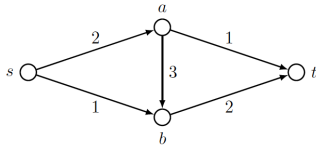


## Network

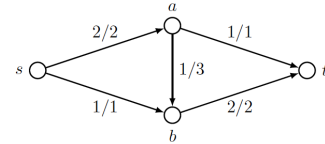


### Definition

A network  $N$  consists of:

- ▶ digraph  $(V, E)$
- ▶ edge capacities  $c : E \rightarrow [0, \infty)$
- ▶ source  $s \in V$ , which has indegree 0, and
- ▶ sink  $t \in V$ , which has outdegree 0.

## Flow



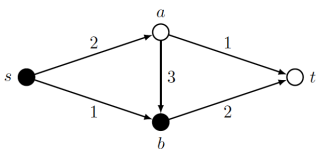
### Definition

A flow is a mapping  $f : E \rightarrow [0, \infty)$  satisfying:

- ▶ [capacity constraints]  $(\forall e \in E) f(e) \leq c(e)$
- ▶ [conservation constraints]  $(\forall v \in V \setminus \{s, t\}) f_{\text{in}}(v) = f_{\text{out}}(v)$   
where  $f_{\text{in}}(v) \doteq \sum_{u \in V: e \doteq (u,v) \in E} f(e)$  and  
 $f_{\text{out}}(v) \doteq \sum_{u \in V: e \doteq (v,u) \in E} f(e)$ .

Value of a flow:  $\nu(f) \doteq f_{\text{out}}(s)$

## Cut



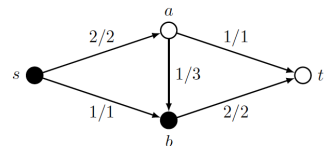
### Definition

An  $st$ -cut is a partition  $(S, T)$  of  $V$  such that  $s \in S$  and  $t \in T$ .

### Note

- ▶  $(S, T)$  is a partition of  $V$  if  $S \cup T = V$  and  $S \cap T = \emptyset$ .
- ▶  $(S, T)$  can alternately be written as  $(S, \bar{S})$  where  $\bar{S} \doteq V \setminus S$ .
- ▶ Term "cut" is sometimes also used to denote the set of edges that cross the cut, i.e.,  $(E \cap S \times T) \cup (E \cap T \times S)$ .

## Invariance Property



### Statement

For every flow  $f$  and  $st$ -cut  $(S, T)$  in a network  $N = (V, E, c, s, t)$

$$f_{\text{out}}(S) - f_{\text{in}}(S) = \nu(f),$$

where  $f_{\text{out}}(S) \doteq \sum_{e \in E \cap S \times T} f(e)$  and  $f_{\text{in}}(S) \doteq \sum_{e \in E \cap T \times S} f(e)$ .

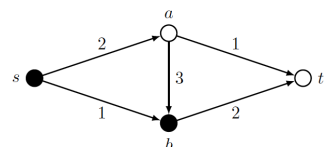
## Proof of Invariance Property

$$\begin{array}{l} \nu(f) \doteq f_{\text{out}}(s) - f_{\text{in}}(s) \\ 0 = f_{\text{out}}(v) - f_{\text{in}}(v) \end{array} \quad v \in S \setminus \{s\}$$

$$\begin{aligned} \nu(f) &= \sum_{v \in S} f_{\text{out}}(v) - \sum_{v \in S} f_{\text{in}}(v) \\ &= \sum_{v \in S} \sum_{e \doteq (v,u) \in E} f(e) - \sum_{v \in S} \sum_{e \doteq (u,v) \in E} f(e) \\ &\stackrel{(*)}{=} \sum_{e^* \in E} c(e^*) f(e^*) \\ &= \sum_{e \in E \cap S \times T} f(e) - \sum_{e \in E \cap T \times S} f(e) \\ &= f_{\text{out}}(S) - f_{\text{in}}(S) \end{aligned}$$

(*) type of $e^* \in E$	$c(e^*)$
$S \times T$	$1 - 0 = 1$
$S \times S$	$1 - 1 = 0$
$T \times S$	$0 - 1 = -1$
$T \times T$	$0 - 0 = 0$

## Cut



### Definition

An  $st$ -cut is a partition  $(S, T)$  of  $V$  such that  $s \in S$  and  $t \in T$ .

### Capacity of an $st$ -cut

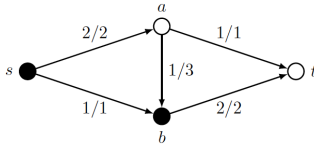
$$c(S, T) \doteq \sum_{e \in E \cap S \times T} c(e)$$

## Weak duality

### Statement

For every flow  $f$  and  $st$ -cut  $(S, T)$  in a network  $N = (V, E, c, s, t)$

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



## Proof of Weak duality

$$\begin{aligned} \nu(f) &= \sum_{e \in E \cap S \times T} f(e) - \sum_{e \in E \cap T \times S} f(e) \\ &\leq \sum_{e \in E \cap S \times T} c(e) - \sum_{e \in E \cap T \times S} 0 \\ &\doteq c(S, T) \end{aligned}$$

### Note

Equality  $\nu(f) = c(S, T)$  holds iff

- ▶  $(\forall e \in E \cap S \times T) f(e) = c(e)$
- ▶  $(\forall e \in E \cap T \times S) f(e) = 0$ .

## Max Flow and Min Cut

### Max flow

Input: network  $N = (V, E, c, s, t)$

Output: flow  $f$  such that  $\nu(f)$  is maximized

### Min cut

Input: network  $N = (V, E, c, s, t)$

Output:  $st$ -cut  $(S, T)$  such that  $c(S, T)$  is minimized

### Weak duality

$$\max_{\text{flow } f} \nu(f) \leq \min_{st\text{-cut } (S, T)} c(S, T)$$

We'll show next lecture that equality always holds (strong duality).

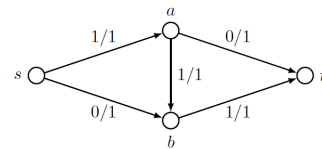
## Path Augmentation - first attempt

### Algorithm

1. Start with  $f \equiv 0$ .
2. While there is an  $st$ -path  $P$  along which more flow can be pushed, additionally push as much flow along  $P$  as possible.
3. Return  $f$ .

### Issue

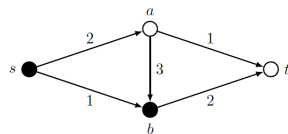
Bad prior choices may block further progress.



## Recap - notions

### Network

- ▶ digraph  $(V, E)$
- ▶ edge capacities  $c : E \rightarrow [0, \infty)$
- ▶ source  $s \in V$  with indegree 0
- ▶ sink  $t \in V$  with outdegree 0



### Flow

A mapping  $f : E \rightarrow [0, \infty)$  satisfying

- ▶ [capacity constraints]  $(\forall e \in E) f(e) \leq c(e)$
- ▶ [conservation constraints]  $(\forall v \in V \setminus \{s, t\}) f_{\text{in}}(v) = f_{\text{out}}(v)$   
where  $f_{\text{in}}(v) \doteq \sum_{u \in V: e \doteq (u, v) \in E} f(e)$  and  $f_{\text{out}}(v) \doteq \sum_{u \in V: e \doteq (v, u) \in E} f(e)$ .

### $st$ -Cut

A partition  $(S, T)$  of  $V$  such that  $s \in S$  and  $t \in T$ .

## Recap - computational problems

### Max flow

Input: network  $N = (V, E, c, s, t)$

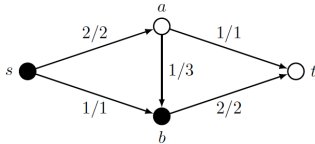
Output: flow  $f$  such that  $\nu(f) \doteq f_{\text{out}}(s)$  is maximized

### Min cut

Input: network  $N = (V, E, c, s, t)$

Output:  $st$ -cut  $(S, T)$  such that  $c(S, T) \doteq \sum_{e \in S \times T} c(e)$  is minimized

## Recap - properties



- ▶ Invariance: For every flow  $f$  and  $st$ -cut  $(S, T)$ 

$$f_{\text{out}}(S) - f_{\text{in}}(S) = \nu(f)$$
- ▶ Weak duality: For every flow  $f$  and  $st$ -cut  $(S, T)$ 

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

Equality  $\nu(f) = c(S, T)$  holds iff

- $(\forall e \in E \cap S \times T) f(e) = c(e)$
- $(\forall e \in E \cap T \times S) f(e) = 0$ .

- ▶ Strong duality (today):

$$\max_{\text{flow } f} \nu(f) = \min_{\text{st-cut } (S, T)} c(S, T)$$

## Residual Network

Consider a flow  $f$  in  $N = (V, E, c, s, t)$ .

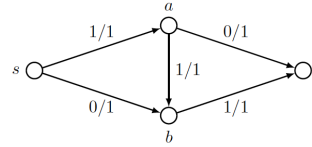
### Definition

The residual network  $N_f = (V, E_f, c_f, s, t)$  has:

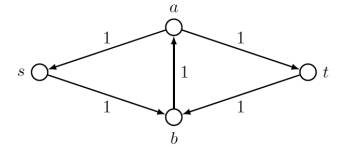
- ▶ For each  $e \in E$  with  $f(e) < c(e)$ , an edge  $e$  in  $E_f$  with  $c_f(e) \doteq c(e) - f(e)$ .
- ▶ For each  $e = (u, v) \in E$  with  $f(e) > 0$ , an edge  $e' \doteq (v, u)$  in  $E_f$  with  $c_f(e') \doteq f(e)$ .

### Unblocking example

flow  $f$ :



residual network  $N_f$ :



## Path Augmentation

### Schema

1. Start with  $f \equiv 0$ .
2. While there is an  $st$ -path in  $N_f$ 
  - ▶ Pick such a path  $P$ .
  - ▶  $f \leftarrow f +$  flow along  $P$  of value  $\min_{e \in P} (c_f(e))$
3. Return  $f$ .

### Soundness

If the algorithm produces an output, it is correct.

- ▶  $f$  always is a valid flow.
- ▶ If there is no  $st$ -path in  $N_f$  then  $\nu(f)$  is maximized.

### Termination

## Soundness

### Theorem

The following are equivalent:

- (1)  $f$  has maximum value.
- (2) There is no  $st$ -path in  $N_f$ .
- (3)  $\nu(f)$  equals the capacity of some  $st$ -cut.

### Proof

- (1)  $\Rightarrow$  (2) Contrapositive follows by path augmentation.
- (2)  $\Rightarrow$  (3) Next slide.
- (3)  $\Rightarrow$  (1) Follows from weak duality.

Corollary: Strong duality

$$\max_{\text{flow } f} \nu(f) = \min_{\text{st-cut } (S, T)} c(S, T)$$

## Construction of Minimum Cut

Suppose that there is no  $st$ -path in  $N_f$ .

$$S \doteq \{v \in V : \text{there exists an } sv\text{-path in } N_f\}.$$

- ▶  $(S, T)$  with  $T \doteq V \setminus S$  is an  $st$ -cut in  $N$ .
  - $s \in S$
  - $t \in T$
- ▶  $(\forall e \in E \cap S \times T) f(e) = c(e)$ 
  - Consider  $e = (u, v) \in E$  with  $u \in S$  and  $f(e) < c(e)$ .
  - Then  $(u, v) \in E_f$  so  $v \in S$ .
- ▶  $(\forall e \in E \cap T \times S) f(e) = 0$ 
  - Consider  $e = (u, v) \in E$  with  $v \in S$  and  $f(e) > 0$ .
  - Then  $e' \doteq (v, u) \in E_f$  so  $u \in S$ .
- ▶  $\therefore (S, T)$  is an  $st$ -cut with  $c(S, T) = \nu(f)$ .
- ▶ By weak duality  $\nu(f)$  is maximized and  $c(S, T)$  minimized.
- ▶ Construction of  $(S, T)$  from  $f$  runs in linear time.

## Path Augmentation - recap

### Scheme

1. Start with  $f \equiv 0$ .
2. While there is an  $st$ -path in  $N_f$ 
  - ▶ Pick such a path  $P$ .
  - ▶  $f \leftarrow f +$  flow along  $P$  of value  $\min_{e \in P} (c_f(e))$
3. Return  $f$ .

### Soundness

If the algorithm produces an output, it is correct.

- ▶  $f$  always is a valid flow.
- ▶ If there is no  $st$ -path in  $N_f$  then  $\nu(f)$  is maximized.

### Termination

## Termination and Running Time

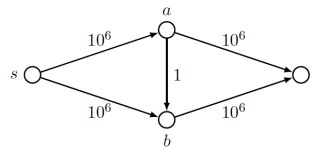
- ▶ Depend on the choice of augmenting path.
- ▶ There exist instances and choices without termination.

### Integral capacities

If all capacities are integral, termination is guaranteed no matter how augmenting paths are picked.

- ▶ # augmentations  $\leq F \doteq \max_{\text{flow } f} \nu(f)$
  - ▶ If each augmenting path is picked using linear-time graph traversal, running time is  $O((n+m) \cdot F)$ .
  - ▶ Running time bounded by polynomial in:
    - size parameters ( $n$  and  $m$ )
    - value of numbers involved (capacities):  $F \leq \sum_{e \in E} c(e)$
- Referred to as *pseudopolynomial* running time.

## Slow convergence



### Bad choices

Every augmentation through edge between  $a$  and  $b$ .

### Good choices

- ▶ Augmenting path of maximum residual capacity
- ▶ Augmenting path with smallest number of edges

## Better Algorithms

### Augmentation along path of maximum residual capacity

- ▶ # augmentations =  $O(m \cdot \log F)$  for integral instances
- ▶ Finding one path:  $O(n+m)$  time
- ▶ Overall running time:  $O((n+m)m \cdot \log F)$
- ▶ Running time is *polynomial*: bounded by polynomial in size parameters and *bitlength* of numbers involved.

### Augmentation along path with smallest number of edges

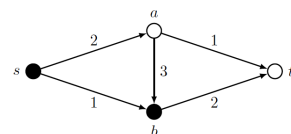
- ▶ # augmentations =  $O(nm)$  for all instances
- ▶ Finding one path:  $O(n+m)$  time (BFS)
- ▶ Overall running time:  $O(nm(n+m))$
- ▶ Running time is *strongly polynomial*: bounded by polynomial in size parameters only.

Other approaches:  $O(nm)$  time for all instances

## Recap - notions

### Network

- ▶ digraph  $(V, E)$
- ▶ edge capacities  $c : E \rightarrow [0, \infty)$
- ▶ source  $s \in V$  with indegree 0
- ▶ sink  $t \in V$  with outdegree 0



### Flow

A mapping  $f : E \rightarrow [0, \infty)$  satisfying

- ▶ [capacity constraints]  $(\forall e \in E) f(e) \leq c(e)$
- ▶ [conservation constraints]  $(\forall v \in V \setminus \{s, t\}) f_{\text{in}}(v) = f_{\text{out}}(v)$   
 where  $f_{\text{in}}(v) \doteq \sum_{u \in V: e \doteq (u,v) \in E} f(e)$  and  $f_{\text{out}}(v) \doteq \sum_{u \in V: e \doteq (v,u) \in E} f(e)$ .

### st-Cut

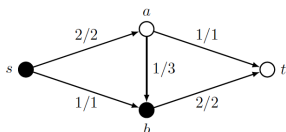
A partition  $(S, T)$  of  $V$  such that  $s \in S$  and  $t \in T$ .



## Recap - notions

### Integral network

- ▶ digraph  $(V, E)$
- ▶ edge capacities  $c : E \rightarrow \mathbb{N}$
- ▶ source  $s \in V$  with indegree 0
- ▶ sink  $t \in V$  with outdegree 0



### Integral flow

A mapping  $f : E \rightarrow \mathbb{N}$  satisfying

- ▶ [capacity constraints]  $(\forall e \in E) f(e) \leq c(e)$
- ▶ [conservation constraints]  $(\forall v \in V \setminus \{s, t\}) f_{\text{in}}(v) = f_{\text{out}}(v)$   
 where  $f_{\text{in}}(v) \doteq \sum_{u \in V: e \doteq (u,v) \in E} f(e)$  and  $f_{\text{out}}(v) \doteq \sum_{u \in V: e \doteq (v,u) \in E} f(e)$ .

### st-Cut

A partition  $(S, T)$  of  $V$  such that  $s \in S$  and  $t \in T$ .

## Bipartite Matching

### Definition

A matching  $M$  in a graph  $G = (V, E)$  is a subset  $M \subseteq E$  such that each  $v \in V$  appears in at most one  $e \in M$ .

### Computational problem

**Input:** bipartite graph  $G = (V, E)$  with bipartition  $(L, R)$ :  
 $E \subseteq L \times R$  where  $L \cap R = \emptyset$

**Output:** matching  $M$  such that  $|M|$  is maximized

### Reduction to max flow

- ▶ Model  $e \in M$  as one unit of flow through  $e$ .
- ▶ Integral network  $N$  such that

$$\begin{array}{ccc} \text{matching } M \text{ in } G & \xleftrightarrow{\text{bijection}} & \text{integral flow } f \text{ in } N \\ |M| & = & \nu(f) \end{array}$$

- ▶ Resulting algorithm runs in time  $O(nm)$ .

## Bipartite Matching – duality

### Capacity of a cut in $N$

- ▶  $c(S, T)$  is finite iff  $E \cap S \times T = \emptyset$   
iff  $G(L \cap S) \subseteq R \cap S$ , (1)  
where  $G(A) \doteq \{v : (\exists u \in A)(u, v) \in E\}$ .
- ▶ If  $c(S, T)$  is finite then  $c(S, T) = |L \cap T| + |R \cap S|$ . (2)

### Matching obstacle

- ▶ Suppose not every  $u \in L$  can be matched.
- ▶  $\nu(f) < |L|$  for every flow  $f$  in  $N$ .
- ▶  $c(S, T) < |L|$  for every min cut  $(S, T)$  in  $N$ . (3)
- ▶ Consider  $A \doteq L \cap S$ .  
 $|G(A)| \stackrel{(1)}{\leq} |R \cap S| \stackrel{(2)}{=} c(S, T) - |L \cap T| \stackrel{(3)}{<} |L| - |L \cap T| = |L \cap S| = |A|$
- ▶ Any  $A \subseteq L$  with  $|G(A)| < |A|$  is obstacle for matching all of  $L$ .

## Marriage theorem

### Definition

A matching  $M$  in  $G = (V, E)$  is perfect if every  $v \in V$  appears in some  $e \in M$ .

### Marriage theorem [Hall]

A bipartite graph  $G = (V, E)$  with bipartition  $(L, R)$  where  $|L| = |R|$  has a perfect matching iff

$$(\forall A \subseteq L) |G(A)| \geq |A|.$$

### Proof

$\Rightarrow$ : Follows from definition of perfect matching.

$\Leftarrow$ : Contrapositive follows from matching obstacle construction.

## Edge-Disjoint Paths

### Computational problem

Input: digraph  $G = (V, E)$ ;  $s, t \in V$

Output: set  $C$  of edge-disjoint  $st$ -paths in  $G$  such that  $|C|$  is maximized

### Reduction to max flow

- ▶ Model each path from  $s$  to  $t$  as one unit of flow.
- ▶ Integral network  $N \doteq (V, E \setminus (V \times \{s\} \cup \{t\} \times V), c \equiv 1, s, t)$   
set  $C$  of edge-disjoint  $st$ -paths in  $G \iff$  integral flow  $f$  in  $N$   
 $|C| = \nu(f)$
- ▶ Resulting algorithm runs in time  $O(nm)$ .

## Edge-Disjoint Paths – duality

### Capacity of a cut in $N$

- ▶  $c(S, T) = |E \cap S \times T|$
- ▶ Removing  $E \cap S \times T$  from  $G$  ensures no  $st$ -path remains.

### Edge connectivity duality [Menger]

The maximum number of edge-disjoint  $st$ -paths equals the minimum number of edges to be removed so no  $st$ -path remains.

### Proof

- ▶  $\ell \doteq$  max number of edge-disjoint  $st$ -paths
- ▶  $r \doteq$  min number of edges to be removed so no  $st$ -path
- ▶  $\max_{\text{flow } f} \nu(f) = \ell \leq r \leq \min_{\text{st-cut } (S, T)} (c(S, T))$
- ▶ By duality, LHS = RHS so  $\ell = r$ .

## Survey Design

### Computational problem

Input:  $n$  customers  $i \in [n]$

$m$  products  $j \in [m]$

$S_i \subseteq [m]$ : products that customer  $i \in [n]$  can survey

$c_i \in \mathbb{N}$ : max number of surveys for customer  $i \in [n]$

$p_j \in \mathbb{N}$ : min number of surveys of product  $j \in [m]$

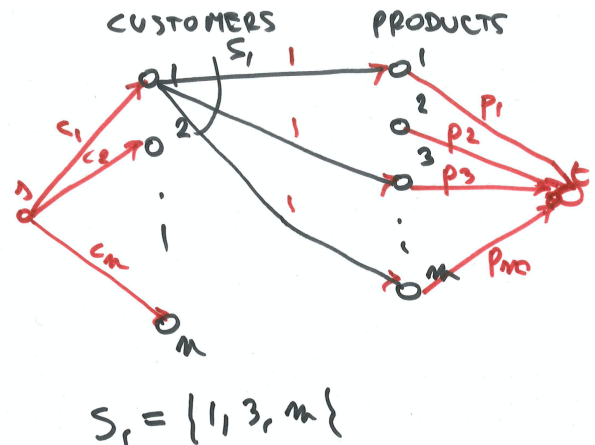
Output: set  $D \subseteq [n] \times [m]$  such that

- $(\forall (i, j) \in D) j \in S_i$
- $(\forall i \in [n]) |\{j \in [m] : (i, j) \in D\}| \leq c_i$
- $(\forall j \in [m]) |\{i \in [n] : (i, j) \in D\}| \geq p_j$

### Model

Represent each  $(i, j) \in D$  as a unit of flow that passes through customer  $i$  and product  $j$ .

## Survey Design – reduction to max flow



## Survey Design – analysis

- ▶ Minimal survey  $D$  satisfies

$$(\forall j \in [m]) |\{i \in [n] : (i, j) \in D\}| = p_j.$$

- ▶ Integral network  $N$  such that

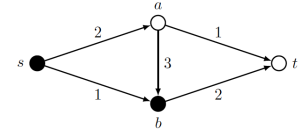
$$\begin{aligned} \text{minimal survey } D &\overset{\text{bijection}}{\longleftrightarrow} \text{integral flow } f \text{ with } \nu(f) = \sum_{j \in [m]} p_j \\ D &= \text{middle edges that carry flow} \end{aligned}$$

- ▶ Resulting algorithm runs in time  $O((n+m)(n+m + \sum_{i \in [n]} |S_i|))$ .

## Recap

### Network

- ▶ digraph  $(V, E)$
- ▶ edge capacities  $c : E \rightarrow [0, \infty)$
- ▶ source  $s \in V$  with indegree 0
- ▶ sink  $t \in V$  with outdegree 0



### $st$ -Cut

A partition  $(S, T)$  of  $V$  such that  $s \in S$  and  $t \in T$ .

### Min cut problem

**Input:** network  $N = (V, E, c, s, t)$

**Output:**  $st$ -cut  $(S, T)$  such that  $c(S, T) \doteq \sum_{e \in S \times T} c(e)$  is minimized

**Complexity:** time  $O(nm)$



## Image segmentation

### Computational problem

**Input:** grid of pixels  $i \in [n]$

$f_i \in [0, \infty)$ : "likelihood" that  $i$  is foreground

$b_i \in [0, \infty)$ : "likelihood" that  $i$  is background

$c \in [0, \infty)$ : penalty for separating neighboring pixels

**Output:** partition of  $[n]$  into foreground  $F$  and background  $B$  maximizing

$$\sum_{i \in F} f_i + \sum_{j \in B} b_j - c \cdot |\{(i, j) \in F \times B : i \sim j\}|$$

### Model

- ▶ Vertex for each pixel, source  $s$ , sink  $t$
- ▶  $S = F \cup \{s\}$  and  $T = B \cup \{t\}$

## Image Segmentation – rewriting objective

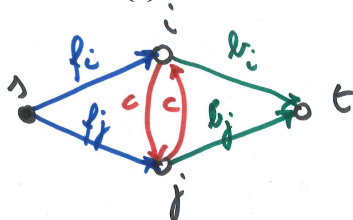
$$\begin{aligned} \max_{F, B} &\left( \sum_{i \in F} f_i + \sum_{j \in B} b_j - \sum_{(i, j) \in F \times B} c \right) \\ &= - \min_{F, B} \left( \sum_{(i, j) \in F \times B} c - \sum_{i \in F} f_i - \sum_{j \in B} b_j \right) \\ &= - \min_{F, B} \left( \sum_{(i, j) \in F \times B} c - \sum_{i \in [n]} f_i + \sum_{i \in B} f_i - \sum_{j \in [n]} b_j + \sum_{j \in F} b_j \right) \\ &= \sum_{i \in [n]} f_i + \sum_{j \in [n]} b_j - \min_{F, B} \left( \sum_{(i, j) \in F \times B} c + \sum_{i \in B} f_i + \sum_{j \in F} b_j \right) \end{aligned}$$

## Image Segmentation – reduction to min cut

- ▶ Need to find partition of  $[n]$  into  $F$  and  $B$  minimizing

$$(*) \doteq \sum_{(i, j) \in F \times B} c + \sum_{i \in B} f_i + \sum_{j \in F} b_j.$$

- ▶ Construct network  $N$  such that  $(*) = c(S, T)$  where  $S = F \cup \{s\}$  and  $T = B \cup \{t\}$ .



- ▶ Resulting algorithm runs in time  $O(n^2)$ .

## Project Selection

### Computational problem

**Input:**  $n$  projects  $i \in [n]$

$m$  tools  $j \in [m]$

$T_i \subseteq [m]$ : tools needed to realize project  $i \in [n]$

$v_i \in [0, \infty)$ : value of project  $i \in [n]$  if realized

$c_j \in [0, \infty)$ : one-time cost of tool  $j \in [m]$  if bought

**Output:** Set of projects  $I \subseteq [n]$  to realize and set of tools

$J \subseteq [m]$  to buy maximizing  $\sum_{i \in I} v_i - \sum_{j \in J} c_j$

such that  $(\forall i \in I) T_i \subseteq J$ .

### Model

- ▶ Vertex for each project  $i \in [n]$  & tool  $j \in [m]$ ; source  $s$ , sink  $t$
- ▶ Project  $i \in [n]$  is realized iff  $i \in S$ .
- ▶ Side of  $st$ -cut determines whether tool  $j \in [m]$  is bought.

### Project Selection – rewriting objective

$$\begin{aligned}
 & \max_{I, J} \left( \sum_{i \in I} v_i - \sum_{j \in J} c_j \right) \\
 &= - \min_{I, J} \left( \sum_{j \in J} c_j - \sum_{i \in I} v_i \right) \\
 &= - \min_{I, J} \left( \sum_{j \in J} c_j - \sum_{i \in [n]} v_i + \sum_{i \in [n] \setminus I} v_i \right) \\
 &= \sum_{i \in [n]} v_i - \min_{I, J} \left( \sum_{j \in J} c_j + \sum_{i \in [n] \setminus I} v_i \right)
 \end{aligned}$$

### Project Selection – reduction to min cut

- ▶ Need to find  $I \subseteq [n]$  and  $J \subseteq [m]$  with  $(\forall i \in I) T_i \subseteq J$  minimizing

$$(*) \doteq \sum_{j \in J} c_j + \sum_{i \in [n] \setminus I} v_i.$$

- ▶ Construct network  $N$  such that  $(*) = c(S, T)$  where  $I$  are the projects in  $S$ .
- ▶ Let  $J$  be the tools in  $S$ .
- ▶ Enforce condition  $(\forall i \in I) T_i \subseteq J$  by including edges  $(i, j)$  with  $c(i, j) = \infty$  for each  $i \in [n]$  and  $j \in T_i$
- ▶ Resulting algorithm runs in time  $O((n+m)(n+m + \sum_{i \in [n]} |T_i|))$ .

### Project Selection – reduction to min cut

