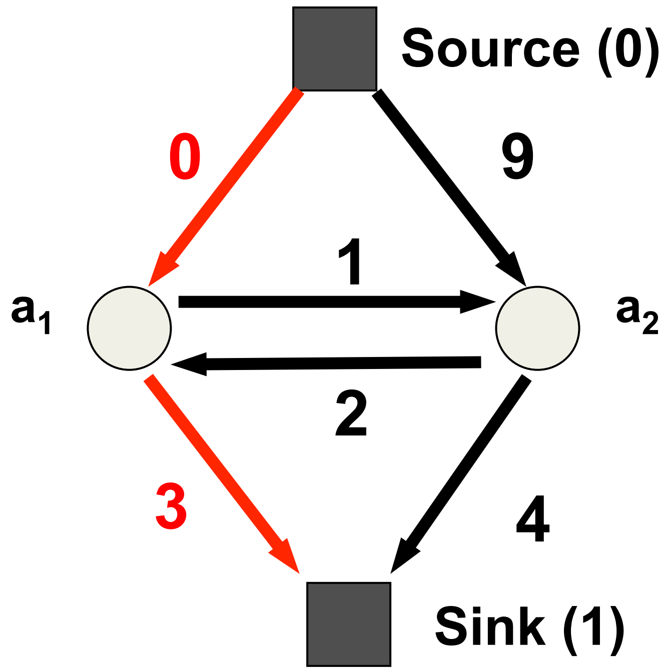
$2a_1 + 5\bar{a}_1$

$= 2(a_1+\bar{a}_1) + 3\bar{a}_1$

$= 2 + 3\bar{a}_1$

Flow and Reparametrization

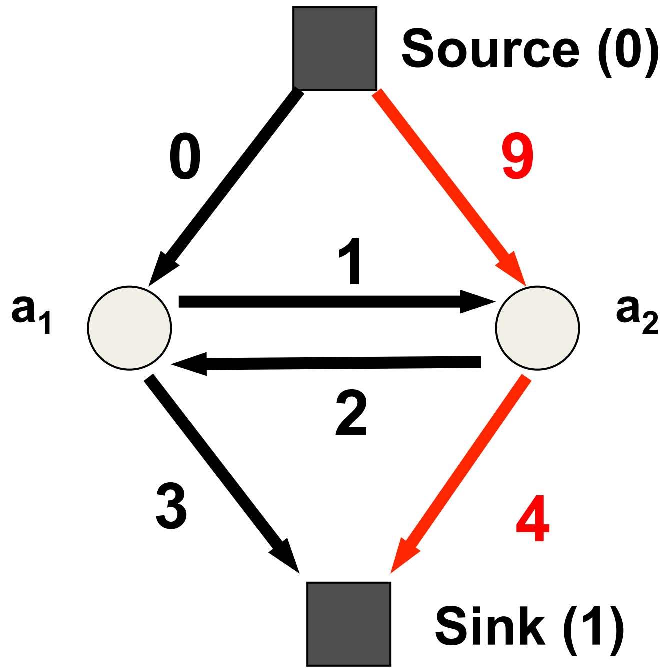
$$E(a_1,a_2) = 2 + 3\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$$



$$2a_1 + 5\bar{a}_1$$
$$= 2(a_1+\bar{a}_1) + 3\bar{a}_1$$
$$= 2 + 3\bar{a}_1$$

Flow and Reparametrization

$E(a_1,a_2) = 2 + 3\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2$



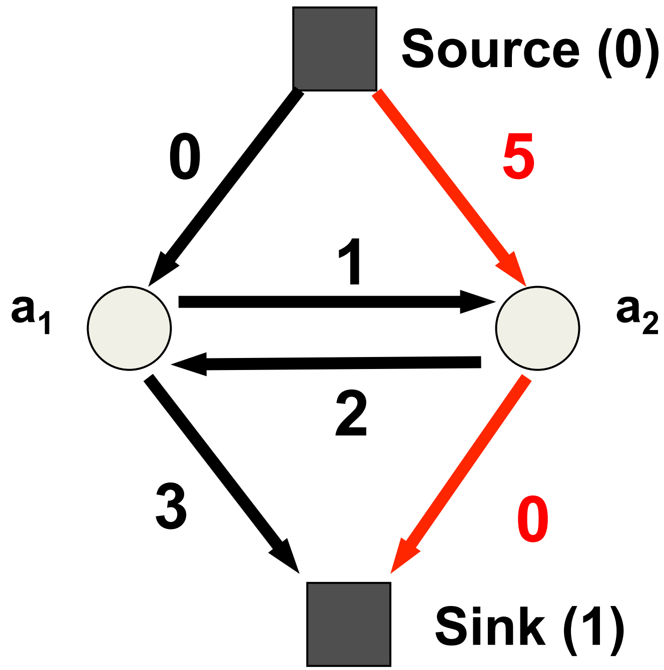
$9a_2 + 4\bar{a}_2$

$= 4(a_2+\bar{a}_2) + 5\bar{a}_2$

$= 4 + 5\bar{a}_2$

Flow and Reparametrization

$E(a_1,a_2) = 2 + 3\bar{a}_1 + 5a_2 + 4 + 2a_1\bar{a}_2 + \bar{a}_1 a_2$



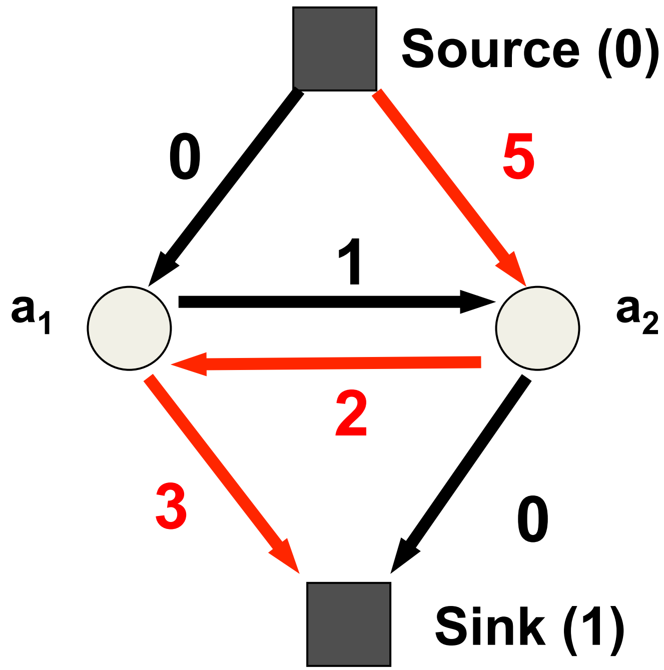
$9a_2 + 4\bar{a}_2$

$= 4(a_2+\bar{a}_2) + 5\bar{a}_2$

$= 4 + 5\bar{a}_2$

Flow and Reparametrization

$E(a_1,a_2) = 6 + 3\bar{a}_1 + 5a_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2$



$3\bar{a}_1 + 5a_2 + 2a_1\bar{a}_2$

$= 2(\bar{a}_1 + a_2 + a_1\bar{a}_2) + \bar{a}_1 + 3a_2$

$= 2(1 + \bar{a}_1a_2) + \bar{a}_1 + 3a_2$

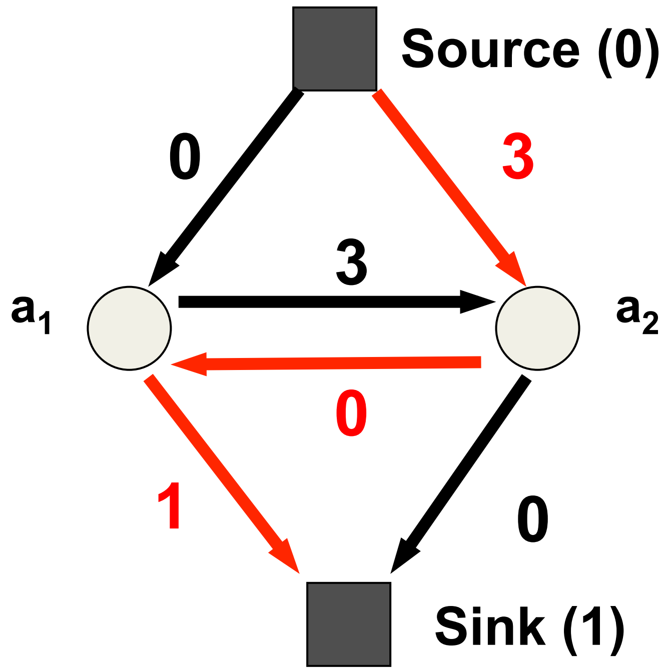
$F1 = \bar{a}_1 + a_2 + a_1\bar{a}_2$

$F2 = 1 + \bar{a}_1a_2$

a_1	a_2	F1	F2
0	0	1	1
0	1	2	2
1	0	1	1
1	1	1	1

Flow and Reparametrization

$E(a_1,a_2) = 8 + \bar{a}_1 + 3a_2 + 3\bar{a}_1a_2$



$3\bar{a}_1 + 5a_2 + 2a_1\bar{a}_2$

$= 2(\bar{a}_1 + a_2 + a_1\bar{a}_2) + \bar{a}_1 + 3a_2$

$= 2(1 + \bar{a}_1a_2) + \bar{a}_1 + 3a_2$

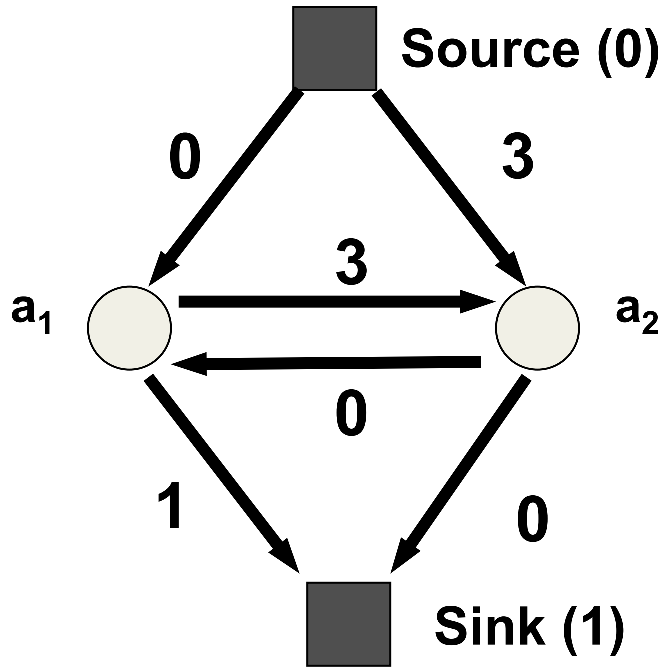
$F1 = \bar{a}_1 + a_2 + a_1\bar{a}_2$

$F2 = 1 + \bar{a}_1a_2$

a_1	a_2	F1	F2
0	0	1	1
0	1	2	2
1	0	1	1
1	1	1	1

Flow and Reparametrization

$E(a_1,a_2) = 8 + \bar{a}_1 + 3a_2 + 3\bar{a}_1 a_2$



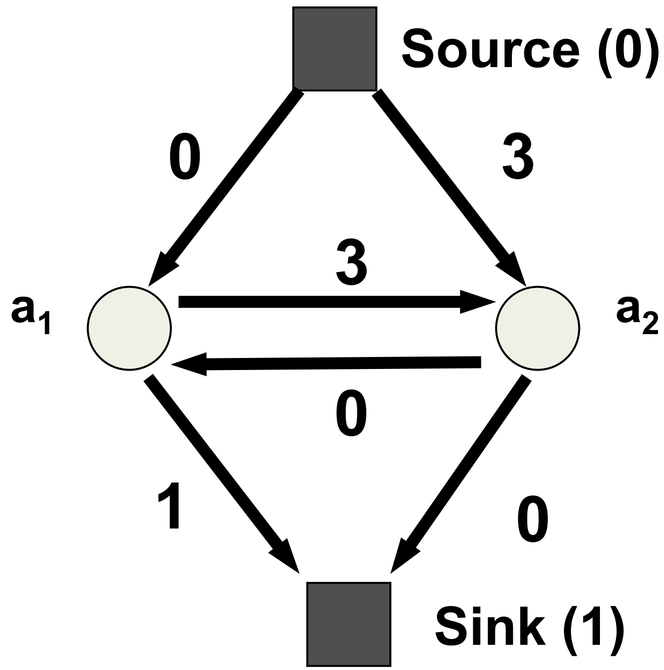
No more augmenting paths possible

Flow and Reparametrization

$E(a_1,a_2) = 8 + \bar{a}_1 + 3a_2 + 3\bar{a}_1 a_2$ → Residual Graph (positive coefficients)

8 → Total Flow

bound on the optimal solution



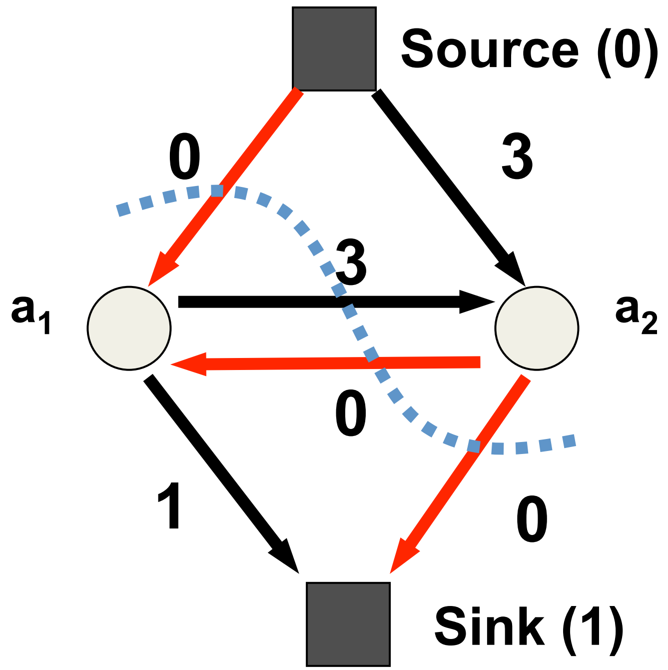
Inference of the optimal solution becomes trivial because the bound is tight

Flow and Reparametrization

$E(a_1,a_2) = 8 + \bar{a}_1 + 3a_2 + 3\bar{a}_1 a_2$ → Residual Graph (positive coefficients)

Total Flow

bound on the optimal solution



st-mincut cost = 8

$a_1 = 1 \quad a_2 = 0$

$E(1,0) = 8$

Inference of the optimal solution becomes trivial because the bound is tight

Example: Image Segmentation

$$E(y) = \sum_i c_i y_i + \sum_{i,j} c_{ij} y_i(1-y_j)$$

$E: \{0,1\}^n \rightarrow R$
$0 \rightarrow fg$
$1 \rightarrow bg$

Global Minimum (y^*)

$$y^* = \arg\min_y E(y)$$

How to minimize E(x)?

How does the code look like?

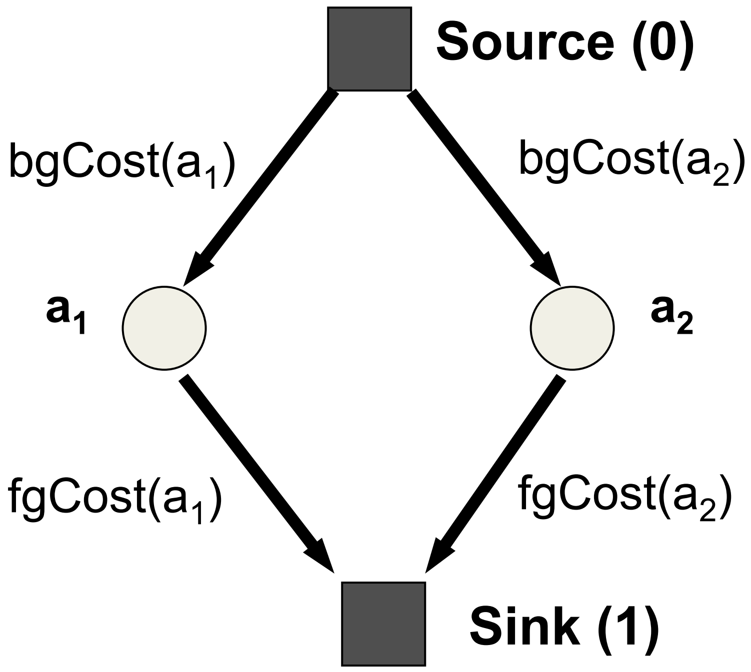
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost);
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```

Source (0)

Sink (1)

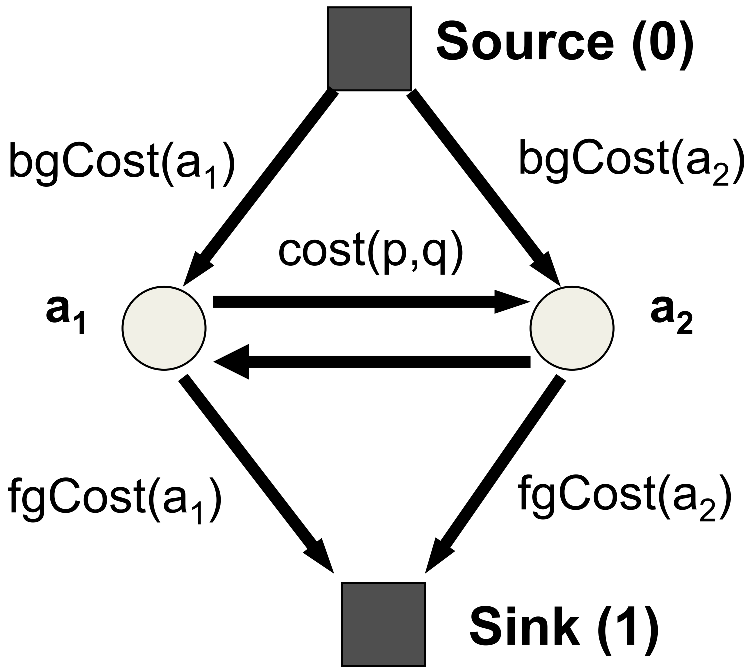
How does the code look like?

```
Graph *g;

For all pixels p

    /* Add a node to the graph */
    nodeID(p) = g->add_node();

    /* Set cost of terminal edges */
    set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
    add_weights(nodeID(p), nodeID(q),  cost);
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



How does the code look like?

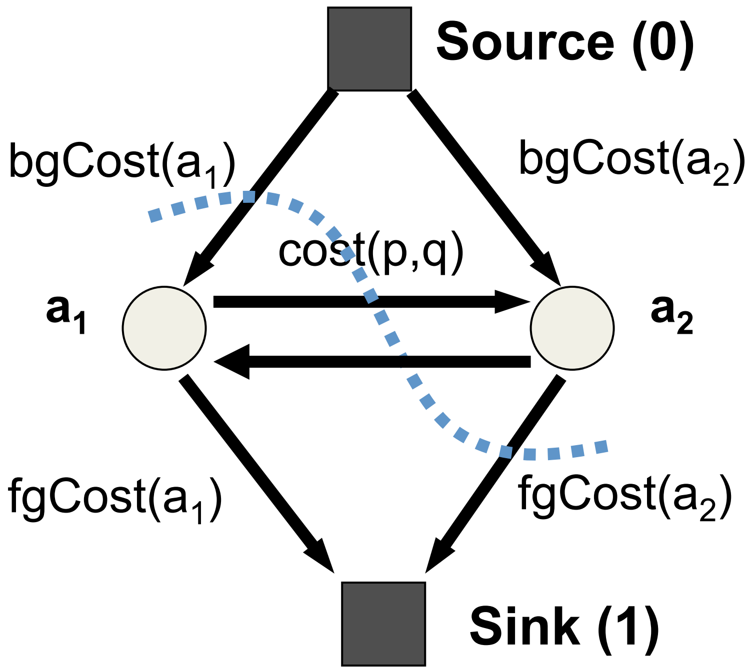
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



How does the code look like?

```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost(p,q));
end
```

```
g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```

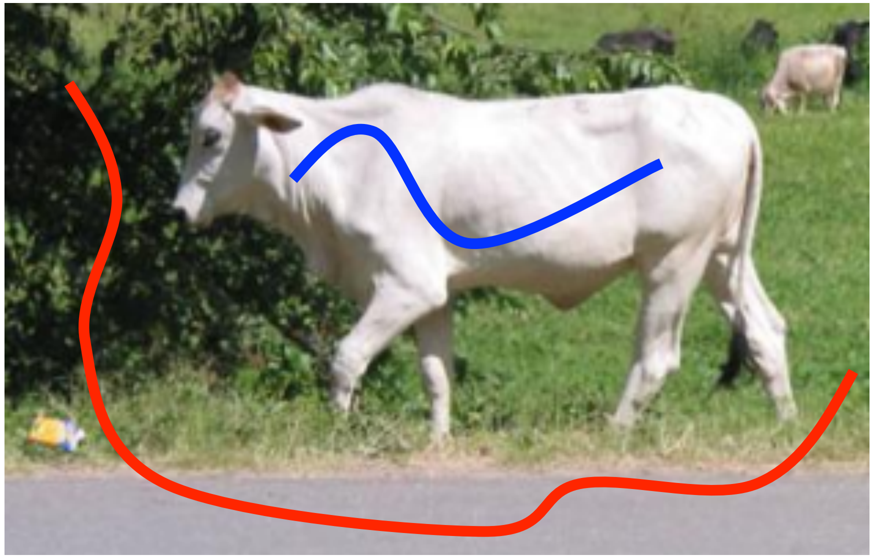


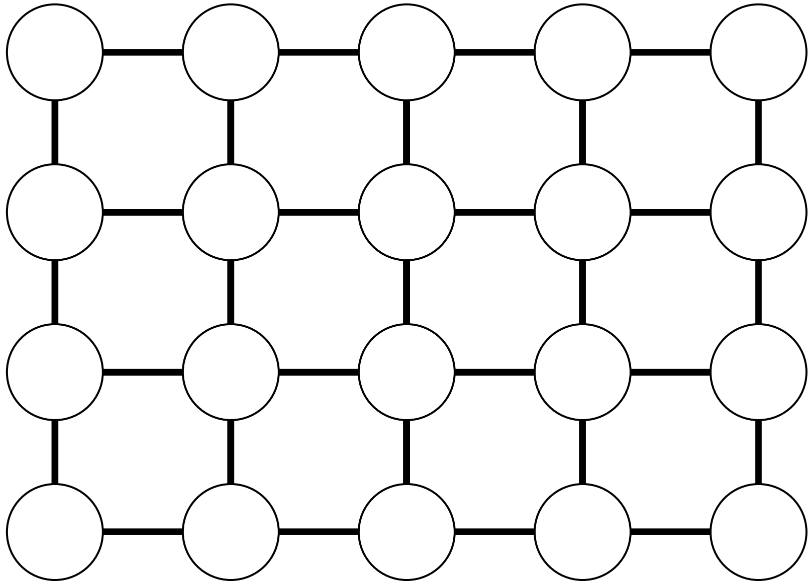
a_1 = bg a_2 = fg

Outline

- Preliminaries
- Maximum Flow
- Algorithms
- **Energy minimization with max flow/min cut**
  - Two-Label Energy Functions

Interactive Binary Segmentation



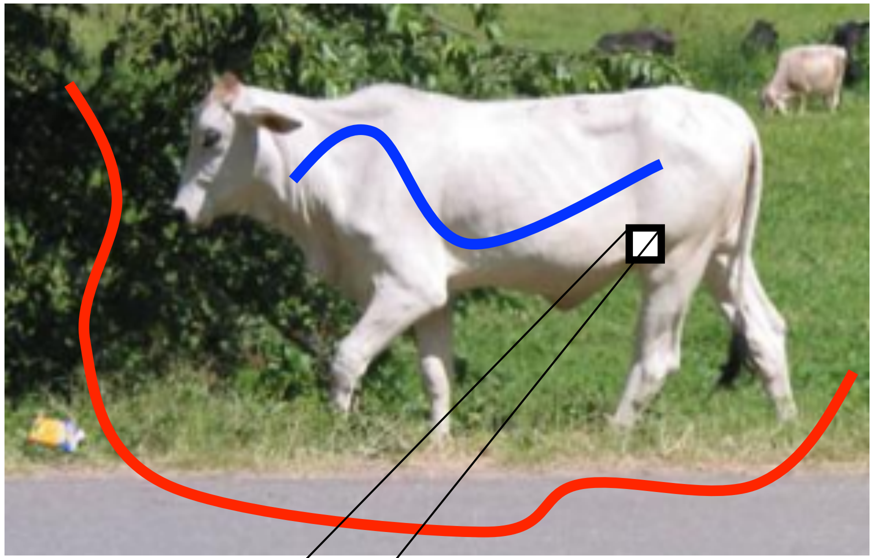


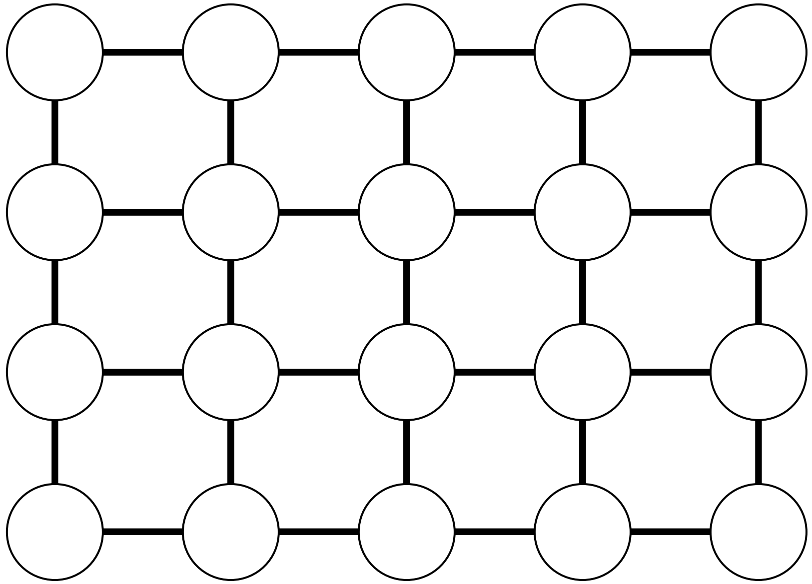
Foreground histogram of RGB values FG

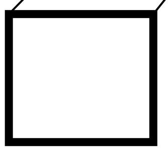
Background histogram of RGB values BG

'1' indicates foreground and '0' indicates background

Interactive Binary Segmentation

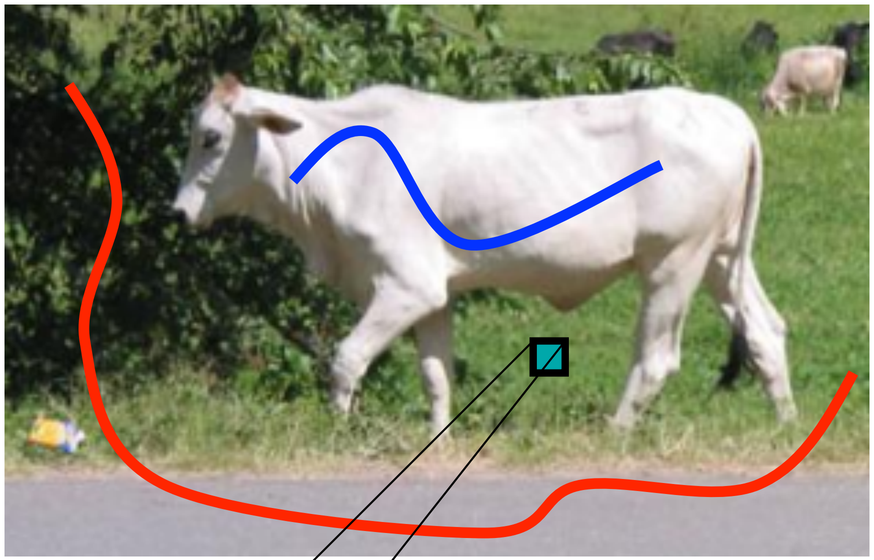


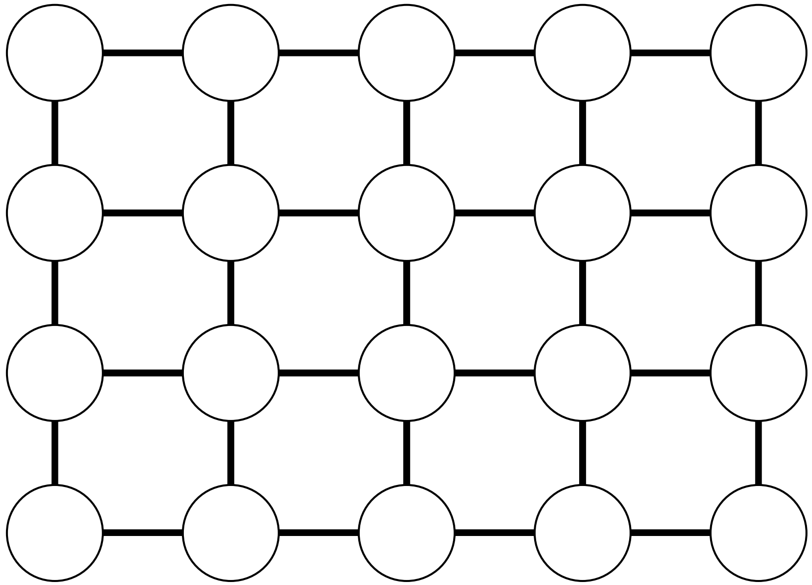




More likely to be foreground than background

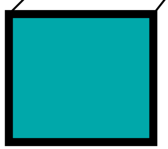
Interactive Binary Segmentation





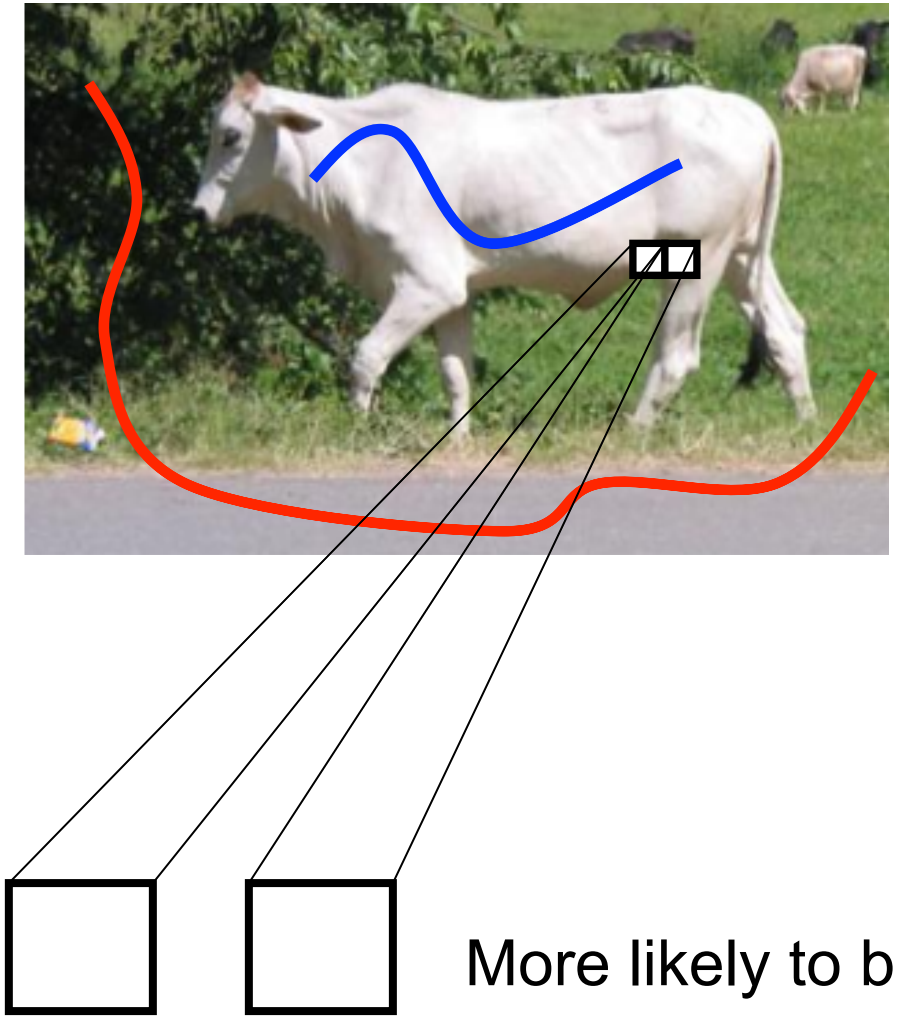
$\theta_p(0)$ proportional to $-\log(BG(d_a))$

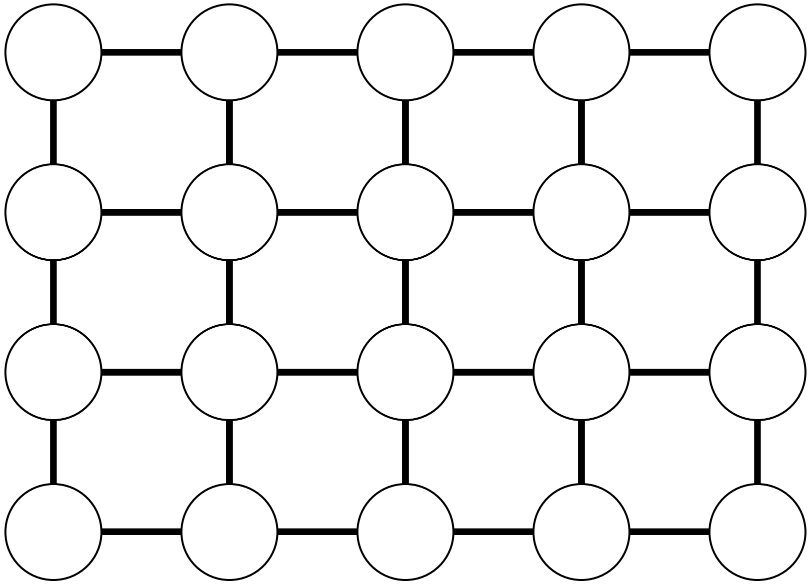
$\theta_p(1)$ proportional to $-\log(FG(d_a))$



More likely to be background than foreground

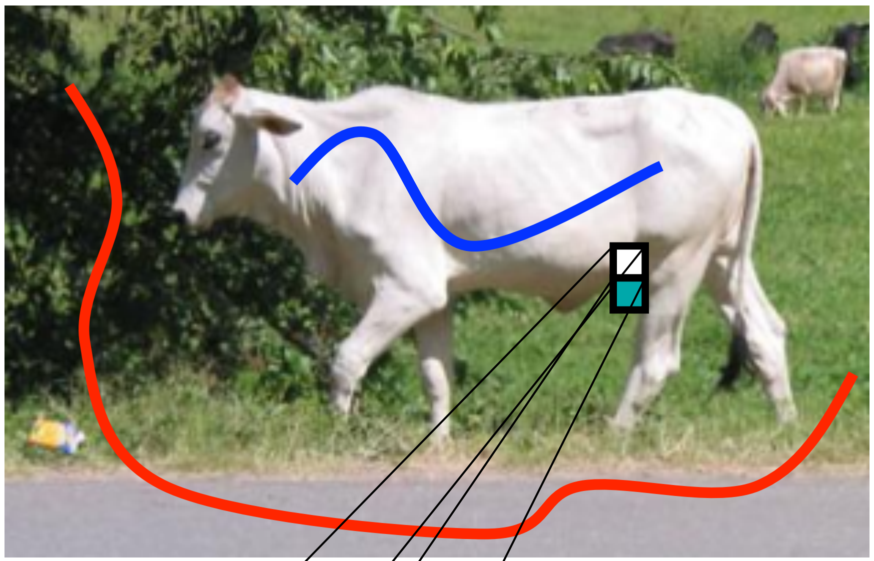
Interactive Binary Segmentation

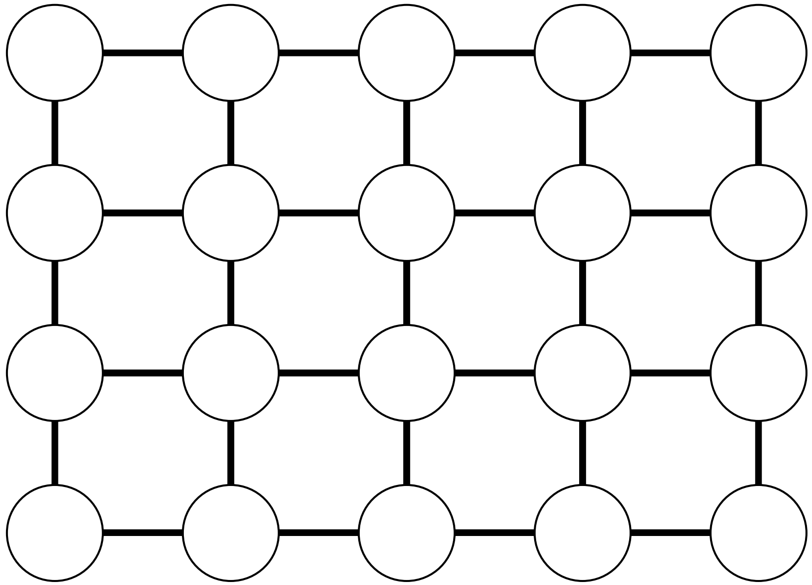




More likely to belong to same label

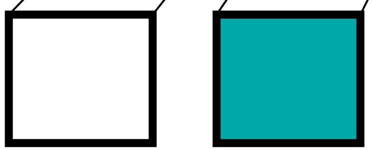
Interactive Binary Segmentation





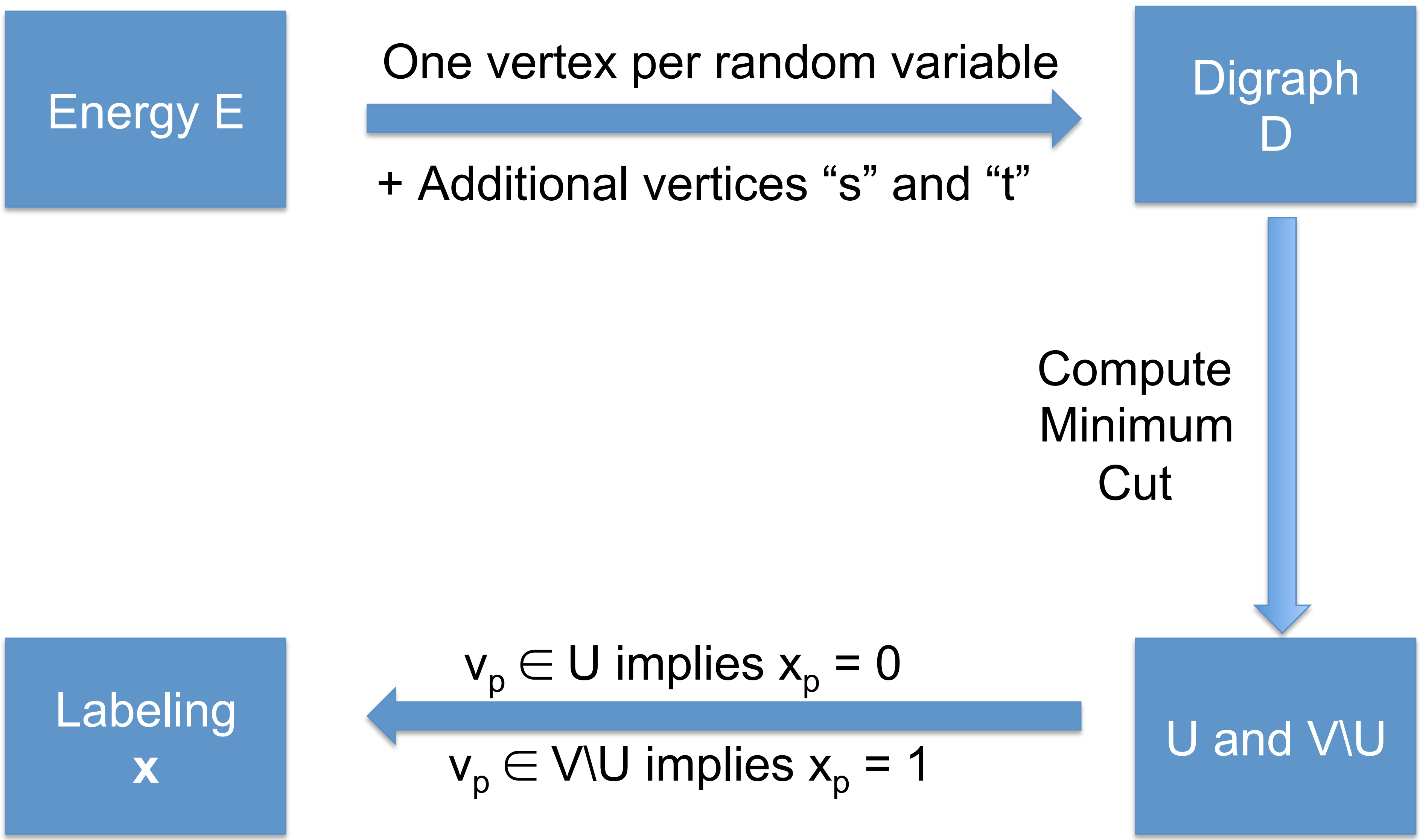
$\theta_{pq}(i,k)$ proportional to $\exp(-(d_a-d_b)^2)$ if $i \neq k$

$\theta_{pq}(i,k) = 0$ if $i = k$



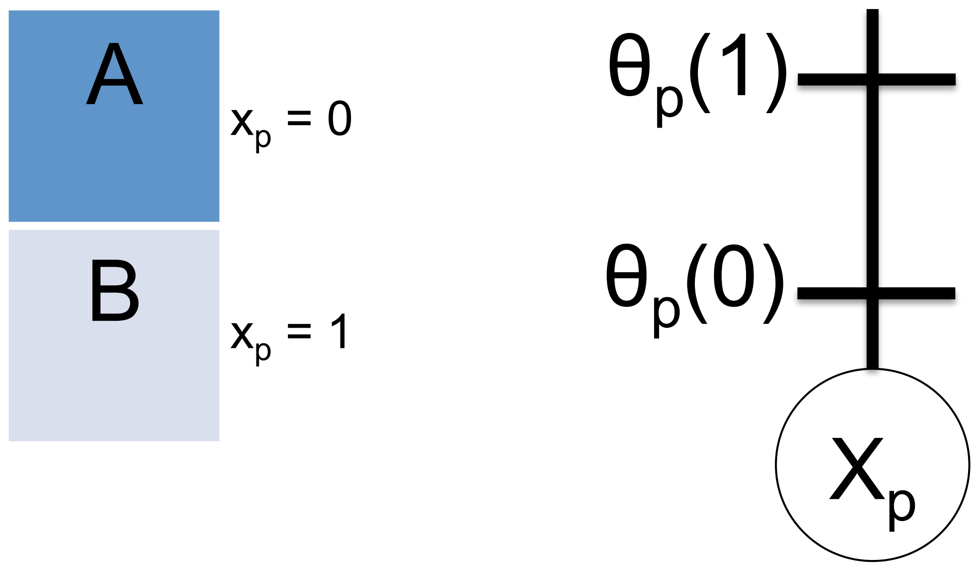
Less likely to belong to same label

Overview

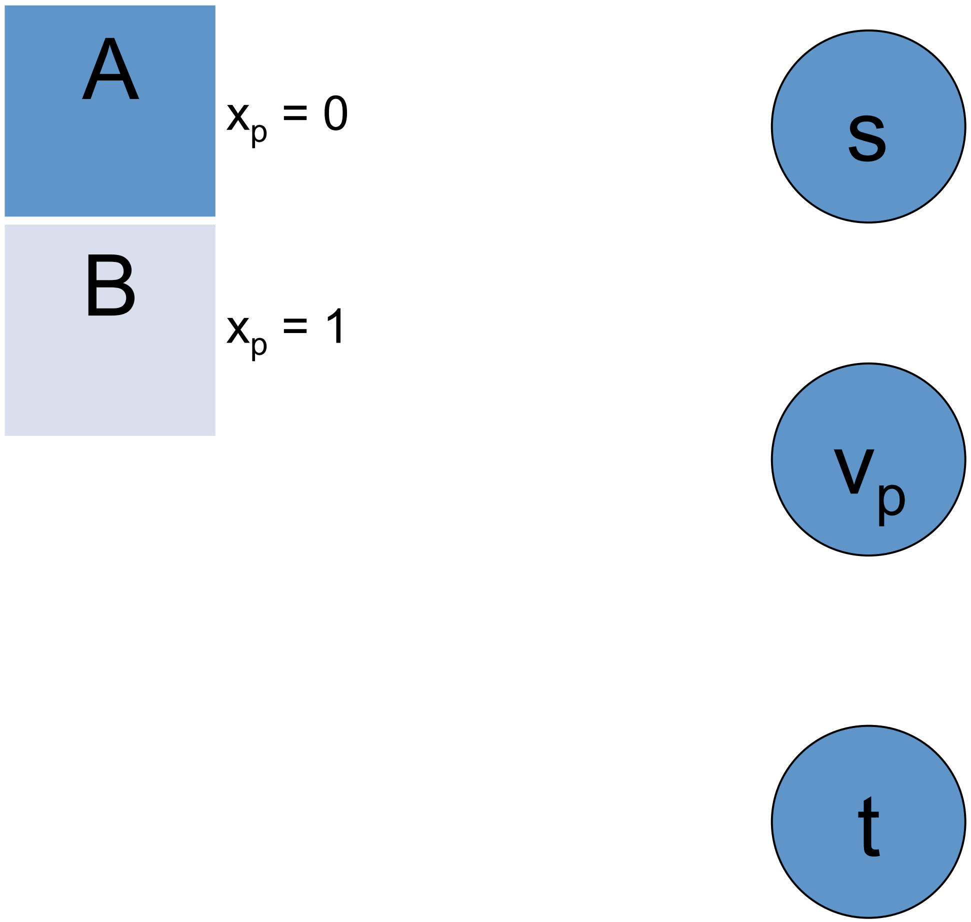


Energy E
One vertex per random variable
+ Additional vertices “s” and “t”
Digraph D
Compute Minimum Cut
U and V\U
$v_p \in U$ implies $x_p = 0$
$v_p \in V \setminus U$ implies $x_p = 1$
Labeling **x**

Digraph for Unary Potentials

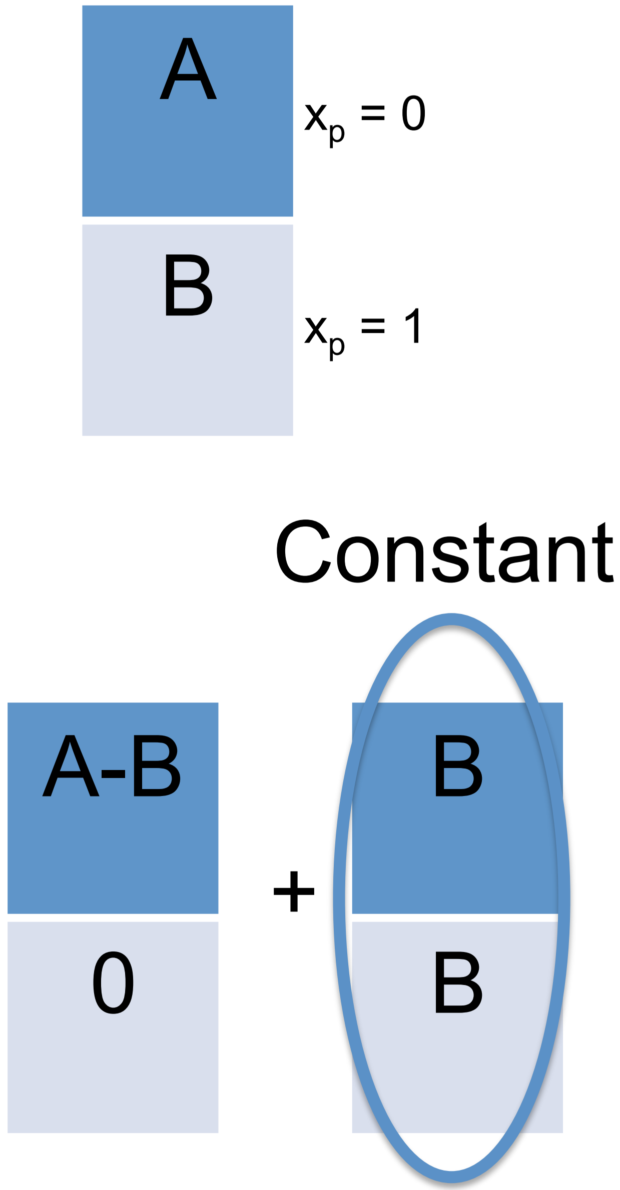


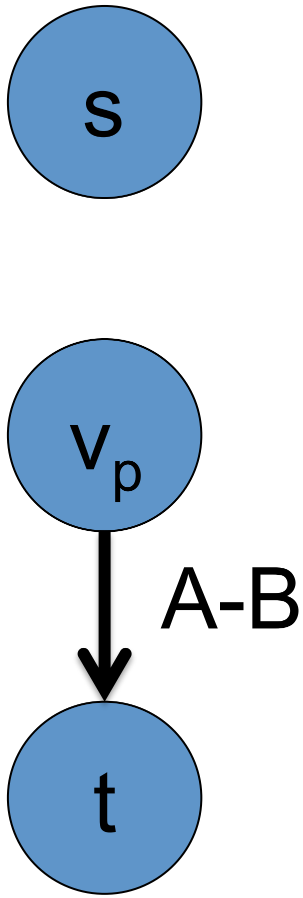
Digraph for Unary Potentials



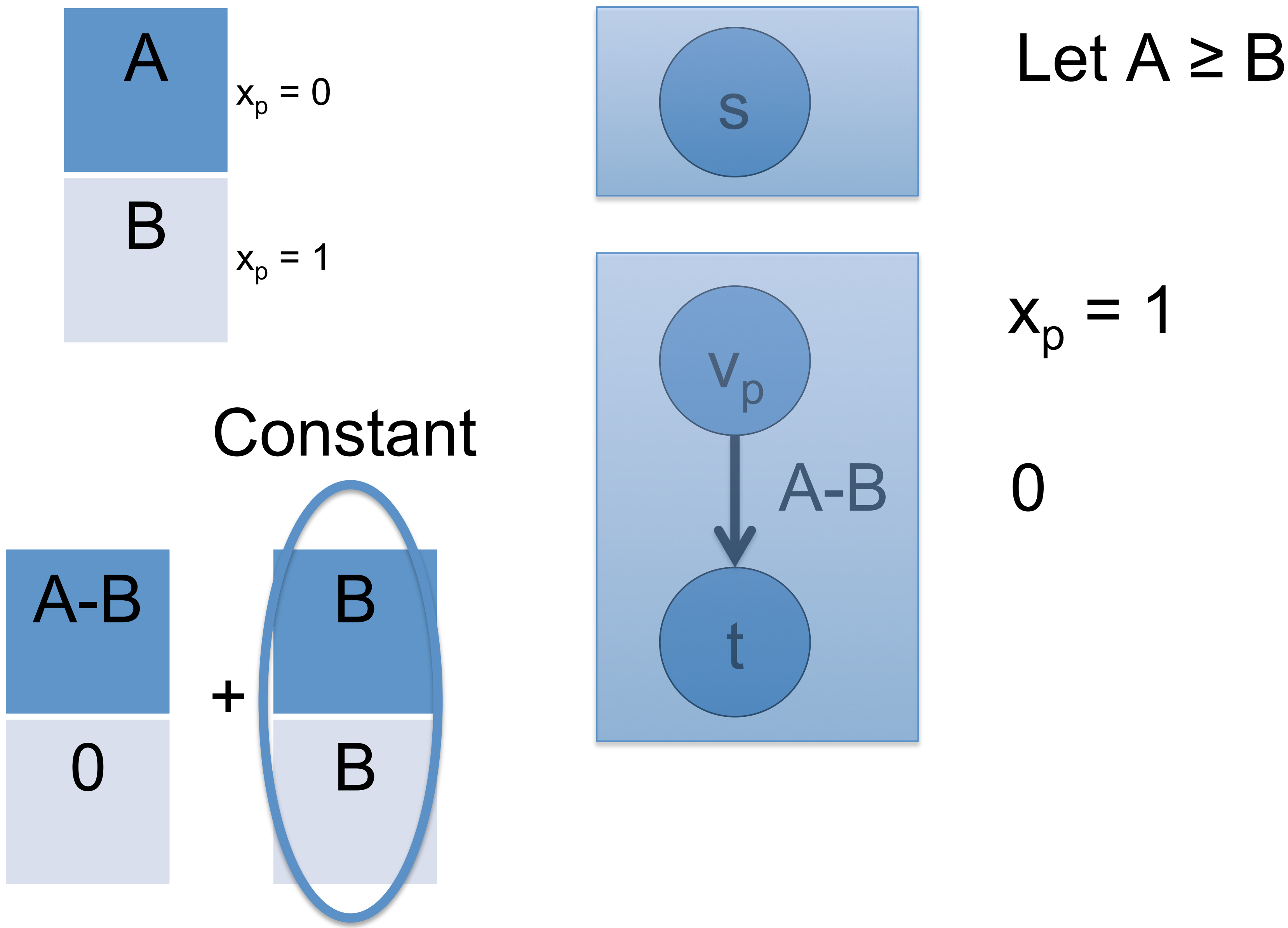
Digraph for Unary Potentials

Let $A \geq B$

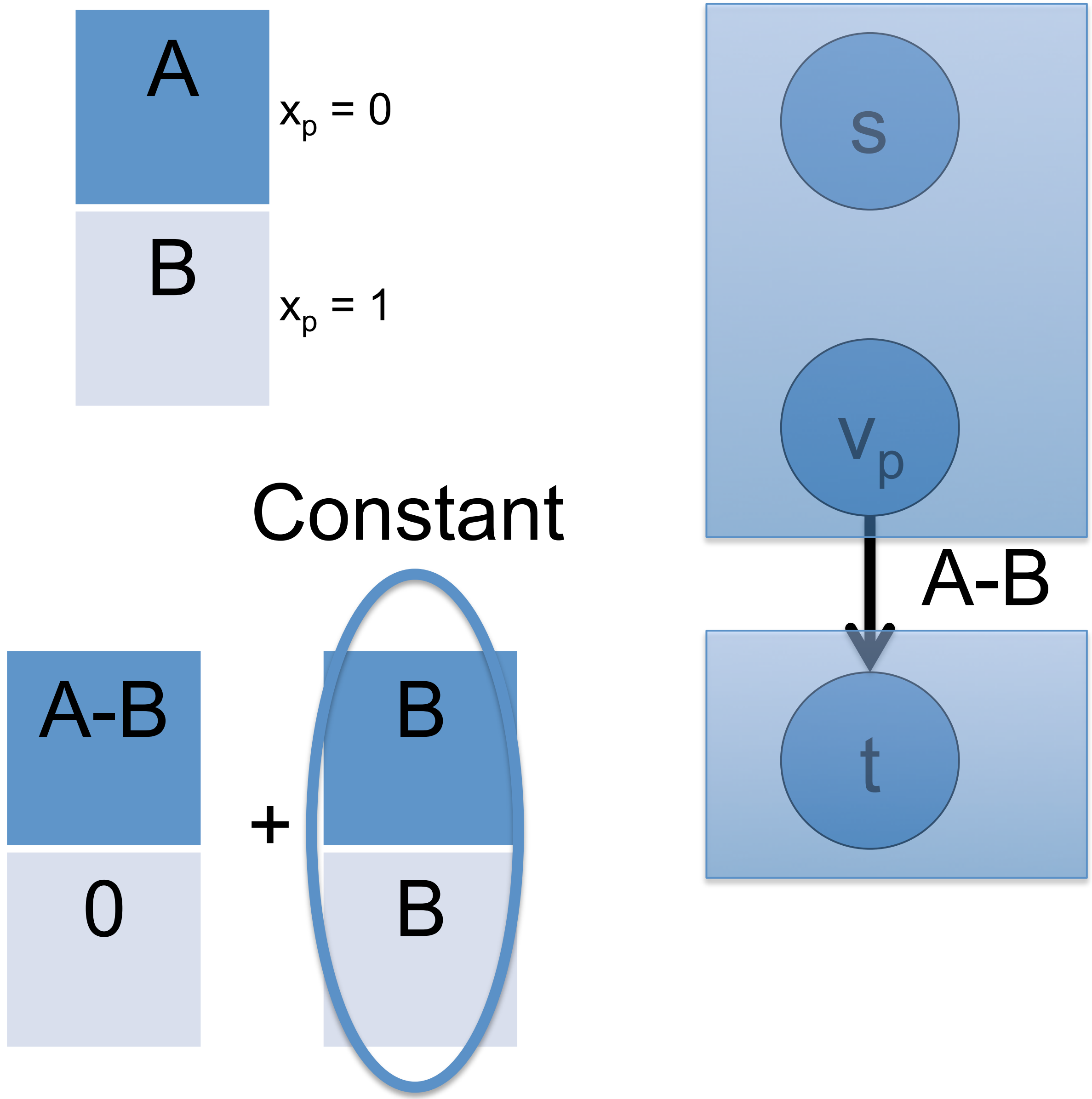




Digraph for Unary Potentials



Digraph for Unary Potentials

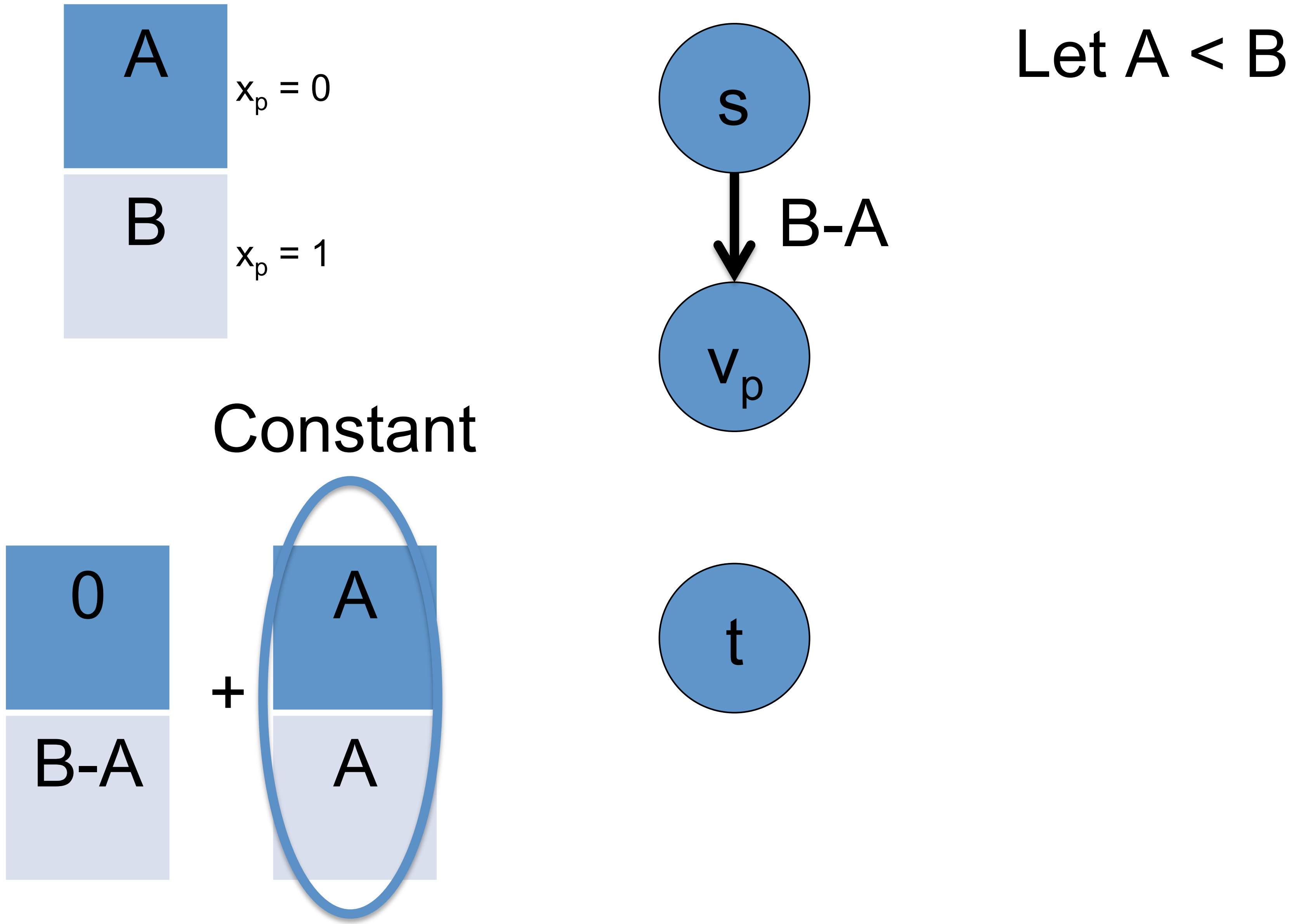


Let A ≥ B

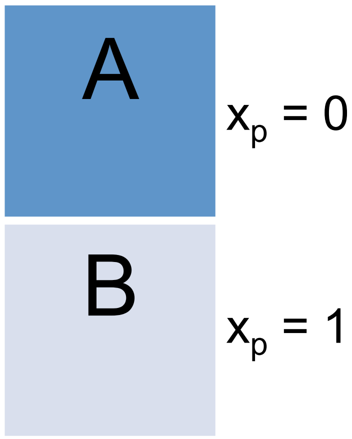
$x_p = 0$

A-B

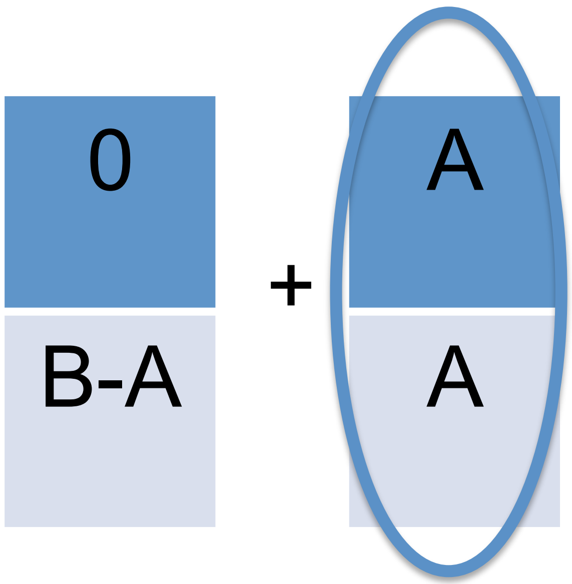
Digraph for Unary Potentials

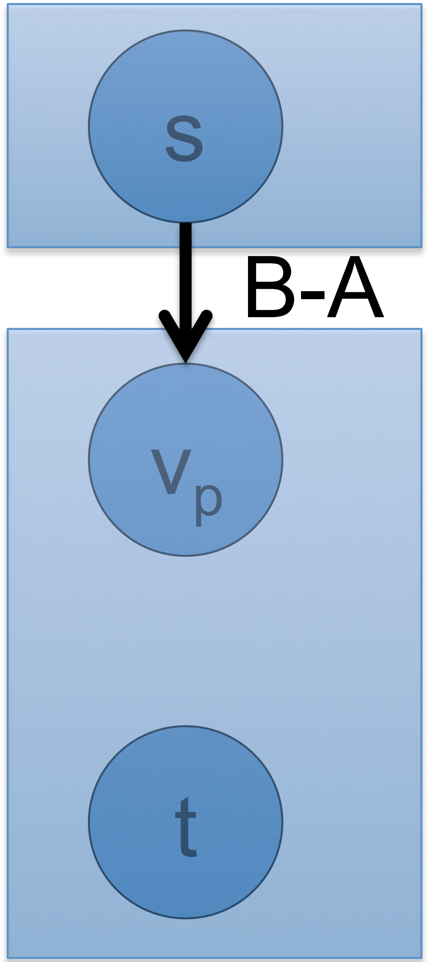


Digraph for Unary Potentials



Constant



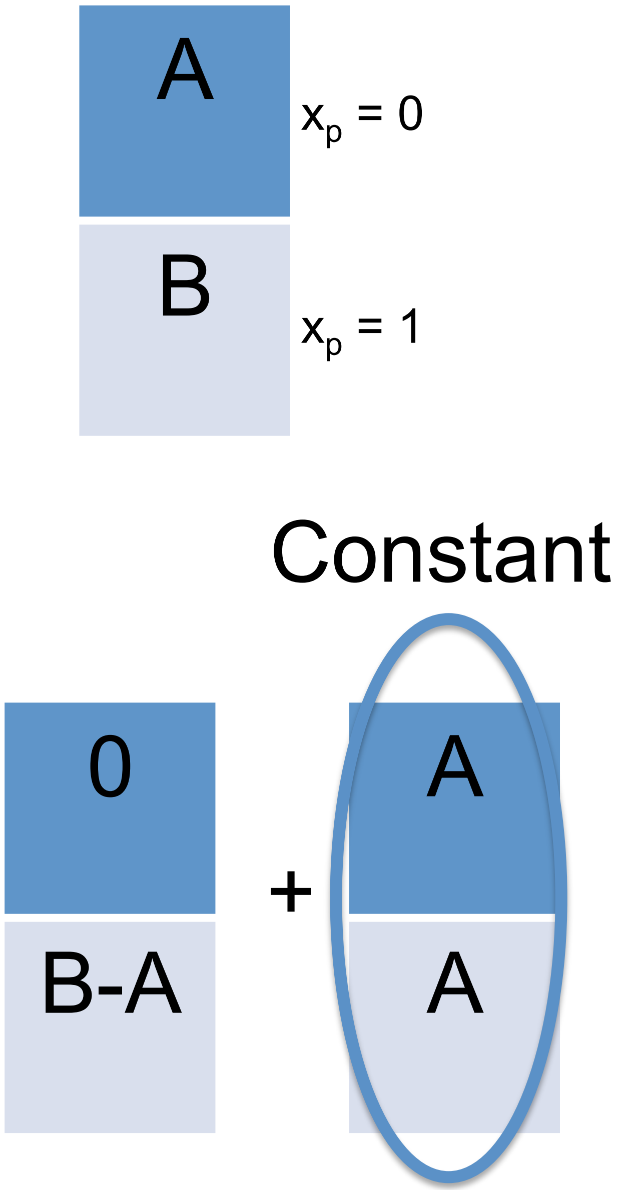


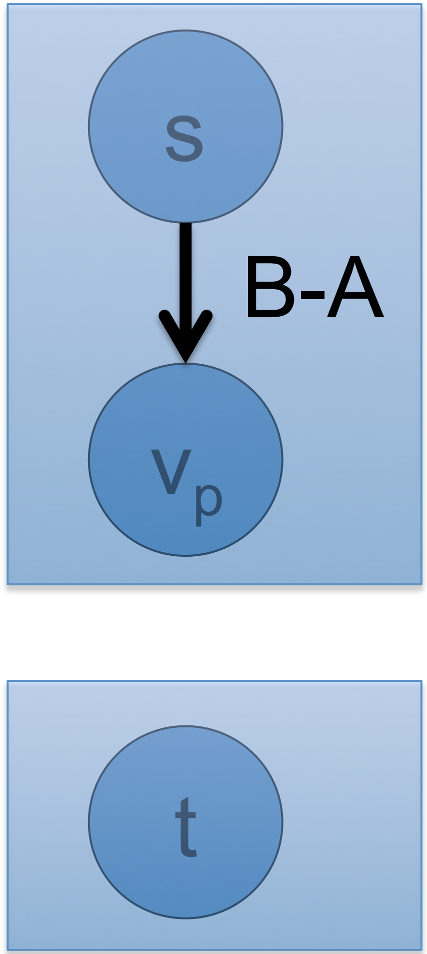
Let $A < B$

$x_p = 1$

B-A

Digraph for Unary Potentials





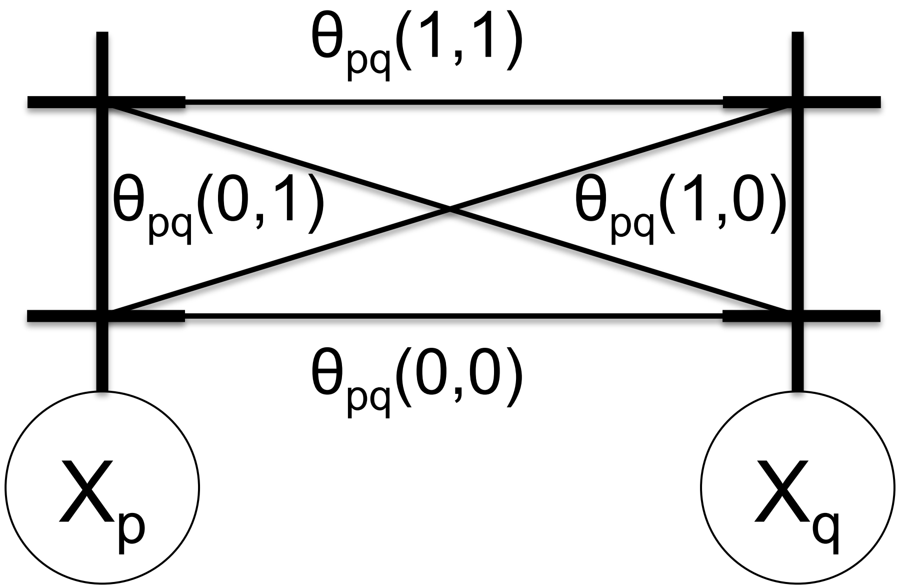
Let A < B

$x_p = 0$

0

Digraph for Pairwise Potentials

	$x_p = 0$	$x_p = 1$
$x_q = 0$	A	C
$x_q = 1$	B	D



A	A
A	A

\+

0	0
B-A	B-A

\+

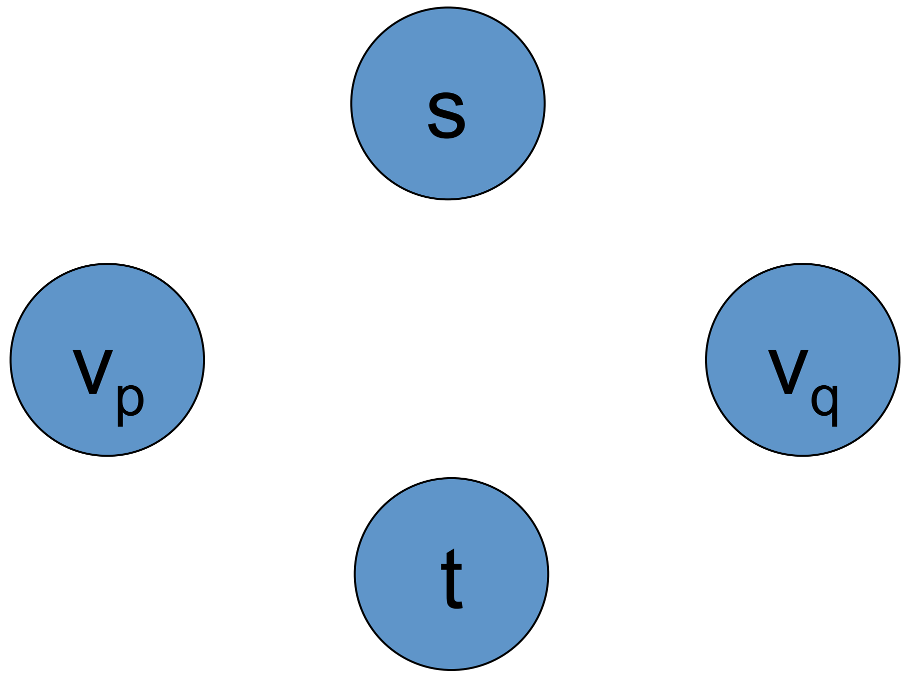
0	D-B
0	D-B

\+

0	C+B-D-A
0	0

Digraph for Pairwise Potentials

	$x_p = 0$	$x_p = 1$
$x_q = 0$	A	C
$x_q = 1$	B	D



Constant

A	A
A	A

\+

0	0
B-A	B-A

\+

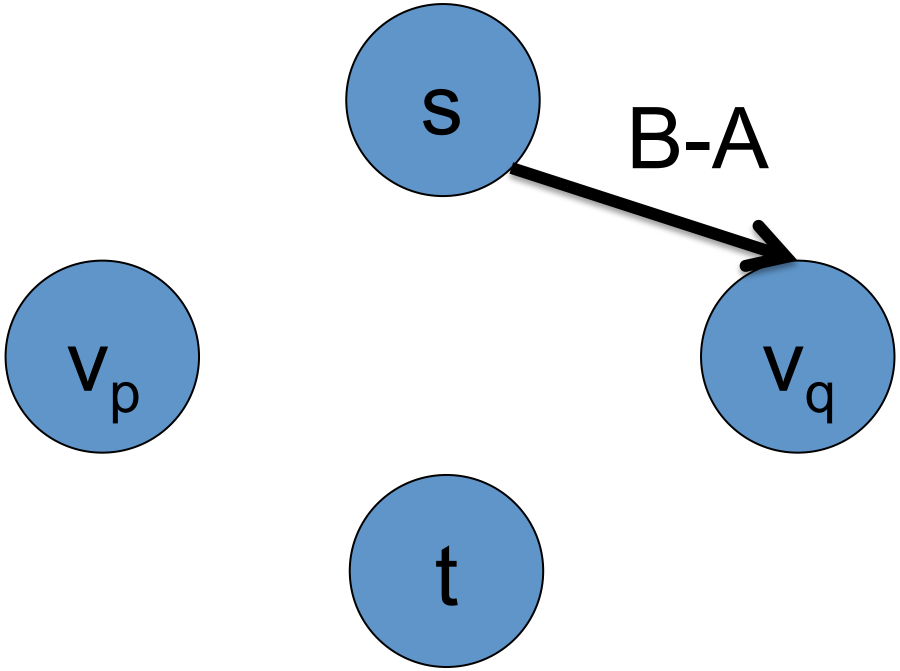
0	D-B
0	D-B

\+

0	C+B-D-A
0	0

Digraph for Pairwise Potentials

	$x_p = 0$	$x_p = 1$
$x_q = 0$	A	C
$x_q = 1$	B	D



Unary Potential

$x_q = 1$

0	0
B-A	B-A

+

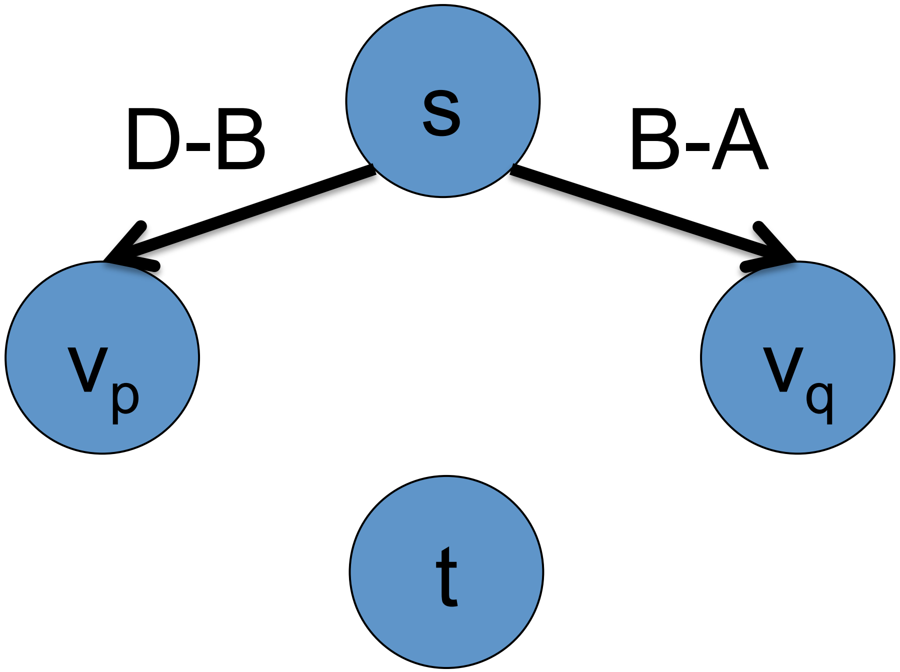
0	D-B
0	D-B

+

0	C+B-D-A
0	0

Digraph for Pairwise Potentials

	$x_p = 0$	$x_p = 1$
$x_q = 0$	A	C
$x_q = 1$	B	D



Unary Potential

$x_p = 1$

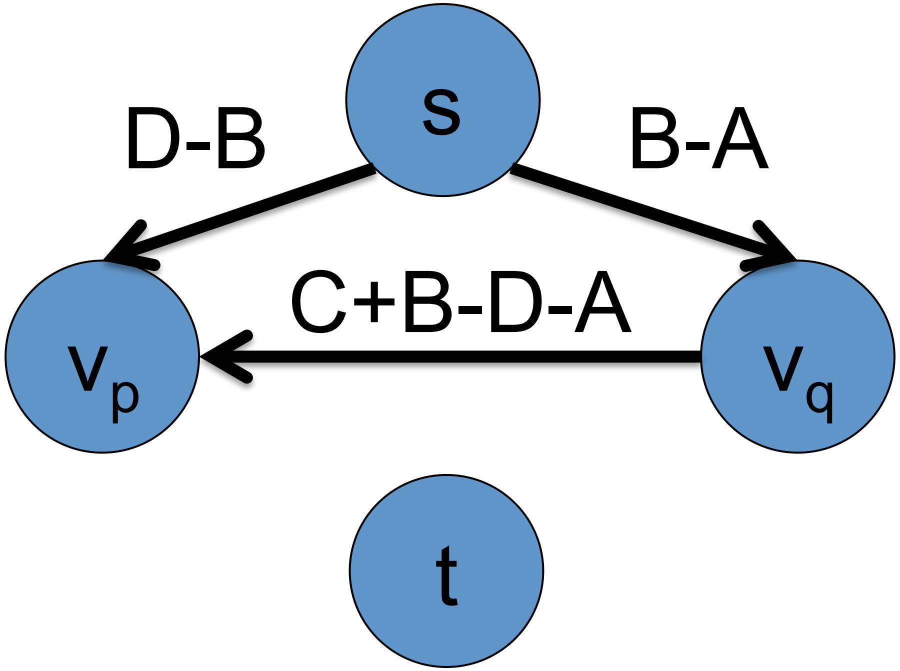
0	D-B
0	D-B

+

0	C+B-D-A
0	0

Digraph for Pairwise Potentials

	$x_p = 0$	$x_p = 1$
$x_q = 0$	A	C
$x_q = 1$	B	D



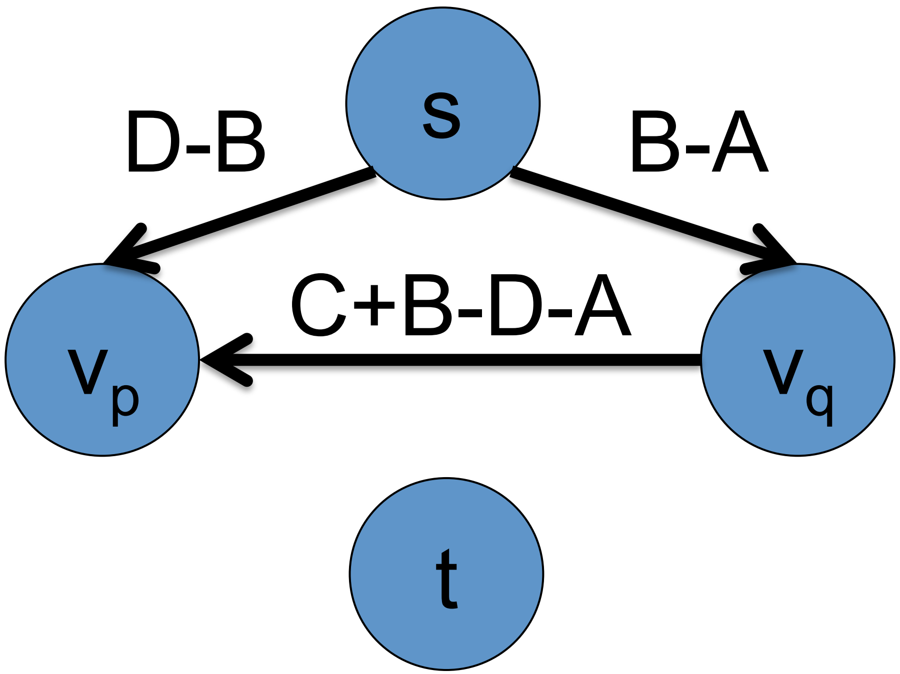
Pairwise Potential

$x_p = 1, x_q = 0$

0	C+B-D-A
0	0

Digraph for Pairwise Potentials

	$x_p = 0$	$x_p = 1$
$x_q = 0$	A	C
$x_q = 1$	B	D



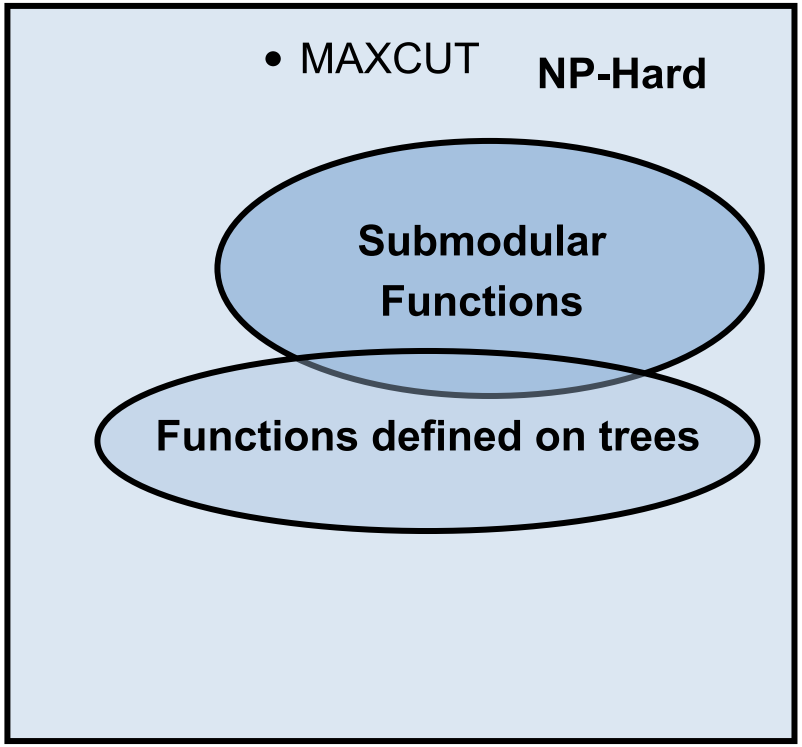
C+B-D-A ≥ 0

Submodular Energy

General 2-label MAP estimation is NP-hard

Minimizing Energy Functions

- **General Energy Functions**
  - NP-hard to minimize
  - Only approximate minimization possible
- **Easy energy functions**
  - Solvable in polynomial time
  - Submodular ~ $O(n^6)$



Space of Function Minimization Problems

Minimizing Submodular Functions

- **Minimizing general submodular functions**
  - $O(n^5 Q + n^6)$ where Q is function evaluation time **[Orlin, IPCO 2007]**

- **Symmetric submodular functions**
  - $E(\mathbf{y}) = E(\mathbf{1} - \mathbf{y})$
  - $O(n^3)$ **[Queyranne 1998]**

- **Quadratic pseudoboolean**
  - Can be transformed to st-mincut
  - One node per variable ($O(n^3)$ complexity)
  - Very low empirical running time

Submodular Pseudoboolean Functions

Function defined over boolean vectors $\mathbf{y} = \{y_1, y_2, \ldots. y_n\}$

Definition

- All functions for one boolean variable ($f: \{0,1\} \rightarrow \mathbb{R}$) are submodular

- A function of two boolean variables ($f: \{0,1\}^2 \rightarrow \mathbb{R}$) is submodular if

$$f(0,1) + f(1,0) \geq f(0,0) + f(1,1)$$

- A general pseudoboolean function $f : 2^n \rightarrow \mathbb{R}$ is submodular if all its projections f^p are submodular i.e.

$$f^p(0,1) + f^p(1,0) \geq f^p(0,0) + f^p(1,1)$$

Quadratic Submodular Pseudoboolean Functions

$$E(y) = \sum_i \theta_i(y_i) + \sum_{i,j} \theta_{ij}(y_i, y_j)$$

For all ij $\quad \theta_{ij}(0,1) + \theta_{ij}(1,0) \geq \theta_{ij}(0,0) + \theta_{ij}(1,1)$

↕ **Equivalent (transformable)**

$$E(y) = \sum_i c_i y_i + \sum_{i,j} c_{ij} y_i (1-y_j)$$ $\quad c_{ij} \geq 0$

i.e. all submodular QPBFs are st-mincut solvable

How are they equivalent?

$A = \theta_{ij}(0,0)$	$B = \theta_{ij}(0,1)$	$C = \theta_{ij}(1,0)$	$D = \theta_{ij}(1,1)$

$$\begin{array}{c|cc} y_i \backslash y_j & 0 & 1 \\ \hline 0 & A & B \\ 1 & C & D \end{array} = A + \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & C-A & C-A \end{array} + \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & D-C \\ 1 & 0 & D-C \end{array} + \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & B+C-A-D \\ 1 & 0 & 0 \end{array}$$

if y_i=1 add C-A **if y_j = 1 add D-C**

$$\theta_{ij}(y_i,y_j) = \theta_{ij}(0,0)$$
$$+ (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, y_i + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, y_j$$
$$+ (\theta_{ij}(1,0) + \theta_{ij}(0,1) - \theta_{ij}(0,0) - \theta_{ij}(1,1))\, (1-y_i)\, y_j$$

B+C-A-D ≥ 0 is true from the submodularity of θ_{ij}

How are they equivalent?

$A = \theta_{ij}(0,0)$ $B = \theta_{ij}(0,1)$ $C = \theta_{ij}(1,0)$ $D = \theta_{ij}(1,1)$

$$
\begin{array}{c|cc} y_i \backslash y_j & 0 & 1 \\ \hline 0 & A & B \\ 1 & C & D \end{array}
= \boxed{A} +
\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & C-A & C-A \end{array}
+
\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & D-C \\ 1 & 0 & D-C \end{array}
+
\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & B+C-A-D \\ 1 & 0 & 0 \end{array}
$$

if y_i=1 add C-A

if y_j = 1 add D-C

$$
\begin{aligned}
\theta_{ij}(y_i,y_j) = \; & \boxed{\theta_{ij}(0,0)} \\
& + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, y_i + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, y_j \\
& + (\theta_{ij}(1,0) + \theta_{ij}(0,1) - \theta_{ij}(0,0) - \theta_{ij}(1,1))\, (1-y_i)\, y_j
\end{aligned}
$$

$B+C-A-D \geq 0$ is true from the submodularity of θ_{ij}

How are they equivalent?

$A = \theta_{ij}(0,0)$	$B = \theta_{ij}(0,1)$	$C = \theta_{ij}(1,0)$	$D = \theta_{ij}(1,1)$

$y_i \backslash y_j$	0	1
0	A	B
1	C	D

= A +

	0	1
0	0	0
1	C-A	C-A

if y_i=1 add C-A

\+

	0	1
0	0	D-C
1	0	D-C

if y_j = 1 add D-C

\+

	0	1
0	0	B+C-A-D
1	0	0

$$\theta_{ij}(y_i,y_j) = \theta_{ij}(0,0)$$
$$+ (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, y_i + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, y_j$$
$$+ (\theta_{ij}(1,0) + \theta_{ij}(0,1) - \theta_{ij}(0,0) - \theta_{ij}(1,1))\,(1-y_i)\, y_j$$

B+C-A-D ≥ 0 is true from the submodularity of θ_{ij}

How are they equivalent?

$A = \theta_{ij}(0,0)$ $B = \theta_{ij}(0,1)$ $C = \theta_{ij}(1,0)$ $D = \theta_{ij}(1,1)$

y_i \ y_j	0	1
0	A	B
1	C	D

= A +

	0	1
0	0	0
1	C-A	C-A

if y_i=1 add C-A

+

	0	1
0	0	D-C
1	0	D-C

if y_j = 1 add D-C

+

	0	1
0	0	B+C-A-D
1	0	0

$$\theta_{ij}(y_i,y_j) = \theta_{ij}(0,0) + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, y_i + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, y_j + (\theta_{ij}(1,0) + \theta_{ij}(0,1) - \theta_{ij}(0,0) - \theta_{ij}(1,1))\,(1-y_i)\, y_j$$

$B+C-A-D \geq 0$ is true from the submodularity of θ_{ij}

How are they equivalent?

$A = \theta_{ij}(0,0)$	$B = \theta_{ij}(0,1)$	$C = \theta_{ij}(1,0)$	$D = \theta_{ij}(1,1)$

y_i \ y_j	0	1
0	A	B
1	C	D

= A +

	0	1
0	0	0
1	C-A	C-A

if y_i=1 add C-A

+

	0	1
0	0	D-C
1	0	D-C

if y_j = 1 add D-C

+

	0	1
0	0	B+C-A-D
1	0	0

$$\theta_{ij}(y_i,y_j) = \theta_{ij}(0,0)$$
$$+ (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, y_i + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, y_j$$
$$+ (\theta_{ij}(1,0) + \theta_{ij}(0,1) - \theta_{ij}(0,0) - \theta_{ij}(1,1))\, (1-y_i)\, y_j$$

B+C-A-D ≥ 0 is true from the submodularity of θ_{ij}

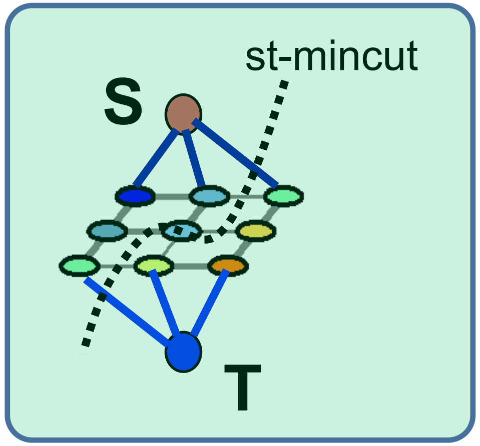
Quadratic Submodular Pseudoboolean Functions

y in $\{0,1\}^n$

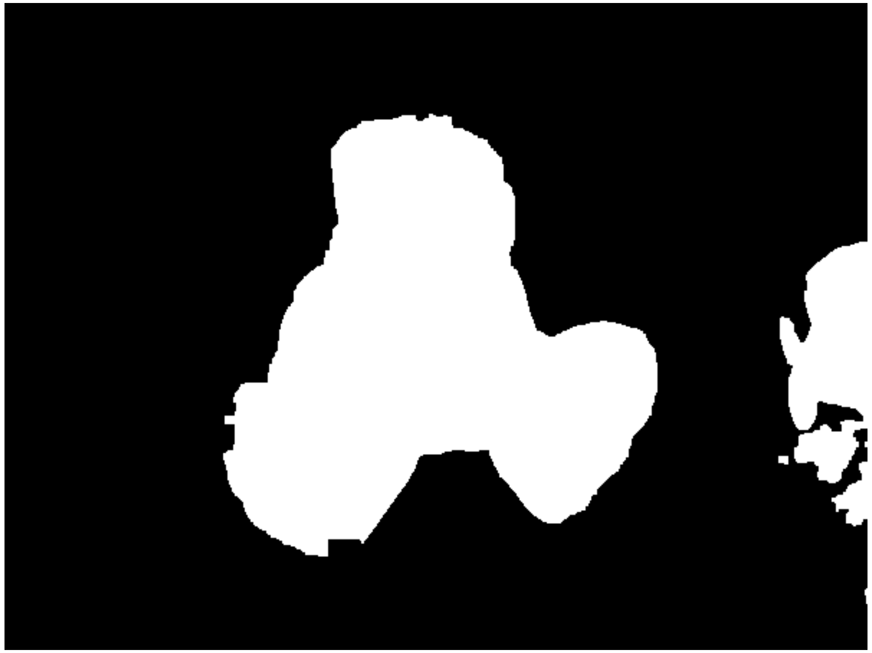
$$E(y) = \sum_i \theta_i (y_i) + \sum_{i,j} \theta_{ij} (y_i, y_j)$$

For all ij $\quad \theta_{ij}(0,1) + \theta_{ij}(1,0) \geq \theta_{ij}(0,0) + \theta_{ij}(1,1)$

Equivalent (transformable)



Results – Image Segmentation



Boykov and Jolly, ICCV 2001

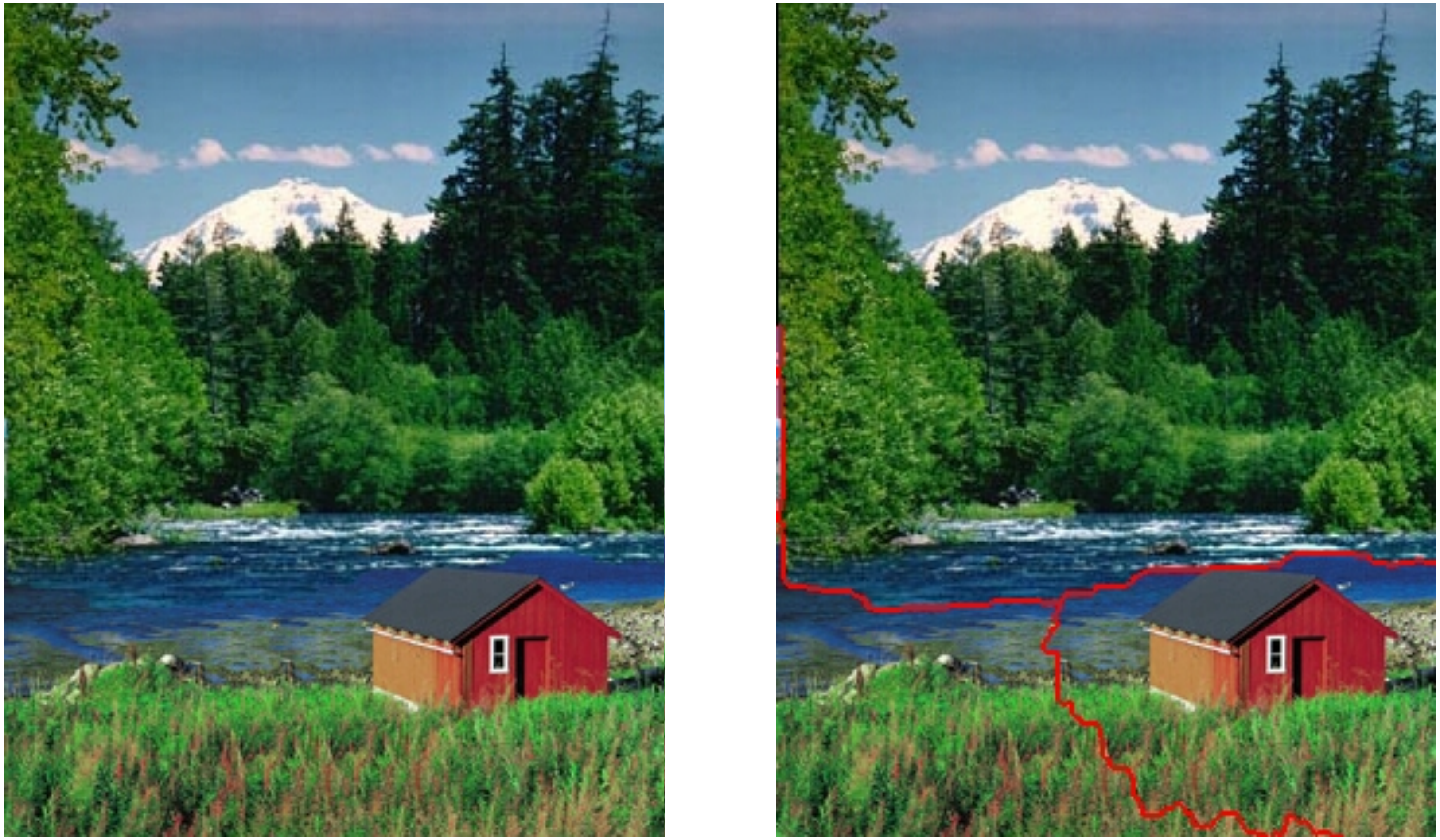
Results – Image Segmentation

Boykov and Jolly, ICCV 2001

Results – Image Segmentation

Boykov and Jolly, ICCV 2001

Results – Image Synthesis



Kwatra et al., SIGGRAPH 2003

Results – Image Synthesis

Kwatra et al., SIGGRAPH 2003