

## Problem definition:

- ▶  $n$  Boolean variables
- ▶  $m$  clauses  $C_1, \dots, C_m$ . For example

$$C_7 = x_3 \vee \bar{x}_5 \vee \bar{x}_9$$

- ▶ Non-negative weight  $w_j$  for each clause  $C_j$ .
- ▶ Find an assignment of true/false to the variables such that the total weight of clauses that are **satisfied** is maximum.

## Terminology:

- ▶ A variable  $x_i$  and its negation  $\bar{x}_i$  are called **literals**.
- ▶ Hence, each clause consists of a set of literals (i.e., no duplications:  $x_i \vee x_i \vee \bar{x}_j$  is **not** a clause).
- ▶ We assume a clause does not contain  $x_i$  and  $\bar{x}_i$  for any  $i$ .
- ▶  $x_i$  is called a **positive literal** while the negation  $\bar{x}_i$  is called a **negative literal**.
- ▶ For a given clause  $C_j$  the number of its literals is called its **length** or **size** and denoted with  $\ell_j$ .
- ▶ Clauses of length one are called **unit clauses**.

# MAXSAT: Flipping Coins

Set each  $x_i$  independently to **true** with probability  $\frac{1}{2}$  (and, hence, to **false** with probability  $\frac{1}{2}$ , as well).

Define random variable  $X_j$  with

$$X_j = \begin{cases} 1 & \text{if } C_j \text{ satisfied} \\ 0 & \text{otw.} \end{cases}$$

Then the total weight  $W$  of satisfied clauses is given by

$$W = \sum_j w_j X_j$$

$$\begin{aligned} E[W] &= \sum_j w_j E[X_j] \\ &= \sum_j w_j \Pr[C_j \text{ is satisfied}] \\ &= \sum_j w_j \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \\ &\geq \frac{1}{2} \sum_j w_j \\ &\geq \frac{1}{2} \text{OPT} \end{aligned}$$

## MAXSAT: LP formulation

- ▶ Let for a clause  $C_j$ ,  $P_j$  be the set of positive literals and  $N_j$  the set of negative literals.

$$C_j = \bigvee_{i \in P_j} x_i \vee \bigvee_{i \in N_j} \bar{x}_i$$

$$\begin{array}{ll} \max & \sum_j w_j z_j \\ \text{s.t.} & \forall j \quad \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \\ & \forall i \quad y_i \in \{0, 1\} \\ & \forall j \quad z_j \leq 1 \end{array}$$

# MAXSAT: Randomized Rounding

Set each  $x_i$  independently to **true** with probability  $y_i$  (and, hence, to **false** with probability  $(1 - y_i)$ ).

## Lemma 2 (Geometric Mean $\leq$ Arithmetic Mean)

For any nonnegative  $a_1, \dots, a_k$

$$\left( \prod_{i=1}^k a_i \right)^{1/k} \leq \frac{1}{k} \sum_{i=1}^k a_i$$



### Definition 3

A function  $f$  on an interval  $I$  is **concave** if for any two points  $s$  and  $r$  from  $I$  and any  $\lambda \in [0, 1]$  we have

$$f(\lambda s + (1 - \lambda)r) \geq \lambda f(s) + (1 - \lambda)f(r)$$

### Lemma 4

Let  $f$  be a concave function on the interval  $[0, 1]$ , with  $f(0) = a$  and  $f(1) = a + b$ . Then

$$\begin{aligned} f(\lambda) &= f((1 - \lambda)0 + \lambda 1) \\ &\geq (1 - \lambda)f(0) + \lambda f(1) \\ &= a + \lambda b \end{aligned}$$

for  $\lambda \in [0, 1]$ .

$$\begin{aligned}
\Pr[C_j \text{ not satisfied}] &= \prod_{i \in P_j} (1 - y_i) \prod_{i \in N_j} y_i \\
&\leq \left[ \frac{1}{\ell_j} \left( \sum_{i \in P_j} (1 - y_i) + \sum_{i \in N_j} y_i \right) \right]^{\ell_j} \\
&= \left[ 1 - \frac{1}{\ell_j} \left( \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \right) \right]^{\ell_j} \\
&\leq \left( 1 - \frac{z_j}{\ell_j} \right)^{\ell_j} .
\end{aligned}$$

The function  $f(z) = 1 - (1 - \frac{z}{\ell})^\ell$  is concave. Hence,

$$\begin{aligned}\Pr[C_j \text{ satisfied}] &\geq 1 - \left(1 - \frac{z_j}{\ell_j}\right)^{\ell_j} \\ &\geq \left[1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}\right] \cdot z_j .\end{aligned}$$

$f''(z) = -\frac{\ell-1}{\ell} \left[1 - \frac{z}{\ell}\right]^{\ell-2} \leq 0$  for  $z \in [0, 1]$ . Therefore,  $f$  is concave.

$$\begin{aligned} E[W] &= \sum_j w_j \Pr[C_j \text{ is satisfied}] \\ &\geq \sum_j w_j z_j \left[ 1 - \left( 1 - \frac{1}{\ell_j} \right)^{\ell_j} \right] \\ &\geq \left( 1 - \frac{1}{e} \right) \text{OPT} . \end{aligned}$$

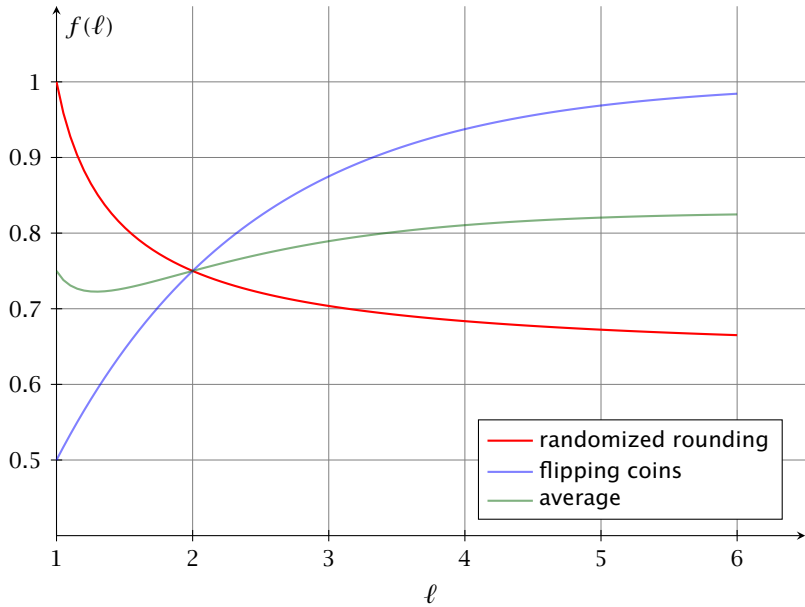
# MAXSAT: The better of two

## Theorem 5

*Choosing the better of the two solutions given by randomized rounding and coin flipping yields a  $\frac{3}{4}$ -approximation.*

Let  $W_1$  be the value of randomized rounding and  $W_2$  the value obtained by coin flipping.

$$\begin{aligned}
 & E[\max\{W_1, W_2\}] \\
 & \geq E[\frac{1}{2}W_1 + \frac{1}{2}W_2] \\
 & \geq \frac{1}{2} \sum_j w_j z_j \left[ 1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j} \right] + \frac{1}{2} \sum_j w_j \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \\
 & \geq \sum_j w_j z_j \underbrace{\left[ \frac{1}{2} \left(1 - \left(1 - \frac{1}{\ell_j}\right)^{\ell_j}\right) + \frac{1}{2} \left(1 - \left(\frac{1}{2}\right)^{\ell_j}\right) \right]}_{\geq \frac{3}{4} \text{ for all integers}} \\
 & \geq \frac{3}{4} \text{OPT}
 \end{aligned}$$



# MAXSAT: Nonlinear Randomized Rounding

So far we used **linear** randomized rounding, i.e., the probability that a variable is set to 1/true was exactly the value of the corresponding variable in the linear program.

We could define a function  $f : [0, 1] \rightarrow [0, 1]$  and set  $x_i$  to true with probability  $f(y_i)$ .



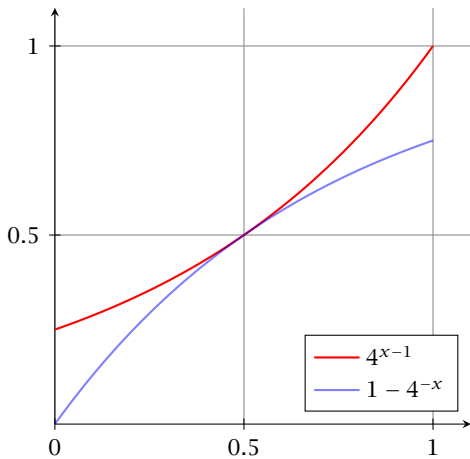
# MAXSAT: Nonlinear Randomized Rounding

Let  $f : [0, 1] \rightarrow [0, 1]$  be a function with

$$1 - 4^{-x} \leq f(x) \leq 4^{x-1}$$

## Theorem 6

*Rounding the LP-solution with a function  $f$  of the above form gives a  $\frac{3}{4}$ -approximation.*



$$\begin{aligned}
\Pr[C_j \text{ not satisfied}] &= \prod_{i \in P_j} (1 - f(y_i)) \prod_{i \in N_j} f(y_i) \\
&\leq \prod_{i \in P_j} 4^{-y_i} \prod_{i \in N_j} 4^{y_i - 1} \\
&= 4^{-(\sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i))} \\
&\leq 4^{-z_j}
\end{aligned}$$

The function  $g(z) = 1 - 4^{-z}$  is concave on  $[0, 1]$ . Hence,

$$\Pr[C_j \text{ satisfied}] \geq 1 - 4^{-z_j} \geq \frac{3}{4}z_j .$$

Therefore,

$$E[W] = \sum_j w_j \Pr[C_j \text{ satisfied}] \geq \frac{3}{4} \sum_j w_j z_j \geq \frac{3}{4} \text{OPT}$$

## Can we do better?

Not if we compare ourselves to the value of an optimum LP-solution.

### Definition 7 (Integrality Gap)

The integrality gap for an ILP is the worst-case ratio over all instances of the problem of the value of an optimal IP-solution to the value of an optimal solution to its linear programming relaxation.

Note that the integrality is less than one for maximization problems and larger than one for minimization problems (of course, equality is possible).

Note that an integrality gap only holds for one specific ILP formulation.

## Lemma 8

Our ILP-formulation for the MAXSAT problem has integrality gap at most  $\frac{3}{4}$ .

$$\begin{array}{ll} \max & \sum_j w_j z_j \\ \text{s.t.} & \forall j \quad \sum_{i \in P_j} y_i + \sum_{i \in N_j} (1 - y_i) \geq z_j \\ & \forall i \quad y_i \in \{0, 1\} \\ & \forall j \quad z_j \leq 1 \end{array}$$

Consider:  $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$

- ▶ any solution can satisfy at most 3 clauses
- ▶ we can set  $y_1 = y_2 = 1/2$  in the LP; this allows to set  $z_1 = z_2 = z_3 = z_4 = 1$
- ▶ hence, the LP has value 4.

## Facility Location

Given a set  $L$  of (possible) locations for placing facilities and a set  $D$  of customers together with cost functions  $s : D \times L \rightarrow \mathbb{R}^+$  and  $o : L \rightarrow \mathbb{R}^+$  find a set of facility locations  $F$  together with an assignment  $\phi : D \rightarrow F$  of customers to open facilities such that

$$\sum_{f \in F} o(f) + \sum_c s(c, \phi(c))$$

is minimized.

In the **metric facility location** problem we have

$$s(c, f) \leq s(c, f') + s(c', f) + s(c', f') .$$

## Integer Program

$$\begin{array}{ll} \min & \sum_{i \in F} f_i y_i + \sum_{i \in F} \sum_{j \in D} c_{ij} x_{ij} \\ \text{s.t.} & \forall j \in D \quad \sum_{i \in F} x_{ij} = 1 \\ & \forall i \in F, j \in D \quad x_{ij} \leq y_i \\ & \forall i \in F, j \in D \quad x_{ij} \in \{0, 1\} \\ & \forall i \in F \quad y_i \in \{0, 1\} \end{array}$$

As usual we get an LP by relaxing the integrality constraints.



## Dual Linear Program

$$\begin{array}{ll} \max & \sum_{j \in D} v_j \\ \text{s.t.} & \forall i \in F \quad \sum_{j \in D} w_{ij} \leq f_i \\ & \forall i \in F, j \in D \quad v_j - w_{ij} \leq c_{ij} \\ & \forall i \in F, j \in D \quad w_{ij} \geq 0 \end{array}$$

## Definition 9

Given an LP solution  $(x^*, y^*)$  we say that facility  $i$  neighbours client  $j$  if  $x_{ij} > 0$ . Let  $N(j) = \{i \in F : x_{ij}^* > 0\}$ .

### Lemma 10

*If  $(x^*, y^*)$  is an optimal solution to the facility location LP and  $(v^*, w^*)$  is an optimal dual solution, then  $x_{ij}^* > 0$  implies  $c_{ij} \leq v_j^*$ .*

Follows from slackness conditions.

Suppose we open set  $S \subseteq F$  of facilities s.t. for all clients we have  $S \cap N(j) \neq \emptyset$ .

Then every client  $j$  has a facility  $i$  s.t. assignment cost for this client is at most  $c_{ij} \leq v_j^*$ .

Hence, the total assignment cost is

$$\sum_j c_{i_j j} \leq \sum_j v_j^* \leq \text{OPT} ,$$

where  $i_j$  is the facility that client  $j$  is assigned to.

## Problem: Facility cost may be huge!

Suppose we can partition a subset  $F' \subseteq F$  of facilities into neighbour sets of some clients. I.e.

$$F' = \bigcup_k N(j_k)$$

where  $j_1, j_2, \dots$  form a subset of the clients.

Now in each set  $N(j_k)$  we open the **cheapest** facility. Call it  $f_{i_k}$ .

We have

$$f_{i_k} = f_{i_k} \sum_{i \in N(j_k)} x_{ij_k}^* \leq \sum_{i \in N(j_k)} f_i x_{ij_k}^* \leq \sum_{i \in N(j_k)} f_i y_i^* .$$

Summing over all  $k$  gives

$$\sum_k f_{i_k} \leq \sum_k \sum_{i \in N(j_k)} f_i y_i^* = \sum_{i \in F'} f_i y_i^* \leq \sum_{i \in F} f_i y_i^*$$

Facility cost is at most the facility cost in an optimum solution.

**Problem:** so far clients  $j_1, j_2, \dots$  have a neighboring facility.  
What about the others?

### Definition 11

Let  $N^2(j)$  denote all neighboring **clients** of the neighboring facilities of client  $j$ .

Note that  $N(j)$  is a set of facilities while  $N^2(j)$  is a set of clients.

### Algorithm 1 FacilityLocation

- 1:  $C \leftarrow D$  // unassigned clients
- 2:  $k \leftarrow 0$
- 3: **while**  $C \neq \emptyset$  **do**
- 4:      $k \leftarrow k + 1$
- 5:     choose  $j_k \in C$  that minimizes  $v_j^*$
- 6:     choose  $i_k \in N(j_k)$  as cheapest facility
- 7:     assign  $j_k$  and all unassigned clients in  $N^2(j_k)$  to  $i_k$
- 8:      $C \leftarrow C - \{j_k\} - N^2(j_k)$



Facility cost of this algorithm is at most OPT because the sets  $N(j_k)$  are disjoint.

### Total assignment cost:

- ▶ Fix  $k$ ; set  $j = j_k$  and  $i = i_k$ . We know that  $c_{ij} \leq v_j^*$ .
- ▶ Let  $\ell \in N^2(j)$  and  $h$  (one of) its neighbour(s) in  $N(j)$ .

$$c_{i\ell} \leq c_{ij} + c_{hj} + c_{h\ell} \leq v_j^* + v_j^* + v_\ell^* \leq 3v_\ell^*$$

Summing this over all facilities gives that the total assignment cost is at most  $3 \cdot \text{OPT}$ . Hence, we get a 4-approximation.

In the above analysis we use the inequality

$$\sum_{i \in F} f_i y_i^* \leq \text{OPT} .$$

We know something stronger namely

$$\sum_{i \in F} f_i y_i^* + \sum_{i \in F} \sum_{j \in D} c_{ij} x_{ij}^* \leq \text{OPT} .$$

## Observation:

- ▶ Suppose when choosing a client  $j_k$ , instead of opening the cheapest facility in its neighborhood we choose a random facility according to  $x_{ijk}^*$ .
- ▶ Then we incur connection cost

$$\sum_i c_{ijk} x_{ijk}^*$$

for client  $j_k$ . (In the previous algorithm we estimated this by  $v_{j_k}^*$ ).

- ▶ Define

$$C_j^* = \sum_i c_{ij} x_{ij}^*$$

to be the connection cost for client  $j$ .

## What will our facility cost be?

We only try to open a facility once (when it is in neighborhood of some  $j_k$ ). (recall that neighborhoods of different  $j'_k$ s are disjoint).

We open facility  $i$  with probability  $x_{ij_k} \leq y_i$  (in case it is in some neighborhood; otw. we open it with probability zero).

Hence, the expected facility cost is at most

$$\sum_{i \in F} f_i y_i .$$

### Algorithm 1 FacilityLocation

- 1:  $C \leftarrow D$  // unassigned clients
- 2:  $k \leftarrow 0$
- 3: **while**  $C \neq \emptyset$  **do**
- 4:      $k \leftarrow k + 1$
- 5:     choose  $j_k \in C$  that minimizes  $v_j^* + C_j^*$
- 6:     choose  $i_k \in N(j_k)$  according to probability  $x_{ij_k}$ .
- 7:     assign  $j_k$  and all unassigned clients in  $N^2(j_k)$  to  $i_k$
- 8:      $C \leftarrow C - \{j_k\} - N^2(j_k)$

## Total assignment cost:

- ▶ Fix  $k$ ; set  $j = j_k$ .
- ▶ Let  $\ell \in N^2(j)$  and  $h$  (one of) its neighbour(s) in  $N(j)$ .
- ▶ If we assign a client  $\ell$  to the same facility as  $i$  we pay at most

$$\sum_i c_{ij} x_{ijk}^* + c_{hj} + c_{h\ell} \leq C_j^* + v_j^* + v_\ell^* \leq C_\ell^* + 2v_\ell^*$$

Summing this over all clients gives that the total assignment cost is at most

$$\sum_j C_j^* + \sum_j 2v_j^* \leq \sum_j C_j^* + 2\text{OPT}$$

Hence, it is at most  $2\text{OPT}$  plus the total assignment cost in an optimum solution.

Adding the facility cost gives a 3-approximation.

## Lemma 12 (Chernoff Bounds)

Let  $X_1, \dots, X_n$  be  $n$  *independent* 0-1 random variables, not necessarily identically distributed. Then for  $X = \sum_{i=1}^n X_i$  and  $\mu = E[X]$ ,  $L \leq \mu \leq U$ , and  $\delta > 0$

$$\Pr[X \geq (1 + \delta)U] < \left( \frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^U,$$

and

$$\Pr[X \leq (1 - \delta)L] < \left( \frac{e^{-\delta}}{(1 - \delta)^{1-\delta}} \right)^L,$$

### Lemma 13

For  $0 \leq \delta \leq 1$  we have that

$$\left( \frac{e^\delta}{(1+\delta)^{1+\delta}} \right)^U \leq e^{-U\delta^2/3}$$

and

$$\left( \frac{e^{-\delta}}{(1-\delta)^{1-\delta}} \right)^L \leq e^{-L\delta^2/2}$$



# Integer Multicommodity Flows

- ▶ Given  $s_i$ - $t_i$  pairs in a graph.
- ▶ Connect each pair by a path such that not too many paths use any given edge.

$$\begin{array}{ll} \min & W \\ \text{s.t.} & \forall i \quad \sum_{p \in \mathcal{P}_i} x_p = 1 \\ & \sum_{p: e \in p} x_p \leq W \\ & x_p \in \{0, 1\} \end{array}$$

# Integer Multicommodity Flows

## Randomized Rounding:

For each  $i$  choose one path from the set  $\mathcal{P}_i$  at random according to the probability distribution given by the Linear Programming Solution.

### Theorem 14

*If  $W^* \geq c \ln n$  for some constant  $c$ , then with probability at least  $n^{-c/3}$  the total number of paths using any edge is at most  $W^* + \sqrt{cW^* \ln n}$ .*

# Integer Multicommodity Flows

Let  $X_e^i$  be a random variable that indicates whether the path for  $s_i$ - $t_i$  uses edge  $e$ .

Then the number of paths using edge  $e$  is  $Y_e = \sum_i X_e^i$ .

$$E[Y_e] = \sum_i \sum_{p \in \mathcal{P}_i; e \in p} x_p^* = \sum_{p: e \in p} x_p^* \leq W^*$$

# Integer Multicommodity Flows

Choose  $\delta = \sqrt{(c \ln n)/W^*}$ .

Then

$$\Pr[Y_e \geq (1 + \delta)W^*] < e^{-W^* \delta^2/3} = \frac{1}{n^{c/3}}$$

# Repetition: Primal Dual for Set Cover

## Primal Relaxation:

$$\begin{array}{ll} \min & \sum_{i=1}^k w_i x_i \\ \text{s.t.} & \forall u \in U \quad \sum_{i:u \in S_i} x_i \geq 1 \\ & \forall i \in \{1, \dots, k\} \quad x_i \geq 0 \end{array}$$

## Dual Formulation:

$$\begin{array}{ll} \max & \sum_{u \in U} y_u \\ \text{s.t.} & \forall i \in \{1, \dots, k\} \quad \sum_{u:u \in S_i} y_u \leq w_i \\ & y_u \geq 0 \end{array}$$

# Repetition: Primal Dual for Set Cover

## Algorithm:

- ▶ Start with  $y = 0$  (feasible dual solution).  
Start with  $x = 0$  (integral primal solution that may be infeasible).
- ▶ While  $x$  not feasible
  - ▶ Identify an element  $e$  that is not covered in current primal integral solution.
  - ▶ Increase dual variable  $y_e$  until a dual constraint becomes tight (maybe increase by 0!).
  - ▶ If this is the constraint for set  $S_j$  set  $x_j = 1$  (add this set to your solution).

# Repetition: Primal Dual for Set Cover

## Analysis:

- ▶ For every set  $S_j$  with  $x_j = 1$  we have

$$\sum_{e \in S_j} y_e = w_j$$

- ▶ Hence our cost is

$$\sum_j w_j = \sum_j \sum_{e \in S_j} y_e = \sum_e |\{j : e \in S_j\}| \cdot y_e \leq f \cdot \sum_e y_e \leq f \cdot \text{OPT}$$



Note that the constructed pair of primal and dual solution fulfills **primal slackness conditions**.

This means

$$x_j > 0 \Rightarrow \sum_{e \in S_j} y_e = w_j$$

If we would also fulfill **dual slackness conditions**

$$y_e > 0 \Rightarrow \sum_{j: e \in S_j} x_j = 1$$

then the solution would be **optimal!!!!**

We don't fulfill these constraint but we fulfill an approximate version:

$$y_e > 0 \Rightarrow 1 \leq \sum_{j:e \in S_j} x_j \leq f$$

This is sufficient to show that the solution is an  $f$ -approximation.

Suppose we have a primal/dual pair

$$\begin{array}{ll} \min & \sum_j c_j x_j \\ \text{s.t.} & \forall i \quad \sum_j a_{ij} x_j \geq b_i \\ & \forall j \quad x_j \geq 0 \end{array}$$

$$\begin{array}{ll} \max & \sum_i b_i y_i \\ \text{s.t.} & \forall j \quad \sum_i a_{ij} y_i \leq c_j \\ & \forall i \quad y_i \geq 0 \end{array}$$

and solutions that fulfill approximate slackness conditions:

$$x_j > 0 \Rightarrow \sum_i a_{ij} y_i \geq \frac{1}{\alpha} c_j$$

$$y_i > 0 \Rightarrow \sum_j a_{ij} x_j \leq \beta b_i$$

Then

right hand side of  $j$ -th  
dual constraint

$$\begin{aligned} \sum_j c_j x_j &\leq \alpha \sum_j \left( \sum_i a_{ij} y_i \right) x_j \\ \text{primal cost} &= \alpha \sum_i \left( \sum_j a_{ij} x_j \right) y_i \\ &\leq \alpha \beta \cdot \sum_i b_i y_i \\ &\quad \text{dual objective} \end{aligned}$$

# Feedback Vertex Set for Undirected Graphs

- ▶ Given a graph  $G = (V, E)$  and non-negative weights  $w_v \geq 0$  for vertex  $v \in V$ .
- ▶ Choose a minimum cost subset of vertices s.t. every cycle contains at least one vertex.

We can encode this as an instance of Set Cover

- ▶ Each vertex can be viewed as a set that contains some cycles.
- ▶ However, this encoding gives a Set Cover instance of non-polynomial size.
- ▶ The  $O(\log n)$ -approximation for Set Cover does not help us to get a good solution.

Let  $C$  denote the set of all cycles (where a cycle is identified by its set of vertices)

### Primal Relaxation:

$$\begin{array}{ll} \min & \sum_v w_v x_v \\ \text{s.t.} & \forall C \in \mathcal{C} \quad \sum_{v \in C} x_v \geq 1 \\ & \forall v \quad x_v \geq 0 \end{array}$$

### Dual Formulation:

$$\begin{array}{ll} \max & \sum_{C \in \mathcal{C}} y_C \\ \text{s.t.} & \forall v \in V \quad \sum_{C: v \in C} y_C \leq w_v \\ & \forall C \quad y_C \geq 0 \end{array}$$

If we perform the previous dual technique for Set Cover we get the following:

- ▶ Start with  $x = 0$  and  $y = 0$
- ▶ While there is a cycle  $C$  that is not covered (does not contain a chosen vertex).
  - ▶ Increase  $y_e$  until dual constraint for some vertex  $v$  becomes tight.
  - ▶ set  $x_v = 1$ .



Then

$$\begin{aligned}\sum_v w_v x_v &= \sum_v \sum_{C:v \in C} y_C x_v \\ &= \sum_{v \in S} \sum_{C:v \in C} y_C \\ &= \sum_C |S \cap C| \cdot y_C\end{aligned}$$

where  $S$  is the set of vertices we choose.

If every cycle is short we get a good approximation ratio, but this is unrealistic.

### Algorithm 1 FeedbackVertexSet

- 1:  $y \leftarrow 0$
- 2:  $x \leftarrow 0$
- 3: **while** exists cycle  $C$  in  $G$  **do**
- 4:     increase  $y_C$  until there is  $v \in C$  s.t.  $\sum_{C:v \in C} y_C = w_v$
- 5:      $x_v = 1$
- 6:     remove  $v$  from  $G$
- 7:     repeatedly remove vertices of degree 1 from  $G$

**Idea:**

Always choose a short cycle that is not covered. If we always find a cycle of length at most  $\alpha$  we get an  $\alpha$ -approximation.

**Observation:**

For any path  $P$  of vertices of degree 2 in  $G$  the algorithm chooses at most one vertex from  $P$ .

### Observation:

If we always choose a cycle for which the number of vertices of degree at least 3 is at most  $\alpha$  we get an  $\alpha$ -approximation.

### Theorem 15

*In any graph with no vertices of degree 1, there always exists a cycle that has at most  $\mathcal{O}(\log n)$  vertices of degree 3 or more. We can find such a cycle in linear time.*

This means we have

$$y_C > 0 \Rightarrow |S \cap C| \leq \mathcal{O}(\log n) .$$

# Primal Dual for Shortest Path

Given a graph  $G = (V, E)$  with two nodes  $s, t \in V$  and edge-weights  $c : E \rightarrow \mathbb{R}^+$  find a shortest path between  $s$  and  $t$  w.r.t. edge-weights  $c$ .

$$\begin{array}{ll} \min & \sum_e c(e)x_e \\ \text{s.t.} & \forall S \in \mathcal{S} \quad \sum_{e:\delta(S)} x_e \geq 1 \\ & \forall e \in E \quad x_e \in \{0, 1\} \end{array}$$

Here  $\delta(S)$  denotes the set of edges with exactly one end-point in  $S$ , and  $\mathcal{S} = \{S \subseteq V : s \in S, t \notin S\}$ .

# Primal Dual for Shortest Path

**The Dual:**

$$\begin{array}{ll} \max & \sum_S y_S \\ \text{s.t.} & \forall e \in E \quad \sum_{S:e \in \delta(S)} y_S \leq c(e) \\ & \forall S \in \mathcal{S} \quad y_S \geq 0 \end{array}$$

Here  $\delta(S)$  denotes the set of edges with exactly one end-point in  $S$ , and  $\mathcal{S} = \{S \subseteq V : s \in S, t \notin S\}$ .

# Primal Dual for Shortest Path

We can interpret the value  $y_S$  as the width of a moat surrounding the set  $S$ .

Each set can have its own moat but all moats must be disjoint.

An edge cannot be shorter than all the moats that it has to cross.

### Algorithm 1 PrimalDualShortestPath

- 1:  $\gamma \leftarrow 0$
- 2:  $F \leftarrow \emptyset$
- 3: **while** there is no  $s$ - $t$  path in  $(V, F)$  **do**
- 4:     Let  $C$  be the connected component of  $(V, F)$  containing  $s$
- 5:     Increase  $\gamma_C$  until there is an edge  $e' \in \delta(C)$  such that  $\sum_{S:e' \in \delta(S)} \gamma_S = c(e')$ .
- 6:      $F \leftarrow F \cup \{e'\}$
- 7: **Let**  $P$  **be an**  $s$ - $t$  **path in**  $(V, F)$
- 8: **return**  $P$



## Lemma 16

*At each point in time the set  $F$  forms a tree.*

### **Proof:**

- ▶ In each iteration we take the current connected component from  $(V, F)$  that contains  $s$  (call this component  $C$ ) and add some edge from  $\delta(C)$  to  $F$ .
- ▶ Since, at most one end-point of the new edge is in  $C$  the edge cannot close a cycle.

$$\begin{aligned}\sum_{e \in P} c(e) &= \sum_{e \in P} \sum_{S: e \in \delta(S)} y_S \\ &= \sum_{S: s \in S, t \notin S} |P \cap \delta(S)| \cdot y_S .\end{aligned}$$

If we can show that  $y_S > 0$  implies  $|P \cap \delta(S)| = 1$  gives

$$\sum_{e \in P} c(e) = \sum_S y_S \leq \text{OPT}$$

by weak duality.

Hence, we find a shortest path.

If  $S$  contains two edges from  $P$  then there must exist a subpath  $P'$  of  $P$  that starts and ends with a vertex from  $S$  (and all interior vertices are not in  $S$ ).

When we increased  $y_S$ ,  $S$  was a connected component of the set of edges  $F'$  that we had chosen till this point.

$F' \cup P'$  contains a cycle. Hence, also the final set of edges contains a cycle.

This is a contradiction.

## Steiner Forest Problem:

Given a graph  $G = (V, E)$ , together with source-target pairs  $s_i, t_i, i = 1, \dots, k$ , and a cost function  $c : E \rightarrow \mathbb{R}^+$  on the edges.

Find a subset  $F \subseteq E$  of the edges such that for every  $i \in \{1, \dots, k\}$  there is a path between  $s_i$  and  $t_i$  only using edges in  $F$ .

$$\begin{array}{ll} \min & \sum_e c(e)x_e \\ \text{s.t.} & \forall S \subseteq V : S \in S_i \text{ for some } i \quad \sum_{e \in \delta(S)} x_e \geq 1 \\ & \forall e \in E \quad x_e \in \{0, 1\} \end{array}$$

Here  $S_i$  contains all sets  $S$  such that  $s_i \in S$  and  $t_i \notin S$ .

$$\begin{array}{ll}
 \max & \sum_{S: \exists i \text{ s.t. } S \in S_i} \gamma_S \\
 \text{s.t.} & \forall e \in E \quad \sum_{S: e \in \delta(S)} \gamma_S \leq c(e) \\
 & \gamma_S \geq 0
 \end{array}$$

The difference to the dual of the shortest path problem is that we have many more variables (sets for which we can generate a moat of non-zero width).

### Algorithm 1 FirstTry

- 1:  $\gamma \leftarrow 0$
- 2:  $F \leftarrow \emptyset$
- 3: **while** not all  $s_i-t_i$  pairs connected in  $F$  **do**
- 4:     Let  $C$  be some connected component of  $(V, F)$  such that  $|C \cap \{s_i, t_i\}| = 1$  for some  $i$ .
- 5:     Increase  $\gamma_C$  until there is an edge  $e' \in \delta(C)$  s.t.  
        $\sum_{S \in S_i: e' \in \delta(S)} \gamma_S = c_{e'}$
- 6:      $F \leftarrow F \cup \{e'\}$
- 7:     Let  $P_i$  be an  $s_i-t_i$  path in  $(V, F)$
- 8: **return**  $\bigcup_i P_i$

$$\sum_{e \in F} c(e) = \sum_{e \in F} \sum_{S: e \in \delta(S)} y_S = \sum_S |\delta(S) \cap F| \cdot y_S .$$

If we show that  $y_S > 0$  implies that  $|\delta(S) \cap F| \leq \alpha$  we are in good shape.

However, this is not true:

- ▶ Take a graph on  $k + 1$  vertices  $v_0, v_1, \dots, v_k$ .
- ▶ The  $i$ -th pair is  $v_0 - v_i$ .
- ▶ The first component  $C$  could be  $\{v_0\}$ .
- ▶ We only set  $y_{\{v_0\}} = 1$ . All other dual variables stay 0.
- ▶ The final set  $F$  contains all edges  $\{v_0, v_i\}$ ,  $i = 1, \dots, k$ .
- ▶  $y_{\{v_0\}} > 0$  but  $|\delta(\{v_0\}) \cap F| = k$ .

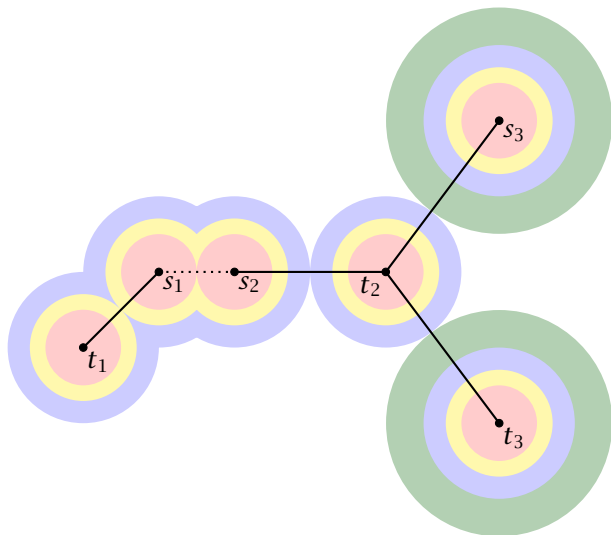
### Algorithm 1 SecondTry

```
1:  $\gamma \leftarrow 0; F \leftarrow \emptyset; \ell \leftarrow 0$ 
2: while not all  $s_i-t_i$  pairs connected in  $F$  do
3:    $\ell \leftarrow \ell + 1$ 
4:   Let  $C$  be set of all connected components  $C$  of  $(V, F)$ 
      such that  $|C \cap \{s_i, t_i\}| = 1$  for some  $i$ .
5:   Increase  $\gamma_C$  for all  $C \in C$  uniformly until for some edge
       $e_\ell \in \delta(C')$ ,  $C' \in C$  s.t.  $\sum_{S: e_\ell \in \delta(S)} \gamma_S = c_{e_\ell}$ 
6:    $F \leftarrow F \cup \{e_\ell\}$ 
7:  $F' \leftarrow F$ 
8: for  $k \leftarrow \ell$  downto 1 do // reverse deletion
9:   if  $F' - e_k$  is feasible solution then
10:    remove  $e_k$  from  $F'$ 
11: return  $F'$ 
```



The reverse deletion step is not strictly necessary this way. It would also be sufficient to simply delete all unnecessary edges in any order.

# Example



## Lemma 17

*For any  $C$  in any iteration of the algorithm*

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|C|$$

This means that the number of times a moat from  $C$  is crossed in the final solution is at most twice the number of moats.

**Proof:** later...

$$\sum_{e \in F'} c_e = \sum_{e \in F'} \sum_{S: e \in \delta(S)} y_S = \sum_S |F' \cap \delta(S)| \cdot y_S .$$

We want to show that

$$\sum_S |F' \cap \delta(S)| \cdot y_S \leq 2 \sum_S y_S$$

- ▶ In the  $i$ -th iteration the increase of the left-hand side is

$$\epsilon \sum_{C \in \mathcal{C}} |F' \cap \delta(C)|$$

and the increase of the right hand side is  $2\epsilon|C|$ .

- ▶ Hence, by the previous lemma the inequality holds after the iteration if it holds in the beginning of the iteration.

## Lemma 18

For any set of connected components  $C$  in any iteration of the algorithm

$$\sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \leq 2|C|$$

### Proof:

- ▶ At any point during the algorithm the set of edges forms a forest (why?).
- ▶ Fix iteration  $i$ .  $e_i$  is the set we add to  $F$ . Let  $F_i$  be the set of edges in  $F$  at the beginning of the iteration.
- ▶ Let  $H = F' - F_i$ .
- ▶ All edges in  $H$  are necessary for the solution.

- ▶ Contract all edges in  $F_i$  into single vertices  $V'$ .
- ▶ We can consider the forest  $H$  on the set of vertices  $V'$ .
- ▶ Let  $\deg(v)$  be the degree of a vertex  $v \in V'$  within this forest.
- ▶ Color a vertex  $v \in V'$  **red** if it corresponds to a component from  $C$  (an active component). Otw. color it blue. (Let  $B$  the set of blue vertices (with non-zero degree) and  $R$  the set of red vertices)
- ▶ We have

$$\sum_{v \in R} \deg(v) \geq \sum_{C \in \mathcal{C}} |\delta(C) \cap F'| \stackrel{?}{\leq} 2|C| = 2|R|$$

- ▶ Suppose that no node in  $B$  has degree one.
- ▶ Then

$$\begin{aligned}\sum_{v \in R} \deg(v) &= \sum_{v \in R \cup B} \deg(v) - \sum_{v \in B} \deg(v) \\ &\leq 2(|R| + |B|) - 2|B| = 2|R|\end{aligned}$$

- ▶ Every blue vertex with non-zero degree must have degree at least two.
  - ▶ Suppose not. The single edge connecting  $b \in B$  comes from  $H$ , and, hence, is necessary.
  - ▶ But this means that the cluster corresponding to  $b$  must separate a source-target pair.
  - ▶ But then it must be a red node.