**Approximation Algorithms** **Scribe: Matt Drescher**
**Instructor: Adrian Vetta** **Fall, 2006**

# Contents

## 1 intro

An approximation algorithm $A$ for an optimisation problem $P$ has the following properties when run on *any* instance $I$ of $P$.

1. it runs in polynomial time in input size. i.e. $poly(|I|)$
2. It always produces a feasible output $S$.
3. - $value(S) \geq \alpha \cdot opt$ for maximization problem $\alpha \leq 1$
   - $value(S) \leq \alpha \cdot opt$ for minimization problem $\alpha \geq 1$
4. $\alpha$ is the performance guarantee of the algorithm.
5. **NB** if $\alpha = 1$ the algorithm *always* gives an optimal solution.

Approximation algorithms are particularly useful for the class of *NP Hard* problems. (no poly-time algorithm exists unless $P = NP$) **Open** $P \neq NP$ ...

## 2 TSP

Given a complete graph $G := (V, E)$ with edge lengths $d_{ij}$ for $e = ij$. Find min length tour in $G$. We shall assume that $d_{ij}$s form a *metric.* i.e.

$$d_{ij} \leq d_{ik} + d_{kj} \forall k$$

. this is the triangle inequality.

**Problem 1 (TSP)** *Find min length Hamelton cycle in G mentioned above.*

### 2.1 Aside

Without the triangle inequality things are hopeless. Take any $G$, put weight 1 on edges, $n \cdot l$ on non edges. If $G$ has a Ham cycle then there is a tour of cost $n$. If $G$ does not have a Hamelton cycle then any tour is atleast $(n-1)1 + nl > nl$. Pick $l > \alpha$ this shows there is no approximation algorithm with factor $\alpha$.

### 2.2 Alg I

1. `find M.S.T`
2. `Double the edges of` $T$

3. `the degree of all vertices is now even` $\Rightarrow$ `Eulerian` $\Rightarrow$ `circuit that uses every edge exactly once.`
4. `Shortcut the circuit to give tour. (uses` $\triangle$ `)`

$$cost(S) \leq 2cost(MST)$$

**Lower Bound on opt**: $S^* - e$ is a spanning tree. Thus

$$cost(S^*) \geq cost(S^* - e) \geq cost(MST)$$

KEY POINT: We don't know the optimal value so we can only compare a solution to an estimate bound on the optimal.

$$cost(S) \leq 2 \cdot cost(MST) \leq 2opt$$

i.e. this is a 2 approximation algorithm.

**From now on:** When not specified non edges are shortest path costs.

### 2.3 Analysis

We have several questions to ask:

- Is our analysis tight w.r.t. our lower bound ? Are there examples where *opt* is $\alpha$ from lower bound? The above is tight. Let $G$ be a path with $d_{ij} = 1\ \forall ij \in E$. $cost(S^*) = 2(n-1) = 2cost(MST)$.
- Is the lower bound any good? Want it as close to opt as possible.
- How does it compare to the lower bound? example shows we are $2 \cdot LB$
- Is the algorithm better than the proof says?(compare solution to *Opt*) in our case: Take a $G$ that is two ladders each of size $n/2$ where each edge has cost 1. $Opt = n$ Our algorithm gives cost $2(n-1)$. i.e. solution is factor 2 from opt.
- Is there another algorithm? In our case yes. Christofides.

### 2.4 Christofides

Observe that a minimum cost tour can be partitioned into 2 perfect matchings(if even). i.e. a minimum cost perfect matching has cost at most $\frac{1}{2}opt$. i.e. $Opt \geq 2cost(min(PM))$. Any minimum cost perfect matching on an even number of vertices has cost $\leq opt$ by $\triangle$ inequality.

1. `Find MST`
2. `Find min cost PM on odd degree vertices.`
3. `Find Euler circuit and shortcut it.`

1. Polytime
2. produces feasible solution
3. $cost(S) \leq cost(MST) + cost(Matching) \leq opt + \frac{1}{2}opt = \frac{3}{2}opt$

**Open:Beat** $\frac{3}{2}$ we are hopeful there is a $\frac{4}{3}$ solution.

Algorithm can give solution $\frac{3}{2}$ from *opt*. Again take a ladder we get $c(S) = \frac{3}{2}n$. $LB = \frac{2}{3}(cost(MST) + cost(matching))$

## 3 Vertex Cover

**Problem 2 (Vertex Cover)** *Given $G := (V, E)$ find minimum number of vertices that intersect every edge.*

### 3.1 Alg

1. `Find any maximal matching` $M$
2. `Pick both end points of each edge in` $M$

Notice that for any maximal matching $opt \geq |M|$ as we must pick one vertex for each edge in the matching. So $2|M| \leq 2opt$.

- How good is $LB$? can be factor 2 off.
- How good is alg? Factor 2 off. Take $K_{n,n} = (A, B)$ $|M| = n$ $|S| = 2n$, $opt = n$.
- **Open: Beat 2 Hardness 1.36**

## 4 Max Clique

**Problem 3 (Max Clique)** *Given $G := (V, E)$ find max size clique*

This is $NP$ hard. It solves vertex cover.

### 4.1 Algorithm

1. `Pick 1 vertex`

Factor $\frac{1}{n}$ approximation. $n - approximation$.
**Open: Beat** $\frac{1}{n}$ Our algorithm is almost as good as it gets.

**Theorem 1** *No $O(n^{1-\epsilon})$ approx algorithm exists unless $NP = ZPP$*

# 5 Cut problems

Given a graph $G := (V, E)$ with edge weights (capacities) $w_e \geq 0$ we have to classical combinatorial optimization problems:

**Problem 4 (min s-t cut)** *minimum $t_1, t_2$ cut. Given $t_1, t_2 \in V$, find a cut $\delta(S)$ of minimum weight $\sum_{e \in \delta(S)} w_e$ s.t. $t_1 \in S, t_2 \in V - S$*

**Problem 5 (min cut)** *Find any cut of min weight.*

Note that *min $s - t$ cut $\Rightarrow$ min cut* (try all pairs). These problems can be generalized:

**Problem 6 (Multiway Cut)** *Given $T := \{t_1, ..., t_k\}$ a multiway cut is an edge set $F \subseteq E$ that disconnects all the $t_i$. Find a min weight multiway cut.*

This is $NP$ hard for $k \geq 3$.

**Problem 7 (k-Cut)** *A k-cut is a set of edges whose removal leaves $k$ components. Find min k-cut.*

Polytime for fixed $k$, $NP$ hard for general $k$.

## 5.1 Multiway cut Algorithm

We give a 2 approximation algorithm.

1. `For each` $t_i$ `find a min weight cut separating` $t_i$ `from all the other terminals` $T - t_i$
2. `Output` $\cup_{i=1}^{k-1} \delta(S_i)$ `where` $\delta(S_k)$ `is the heaviest of` $\delta(S_i)$

As usual we need to show

- **Feasibility**: Can not walk from $t_i$ to $t_j$ for any pair. Since WLOG $t_i \neq t_k$ so $\delta(S_i)$ separates $t_i$ from $t_j$.
- **Polytime**: Contract $T - t_i$ into 1 vertex $\hat{t_2}$, find a min $t_1 - \hat{t_2}$ *cut*.
- **approx guarantee**: Let $F^*$ be an optimal solution. So $G - F^*$ leaves $k$ components $T_1, ..., T_k$, $t_i \in T_i$. $w(F^*) = \sum_{e \in F^*} w_e = \frac{1}{2} \sum_{i=1}^{k} w(\delta(T_i))$. But $\delta(T_i)$ separates $t_i$ from the other terminals, so $w(\delta(T_i)) \geq w(\delta(S_i))$ . So $w(F) \leq \sum_{i=1}^{k-1} w(\delta(S_i)) \leq \sum_{i=1}^{k-1} w(\delta(T_i)) \leq (1 - \frac{1}{k}) \sum_{i=1}^{k} w(\delta(T_i)) = 2(1 - \frac{1}{k}) w(F^*)$ ■

## 5.2 k-Cut problem

We give an approximation algorithm based on Gomory-Hu trees.

**Definition 1 (Gomory-Hu)** *Given $G = (V, E)$ with weights $w_e \geq 0$, a $G - H$ tree $T^{GH}$ is a tree on $V$ such that ($T$ does not need to use edges in $E$)*

- *For $e = (u, v) \in T^{GH}$ we have that $\delta(S_e)$ is a min $u - v$ cut in $G$.*

It can be shown that

**Property 1 (*)** *For any $s, t \in V$ the min $s-t$ cut is the smallest weight cut given by the $s-t$ path in $T^{GH}$ i.e. min $w(\delta(S_e))e \in P_{st}$*

NOTE:*this is the fastest way to find all min $t_1 - t_2$ cuts $\forall$ pairs $t_1, t_2$*
for our purposes we only care that:

**Theorem 2** *G-H trees exist and can be found in polytime!*

and shall not spend time proving it.

## 5.3 Alg

1. `compute` $T^{GH}$ `for` $G$
2. `let` $e_1, ..., e_{k-1} \in T^{GH}$ `give the` $k-1$ `cheapest cuts in the tree:` $\delta(S_{e_i})$ `which we write as` $\delta(S_i)$
3. OUTPUT: $F = \cup_{i=1}^{k-1} \delta(S_i)$

**Theorem 3** *This is a 2 approximation algorithm*

**Proof.** Again let $F^*$ be an optimal solution with components $T_1, ..., T_k$. Let $\delta(T_i)$ be the heaviest of these. So

$$w(F^*) = \frac{1}{2} \sum_{i=1}^{k} w(\delta(T_i))$$

CLAIM: $w(\delta(S_i)) \leq w(\delta(T_i))$ **if cuts ordered by cost** $1 \leq i \leq k-1$ Contract the $T_i$, we have at least $k-1$ $GH$ edges between these $k$ vertices. Pick a tree $R$ from this contracted graph with respect to the $GH$ edges. $R \subseteq T^{GH}$ So by definition

$$\sum_{i=1}^{k} w(\delta(S_i)) \leq \sum_{e \in R} w(\delta(S_e))$$

Given $R$ orient the edges toward $T_k$. Let $r_j \in R$ be the edge leaving $T_j$. Let $r_j = (u_j, v_j) \in T^{GH}$ so $\delta(S_{r_j})$ is a min $u_j - v_j$ cut in $G$. So $w(\delta(S_{r_j})) \leq w(\delta(T_j))$ as $T_j$ separate $v_i - v_j$ too. So

$$\sum_{i=1}^{k-1} w(\delta(S_i)) \leq \sum_{e \in R} w(\delta(S_e)) = \sum_{j=1}^{k-1} w(\delta(S_{r_j}))$$

$$\leq \sum_{j=1}^{k-1} w(\delta(T_j)) \leq (1 - \frac{1}{k}) \sum_{j=1}^{k} w(\delta(T_j)) \leq 2(1 - \frac{1}{k}) opt$$

■

The analysis for both cut algorithms are tight. There are better algorithms: $\frac{3}{2}$ for both.

## 6 Background

- Decision problems: yes, no
- in $P$ if there is a polytime algorithm to solve it
- in $NP$ if a solution can be polynomialy certified that it is valid.
- A problem is in $CO-NP$ : $I$ is NO instance iff $I$ has polylength no certificate. e.g. validity: A boolean formula is valid if it is satisfied by all possible assignments. No certificate = an assignment for which formula is not valid.

We say a problem is well characterized if it has both yes and no certificates. Typically problems in $NP \cap CO-NP$ are in $P$. For example the matching problem: Yes $\rightarrow PM$, NO $\Rightarrow$ Tutte $f-factor$, odd components. **Open:Prime FActorization (is there a prime factor $\leq t$)**

1. Yes $prime \leq t$ with $t|n$
2. the prime factorization(in $P$)

## 7 Set Cover

We are given $n$ items $V := \{v_1, ..., v_k\}$, and $t$ sets $S_1, ..., S_t \subseteq V$ with cost $c(S_i)$.

**Problem 8 (Set Cover)** *Find min cost collection of sets $X$ that cover every item. i.e. min* $c(X) = \sum_{S_j \in X} c_j$

### 7.1 Greedy Algorithm

```
1. $X = \emptyset$
2. Pick set $S_j$ that covers uncovered items for the lowest average cost
    • avg cost: $c_j/S'_j$
    • $S'_j = S_j - X$
3. $X := X \cup S_j$
4. Repeat until $X$ covers all items
```

**Theorem 4** *Greedy is a $Olog(n)$ approximation algorithm.*

**Proof.** Let $X^* := \{S_1^*...S_i^*$ Greedy $:= \{S_1, ..., S_r\}$ Label elements $v_1, ..., v_n$ according to the order they are first covered by greedy. Consider $v_i$ There are at least $n-i+1$ uncovered items just before $v_i$ is covered. Let $opt := \sum_{S_j \in X^*} c(S^*)_j$ so some set in $X^*$ has average cost *at this point* of at most $\frac{opt}{n-i+1}$. Greedy is cheaper than this.

$$c(Greedy) = \sum_{j=1}^{r} c(S_j) = \sum_{i=1}^{n} (avg\ cost\ to\ cover\ item\ v_i\ by\ greedy)$$

$$\leq \sum_{i=1}^{n} \frac{opt}{n-i+1} = opt \sum_{i=1}^{n} \frac{1}{n-i+1} = opt \sum_{i=1}^{n} \frac{1}{i}$$
$$= H_n \cdot opt$$

Observe:$\lg(n) \leq H_n \leq \lg(n) + 1$ ($\int \frac{1}{x} dx = log(x)$) ■

How bad is **Greedy** really? Basically this analysis is tight.

**Problem 9 (hitting set)** *We have a set of elements $\{e_1, ..., e_k\}$ and a collection of sets $\{T_1, ..., T_n\}$. Each element $e_i$ has a cost $c_i$ and we want a min cost set of elements that intersects(hits) all the sets*

Examples

- minimum spanning tree picks edges that hits all cuts $\delta(S)$.
- Shortest $s-t$ path , edges, hit all $s-t$ cuts.
- Graph bipartization. Pick edges that intersect every odd cycle. Solves MAX-CUT which is $NP$ hard.
- Multicut $\{t_1, ..., t_k\}$ edges, all paths from $t_i$ to $t_j \forall i, j$
- vertex cover

Set cover = Hitting set.

## 7.2 Randomized Rounding

We have the following **IP** for $H.S.$

$$min \sum_{i=1}^{t} c_i x_i$$
$$s.t. \sum_{i:e_i \in T_j} x_i \geq 1 \forall T_j$$
$$x_i \in \{0,1\} \; i = 1...t$$

Take the **LP** relaxation

$$min \sum_{i=1}^{t} c_i x_i$$
$$s.t. \sum_{i:e_i \in T_j} x_i \geq 1 \forall T_j$$
$$0 \leq x_i \leq 1 \; i = 1...t$$

Linear programming is polytime in the number of constraints and the number of variables, $poly(t, n)$. So we get a fractional solution. 'Round' to integral solution. Suppose $max_j |T_j| = k$ So there exists $e_i \in T_j$ with $x_i \geq \frac{1}{k}$. Round all $x_i \geq \frac{1}{k}$ to 1, $x_i < \frac{1}{k}$ to 0. Worst case cost increases by factor $k$. Then we have a hitting set. This is an $O(k)$ approximation algorithm. e.g. vertex cover has $k = 2$.

Let $x$ be an optimal solution to $LP$ we set $e_i = 1$ with probability $x_i$, $P(e_i = 0) = 1 - x_i$. Expected cost of chosen set is $\sum_{i=1}^{t} c_i x_i = LP(opt) \leq opt$. But our output may not be a hitting set.

**Lemma 1 (Arithmetic-geometric means ineq)**

$$(\frac{\sum_{i=1}^{n} x_i}{n})^n \geq \prod_{i=1}^{n} x_i$$

**Proof.** WLOG $x_1 \leq ... \leq x_n$ if $x_1 < \frac{\sum x_i}{n}$, $x_n > \frac{\sum x_i}{n}$ $x_1 + \epsilon)(x_n - \epsilon) = x_1 x_n + \epsilon(x_n - x_1) + \epsilon^2 > x_1 x_n$ ■

**Lemma 2** *Probability that $T_j$ is hit is at least* $1 - \frac{1}{e}$

**Proof.**

$$\sum_{e_i \in T_j} x_i \geq 1$$

$$P(we\ miss\ T_j) = \prod_{i: e_i \in T_j} (1 - x_i)$$

$$\leq (1 - \frac{1}{|T_j|})^{|T_j|}$$

$$\leq e^{-1}$$

■

**Idea** Repeat the algorithm $p \cdot logn$ times. Take the union. By independence the probability we miss

$$T_j \leq \frac{1}{e}^{p \lg(n)} = \frac{1}{n^p}$$

the probability we miss *any* set is at most $n \cdot \frac{1}{n^p} = \frac{1}{n^{p-1}}$. So with high probability we find a hitting set of expected cost $O(opt \cdot \lg(n))$

Back to set cover. We can tighten the analysis a bit. Set $k = max|S_j|$

**Theorem 5** *Greedy is $O(\log(k))$ approximation algorithm for set cover*

**Proof.** $e^* = (S_1^*, ..., S_i^*)$ order $v_1, ..., v_n$ according to order covered by greedy. When $v_j \in S_i^*$ was first covered by $S_m$ say we could have chose $S_i^*$. So

$$\frac{c(S_m)}{|S'_m|} \leq \frac{c(S_i^*)}{n_i + 1 - j}$$

$n_i$ size $|S_i^*|$

$$c(X) \leq \sum_{i=1}^{l} \sum_{v_j \in S_i^*} \frac{c(S_i^*)}{n_i + 1 - j}$$

$$\leq \sum_{i=1}^{l} c(S_i^*) \sum_{j=1}^{n} \frac{1}{n_i + 1 - j}$$

$$(1 + H_k) \sum_{i=1}^{l} c(S_i^*)$$

$$\leq (1 + H_k)opt$$

**Theorem 6** *There exists ccl such that i and $P \neq NP$ there is no approximation algorithm better then $c\lg(n)$.*

## 8 Max Sat

Have a set $x_1, ..., x_n$ of boolean variables and clauses $C_1, ..., C_k$ or the form $C_1 = (x_1 \vee \bar{x_2} \vee \bar{x_7})$

**Problem 10** *SAT IS there a T/F assignment of the variables such that all clauses are satisfied?*

**Problem 11** *Max SAT how many clauses can we satisfy?*

### 8.1 Randomized Rounding Alg

1. `for each` $x_i$ `set` $x_i := T$ `with probability` $\frac{1}{2}$

If $C_j$ has $k$ literals then $P(C_j\ is\ satisfied) = 1 - \frac{1}{2^k}$ worse case $k = 1 \Rightarrow P(C_j\ is\ satisfied) = \frac{1}{2}$.

**Corollary 1** $\frac{1}{2}$ *approximation algorithm*

### 8.2 LP rounding

Set up an $IP$ with $y_i = 1(0)\ iff\ x_i = T(\bar{x}_i = T) \forall x_i$ and $\forall$ clauses $C_j \exists z_j : z_j = 1\ if\ C_j\ satisfied\ 0\ otherwise$

$$max \sum_j z_j$$

$$\sum_{x_i \in C_j^+} y_i + \sum_{x_i \in C_j^-} y_i \geq 1 \forall j$$

$$z_j, y_i \in \{0, 1\}$$

where $C_j^+$ is the set of variables in $C_j$ in uncomplemented form $x_i$ not $\bar{x}_i$.

Relax $0 \leq y_i \leq 1$, $0 \leq z_j \leq 1$ and solve $LP$ Set $x_i = T$ with probability $y_i$ otherwise false.

**Lemma 1** *If $C_j$ has $k$ literals then it is satisfied with probability at least*

$$(1 - (1 - \frac{1}{k})^k z_j \geq (1 - \frac{1}{e}) z_j$$

**Proof.** Assume WLOG $\bar{C_j} = \emptyset$. $C_j = X_1 \vee X_2 ... \vee X_k$. $C_j$ is unsatisfied with probability

$$\prod_{i=1}^{k}(1-y_i) \leq_{AGMI} (1-\frac{\sum_{i=1}^{k} y_i}{k})^k \leq (1-\frac{z_j}{k})^k$$

by the $LP$.

$$max \sum_j z_j$$

$$\sum_{x_i \in C_j} y_i \geq z_j^* \; \forall j$$

$$z_j, y_i \in \{0,1\}$$

it is satisfied with probability atleast

$$1-(1-\frac{z_j}{k})^k$$

$$\geq (1-(1-\frac{1}{k})^k)z \oplus$$

$$\geq (1-\frac{1}{e})z_j$$

to prove $\oplus$ $f''(x) \leq 0$

**Lemma 2** *let $f(x)$ be concave on $[0,1]$ with $f(0) \geq b$ and $f(1) \geq a+b$. The $f(x) \geq ax+b$ on $[0,1]$*

we have $f(z_j = (1-\frac{z_j}{k})^k$ convex $f''(x) \geq 0$,

- $f(0) = 1-(1-\frac{0}{k})^k = 0 = b$
- $f(1) = 1-(1-\frac{1}{k})^k = [1-(1-\frac{1}{k})^k_{:=a}] \cdot 1 + 0$

So $E(\#clauses\ set$

$$\geq \sum_j z_j(1-\frac{1}{e}) = (1-\frac{1}{e})\sum_j z_j \geq (1-\frac{1}{e})opt$$

Note that the bound gets worse as $k \to \infty$.

### 8.2.1 R3

1. `Run both algorithms and take the best one.`

**Theorem 7** *R3 is $\frac{3}{4}$ approximation algorithm.*

**Proof.** $E(max(N_1,N_2)) \geq E(\frac{N_1+N_2}{2}) =$

$$\frac{1}{2}\sum_{C_j} P(C_j\ sat\ by\ R_1) + P(C_j\ sat\ by\ R_2)$$

$$\geq \frac{1}{2}\sum_{C_j}(1-\frac{1}{2}^{k_j}) + (1-(1-\frac{1}{k_j})^{k_j})z_j$$

$$\geq \frac{1}{2}\sum_{C_j}(1-\frac{1}{2^{k_j}})z_j + (1-(1-\frac{1}{k_j})^{k_j})z_j$$

$$\frac{1}{2}\sum z_j(1-\frac{1}{2^{k_j}} + (1-(1-\frac{1}{k_j})^{k_j}))$$

- $k := 1\ \frac{1}{2}(1-\frac{1}{2}+1) = \frac{3}{2}$
- $k := 2\ \frac{1}{2}(\frac{3}{4}+\frac{3}{4}) = \frac{3}{4}$
- $k \geq 3 \geq \frac{1}{2}(\frac{7}{8}+(1-\frac{1}{e})) \geq \frac{3}{4}$

$\blacksquare$

### 8.3 non-linear randomized rounding

We dont have to round using the exact $LP$ values. e.g. suppose there is a function $f$ such that

$$1-\frac{1}{4^x} \leq f(x) \leq 4^{x-1}\ \forall x \in [0,1]$$

Round according to $f(y_i)$ rather than $y_i$. Take $C_j$

$$P(C_j\ is\ sat) = 1 - \prod_{x_j \in C_j^+}(1-f(y_i)) \prod_{x_j \in C_j^-} f(y_i)$$

$$\geq 1 - \prod_{C_j^+}(1-(1-\frac{1}{4^{y_i}})) \prod_{\bar{C}_j} 4^{y_i-1}$$

$$\geq 1 - \prod_{C_j^*} 4^{-y_i} \prod_{C_j^-} 4^{y_i-1}$$

$$= 1 - 4^{[-\sum_{x_i \in C_j^+} y_i - \sum_{x_i \in C_j^-}(1-y_i)]}$$

$$\geq 1 - 4^{-z_j} \geq \frac{3}{4}z_j$$

$\blacksquare$

now $4^{-x}$ is convex. $1-4^{-x}$ is concave.

$$1-4^{-0} = 0_{:=b} = 0\frac{3}{4}+0$$

$$1-4^{-1} = \frac{3}{4}_{:=a} = \frac{3}{4}\cdot 1 + 0$$

$$f(x) \geq \frac{3}{4}x$$

Can we do better than this $LP$ ? Can we do better with this $LP$ ?

$$x_1 \vee x_2, x_1 \vee \bar{x_2}, \bar{x_1} \vee x_2, \bar{x_1} \vee \bar{x_2}$$

$opt = 3$, $LP = 4$ $x_i = \frac{1}{2}$ $i = 1, 2$ $z_j = 1$ $j = 1, ..., 4$ Can do better using semidefinite programming upperbound get 0.78.
**Open: beat** 0.78 Interestingly **max(2 SAT)** is $NP - hard$ but $2 - SAT$ is polytime. EXERSIZE: solve $2 - sat$.

## 9 Min Congestion Flow

**Problem 12** *MCFP Given $G := (V, A)$ and source terminal pairs, $s_i, t_i$, $i \in \{1, ..., k\}$ We want to find a collection of $s_i - t_i$ paths such that the max congestion on any arc is minimized.*

- ***congestion:*** *the flow on arc $a$ is equal to the number of paths $P_i$ using $a$.*

We would like to use randomized rounding. So we need an $LP$ formulation.

$$min\ C$$

$$\sum_{a \in \delta^+(v)} f_a^i - \sum_{a \in \delta^-(v)} f_a^i = 0_{\neq si,ti}, 1_{=s_i}, -1_{=t_i}\ \forall commodity\ i, \forall v$$

$$\sum_{i=1}^{k} f_a^i \leq C \forall a$$

$$f_a^i \geq 0, \forall a, i$$

$$C \geq 0$$

We can solve the $LP$ in polytime, since there is a polynomial number of constraints and variables. What do we do with the $LP$ solution?

- If we round edges we dont get a flow.
- **Flow Decomposition:** Any flow $f^i$ of value 1 can be decomposed into at most $m$ paths of weights $\lambda_1^i, \lambda_2^i, ..., \lambda_m^i$ such that $\sum_{j=1}^{m} \lambda_j^i = 1$

**Proof.**

1. - Let $G^i$ be a graph with positive $f_a^i$. Find $s_i - t_i$ path $P$ in $G_i$ by BFS, DFS.
   - Let $\hat{a} = min_{a \in P} f_a^i$. Pick $P$ with $\lambda^P = f_{\hat{a}}^i$.
   - Let $\chi^P$ be the incident vector for $P$. $\chi_a^P = 1_{a \in P}, 0_{a \notin P}$. $f^i$ is still a flow of value $1 - \lambda^P$.
2. Repeat

At each stage the constraints are not violated. Note that $G^i$ now has at least 1 less arc. We repeat at least $m$ times therefore we obtain at least $m$ paths.

$$\sum_p \lambda_p^i = 1$$

Do this for all $k$ commodities. Let $P_i^+ :=$ the set of $s_i, t_i$ paths with positive flow. For each $i$

$$\sum_{p \in P_i^+} \lambda_p^i = 1$$

Use $\lambda_p^i$ as the probability to choose one path/commodity. i.e. choose $p_j \in P_j^+$ with probability $\lambda_{p_j}^i$. Expected Congestion on arc $a$ is

$$\sum_{i=1}^k \sum_{p_j^i \in P^{i+}, a \in p_j^i} P(p_j^i \textit{ is chosen})$$

$$= \sum_{i=1}^k \sum_{a \in p_j^i, p_j^i \in P_i^+} \lambda_j^i$$

$$= \sum_{i=1}^k f_a^i \leq C$$

this is just the expectation on one edge. We want to approximate $Cong = max_a \; cong(a)$. We want the probability that the maximum congestion is large to be small. To do this let $X_i^a$ be the event that path $P_i^*$ contains $a$. So the congestion on $a$ is $\sum_i X_i^a$

$$P(X_a^i = 1) = \sum_{a \in P_j^i, P_j^i \in P_i^{j+}} \lambda_j^i = p_a^i$$

**Theorem 8** *Chernoff Bounds Let $X_1, ..., X_k$ be independent bernouilli trials such that $P(X_i = 1) = p_i$, then if $X := \sum_i X_i$ we have*

$$P(X > (1+\delta)E(X)) \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{E(X)}$$

**Proof.**

$$P(X > (1+\delta)E(X)) = P(e^{tX} > e^{t(1+\delta)}E(X))$$

By Markov inequality

$$= P(e^{tX} > \frac{e^{t(1+\delta)E(X)}}{E(e^{tX})} \cdot E(e^{tX})) \leq \frac{E(e^{tX})}{e^{t(1+\delta)E(X)}}$$

now

$$E(e^{tX}) = E(e^{t \sum X_i}) = \prod_i E(e^{tX_i}) = \prod_i (1 - p_i + p_i e^t)$$

$$= \prod_i (1 + p_i(e^t - 1))$$

since $1 + x \leq e^x$,

$$\leq \prod_i e^{p_i(e^t-1)}$$

$$= e^{(e^t-1)\sum p_i} = e^{E(X)(e^t-1)}$$

So we have

$$P(X \geq (1+\delta)E(X)) \leq \frac{e^{(e^t-1)E(X)}}{e^{t(1+\delta)E(X)}}$$

Choose $t$ to minimize this probability $t = h(1+d)$

$$P(X > (1+\delta)E(X)) \leq (\frac{e^\delta}{(1+\delta)^{1+\delta}})^{E(X)}$$

■

Set $1 + \delta \geq 2e$, $X = \sum_{i=1}^k X_i^a$

$$P(X \geq 2eE(X)) \leq (\frac{e}{1+\delta})^{(1+\delta)E(X)} \leq \frac{1}{2^{(1+\delta)E(X)}}$$

Lets take $\delta$ such that $(1+\delta)\mu \geq 3\lg(n)$

$$P(X \geq (1+\delta)E(X)) \leq \frac{1}{n^3}$$

There exists at most $n^2$ arcs. So the probability any has congestion greater than $(1+\delta)E(X) = max(2eE(X), 3\lg(n)) \leq \frac{n^2}{n^3} = \frac{1}{n}$

So our 'approx' guarantee is $Cong \leq 2e \cdot opt + 3\lg(n)$. i.e. $O(\lg(n))$ approximation algorithm. But $O(1)$ approximation algorithm if $opt \geq \Omega(\lg(n))$

## 10 Shortest Superstring

**Problem 13 (Shortest Superstring)** *Given $n$ strings $s_1, ..., s_n$, find a **superstring** containing all the $s_i$ of minimum length.*

### 10.1 Applications

- Given pieces of $DNA$, what is original string
- Data compression

Without loss of generality we can assume $s_1...s_n$ are ordered in $S$, and that no $s_i \subseteq s_j$. Let $p_{12} := prefix(s_1, s_2) = s_1 - O_{12}$ where $O_{12} := overlap(s_1, s_2)$. So $opt = p_{12} + p_{23} + ... + p_{n,1} + O_{n,1}$, thus

$$opt \geq \sum_{i=1}^n p_{i,i+1}$$

How can we use this? We can find $p_{ij}$

- **Prefix Graph**
  - $G^*$ has vertices $s_1, ..., s_n$
  - we have arcs $(s_i, s_j)$ of length $p_{ij}$.
- So $G^*$ has a Hamelton cycle of cost at most *opt*.

But we can find a minimum cost *cycle cover.*

**Problem 14 (Cycle Cover)** *Find collection of disjoint cycles covering all the vertices of minimum cost.*

This is a matching problem. Solution has $deg^-(v) = deg^+(v) = 1\ \forall v$. Consider the bipartite graph $B := (V, V')$ and have an arc $ij'$ for each arc $ij$. Finding a min cost $\mathcal{PM}$ in poly time solves the problem.

So we run *cycle cover* and get cycles $C_1, ..., C_k$. Let $r_i \in C_i$ be a representive string of $C_i$. We break the cycles at their reps and then take the ordering given by the union. This solution has cost at most

$$\sum_{i=1}^{k} |C_i| + |r_i|$$

$$\leq opt + \sum_{i=1}^{k} |r_i|$$

So we need to bound $\sum |r_i|$. If the $|r_i$ are small this is easy, but the cycles could be small.

The key to bounding $\sum |r_i|$ is the following

**Lemma 3** *Take $C_i$ and $C_j$ with reps $r_i$, $r_j$ then $O_{ij} < |C_i| + |C_j|$*

Using this lemma we can prove

**Theorem 9** *The algorithm is a 4 approximation algorithm*

**Proof.** Observe that

$$opt \geq \sum_{i=1}^{k} |r_i| - \sum_{i=1}^{k-1} O_{i,i+1}$$

by the lemma

$$> \sum |r_j| - \sum_{i=1}^{k-1} (|C_i| + |C_{i+1}|)$$

$$\geq \sum |r_i| - 2 \sum_{i=1}^{k} |C_i|$$

So

$$\sum |r_i| \leq opt + 2 \sum |C_i| \leq 3opt$$

Our solution has cost at most

$$opt + \sum |r_i| \leq 4 \cdot opt$$

■

We now prove the lemma. We need some notation. Let $\gamma^N = \gamma \circ \gamma \circ \ldots \circ \gamma$ where $\circ$ denotes concatenation. **Claim:** If $t_1, ..., t_p$ are subsets of $\gamma^\infty$ then there is a cycle in the prefix graph of size $\gamma$ covering $t_1, ..., t_p$. **Proof.** Pick a starting point of $t_j$ on cycle of size $\gamma$ this proves the claim ■

**Proof.** (of lemma) Assume $O_{ij} \geq |C_i + |C_j|$ let $C_i = string\ \alpha$, $C_j = string\ \beta$ then $\alpha \subset \beta^\infty$, $\beta \subseteq \alpha^\infty$, $O_{ij} \subseteq \alpha^\infty$ $O_{ij} \subseteq \beta^\infty$. Without loss of generality $\beta \leq \alpha$ need $|\alpha| + |\beta| \leq O_{ij}$. Pick first $|\beta|$ elements in second $\alpha$. So $\alpha \circ \beta = \beta \circ \alpha$. Consequently $\alpha^n \circ \beta^n = \beta^n \circ \alpha^n$ (swap $\alpha\beta$ pairs in turn). THis means that the first $|\beta|^n$ entries of $\beta^n$ and $\alpha^n$ are the same! So $\alpha^\infty = \beta^\infty$. By the previous claim all strings in $C_j$ and $C_i$ are substrings of $\beta^\infty$. So we dont need the $\alpha$ cycle. i.e. we didnt have a minimum cost cycle cover. ■

- **Open:Find factor 2 approximation algorithm(best is 2.5)**
- In particular the Greedy alg that repeatedly merges 2 strings with largest overlap is widely used in bioinformatics. Is this a $2-approximation$?

## 11 The $k$- center problem

**Problem 15 (k-center)** *Given G and edge weights $d_e$, choose k vertices to $min_{S:|S|=k} max_{v \in V} d(v, S)$ where $d(v, S) := min_{u \in S} d(v, u)$.*

i.e. pick $k$ vertices such that all vertices are “near” some center.

- For general $d$ this is hopeless.

**Problem 16 (Dominating Set)** *given $G := (V, E)$ is the set of vertices $T \subseteq V$ such that all vertices in $V - T$ are adjacent to some vertex in $T$.*

This is *NP-hard*. Suppose we have an $\alpha$ approximation algorithm for $k$-center. Then we can solve dominating set! Take $G$, $d_e = 1\ \forall e \in E$, $d_e = L$ where $L > \alpha\ \forall e \notin E$. There exists a dominating set of size $k$ iff there exists a solution to $k$ center with value 1. So approximation algorithm gives the optimal solution if there exists a dominating set of size $k$.

**Corollary 2** *if d is a metric then there is no approximation algorithm with guarantee $2 - \epsilon$.*

**Proof.** set $L < 2 - \epsilon$. This satisfies the triangle inequality. ■
We can find a 2 approximation algorithm in this case:

1. let $s_1$ be any vertex
2. Given $S := \{s_1, s_2, ..., s_{i-1}\}$ $i \leq k$, let $s_i \in V - S$ be the vertex that maximize
$$max_{v \in V-S} d(v, S)$$
3. output $S := \{s_1, ..., s_k\}$ $max_{v \in V-S} d(v, S)$

**Theorem 10** *This is a 2 approximation algorithm*

**Proof.** Let $s_{k+1}$ be furthest from $S$. Our solution is $d(s_{k+1}, S)$. Note the $d(s_i, S)$ are decreasing. Consider $\{s_1, s_2, ..., s_k, s_{k+1}\}$ since we have only $k$ centers, and $k+1$ vertices, we must have $s_j, s_{j'}$ which are ‘served’ by the same center $x$. Thus by the triangle inequality $d(x, s_{j'}) \geq \frac{1}{2} d(s_{k+1}, S)$ ■

## 11.1 Asymmetric $k$-center

What if the graph is directed? Choose $S$ such that $|S| = k$ such that there exists a short dipath from some vertex in $S$ to any other vertex. Assume we satisfy the triangle inequality and therefore we can assume $G$ is complete graph by filling in shortest paths. $d_{ij} \leq d_{iv} + d_{vj}\ \forall v, i, j$ We can assume we know the optimal distal $R = min_{S:|S|=k} max_v d(S, v)$ There are only $n^2$ possible solutions( the edges). We try all $n^2$ until one works. ( By the triangle inequality: keep edges $\leq R$. If we have correct $R$ then there exists a dominating set (directed sense) of size $k$ in this graph.) Assume now all distances are equal to $1 (= R)$. i.e. just want to find stars(unweighted).

**Problem 17 (Combinatorial Problem)** *Given $G := (V, A)$ find $k$ directed stars that cover all vertices.*

this is a bicriteria result.

**Lemma 4** *If $G$ has a $[k, \cancel{R}_{=1}]$ solution then there exists a polytime algorithm to give a $[2k, \lg^*(n)\ /\ R]$ solution*

■

**Proof.** We can view this as a set cover problem. There are $n$ elements $v_1, ..., v_n$ and $n$ sets(stars) where $S_v = \{u : d(u, v) \leq 1\} = \{v\} + \Gamma^+(v)$ where $\Gamma^+$ means out neighbours . So we know there exists a solution of size $k$. We have "seen" that there exists an approximation algorithm giving solution $S$ of size $(k \cdot \lg(\frac{n}{k}))$ $(\frac{n}{k} = \frac{\#elements}{Opt})$. But suppose we cover $S_1$. If we do this then we cover all elements at cost at most 2. But $|S_1| < n$ so we find a solution of size $(k \lg \frac{k \lg(n/k)}{k}) = (k \lg\lg \frac{n}{k})$ Repeat $S_3$ $(k \lg\lg\lg \frac{n}{k})$... We get set $S$ of size $\frac{k \lg ... \lg \frac{n}{k}}{\lg^* n} \leq 2k$ that $\lg^* n$ covers $V$. ■

How do we use this for an approximation algorithm?

### 11.1.1 Phase 1

We call a vertex $v$ "center-capturing"(ccv) if $\Gamma^-(v) \subseteq \Gamma^+(v)$. Now $v \cup \Gamma^-(v)$ has a center therefore $v \cup \Gamma^+(v)$ has a center. Choose $v$ to be alg center as it hits all of $\Gamma^+(v)$ too. Repeat on $G - \Gamma_2^+(v)$ vertices at distance $0, 1$ or $2$ from $v$. Stop when there are no ccvs. Let $X$ be set of chosen vertices. Clearly $|X| \leq k$. So there is a set $P$ of $k - |X|$ vertices that cover the remaining vertices.(i.e. those not 2-covered by $X$).

**Claim:** *there exists $\frac{|P|}{2}$ vertices $P'$ such that $X \cup P'$ 5-covers $V$.*

**Proof.** Consider *opt* : $S = (S_1, S_2, S_3)$

- $S_1$: centers in $\Gamma_2^+(x) = \bar{A}$
- $S_2$: centers $x \in A$ such that $\Gamma_2^-(x) \cap \bar{A} \neq \emptyset$
- $S_3$: centers $x \in A$ such that $\Gamma_2^-(x) \cap \bar{A} = \emptyset$

Let $S_3 = \{x_1, ..., x_q\}$. $x_i \in S_3$ is not *ccv*. So there exists $y_i \in \Gamma^-(x_i) - \Gamma^+(x_i)$. Thus $x_i$ is at least 3 from $\bar{A}$. So $y_i \geq 2$ from $\bar{A}$. So $y_i$ is covered by $S_2 \cup S_3 = \{\binom{x_1}{x_{q+1}} \dots \binom{x_1}{x_p}\}$

**Claim:** *Thereare $\frac{|P|}{2}$ vertices in $S_2 \cup S_3$ that cover the vertices covered by $S_3$* **Proof.**

Consider an auxiliary graph $G$ with vertices $S_2 \cup S_3$. There exists an arc $(x_i, x_j)$ if $1 \leq j \leq q$ $y_j$ is covered by $x_i$.

**Any digraph $H$ has a subset of size $\frac{|H|}{2}$ that 2 covers any vertex with indegree atleast** 1. **Proof.** EXORCISE
So there exists a set of size $\frac{p}{2}$ that 2 covers $\{x_1, ..., x_q\}$ in $H$. So they 4-cover $x_1, ..., x_q$ via $y_i$ in $G$. So we 5 cover the vertices covered by $S_3$. ■

Given the claim we can apply our bicriteria result to find $2\frac{|P|}{2}$ vertices that with $X$ $S \lg^*(n)$-cover everything. Thus we have

**Theorem 11** *there exists an $O(\lg^*(n))$ approximation algorithm for asymmetric $k$-center problem.*

■

**Theorem 12** *There does not exists an approximation algorithm with performance better than $O(\lg^* n)$.*

■

## 12 The Knapsack Problem

**Problem 18 (Knapsack)** *Given $n$ objects with*

- *values: $v_1, ..., v_n$*
- *weights: $w_1, ..., w_n$*

*Finde a collection of objects of maximum value that fit into the bag. We have the IP:*

$$max \sum_{i=1}^{n} t_i v_i$$

$$s.t. \sum_{i=1}^{n} t_i \cdot w_i \leq W$$

$$t_i \in \{0, 1\}$$

This problem is $NP$ hard. Reduction from SUBSET-SUM. However this is the "easiest" type of *NP-complete* problem in terms of approximation.

### 12.1 Approximation Schemes

**Definition 2 (PTAS)** *An algorithm $A$ is a polytime approximation scheme(PTAS) for a problem $P$ if for any instance $I$ and fixed $\epsilon$*

- *$A$ gives a solution within $\epsilon$ of opt.*
- *$A$ runs in polynomial time in the size of $I$.*

**Definition 3 (FPTAS)** *An algorithm A is fully polytime approximation scheme(FPTAS) if is PTAS and*

- *A runs in polytime in $|I|$ and $\frac{1}{\epsilon}$*

We will give *FPTAS* for Knapsack. First we give pseudo-polytime algorithm (poly in input and values of data) . We use dynamic programming: Let

- $s(i, V) =$ min weight of subset of $\{1, 2, ..., i\}$ that give a value of $V(\infty$ if it doest exist).
- These can be found recursively

$$w(i, V) = min[w(i-1, V), w_i + w(i-1, V - v_i)]$$

Base cases: $w(i, 0) = 0$, $w(i, -N) = \infty$

Finding $w(i, V)$ is constant time. So running time is $\#w(i, V)$. For $i = 1, ..., n$ let $v^* := max_i v_i$ so $V \leq nv^*$ . Therefore run time is $O(n^2 v^*)$(pseudo- polytime) gives us the *exact answer.*

### 12.1.1 FPTAS

If $v^*$ is small this is fine. So we try to make $v^*$ small.
**Algorithm**

1. `Run D.P with value` $\hat{v_i} := \lfloor \frac{v_i}{k} \rfloor$`(`$k$ `to be decided)`
2. Output $S$, the solution to $D.P$

**Claim:** *S has value within $\epsilon$ of opt.* **Proof.** Let $S^*$ be optimal, then

$$v(S) \geq k\hat{v}(S) \geq k\hat{v}(S^*) = k \sum_{i \in S^*} \hat{v_i}$$

$$= k \sum_{i \in S^*} \lfloor \frac{v_i}{k} \rfloor$$

$$\geq \sum_{i \in S^*} (v_i - k) = v(S^*) - k|S^*| \geq v(S^*) - nk$$

so

$$v(S) \geq v(S^*) - nk$$

i.e. set $k = \frac{\sum v^*}{n}$

$$v(S) \geq v(S^*) - n\frac{\epsilon v^*}{n}$$

$$v(S^*) - \epsilon v^*$$

$$\geq v(S^*) - \epsilon v(S^*)$$

we know that $v^* \subset v(S^*)$. ■

**claim 2** *This is polytime in $|I|\alpha\frac{1}{\epsilon}$* **Proof.** $\hat{v} \leq \frac{v^*}{k} \approx \frac{n}{\epsilon}$ so The run time $O(n^2\hat{v}^*) = O(n^3\frac{1}{\epsilon}$ ■

## 13 Bin Packing

**Problem 19 (Bin Packing)** *Given $n$ items of sizes $a_1, ..., a_n \in [0, 1]$ find a packing of the items into as few unit sized bins as possible.*

### 13.1 2 Approx Algo

1. `Place an item into the first bin it fits into`
    - `If it fits into no bins then start a new bin`

**Theorem 13** *This is a 2 approximation algorithm*

**Proof.** Suppose we use $m$ bins. The first $m - 2$ bins are at least half full.(otherwise we could combine them). Bins $m$, and $m - 1$ must contribute at least 1 unit, otherwise we would not need to start a new bin. We have

$$\sum_{i=1}^{n} a_i \geq \frac{1}{2}(m-2) + 1$$

$$= \frac{1}{2}m$$

Thus

$$OPT \geq \frac{1}{2}m$$

■

**Theorem 14** *There is no approximation algorithm with guarantee $\frac{3}{2} - \delta$ unless $P = NP$*

**Problem 20 (Partition Problem)** *Given $a_1, ..., a_n$ can we partition the numbers into two groups with equal sum?*

**Proof.** Does the *partition problem* return true when applied to our items $a_1, ..., a_n$? It does iff the items can be placed into 2 bins of size

$$\frac{\sum_{i=1}^{n}}{2}$$

So a $\frac{3}{2} - \delta$ approximation algorithm for bin packing solves this problem. A $\frac{3}{2} - \delta$ approximation will only output at least 3 if the solution is greater than 2. This applies if $OPT$ is small. In some ways this is misleading as the bound comes from instances with small solutions, but a large number of items. In fact, we "almost" get a PTAS for this problem. The idea is to partition the items into *big* $:\geq \epsilon$ and *small* $< \epsilon$.

**Lemma 5** *There exists a* PTAS *for bin packing if every item has size at least $\epsilon$*

**Proof.**

- Sort the items by size.

- Partition them into $b := \lceil \frac{1}{\epsilon^2} \rceil$ groups of cardinality $\leq S := \lfloor n\epsilon^2 \rfloor$.
- Create 2 new instances $I_1, I_2$ by rounding up(respectively down) the size of each item in a group to the size of the max(resp min).

**Claim:***Problems $I_1, I_2$ can be solved optimally in polynomial time.* **Proof.**

- We have $k$ distinct sizes.
- All items have size at least $\epsilon$
- Any bin has at most $y := \lfloor \frac{1}{\epsilon} \rfloor$ items
- The number of different configurations in a single bin is at most $X := \binom{y+k}{y}$. We can think of the $k+1$ element which means we dont use any of the $k$ sizes. Drop $y$ balls into $k+1$ boxes (+1 means ball not used) there are $\binom{y+k}{k}$ ways to do this.
- So we have at most $X$ types of bins. There are at most $n$ bins(# of items). So the total number of configurations os at most

$$Z := \binom{n+X}{n} = \binom{n+X}{X}$$

Since $X, y, k$ are constant we have $Z \in \mathit{poly}(n)$ sop we have only a polynomial amount of configurations to try. ■

Note that an optimal packing for $I_1$ is feasible for $I$ since sizes are bigger. We want to show that

$$OPT(I_1) \leq (1+\epsilon)OPT(I)$$

To show this consider $I_2$. Observe that the $n - S$ (where $S := \lfloor n\epsilon^2 \rfloor$) largest items in $I_2$ dominate the $n - S$ smallest items in $I_1$. thus

$$OPT(I_1) \leq OPT(I_2) + S$$

where we are stupidly just adding $S$ boxes for the top discarded $S$ items from $I$ . Finally each item has size $\epsilon$ implies

$$OPT(I) \geq n\epsilon \geq \frac{S}{\epsilon}$$

So

$$S \leq \epsilon OPT(I)$$

therefore

$$OPT(I_1) \leq OPT(I)(1+\epsilon)$$

■How do we deal with the small items?

**Theorem 15** *There is a poly time algorithm for bin packing that gives you a solution with at most $(1+2\epsilon)OPT + 1$ bins .(this +1 is bad if $OPT$ is small but negligable if $OPT$ is big).*

**Proof.**

- Run previous algorithm on big items
- Greedily add in the small items

If no new bins opened then we use at most $(1+\epsilon)OPT$ bins. Suppose $m$ bins are used. The first $m-1$ have less then $\epsilon$ space left. So

$$OPT \geq (m-1)(1-\epsilon)$$

therefore

$$m \leq \frac{OPT}{1-\epsilon} + 1 \leq (1+2\epsilon)OPT + 1$$

## 14 Unsplittable Flow Problem

**Definition 4 (Unsplittable Flow)** *An unsplittable flow is a flow that satisfies each of the demands using a single path and it satisfies arc capacities.*

**Problem 21 (Unsplittable Flow)** *Given directed graph $G = (V, A)$ with arc capacities $u_a$, there is a set $(s_i, t_i), i = 1, ..., k$ of source-terminal pairs with demand $d_i$ units from $s_i$ to $t_i$.*

**Problem 22 (Congestion Problem)** *What is the smallest $\alpha \geq 1$ such that there is a feasible unsplittable flow if we multiply all arc capacities by $\alpha$*

We would like our approximation algorithm to find $\alpha$.

- We assume here $s_i = s \; \forall s$

Solves **partition problem** if $\alpha = 1$ on two nodes $s, t$ with two arcs from $s$ to $t$ each with half the capacity.

**Theorem 16** *Any feasible fractional flow can be converted into an unsplittable flow with at most twice the congestion provided that $d_{max} \leq u_{min}$*

**EX**: Find min congestion feasible fractional flow using an LP

### 14.1 Algorithm(single source)

1. `Find fractional flow` $f$
2. **Preprocess** $f$
   - Assume acyclic
   - Given $f$, greedily move back terminals towards source if $d_i \leq f_a$ on an incoming arc.

   After preprocessing all terminals are called *regular:* $d_i > u_a$ for all in coming arcs.
3. **Augmenting Cyles**
   - Pick any vertex and grow *forward* path to a *sink*= *outdegree* $= 0$ in $G$. There is a sink since we have no cycles.

- Grow a backward path along *singular arcs*: $(u, v)$ is a singular arc if there is a unique path from $v$ to a unique sink.(All out degrees from $v$ to sink are 1).
- Repeat until reach vertex we have seen. i.e. cycle

4. **Augment Cycle**

   - decrease flow by $\epsilon$ on forward arcs.
   - increase flow by $\epsilon$ on backward arcs.
   - $\epsilon = min[\epsilon_1 := min_{a\ forward} f_a, \epsilon_2 := min_{a\ backward}(d_i - f_a)]$
   - So we either remove arc or get $d_i = f_a$ on a singular arc. Note we maintain flow constraints.
   - After Augment we:Priority
     (a) $a$ singular and $f_a = d_i$
     (b) $a$ non singular and $f_a \geq d_i$
   - we then find another cycle and repeat until all terminals at source.

*Singular arcs remain singular.*

**Theorem 17** *if $v$ contains an irregular terminal then*

1. *$deg^{-}(v) = 0$ (outdegree is zero)*
2. *$v$ has no other irregular terminal*
3. *$v$ contains a regular terminal*

We have $d_j \leq f_a$ as $j$ is irregular. Hence $f_a > d_j$ or it would have been moved back by **(a)**. $a$ is singular by **(b)** or it would have been moved back by **(a)**.

**Claim:** *To create an irregular terminal $j$ must have been moved back along a singular arc say $(v, w)$.*

By $\epsilon_2$, $f_a$ can't have gone from below $d_j$ to above $d_j$ whilst $j$ was at $v$. THerefore $f_a > d_j$ when $j$ is moved to $v$. Therefore $a$ is singular at this time or make move (2). Therefore $\hat{a}(v, w)$ was singular at this point. Therefore $f_{\hat{a}} = d_j$ So $f_{\hat{a}}$ becomes 0 and is removed. $v$ had out degree 1 before so $outdegree(v) = 0$ **(A)**. **(B)** So when irregular terminal moves to $v$ we then get $outdegree(v) = 0$ so no other irregular terminals can move to $v$. **(C)** $v$ contains a regular terminal since we have $f_a > d_j$ but $outdegree(v) = 0$ therefore there exists another terminal at $v$ therefore by **(B)** this terminal is regular. ■

**Corollary 3** *At start of phase the in degree of any vertex with a terminal is at least* 2.

**Proof.** it contains a regular terminal. ■

**Lemma 6** *Alg gives an unsplitable flow*

**Proof.** Forward paths hit sink. The sink has a regular terminal therefore indegree is atleast 2. So we can find backward path of singular arcs. So find cycle. ■

**Proof Of Main theorem: Proof.** The flow increases only on singular arcs. Once an arc is singular but since $f_a = d_j$ we then remove $a$. Therefore we can only send 1 extra packet above its capacity $\leq d_{max} \leq u_a$ so $total\ flow \leq u_a + d_{max} \leq 2u_a$. ■

- 5 approximation algorithm if $u_a < d_{max}$
- Demand max: Max sum of demands satisfied without exceeding capacities $4 - 43$ approx
- Round min use $k$ phases to satisfy demands $k = 5$

## 15 Confluent Flows

Chen et al.

**Definition 5 (Confluent)** *A flow is confluent if all packets leaving a vertex go out along the same arc if they have the same destination.*

- *If there is a single destination $t$ then a confluent flow is an aborescence.*

This is interesting since most internet routing is destination-based. e.g. shortest path routing.

Confluent flows do not seem good for minimizing congestion. How good/bad are they? **Single Commodity:**i.e. one sink $t$.

- Uniform capacities $u_a = 1$.
- Remove $t$ to give "sinks" $t_1, ..., t_k$.
- special case : $d_i := 1 \forall i$.
- Find a spanning tree with root $t$ such that the max size of a subtree is minimized.

We are interested in "node" congestion at $t_1, t_2, ..., t_k$. Worst node congestion here equals worst arc congestion before.

### 15.1 Algorithm

We start with a splittable (fractional) flow that minimizes node congestion. (WLOG $congestion = 1$). i.e.

$$flow\ in\ v + d_v = \ flow\ out\ of\ v$$

We take an LP formulation

$$\sum_{a \in \delta^-(v)} f_a + d_v = \sum_{a \in \delta^+(v)} f_a$$

$$\sum_{a \in \delta^+(v)} \leq 1$$

Want to turn this into a low congestion confluent flow. We use 3 operations: We assume $f$ is acyclic.

1. Node Aggregation: If node $v$ only has arcs going to one sink $t_i$ ($T_i$) we contract $v$ into $t_i$ using only one of its arcs. This does not change the max congestion which is at $t_1$

2. Breaking saw-tooth cycles: We look for cycles between "frontier" nodes and sinks.

   - Add a reverse arc $(t_i, v)$ for any arc $(v, t_i)$ and look for a dicycle of length greater than 2.

   Blue paths long, yellow have 1 edge.

   - increase the flow by $\epsilon$ on yellow arcs (rev of yellow arc).
   - decrease the flow on blue arcs by $\epsilon$.

   We still have flow constraints on non-sinks. Congestion falls/stays the same on these nodes. If we extend this into $t_i$ then we maintain $\epsilon$ flow. The congestion is the same at root (or where blue, yellow paths meet) so max congestion still does not go up.

3. Sink Deactivation: If we can not do (1) or (2) then there must be a sink $t_i$ with only one neighbour $v$ (check this)

Let $b_i$ be the congestion of $t_i$.

- if $b_i + f(v, t_j) < b_j - f(v, t_j)$ we remove arc $(v, t_j)$ and send the flow to $t_i$.
- Otherwise we remove arc $(v, t_i)$ and send its flow to $t_j$. "Deactivate" $t_i$.

Repeat these steps until we have a confluent flow. observe that (1) and (2) do not increase max congestion. (congestion never increases for non-sink nodes, even for S. i.e. congestion $\leq 1$) To evaluate (3) we use a *potential function*

$$\phi(f) = \sum_{t_i \ active} 2^{b_i}$$

initially

$$b_i \leq 1 \ so \ \phi(f) \leq k \cdot 2^1 = 2k$$

We will show that $\phi$ only decreases.

$$\phi(f') \leq 2k$$

so

$$\sum_{active} 2^{b_i} \leq 2^k \Rightarrow b_i \leq O(\log k)$$

- $$b_i + f(v, t_j) < b_j - f(v, t_j)$$

  $$2^{b_i} + 2^{b_j} \to 2^{b_i + f(v, t_i)} + 2^{b_i - f(v, t_j)}$$

  By convexity this is smaller.

- $t_2$ is deactivated so $2^{b_i} \leq \phi(f) \leq 2k$ So $b_i \leq O(\log k)$.

- For active nodes the change $\phi(f') - \phi(f)$

$$-2^{b_i} - 2^{b_j} + 2^{b_j+f(v,t_v)} \leq 2^{b_j+f(v,t_i)} - 2^{b_j} - 2^{b_j-2f(v,t_i)-i}$$

$$b_i > b_j - 2f(v,t_j)$$

$$\leq 2^{b_j+f(v,t_i)} - 2^{b_j+1-f(v,t_j)}$$

  Convexity

$$2^{b_j} + 2^{b_j-2f(v,t_j)} \geq 2 \cdot 2^{b_j-f(v,t_j)}$$

  as long as $v \leq 1$ we have

$$f(v,t_i) + f(v,t_j) \leq 1$$

  $(f \to f')$.

**Corollary 4** *This is an $O(\log k)$ approximation algorithm.*

It is $NP$ *hard* to do better than $O\log k$. This $LP$ analysis is tight.

## 16 Semi Definite Programming

We often use an LP relaxation to approximate the set we are optimizing over. Problems:

- Sometimes relaxation is not"tight" enough. Here we look at non linear relaxations that may be tighter.

### 16.1 Linear Algebra

The following are equivalent :

- $X$ is positive semidefinite $\preceq 0$
- $y^T X y \geq 0 \; \forall y \in R^n$
- All eigenvalues of $X$ are non-negative
- $X := V^T V$ for some $m \times n$ matrix $V$.

**Proof.** Exercise
Given $A, X(n \times n)$ set $A \circ X := \sum_{i,j} a_{ij} \cdot x_{ij} = trace(A^T X)$ Then a *semi definite program* is formulated as

$$max \; C \circ X$$

$$s.t. \; A^i \circ X = b^i$$

$$X \succeq 0, X \; symmetric$$

or

$$\sum_{ij} c_{ij} x_{ij}$$

$$s.t.\ \sum_{ij} a_{ij}^k x_{ij} = b^k$$

$$(x_{ij}) \succeq 0$$

$$x_{ij} = x_{ji}$$

If $C, A^k$ are diagonal matrices we get an $LP$. Semi definite programs ca be solved in polytime (also in $\log \frac{1}{\epsilon}$ by ellipsoid method, or interior point methods to within $\epsilon$ of $Opt$.

- **We have now Covered *ellipsoid method***

Given $X$ is positive semi definite we can find $V$ such that $X = V^T V$ in polytime (within $\epsilon$). Semi definite programming is equivalent to vector programming.

$$max\ \sum_{i,j} c_{ij}(v_i \cdot v_j)$$

$$s.t.\ \sum_{ij} a_{ij}^k (v_i \cdot v_j) = b^k \quad \forall k$$

$$v_i \in R^k$$

because $X$ is positive semidefinite and symmetric $X = V^T V$

## 17 Max Cut

Given $G = (V, E)$ and weights $c_e\ \forall e \in E$ find $S \subseteq V$ such that $\sum_{e \in \delta(S)} c_e$ is maximized. Unlike min-cu, max-cut is $NP$ hard. For a long time factor 2 was the best known approx alg.

### 17.1 Goemans , Williamson

$$max\ \sum_{i<j} c_{ij}(\frac{1 - v_i v_j}{2})$$

$$V_i \in \{-1, 1\}\ \forall i \in V$$

So we have the vector program relaxation

$$max\ \frac{1}{2} \sum_{i<j} c_{ij}(1 - v_i v_j$$

$$s.t.\ v_i \cdot v_i = 1$$

$$v_i \in R^n$$

So we solve this $VP$ to get $v_1, v_2, ..., v_n$. What do we do with these? Easy pick a random vector $r$ and let $S = \{i : r \cdot v_i \geq 0\}$ picking a random hyperplane to divide the vectors into 2. By linearity of expectation we can focus on a single edge $(i, j)$. What matters is just the probability that $v_i, v_j$ are separated by $H$. This depends only on the orthogonal projection of $r$ onto the plane given by $v_i$ and $v_j$. As $r \cdot v_i = (r_1 + r_2) \cdot v_i = r_1 \cdot v_i$ , $r \cdot v_j = r_1 \cdot v_j$.

**Lemma 7** *$P(i,j \text{ separated}) = \frac{1}{\pi} arccos(v_i \cdot v_j) = \frac{\phi_{ij}}{\pi}$*

**Corollary 5** *SDP gives*

$$E(SDP) = E(\sum_{e \in \delta(S)} c_e) = \sum_{i<j} c_{ij} \frac{\phi_{ij}}{\pi}$$

Is this any good?

$$\frac{\sum_{i<j} c_{ij} \frac{\phi_{ij}}{\pi}}{\frac{1}{2} \sum_{i<j} c_{ij}(1 - v_i v_j)}$$
$$\geq min_{i<j} \frac{\phi_{ij}}{\pi} \cdot \frac{1}{\frac{1}{2}(1 - v_i v_j)}$$
$$= \frac{2}{\pi} min_{i<j} \frac{\phi_{ij}}{1 - cos\phi_{ij}}$$
$$\geq \frac{2}{\pi} min_{0<\phi<\pi} \frac{\phi}{1 - cos(\phi)}$$
$$\geq 0.878$$

**Lemma 8** *For 5 cycle $\frac{Opt}{UpperBound} \approx 0.884$*

In fact there are examples where the gap is 0.878.

**Theorem 18 (Haastad)** *there is no 0.941 approximation algorithm for max cut*

## 18 Applications

- **Max 2-SAT**

### 18.1 Vertex Coloring

Given $G$. Assign min number of colors to vertices such that $i, j$ have different colors if $(i, j) \in E$. This is notoriously hard to approximate

**Theorem 19** *Unless $P = NP$ not approximable to within $n^{\frac{1}{7}-\epsilon}$ , $n^{1-\epsilon}$ unless $P = ZPP$.*

In certain cases we do okay. i.e. can test in polytime if $\chi(G) = 2$.

**Theorem 20 (Widgeson)** *If $G$ is 3 colorable there is a $O(\sqrt{n})$ approximation algorithm*

**Proof.** see HW #1. ■

To beat this we use a SDP approach. We can partition the graph into 3 stable sets. Assign each group to a vector $y_1, y_2, y_3$. We will insist if $(i, j) \in E$ then $v_i, v_j$ are at 120°. This motivates the following vector progam:

$$min\ \lambda$$

$$v_i \cdot v_j \leq \lambda\ \forall (i, j) \in E$$

$$v_i \cdot v_j = 1 \ \forall i \in V$$

$$v_i \in R^n \ \forall i \in V$$

**Ex:** FORMULATE AS AN SDP

$\phi_{ij} \geq arccos(\lambda)$ . We know $\lambda \leq -\frac{1}{2}$. So we have $v_1, ..., v_n$. Pick $t$ random vectors $r_1, ..., r_t$ where $t := 1 + \log_3(\Delta)$. Each vector $v_i$ has $v_i \cdot r_j \leq 0 : -$ or $v_i \cdot r_j > 0 : +$. So $v_i$ has a type $(+ + - - + - + + + - - -)_{length\ t}$ there are $2^t$ types $2^{1+\log_3(\Delta)} = 22^{\log_3(\Delta)} = 2\Delta^{\log_3(2)}$. We use 1 color per type. The problem is that this may not be feasible. If there are not many bad edges this is not a problem. Suppose $\leq \frac{n}{3}$ edges are bad. Then at least $\frac{n}{3}$ vertices $S$ which are not adjacent to a bad edge. Recurse on $G - S$ using new set of colors. Therefore edges in $\delta(S)$ are good. We get a valid coloring on $G[V - S]$. Use $O(\log(n)\Delta^{\log_3(2)})$ colors in total.

How many edges are bad?

$$P(ij\ is\ bad) = (1 - \frac{\phi_{ij}}{\pi})^t$$

$$\leq (1 - \frac{\arccos(\lambda)}{\pi})^t$$

$$\leq (1 - \frac{\arccos(-\frac{1}{2})}{\pi})^t$$

$$= 1 - \frac{2\pi}{3}\frac{1}{\pi} = \frac{1}{3}^t = (\frac{1}{3})^{1+\log_3 \Delta} = \frac{1}{3\Delta}$$

**Lemma 9** *$E(\#\ bad\ edges) \leq \frac{n}{6}$*

**Proof.** There exist $m$ edges with $m \leq \frac{n\Delta}{2}$ thus $\frac{n\Delta}{2}\frac{1}{3\Delta} = \frac{n}{6}$ ■

By Markovs inequality the probability there are at least $\frac{n}{3}$ bad edges is $\leq \frac{1}{2}$. Repeat $c \cdot \log(n)$ times. Probability we get $\frac{n}{3}$ bad edges is $(\frac{1}{2})^{c\log(n)} \approx \frac{1}{n^c}$.

**Corollary 6** *There exists $O^*(\Delta^{\log_3(2)})$ approximation algorithm to color 3 colorable graphs.*

**Proof.** $O^*$ means ignore lower order terms such as $\log(s)$ ■. But if $\Delta = n$ this is $O^*(n^{.63})$. How do we fix this? Run hybrid with Widgerson. Apply$O(\frac{n}{p})$ iterations of Widgeson. For the case degree $\geq p$. Then apply SDP to use $O(p^{\log_3(2)}$ colors. ■

**Corollary 7** *there is a $O^*(n^{.387})$ approximation algorithm.*

**Proof.** Use $O(\frac{n}{p} + p^{\log_3(2)})$ colors. This is minimized $n = p^{1+\log_3(2)} \Rightarrow p = n^{0.613}$ ■

We can improve this.

**Theorem 21** *There exists an $O^*(n^{\frac{1}{4}})$ approximation algorithm if $\chi(G) = 3$.*

**Proof.**

- Solve SDP

- Pick $t = O^*(\Delta^{\frac{1}{3}})$ round $v_i$ to closest $r_j$
- Ignore bad edges and recurse.

It can be shown that $P(ij\ bad) \leq O^*(\frac{1}{\Delta})$... This gives $O^*(\frac{n}{p} + p^{\frac{1}{3}})$ coloring. $n = p^{\frac{4}{3}} \Rightarrow O^*(n^{\frac{1}{4}})$ colors. ■

These methods extend to $\chi(G) = k$. ■

## 19 LP Duality

Take an $LP$ in standard form:

$$min\ cx$$

$$s.t.\ Ax \geq b$$

$$x \geq 0$$

We would like to know if we can get a lower bound(any feasible solution gives an upper bound) , take for example

$$min\ 7x_1 + x_2 + 5x_3$$

$$s.t.\ x_1 - x_2 + 3x_3 \geq 10$$

$$5x_1 + 2x_2 - x_3 \geq 6$$

$$x_1, x_2, x_3 \geq 0$$

by adding constraints together we can derive other inequalities e.g.

$$7x_1 + x_2 + 5x_3 \geq 6x_1 + x_2 + 2x_3 \geq 16$$

More systematically we are choosing multiples $y_i$ of row $i$, such that

$$\sum_i y_i a_{ij} \leq c_i \quad \forall j$$

and

$$y \geq 0$$

this gives us lower bound

$$\sum_i b_i y_i$$

We want the maximum lower bound so in general we have

$$max\ yb$$

$$s.t.\ yA \leq c$$

$$y \geq 0$$

which we call the dual.

**Theorem 22 (Weak Duality)** *for any feasible $x, y$ for primal, dual programs we have $yb \leq cx$*

**Proof.**

$$yb \leq y(Ax) = (yA)x \leq cx$$

■

Perhaps, surprisingly we have something stronger

**Theorem 23 (Strong Duality)** *If the primal and dual are feasible then their objective optimal values are the same.*

$$cx^* = y^*b$$

**Proof.** see combinatorial optimization notes. ■

### 19.1 Complimentary Slackness

Weak duality proof tells us that feasible solutions $x, y$ are optimal iff the following hold

**Theorem 24 (Primal complimentary slackness)** $(yA)x = cx$, *i.e.* $x_j > 0 \Rightarrow \sum_{i=1}^{m} a_{ij}y_i = c_j$

similarly

**Theorem 25 (Dual complimentary slackness)** $yb = xAy$ *i.e.* $y_i > 0 \Rightarrow \sum_{j=1}^{n} a_{ij}x_i = b_i$

Many classical results in combinatorial optimization can be derived from combinatorial optimization.

## 20 Linear Programming and Approximation Algorithms

Here we see how $LP$ duality can be used to evaluate the performance of an algorithm.

### 20.1 Meta Method

1. Formulate problem as an integer program
2. Relax to $LP$(primal)
3. Use the dual as lower bound and compare against solution by algorithm (any dual feasible solution can be used)

Lets apply this to set cover

#### 20.1.1 Set Cover

sets $S_1, ...., S_t$ costs $c_1, ..., c_t$.

1. $\mathcal{C} := \emptyset$
2. `Pick set` $S_j$ `which covers uncovered elements at the lowest average cost`
3. $\mathcal{C} := \mathcal{C} \cup \mathcal{S}_|$

4. `repeat until` $\mathcal{C}$ `is a set cover`

let $\mu_j = \frac{c_j}{\# \ of \ items \ first \ covered \ by \ S_j}$ We have (P)

$$min \sum_{j=1}^{t} c_j x_j$$

$$s.t. \sum_{S_j : v_j \in S_j} x_j \geq 1 \forall i$$

$$x_j \geq 0$$

Note that $A$ has a row for each element and a column for each set. So we have the Dual

$$max \sum_{i=1}^{n} y_i$$

$$s.t. \sum_{v_i : v_i \in S_j} y_i \leq c_j \quad \forall S_j$$

$$y_i \geq 0$$

**recommends that we practice taking the dual**.

Let Greedy give integral solution $x$ of cost $cost(x)$. We want a dual solution $y$ such that

$$cost(x) \leq \alpha dualcost(y) \leq \alpha dualopt = \alpha primalopt \leq \alpha opt$$

i.e. an $\alpha$ approximation algorithm. To get this define $y$ as

$$y_i = \frac{1}{H_n} \mu_{j(i)}$$

where $v_i$ is first covered by $S_{j(i)}$ in greedy, where $j(i)$ is the first set which covers $i$ in greedy. Lets set that $y$ is dual feasible. $y_i \geq 0$. We want to show that

$$\sum_{v_i : v_i \in S_j} y_i \leq c_j \quad \forall S_j$$

Let $S_j := \{v_i, ..., v_p\}$ in the order that they were covered by greedy. Take $v_r \in S_j$. At this point $v_r, ..., v_p$ are uncovered,so $S_j$ could be chosen with $\mu_j \leq \frac{c_j}{p-r+1}$. So $y_r \leq \frac{1}{H_n} \mu_{j(r)} \leq \frac{1}{H_n} \mu_j = \frac{1}{H_n} \frac{c_j}{p-r+1}$ therefore

$$\sum_{r=1}^{p} y_r \leq \frac{1}{H_n} \sum_{r=1}^{p} \frac{c_j}{p-r+1}$$

$$= \frac{c_j}{H_n} \sum_{r=1}^{p} \frac{1}{r} = c_j \frac{H_p}{H_n}$$

$$\leq c_j \frac{H_n}{H_n} = c_j$$

now

$$cost(y) = \sum_i y_i = \frac{1}{H_n} = \frac{1}{H_n} \sum_i \mu_{j(i)}$$

$$= \sum_{S_j \in \mathcal{C}} \frac{1}{H_n} \sum_{v_i \ first \ covered \ by \ S_j : j(i)=j} \mu_j$$
$$= \frac{1}{H_n} \sum_{S_j \in \mathcal{C}} c_j$$
$$= \frac{1}{H_n} cost(x)$$

i.e. $O \log(n)$ approximation algorithm.

## 21 The Primal Dual Method

Lets first see how to use $LP$ duality to solve LPs. This is the Primal Dual method due to Dantzig, Ford and Fulkerson.

$$(P) \ \min \ cx$$
$$s.t. \ Ax \geq b$$
$$x \geq 0$$

$$(D) \ \max \ by$$
$$yA \leq c$$
$$y \geq 0$$

We know feasible $(x, y)$ are optimal if they satisfy complimentary slackness conditions:

$$(P) \ x_j > 0 \Rightarrow \sum_{j=1}^{m} a_{ij} y_i = c_j$$

$$(D) \ y_i > 0 \Rightarrow \sum_{i=1}^{n} a_{ij} x_j = b_i$$

The idea is:

- Start with a feasible $y$ to dual.
- Either find $x$ such that $(x, y)$ satisfy complimentary slackness or find $y'$ with higher dual value than $y$.
- repeat

How do we do this? Given $y$ lets try to find an $x$ that satisfies complimentary slackness with $y$. We use an LP.

$$(ResP) \ \min \sum_{i : y_i > 0} s_i + \sum_{j : \sum_{i=1} a_{ij} y_i < c_j} x_j$$

$$s.t.\ \sum_{j=1}^{n} a_{ij}x_j \geq b_i$$

$$\sum_{j=1} a_{ij}x_j - s_i = b_i$$

$$s_i \geq 0$$

$$x_j \geq 0$$

This imposes dual complimentary slackness via constraints, and the objective function enforces primal conditions when $x_j = 0$. Observe if $Opt(ResP) = 0$ then $(x, y)$ are optimal in original $P$ and so we are done. So assume $Opt(ResP) > 0$ and

$$(ResD)\ \max \sum_{i=1}^{n} b_i\hat{y}_i$$

$$s.t.\ \sum_{i=1}^{m} a_{ij}\hat{y}_i \leq 0\ \ \forall j : \sum_{i=1}^{n} a_{ij}y_i = c_j$$

$$\sum_{i=1}^{m} a_{ij}\hat{y}_i \leq 1\ \forall j : \sum_{i=1}^{n} a_{ij}y_i < c_j$$

$$\hat{y} \geq -1\ \forall i : y_i > 0$$

$$\hat{y}_i \geq 0\ \forall i : y_i = 0$$

Exercise : check that (ResD) is the dual of (ResP)

since $Opt(ResP) > 0$ this implies $ResD$ has $Opt > 0$. Take a solution $\hat{y}$ with value greater than 0. $y' = y + \epsilon\hat{y}$. Want $y'$ feasible and of higher value than $y$ in $D$.

$$b^T y' = b^T(y + \epsilon\hat{y}) = b^T y + \sum b^T\hat{y} > b^T y$$

So we want $\epsilon$ such that $y'$ is feasible in $D$. We know $y \geq 0$.

$$y' = y + \epsilon_1\hat{y} \geq 0$$

This is true since if $-1 \geq \hat{y} < 0$ then $y_i > 0$ so pick $\epsilon_1$ to give tightest constraint. (if $\hat{y} \geq 0$ there is nothing to show).

Similarly $\sum_{i=1}^{m} a_{ij}y'_i \leq c_j$ for some $\epsilon_2 > 0$ since if $\sum a_{ij}\hat{y}_i > 0$ then $\sum a_{ij}c_j < c_j$.

- Take the most restrictive
- Set $\epsilon := \min(\epsilon_1, \epsilon_2)$

So $y'$ is feasible. So $P - D$ method leads to optimal solution. But we have reduced $LP$ to a collection of $LP$s! There are two advantages:

- 
  - Objective function in $P$ is $0, 1$.

  - Constraints in ResD have $0-1$ on RHS. i.e. no $c$.

- - The unweighted problems can often be solved combinatorially

Many classical combinatorial optimization algorithms can be understood in terms of the primal dual method. i.e. Hungarian method for assignment problem(Kuhn), Edmonds matching algorithm, Ford Fulkersons network flow algorithm. Even Dijkstra's shortest path algorithm.

In general it is not practical for solving $LP$s. However the ideas are very useful for approximation algorithms.

## 21.1 Primal Dual Method for Approximation Algorithms

Maybe not obvious since approximation algorithms use $I.P$s not $LP$s! These don't usually have nice max-min results for integral solutions. We also need integral solutions to primal. What do we do? we relax the complimentary slackness conditions.

Let $x, y$ be feasible primal dual solutions satisfying for $\alpha, \beta \geq 1$.

- Approximate Primal complimentary slackness conditions

$$x_j > 0 \Rightarrow \sum_{j=1}^{m} a_{ij} y_i \geq \frac{1}{\alpha} c_j$$

- Approximate Dual Complimentary slackness conditions

$$y_i > 0 \Rightarrow \sum_{i=1}^{n} a_{ij} x_j \leq \beta \cdot b_i$$

**Lemma 10** *If so*

$$val(x) \leq \alpha\beta val(y) \leq \alpha\beta opt(LP) \leq \alpha\beta opt(INT)$$

**Proof.**

$$c^T x \leq \alpha (A^T y)^T x = \alpha y^T A x \leq \alpha\beta b^T y$$

■So if $x$ is integral and we find such a $Y$ (can be fractional) then $x$ is an $\alpha\beta$ approximate solution.

Lets see how we can apply this to SET COVER :

$$\min \sum_j c_j x_j$$

$$s.t. \sum_{S_j : v_i \in S_j} x_j \geq 1 \quad \forall v_i$$

$$x_j \geq 0$$

Take the dual we have

$$\max \sum_i y_i$$

$$s.t. \sum_{v_i : v_i \in S} y_i \leq c_j \quad \forall S_j$$

$$y_i \geq 0$$

These types of primal -dual problems are known as *packing - covering* . Let $f$ be the maximum number of sets any element is in. Now set $\alpha = 1,\ \beta = f$.

So we need to satisfy

$$x_j > 0 \Rightarrow \sum_{i=1}^{m} y_i = c_j$$

and

$$y_i > 0 \Rightarrow \sum_{j=1}^{n} x_j \leq f$$

### 21.1.1 The Primal Dual Algorithm

1. typically start with $x = y = 0$ is a dual feasible solution
2. increase $y$ , keeping it feasible and try to improve the "feasibility" of $x$.
3. We do this maintaining approximate complimentary slackness conditions until $x$ is feasible

Lets apply this to **Set Cover**:

```
1. $x := 0$, $y := 0$
2. if $x$ not feasible then there is some element $v_i$ which has not been covered.
3. While $x$ is not feasible DO:
    (a) Pick uncovered $v_i$
    (b) Increase $y_i$ until some set $S_j \ni v_i$ becomes tight in the dual.
    (c) So picking $S_j$ covers at least one new element
    (d) Update as covered any newly covered element.
```

**Theorem 26** *This is an $f$ approximation algorithm*

**Proof.** We have produced an integral feasible $x$ in polytime. The primal complimentary slackness conditions are tight. Since we have produced a $0, 1$ solution, by the definition of $f$ we have satisfied the dual complimentary slackness conditions. ■

This analysis is tight.

### 21.1.2 Multi commodity Flows and Multicuts

**Definition 6 (Multicut)** *Given $G := (V, E)$ with edge capacities $u_e$ , wource sink pairs $(s_1, t_1), ..., (s_k, t_k)$ Multicut is a set of edges whose removal disconnects all pairs $s_i, t_i$.*

**Problem 23 (Min Multicut)** *Find a minimum capacity multicut*

This simple case is still $NP-hard$. To see this take an instance of vertex cover on $Star = v^{centre}, v_1, ..., v_n$. $(v_i, v_j)$ is source-sink pair if and only if $(v_i, v_j) \in E$. An edge set $(v^{centre}, v_i)_{i \in C}$ is a multicut if and only if $\{v_i\}_{i \in C}$. is a vertex cover.

Let $P_i$ be a path from $s_i$ to $t_i$ in $T$.

$$\min \sum u_e x_e$$

$$s.t. \sum_{e \in P_i} x_e \geq 1 \quad \forall i = 1 \dots k$$

$$x_e \geq 0$$

The Dual is

$$\max \sum y_i$$

$$s.t. \sum_{P_i : e \in P_i} y_i \leq u_e \quad \forall e$$

$$y_i \geq 0$$

An integral solution to the dual is an integral multi commodity flow. Where $y_i$ is the number of units of flow from $s_i$ to $t_i$. So fractional multicut equals fractional flow for LP optima.

We have (PCS)

$$x_e > 0 \Rightarrow \sum_{i : e \in P_i} y_i = v_e$$

and (DCS)

$$y_i > 0 \Rightarrow \sum_{e : e \in P_i} \leq 2 \cdot 1 = 2$$

Thus we will have a 2 approximation algorithm for minimum multicut on trees. We change $y$ integrally so we get a 2 approximation algorithm for integral multi commodity flow as well.

**Algorithm**

```
1. $C := \emptyset, F := \emptyset$
2. Root $T$ at some vertex $r$
3. PHASE 1: Work up from leaves
   (a) Given $v$ for all $(s_i, t_i)$ with least common ancestor $(lca(s_i, t_i) = v$ greedily route
       flow integrally from $s_i$ to $t_i$
   (b) Add to $C$ all edges that becomes tight
4. PHASE 2: CLEAN UP let $C := \{e_1, ..., e_i\}$ in order.
   (a) For $j = i$ to $1$ (reverse order)
   (b) if $C - e_j$ is a multicut set $C := C - e_j$
```

**Lemma 11** *Let $(s_i, t_i)$ have non zero flow. Then $|P_i \cap C| \leq 2$ which means we satisfy (DCS).*

**Proof.** Let $lca(s_i, t_i) = v$. We show that at most 1 edge from $s_i - v$ is chosen, and at most 1 edge from $t_i - v$ is chosen. Suppose $e, e'$ are chosen on $s_i - v$ with $e$ deeper. Consider reverse delete step when $e$ is examined. It is not removed so there is some $(s_j, t_j)$ with $P_v \cap C = \{e\}$. So $e' \in P_j$ implies that $lca(s_j, t)$ are below $e'$.

So $v$ is above $v'$ too. So for phase examining $v'$ to end there is some $e'' \in P_j$ that hits capacity. There is flow from $s_i$ to $t_i$. We can only add this flow in phase $v = lca(s_i, t_i)$. So $e$ was added to $C$ in phase $v$ or later. So $e$ is added after $e''$ so $e'' \in P_j \cap C$ when we test $e$ in reverse delete. $\Rightarrow\Leftarrow$ ■

There are $O\log(k)$ approximation algorithms for multicut in general graphs. There no non trivial approximation algorithms of integral multi commodity flow in general graphs. There are big integrality gaps for typical LPs.

## 22 The Steiner Forest Problem

**Problem 24 (Steiner Forest)** *Given $G := (V, E)$ with edge costs $c_e$ and disjoint vertex sets $V_1, ..., V_k \subseteq V$. Find a min cost sub graph such that all vertices in $V_i$ are connected for all $i$.*

To formulate as an LP let $r_{ij} = 1$ if $i, j \in V_p$ , 0 otherwise. Let $f(S) = 1$ if we have $i \in S, j \notin S$ with $r_{ij} = 1$. 0 otherwise.

$$\min \sum_{e \in E} c_e x_e$$

$$s.t. \sum_{e \in \delta(S)} x_e \geq f(S) \quad \forall S \subseteq V$$

$$x_e \geq 0$$

Dual

$$\max \sum_{S} f(S) y_S$$

$$\sum_{S: e \in \delta(S)} y_S \leq c_e \; \forall e$$

$$y_S \geq 0$$

The primal dual algorithm usually wants : PCS $x_e > 0 \Rightarrow \sum_{S: e \in \delta(S)} y_S = c_e$ and DCS: $y_S > 0 \Rightarrow \sum_{e \in \delta(S)} x_e \leq 2f(S)$ However in our case making this work is an open problem.

Relax DCS

- Instead of a chosen $S$ having at most 2 edges
- A chosen $S$ has at most 2 edges 'on average'

One obstacle is that there are an exponential amount of cuts. During the algorithm we say $S$ is unsatisfied if

- $f(S) = 1$
- $|F \cap \delta(S)| = 0$ where $F$ is current solution.

We say that $S$ is active if it is a minimal unsatisfied set. Observe $S$ is active if and only if it is a component of $F$ and $f(S) = 1$. We can find these in linear time by BFS. a component is active if $|S \cap V_i|, |\bar{S} \cap V_i| \geq 1$ for some $i$. If there are no active sets then $x$ must be feasible.

**Algorithm**

```
1. $F := \emptyset, y_S := 0 \ \forall S$
2. Repeat until no unsatisfied sets(no active sets)
    (a) Raise $y_S$ for all active sets until some edge $e$ is tight.
    (b) $F := F \cup e$
3. Return $\hat{F} := \{e \in F : F - e \text{ is infeasible}\}$
```

$\hat{F}$ is feasible. In the first phase we never add an edge with a component. (only add one edge at a time). So $F$ is a forest. Therefore there is a unique path from $i$ to $j$ for $i, j \in V_p$. So take $R \in P_{ij}$, $F - e$ disconnects $i$ and $j$ so keep $e \in \hat{F}$. So algorithm is polytime and gives feasible solution. Lets analyse it.

**Theorem 27** *This is a 2 approximation algorithm*

**Proof.** We want to show that $\sum_{e \in F} c_e \leq 2 \sum_S y_S$

$$\sum_{e \in \hat{F}} c_e = \sum_{e \in \hat{F}} ( \sum_{s: e \in \delta(S)} y_S)$$

$$= \sum_S \sum_{e \in \delta(S) \cap \hat{F}} y_S = \sum_S y_S |\hat{F} \cap \delta(S)|$$

**Claim:** *Let $\Delta_i :=$ the raised dual weights in iteration i of the first phase. Then $\Delta_i \sum_{S \text{ fl } in\ phase\ i} |\hat{F} \cap \delta(S)| \leq 2\Delta_i \cdot$ number of active sets in phase $i$* **Proof.** We want to show

$$\sum_{S\ active\ phase\ i} \frac{|\hat{F} \cap \delta(S)|}{\#\ active\ sets\ phase\ i} \leq 2$$

Look at $H = (V, \hat{F})$ and contract the components of $F_i$ at iteration $i$. Each active node $S$ in iteration $i$ must have $|\delta(S) \cap \hat{F}| \geq 1$. So isolated nodes are non active. Remove all non-active nodes. The remaining non active nodes share $|\delta(S') \cap \hat{F}| \geq 2$ (as not leaf) Suppose $\delta(S') \cap \hat{F} = e$ it is non redundant so $e$ needed to connect $i, j \in V_r$ so WLOG $i \in S', j \in S' \Rightarrow S'$ is active as $r_{ij} = 1$.

So non active nodes have degree at least 2 in a Forest. The average degree in a forest is at most $\frac{2(n-1)}{n} \leq 2$. Therefore active nodes have average degree less than 2. i.e.

$$\frac{\sum_{S\ active\ phase\ i} |\hat{F} \cap \delta(S)|}{\#\ active\ sets\ phase\ i} < 2$$

■(this is tight)

## 23 Facility Location Problem

We have a set $F$ of facilities and a set $C$ of cities.

- Cost $c_f$ to open facility $f \in F$
- Costs $d_{i_f}$ to connect city $i$ to facility $f$ (service $i$ from $f$).

**Problem 25 (Facility Location)** *We want a minimum cost solution subject to the constraint that every city is connected to some facility.*

Variant:

- can open at most $k$ facilities. This is $k$ median problem

We formulate the FLP as an IP.

$$\min \sum_{i \in C} \sum_{f \in F} d_{if} x_{if} + \sum_{f \in F} c_f y_f$$

$$s.t. \sum_{f \in F} x_{if} \geq 1$$

$$x_{if} \leq y_f$$

$$x_{if} \in \{0, 1\}$$

$$y_f \in \{0, 1\}$$

taking the linear relaxation we have (P)

$$\min \sum_{if} d_{if} x_{if} + \sum_{f} c_f y_f$$

$$s.t. \sum_{f} x_{if} \geq 1 \quad \forall i$$

$$y_f - x_{if} \geq 0 \quad \forall i, f$$

$$x_{if} \geq 0$$

$$y_f \geq 0$$

and (D)

$$\max \sum_{i \in C} \alpha_i$$

$$s.t. \ \alpha_i - \beta_{if} \leq d_{if} \quad \forall i, f$$

$$\sum_{i \in C} \beta_{if} \leq c_f \quad \forall f$$

$$\alpha_i \geq 0 \quad \forall i$$

$$\beta_{if} \geq 0 \quad \forall i, f$$

What are the complimentary slackness conditions? PCS

1. $x_{if} > 0 \Rightarrow \alpha_i - \beta_{if} = d_{if}$
2. $y_f > 0 \Rightarrow \sum_{i \in C} \beta_{if} = c_f$

DCS

1. $\alpha_i > 0 \Rightarrow \sum_f x_{if} = 1$
2. $\beta_{if} > 0 \Rightarrow y_f = x_{if}$

Suppose $F^* \subseteq F$ is an optimal integral solution, and $\sigma : C \to F^*$ is corresponding connection. We view $(\alpha, \beta)$ as "paying" for the primal solution.

We again want a dual solution that "approximately" pays for an integral primal solution. Unlike the other primal dual problems we have seen, now

- there are different types of constrains
- Not packing/covering
- there are negative coefficients

Here we relax PCS, but not DCS.

**Algorithm**

1. We raise $\alpha_i, \beta_i$ dual variables over time $t$
2. Raise $\alpha_i$ by 1 at time 1 etc...
3. If we reach $f$ then also raise $\beta_{if}$.
4. We do this simultaneously for each city. e.g. more than 1 city pay for a facility.
5. We stop growing $\alpha_i$, $\beta_{if}$ when
6. $i$ is connected to some facility that is completely "paid for"(has reached). We say such a facility is "temporarily open"
7. $i$ is connected to the first "temp open" facility $f$ it has reached $c(i) = f$. (in this case we say $f$ is a witness for $i$)
8. Algorithm stops when all the duals have stopped growing.

The problem is that a city may pay $\beta_{if} > 0$ to lots of facilities. So the dual does not actually pay for the integral solution we found. If you only contribute to one facility this would work. Let $\hat{F}$ = temporary open facilities. Take $\hat{F}$ we have $(f, f') \in E$ if there is some $i$ that contributes to both $f$ and $f'$. Take a *maximal* stable set $S \subseteq \hat{F}$. clearly no city contributes to more than 2 facilities in $S$. But most cities may not be connected. So for PHASE 2:

1. Connect $i$ to $f \in S$ if $\sigma(i) = f$ i.e. $f$ is a witness for $i$.
2. if $\sigma(i) = f' \notin S$ then there is some $(f, f')$ edge so connect $i$ to $f$.

How good is this algorithm? To get a 3 approximation algorithm APPROX PCS:

1. $x_{if} > 0 \Rightarrow \frac{1}{3}d_{if} \leq \alpha_i - \beta_{if} \leq d_{if}$
2. $y_f > 0 \Rightarrow \frac{1}{3}c_f \leq \sum_i \beta_{if} \leq c_f$

We prove something stronger.

- (2) is not relaxed.
- (1) relax indirectly connected cities. Not relaxed for directly connected cities.
- $i$ indirectly connected $\Rightarrow \beta_{if} = 0 \ \forall f$
- Directly connected cities pay for $S$.

$$\sum_{i:\sigma(i)=f} \beta_{if} = c_f$$

- Directly connected $i \Rightarrow \alpha_i = d_{i,\sigma(i)} + \beta_{i,\sigma(i)}$

**Lemma 12** *$i$ indirectly connected to $f \Rightarrow d_{if} \leq 3\alpha_i$*

**Proof.** $i'$ contributes $\beta_{i'f}, \beta_{i'f} > 0$. This means they both open after time $\max[d_{i',\sigma(i)}, d_{i',f}] \leq \alpha'_i \leq \min(t', t) \leq \alpha_i$. $\sigma(i)$ is a witness for $i$ , so $\alpha_i \geq t'$. We assume triangle inequality in $d$(otherwise its hopeless). So $d_{i'f} \leq 3\alpha_i$. ■

Therefore this is a 3 approximation algorithm.

## 24 Euclidian TSP

There is no PTAS for metric TSP unless $P = NP$. What if distances satisfy a stronger property than the triangle inequality? Here we take points on the plane(more generally in $R^d$).

**Theorem 28 (Arora)** *There is a PTAS for Euclidean TSP*

Assume:

- the maximum distance is $n^2 = L$ where $n$ is a power of 2(by scaling)
- We can round each vertex to the nearest unit grid point. Let $d(v, v')$ be the distance from $v$ to the nearest point on the grid. We have

$$Opt' \leq Opt + 2\sum_v d(v, v')$$

$$\leq Opt + 2n$$

$$Opt \geq 2n^2$$

i.e.

$$Opt' \leq (1 + \frac{1}{n})Opt$$

We want to use Dynamic programming. To do this we want to show there is a "near" optimal tour with a nice structure. This nice structure will allow us to search for it using dynamic programming(i.e. search all tours with this nice structure).

1. First we partition the grid up as follows. Draw two orange 'type 0' lines to divide up the grid into 4 equal regions. For each region repeat and draw two pink 'type 1' lines to divide the region into 4 equal regions. In general a level $i+1$ line is half the length of a level $i$ line.(although sometimes we will abuse notation and consider the union of level i+1 lines.)
2. We add 'portals' on each line.($m = \frac{2\log(n)}{\epsilon}$) equidistant $\frac{L}{2^i m} = \frac{\epsilon L}{2^i 2\log(n)}$

We say a tour if 'nice' if:

- It visits each vertex
- It crosses lines only at portals(or vertices)
- Portals can be used many times but no other point can.

We a tour is *very nice* if

- It visits portals at most twice

**Lemma 13** *Given a nice tour, there is a very nice tour of at most the same length*

**Proof.** Suppose a portal is used at least 3 times. Then we can short cut it. If you create a crossing at an interior point then shortcut that. ■

**Theorem 29** *The best very nice tour can be found by dynamic programing in time* $2^{O(m)} = n^{O(\frac{1}{\epsilon})}$

**Proof.** Any unit square has at most $4m$ portals. So there are at most $8m$ visits to these portals. This corresponds to at most $4m$ non-crossing paths between portals. Each portal is used $0, 1$ or $2$ times. i.e. $3^{4m} = n^{O(\frac{1}{\epsilon})}$ possibilities. There are $n^4$ squares. Given the portals(there are an even number of which say $r$) we have at most $2^{3r} \leq 2^{8m} = n^{O(\frac{1}{2})}$ valid pairings.

$2^{2r}$ can be shown using Catalan numbers. There are $p$ pairs of brackets so $\frac{1}{p+1}\binom{2p}{p} < 2^{2p}$. We store all these $n^{O(\frac{1}{\epsilon})}$ possible solution. For each square, given a bigger square $S$. There are $n^{O(\frac{1}{\epsilon})}$ enter/exit possibilities. What are their costs? The pairing must be consistent with the routings in $S_1 S_2 s_3 S_4$. We check all possibilities. Which we can do as we only have $n^{O(\frac{1}{\epsilon})}$ table entries for each of the 4 squares.

Finally at the top square the pairing should give a tour. ■

So we can find the best very nice tour. But this may not be near *Opt*. But there is if we randomly shift the partition by $(a, b)$. Let $\tau$ be an optimal tour. Let $N(\tau)$ be the number of times it crosses a grid line.

**Lemma 14** $N(\tau) \leq 4Opt$

**Proof.** Let $e \in \tau$ have distance $d_e \geq 1$. Say for $(x_1, y_1)$ to $(x_2, y_2)$ it crosses at most

$$|x_1 - x_2| + 1 + |y_1 - y_2| + 1 \leq 4d_e$$

■

**Lemma 15** *Given $\tau$ and a random shift, there is a very nice tour with expected cost at most $(1+4\epsilon)Opt$.*

**Proof.** When we move an edge to cross at a portel, the cost increases by at most the interportal distance. This distaince is $\frac{L}{2^i m}$ on level $i$ line. There are $2^i$ horizontel level $i$ lines, $L$ lines in total. So expected cost of moving to a portal

$$\sum_{level\ i} \frac{2^i}{L} \cdot \frac{L}{2^i m} = \sum_{level\ i} \frac{1}{m}$$

$$= \frac{2\log(n)}{\frac{2\log n}{\epsilon}} = \epsilon$$

We have at most $4Opt$ crossings . So total increase in cost is at most $\epsilon 4Opt$ ■

So a random one will be good. Or try all $n^4$ possibilities.

## 25 2 Connectivity

Assignment #4 question 5 should be $G-S$ is a tree, not bipartite.

A Graph $G$ is 2 edge connected if there are 2 edge disjoint paths between any pair of vertices. This is equivalent to saying $G$ has minimum cut at least 2 by Mengers theorem.

**Problem 26 (2EC)** *Given 2 edge connected graph $G$ find a minimum sized spanning subgraph $H \subseteq G$ such that $H$ is 2 edge connected.*

This is $NP$ hard. Includes Hamelton cycle as a special case.

To obtain an approximation algorithm we use a **DFS Tree**

- Back edge connects ancestor to descendent
- there are no cross edges.

Let $T_i$ be subtree of $T$ rooted at vertex $i$. Let $b_i$ be the deepest back edge emanating from $T_i$ (goes as close to $r$ as possible). This $b_i$ exists or $e_i$(the edge closest to $r$ touching $v_i$ is a cut edge.

**Algorithm**

```
1. Find $DFS$ tree $T$.  Set $H := T$
2. Work up from leaves.  At vertex $i$:  if $e_i$ is a cut edge in $H$ set $H := H \cup b_i$
```

This gives a feasible solution. $H$ is 2 edge connected. Take $e \in H$ $e = e_i$ tree edge. ($e_i$ is not a cut edge or we would have already added $b_i$). $e = b_i$ then cant be cut edge as $a-b$ path is in $T$. This is clearly Polytime.

This is a 2 approximation algorithm.

- $|T| = n-1$
- the number of back edges is at most $n-1$.

- *Opt* $\geq n$ (Hamelton cycle is best possible)

In fact the algorithm is better than this. We need to tighten the lower bound. Say we add $k$ back edges. Call these $b_{v_1}, b_{v_2}, ..., b_{v_k} = b_1, ..., b_k$. Let the corresponding tree edges be $e_1, ..., e_k$. Observe that $T - \{e_1, ..., e_k\}$ contains exactly $k+1$ components.Contracting components $V_1, ..., V_k$ to nodes gives a tree $\hat{T}$ with $k+1$ vertices. We claim that $\hat{T}$ is a **tree carving**

**Definition 7 (Tree Carving)** *A partition of $V$ into $W_1, ..., W_k$ such that*

- *Each $W_i$ is a node in a tree $\Gamma$.*
- *if $(x, y) \in E(G)$ and $x \in W_i$ then either $y \in W_j = W_i$ or $y \in W_j$ where $(W_i, W_j) \in E(\Gamma)$*

**Lemma 16** *$\hat{T}$ is a tree carving*

**Proof.** Take a non tree edge $(x, y) \in G$. Let $x_i \in V_i, y \in V_j$ ($V_i$ named after its top vertex $v_i$). But when we removed $e_i$ we added $b_i$ but this does not go as high as $y$. ∎

**Lemma 17** *If $G$ has a tree carving with $k+1$ pieces then*

$$Opt \geq 2k$$

**Proof.** Each edge $f \in \Gamma$ cuts the graph into two parts: $(S_f, \bar{S_f})$. Any Solution has to have at least 2 edges across a cut. No edge is in more then 1 of these cuts as we have a tree carving. i.e. *Opt* $\geq 2k$. ∎

**Theorem 30 (Khuller , Vishkin '94)** *Our Algorithm is a $\frac{3}{2}$ approximation algorithm.*

**Proof.** We use $n - 1 + k$ edges. *Opt* $\geq \max(n, 2k)$. So

$$\frac{|H|}{Opt} \leq \frac{n+k-1}{\max(n, 2k)} \leq \frac{(3/2)n - 1}{n} \text{ when } n \geq 2k, \ \frac{3k-1}{2k} \text{ else}$$

∎This approach can be used on connectivity problems as we will see on assignment 5...

## 25.1 Iterative Rounding

We consider the following edge connectivity problem

**Problem 27 (Generalized Survivable Network Design Problem)** *Given $G := (V, E)$ with edge cost $c_e$, requirments $r_{ij}$ for pairs $i, j \in V$. Find a minimum cost subgraph $H$ with $r_{ij}$ edge disjoint paths from $i$ to $j$ for all $i, j$.*

This is clearly $NP$ hard, since if we set $c_e = 1$, $r_{ij} = 2\ \forall ij$ we get the $2EC$ problem.
For any $S \subseteq V$ set

$$f(S) = \max_{i \in S, j \notin S} r_{ij}$$

By max flow min cut we can then model the problem as

$$\min \sum_{e \in E} c_e x_e$$

$$s.t. \sum_{e \in \delta(S)} x_e \geq f(S) \; \forall S \subseteq V$$

$$x_e \in \{0,1\}$$

So $f(S) \geq r_{ij}$ for any $S$ such that $i \in S, j \notin S$

Cant solve this so take the $LP$ relaxation.

Given $x$ does it satisfy the $LP$ constraints . If not give a violated constraint. For all $i, j$ is the max $-j$ flow(with capacities of $e = x_e$) at least $r_{ij}$. If no then there is some $i - j$ cut $S$ with capacity $< r_{ij} \leq f(S)$.

## 25.2 SVNP cont

$$\min \sum_{e \in E} c_e x_e$$

$$s.t. \sum_{e \in \delta(S)} x_e \geq f(S) \; \forall S \subseteq V$$

$$0 \leq x_e \leq 1$$

**Theorem 31 (Jain 00)** *The LP has an edge of weight at least $\frac{1}{2}$ ($x_e \geq \frac{1}{2}$).*

In fact, we have a stronger result. This theorem holds for any weakly supermodular function $f$.

**Definition 8 (weakly supermodular)** *A set function $g$ is weakly supermodular if $g(V) = 0$ and $\forall S, T \subseteq V$ either $g(S) + g(T) \leq g(S \cap T) + g(S \cup T)$ or $g(S) + g(T) \leq g(S - T) + g(T - S)$*

**Lemma 18** *$f$ is weakly supermodular.*

**Proof.** Notice we have 6 types of edges classified according to where their end points lie with respect to $S, T$. $f(S) = \max \; r_1 \; r_3 \; r_4 \; r_6$ where $r_i = \max_{p \textit{ a vertex pair following type } i} r_p$
$f(T) = \max \; r_1 \; r_2 \; r_3 \; r_6$ ... case analysis shows that $f$ is indeed supermodular. ■

## 25.3 Iterative Rounding

Solve $LP$ in iteration $i$. Select all edges $A_i$ such that $A_i = \{e_i : x_e \geq \frac{1}{2}\}$. $f^{i+1}(S) := f(S)^i - |\delta(S) \cap A_i| \forall S \subseteq V | \forall S \subseteq V$ (repeat until feasible)

**Lemma 19** *$f^{i+1}$ is weakly supermodular.*

$|\delta|$ is submodular $|\delta(S)| + |\delta(T)| \geq |\delta(S \cup T)| + |\delta(T \cap S)| \geq |\delta(S - T)| + |\delta(T - S)|$.

How do we update all $f^i(S)$ in polytime? We can still solve the new LP with respect to $f^{i+1}$. All we do is perform max flows with capacity if $e = 1$ if $e \in A_i$.

**Theorem 32** *This is a 2 approximation algorithm*

**Proof.** Induction on the number of iterations. Base case: $A_1$ is feasible:

$$c(A_1) \leq 2 \sum_{e \in A_1} c_e x_e \leq 2 \sum_{e \in E} c_e x_e = |LP\ sol| \leq 2Opt$$

Try

$$c(A_1 \cup ... \cup A_{r+1}) = c(A_1) + c(A_2 + \cup ... \cup A_{r+1})$$
$$\leq 2 \sum_{e \in E - A_1} x'_e c_e$$
$$\leq 2 \sum_{e \in E} c_e x_e + 2 \sum_{e \in E - A_1} x'_e c_e$$

But $[1, y]$ is feasible in iteration 2.

$$\leq 2 \sum_{e} c_e x_e$$

■

Assume $0 < x_e < 1$. If $x_e = 0$ throw $e$ away. If $x_e = 1$ we are done. We say set $S$ is tight if

$$x(\delta(S)) := \sum_{e \in \delta(S)} x_e = f(S)$$

(LP has a basis of $m$ tight linearly independent sets)Suppose $S, T$ cross, $S - T, T - S, S \cap T \neq \emptyset$.

**Theorem 33** *If $S, T$ are tight then either $S \cup T$, $S \cap T$ are tight or $S - T, T - S$ is tight.*

**Proof.** Assume $f(S) + f(T) \leq f(S - T) + f(T - S) \leq x(\delta(S - T)) + x(\delta(T - S))$

$$\leq x(\delta(S)) + x(\delta(T))$$
$$= f(S) + f(T)$$

So the first inequality is tight. $x_e > 0 \forall e$ so there are no edges of type 1, therefore $\chi_S + \chi_T = \chi_{S-T} + \chi_{T-S}$. ■

$L$ is laminar if no sets in $L$ cross.

**Theorem 34** *there is a laminar family of $m$ linearly independent tight sets .*

**Proof.** Take a maximal laminar set $L$ of tight sets. Suppose there is a tight set that is not spanned by $L$. Take such an $S$ such that it crosses the fewest number of sets in $L$. So $S$ crosses $T \in L$ . Assume $\chi_S = \chi_{S-T} + \chi_{T-S} - \chi_T$, $S - T, T - S$ are tight. Without loss of generality $S - T \notin span(L)$ But $S - T$ crosses fewer sets in $L$ than $S$ which is tight. ■

**Lemma 20** *There is a set $S$ in $L$ with $|\delta(S)| \leq 3 (\Rightarrow \exists e\ s.t. x_e \geq \frac{1}{3})$.*

**Proof.** Given $L$ we have a forest given by containment. There are $m$ edges and $2m$ endpoints. If the theorem is fals we assign end points to sets such that all sets have 2 endpoints, the roots have 4 endpoints. Assign endpoint to smallest set $S \in L$ that contains it. Base case: Leafs get at least 4 end points. If root $R$ has at least 2 children take 2 end points from both. If $R$ has only 1 child then if $R$ has exactly 1 end point assigned to it then $\delta(C), \delta(R)$ differ in 1 edge (as $f$ is integral) which is a contradiction. If it has no end points then $\chi_R = \chi_C$ i.e. not linearly independent for contradiction. ■

## 25.4 Vertex Connectivity

**Problem 28 (Vertex Connectivity)** *$SNDP - VC$ see edge version*

Let $r_{ij} :=$ the number of vertex disjoint paths between $i$ and $j$. Vertex connectivity is notoriously more difficult than edge connectivity.

**Theorem 35 (KKC)** *There is no approximation algorithm for $SNDP-VC$ with guarantee better than $2^{\log^{1-\epsilon} n}$ unless $NPL \subseteq DTIME(n^{polylog(n)})$.*

What about the special case of $k$ connectivity?

**Theorem 36 (CVV)** *There is a $O\log(k)$ approximation algorithm for $k$ vertex connectivity if $n \geq 6k^2$*

First lets formulate it as an $IP$. We say $(W_t \subseteq V, W_h \subseteq V)$ is a set pair if $W_t \cap W_h = \emptyset$. Let $\delta(W)$ be the edges between $W_t$ and $W_h$.

$$\min \sum_e c_e x_e$$

$$\sum_{e \in \delta(W)} x_e \geq k - (n - |W_t| - |W_h|) = f(W) \quad \forall set\ pairs\ W$$

$$x_e \in \{0,1\}$$

Notice that any $k$ connected subgraph must have $f(W)$ edges across $\delta(W)$. Moreover a solution to the $IP$ is $k$ connected.

We say the $LP$ relaxation is $LP(k)$ with solution $Opt(k)$. To describe the algorithm we use the following problem.

**Problem 29 (k-Outconn)** *Given root $r \in V$ . We want a subgraph $H \subseteq G$ that is $k - outconn$ from $r$. i.e. there are $k$ vertex disjoint paths from $r$ to any other vertex $v$.*

Interestingly the **directed** version of this can be solved as the following $LP$ is integral. let $\bar{W} := V - W_t - W_h$.

$$\min \sum_{a \in A} c_a x_a$$

$$s.t. \quad \sum_{a \in \delta^+(W)} x_a \geq k - (n - |W_t| - |W_h|) \ \forall W \ s.t. \ r \in W_t$$

$$0 \leq x \leq 1$$

e.g. the ellipsoid method gives an optimal solution in polytime(by max flow algorithms).

**Corollary 8** *There is a 2 approximation algorithm for undirected $k$ outconn*

**Proof.** Given $G$, get $D$ by replacing each undirected edge with two directed edges. Solve directed version. Clearly

$$Opt(D) \leq 2Opt(G)$$

Given $H^D$ solution to $D$, pick $H^U$, the underlying undirected graph for $H^D$.

$$cost(H^U) \leq cost(H^D) \leq 2Opt(G)$$

Note $cost(H^U) \leq 2Opt(k)$. ■

Finally a result from graph theory:

**Definition 9 (3 critical)** *A graph $G$ is 3 critical if every subset $S \subseteq V, |S| \leq 3$ is contained in some minimum vertex cut (separator). i.e. $\kappa(G - S) = \kappa(G) - |S|$.*

**Theorem 37 (Mader)** *A 3 critical graph with vertex connectivity $= k$ has $\leq 6k^2$ vertices.*

**Algorithm**

1. Start with $H_1 = MST$
2. Given $H_i$
   (a) Set $c_e = 0$ if $e \in H_i$
   (b) Find a non-critical set $R = \{r_1, r_2, r_3\}$ in $H_i$
   (c) Find a $i+1$ outconn subgraphs $R_1, R_2, R_3$ from $r_1, r_2, r_3$
   (d) Set $H_{i+1} = H_i \cup R_1 \cup R_2 \cup R_3$
3. Output $H_k$

To find $R$: there are $n^3$ possibilities. Test if $\kappa(H_i) = \kappa(H_i - R) + |R|$

**Lemma 21** *The algorithm is polytime and gives feasible solution.*

**Proof.** WTS $H_{i+1}$ is $i+1$ connected. Suppose not. Then there is some in-eparator $T$ in $H_{i+1}$. Some vertex of $R$, say $r_1$ is not in $T$. But $R_1$ is $i+1$ outconn therefore there is at least 1 edge across set pair. ■

CLAIM*$cost(R_1) \leq \frac{2Opt(k)}{k-i}$ for phase $i$ .*

If true

$$total\ cost \leq 6(1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k})Opt(k) \leq 6H_k \cdot Opt$$

as we will see in homework $cost(MST) \leq \frac{2Opt(k)}{k}$.

**Proof.** $R_1$ is $k+1$-outconn subgraph with cost at most $2Opt(i+1)$. Let $x$ be an optimal solution to $LP(k)$ and $x'$ feasible for $LP(i+1)$. Where $x'_e = 1$ for $e \in H_i$, $= \frac{x_e}{k-i}$ $e \notin H_i$. The result will follow. Take a set pair $W$.

Case 1 $H_i \cap \delta(W) = \emptyset$(other case similar). Let $q = |\bar{W}| = n - |W_t - |W_h|$. $q \geq i$ as $H_i$ is $i$ connected. Want

$$\sum_{e \in \delta(W)} x'_e \geq (i+1) - 1$$

Therefore $q = i$. So now

$$\sum_{e \in \delta(W)} x_e \geq k - q = k - i$$

So

$$\sum_{e \in \delta(W)} x'_e \geq \frac{k-i}{k-i} = 1$$

■

## 26 Minimum Degree Constrained Spanning Tree Problem

So far we have assumed that a $FPTAS/PTAS$ is the best approximation algorithm we can get for an $NP$ hard problem. However, we can do better if we allow "additive" approximation algorithms. For example there are 2 well known $NP$ hard problems for which we can get within one unit of *Opt*

- Coloring a planar graph . Try one, two colors, otherwise 4 color it.
- Edge coloring a graph. $Opt \geq \Delta(G)$ max degree. Vizing showed algorithmically $Opt < \Delta(G) + 1$

**Problem 30 (MDST)** *We want a spanning tree $T$ in $G := (V, E)$ such that the max degree in $T$ is minimized.*

This is $NP$ hard since it includes Hamelton Path. Thus it is likley that no $(\frac{3}{2} - \epsilon)$ approximation algorithm exists.

The key thing here is getting a good lower bound.

**Lemma 22** *Let $W \subseteq V$ and $G - W$ have $\gamma(W)$ components $comps(G - W)$. Then*

$$Opt \geq \lceil \frac{|W| + \gamma(W) - 1}{|W|} \rceil$$

**Proof.**
Contract the components of $G - W$. We have $\gamma(W)$ component nodes plus $|W|$ singleton nodes. i.e. $\gamma(W) + |W|$ nodes. Ao any tree needs $|W| + \gamma(W) - 1$ edges to connect these nodes. All these edges hits some vertex in $W$. ■

What will our $W$ be? $W := S_k + B$ where $B \subseteq S_{k-1}$ and $S_j = \{v : deg_T(v) = j\}$ where $k :=$ max degree in $T = \Delta(T)$.

**Theorem 38** *If $comps(G - W) = comps(T - W)$ then $Opt \geq k - 1$*

**Proof.** So there are no edges in $G$ between components in $T - W$. So the lemma applies:

$$Opt \geq \lceil \frac{|W| + \gamma(W) - 1}{|W|} \rceil$$

Let $|W| = n_k = |S_k| + n_{k-1} = |B|$

$$\gamma(W) \geq 1 + |E(W)| - |W|$$

$E(W) = \{e \in T | e \textit{ is adjacent to some vertex in } W\}$

$$|E(W)| \geq kn_k + (k-1)n_{k-1} - (|W| - 1)$$

here $|W| - 1$ is the maximum number of tree edges inside $W$.

$$= (k-1)n_k + (k-2)n_{k+1} + 1$$

$$\gamma(W) \geq (k-2)n_k + (k-3)n_{k-1} + 1$$

$$Opt \geq \lceil \frac{|W| + \gamma(W) - 1}{|W|} \rceil \geq \lceil \frac{(k-1)n_k + (k-2)n_{k-1} + 1}{n_k + n_{k-1}} \rceil$$

$$= (k-2) + \lceil \frac{n_k + 1}{n_k + n_{k-1}} \rceil \geq (k-1)$$

■

So we want to find a $(T, W)$ with this property.

Idea: if *comp*$(G - W) \neq$ *comp*$(T - w)$ then there is some edge $e = (u, v)$ between components of $T - W$. Suppose $deg_T(u), deg_T(v) \leq k - 2$ and there is some $w \in W \cap C$ such that $deg_T(w) = k$. Then add $e$ and remove an edge on $C$ incident to $V$. Therefore $|S_k|$ has decreased. (this is an improving move)

**Algorithm**

1. Start with $W = S_k + B$ where $B = S_{k-1}$. Then
   (a) Find $T$ with smaller $|S_k|$(improving move)
   (b) Or Reduce the size of $B \subseteq S_{k-1}$
   (c) Or *comps*$(T - W) =$ *comps*$(G - W)$

This is polytime as the number of iterations is polytime. i.e. within 1 of *Opt*. If there is an improving move, we have (a). As (c) is not true there is some $e = (u, v)$ connecting components of $T - W$. So $F_u, F_v$ are subtrees of $T$ in components containing $u, v$ respectively. We claim, by induction, that we can "redirect" $F_u, F_v$ so that $u, v$ have degree at most $k - 2$ in $T$. This is clearly true at the start since $B = S_{k-1}$.

Now cycle $C$ in $T' \cup e$ has only vertices in $B$, say $B' \subseteq B \subseteq S_{k-1}$. We set $B := B - B'$. ($|B'| \geq 1$) Adding $B'$ to *comps*$(T - W)$ creates a bigger component $F$. Take $w \in B' \subseteq F$. $w$ has degree $k - 1$. It is also on $C$ so if we need to we can remove an edge on $C$ adjacent to $w$ sch that $deg_T(w) \leq k - 2$. This maintains induction. i.e. $v \in F_v \Rightarrow$ redirect such that $deg_{T'}(v) \leq k - 2$ ■

## 27 2 Edge connected

$\frac{4}{3}$ approximation algorithm . Can assume that the graph is 2 vertex connected, otherwise we can paste together on a cut vertex.

## 28 Multicut and Sum Multicommodity flow

Given $G := (V < E)$ with edge capacities $v_e$ and pairs $(s_1, t_1), ..., (s_k, t_k)$

**Problem 31 (Multicut)** *Remove a minimum capacity set of edges that disconnects all pairs.*

**Problem 32 (Sum Multicommodity Flow)** *Find a flow that maximizes the sum of the satisfied pairs.(fractional and integral versions) We will be considering the fractional version.*

$P_i :=$ set of $s_i$ $t_i$ paths. $\mathbf{P} := \cup_i P_i$. We can formulate this as an $LP$:

$$\max \sum_{P \in \mathbf{P}} f_P$$

$$s.t. \sum_{P : e \in P} f_P \leq u_e$$

$$f_P \geq 0$$

The dual problem is

$$\min \sum_e u_e d_e$$

$$s.t. \sum_{e \in P} d_e \geq 1 \; \forall P \in \mathbf{P}$$

$$d_e \geq 0$$

The dual constraint says : for each pair $s_i, t_i$, the length of any $s_i, t_i$ path is at least 1. So the length of a shortest $s_i, t_i$ path is at least 1. Since we can find shortest paths in polynomial time, we have a separation oracle:

- Given $d$ check that each shortest path is at least 1.

Notice that an integral solution to the dual is a multicut. How can we turn a fractional solution into an integral one? (Intuitively the edges with large $d_e$ are more useful.) Given a dual solution we work on

- $G = (V, E)$
- Edge distances $d_e$
- Edge weights $w_e := u_e d_e$
- weights $w(s_i)$

Our algorithm takes at most $k$ phases. In each phase we find a cut that separates some pair $s_i, t_i$. We do this by "growing a region" around some $s_i$. This region $R_i$ induces the cut $\delta(R_i)$

1. Pick $s_i$
2. At time $t$: $R_i = \{v : d(s_i, v) \leq t\}$
3. .

4. .

$$Cap(R_i) = \sum_{e \in \delta(R_i)} u_e$$

$$w(R_i) = \sum_{e \in E} w_e \lambda_e + w(s_i)$$

Where for each $e = (u, v)$ , $\lambda_{uv} = 1 \ if \ u, v \in R_i \ \ = 0 \ if \ u, v \notin R_i$ and $\lambda_{uv} = \frac{t - d(s_i,u)}{d(s_i,v) - d(s_i,u)} \ e \in \delta(R_i), u \in R_i, v \notin R_i$ corresponds to the 'fractional' amount of the edge in the cut.

1. Pick $s_i$
2. At time $t$: $R_i = \{v : d(s_i, v) \leq t\}$
3. We stop growing when $Cap(R_i) \leq Mw(R_i)$ for some chosen $M$.

We claim that this process terminates before $t = \frac{t}{2}$. Hence $R_i$ separates $s_i$ and $t_i$ $(d(s_i, t_i) \geq 1)$. More over $R_i$ contains at most one of $\{s_j, t_j\}$ as otherwise

$$d(s_j, t_j) \leq d(s_i, s_j) + d(s_i, t_j) < \frac{1}{2} + \frac{1}{2} = 1$$

We repeat on $G_2 := G - R_i$ and pick some $s_r$ to be the new source, where $s_r, t_r \in G - R_i$. Observe

$$d_G(s_i, t_i) \leq d_{G_i}(s_j, t_j)$$

so using distances in $G$ is OK. No $R_i$ can contain a pair.

**Corollary 9** *$\cup\delta(R_i)$ is a multicut.*

It remains to show

- there is some $M$ such that the region stops growing before $t = \frac{1}{2}$.
- What approximation guarantee do we get.

**Proof.** of (1). Suppose it does not terminate before $t = \frac{1}{2}$. Then $\forall t \in [0, \frac{1}{2}]$

$$Cap(R_i > Mw(R_i)$$

This means we can observe:

$$dw(R_i) \geq cap(R_i)dt > Mw(R_i)dt$$

To see this take $e = (u, v) \in \delta(R_i)$. Notice that $u \in R_i$ at time $t = d_{G_i}(s_i, u)$ $v$ is added to $R_i$ at time $d_{G_i}(s_i, v)$. So

$$dw(R_i) = \sum_e w_e d\lambda_e =^\star \sum_{e \in \delta(R_i)} w_e \frac{1}{d(s_i, v) - d(s_i, u)} dt$$

So

$$dw(R_i) \geq \star \geq \sum_{e \in \delta(R_i)} \frac{w_e}{d_e} dt = \sum_{e \in \delta(R_i)} u_e dt$$

$$= Cap(R_i)dt$$

So

$$w(R_i) \textit{ increases exponentially } > Mw(R_i)dt$$

■

Technicality: $w(R_i) = w(s_i) + \sum_{e \in E} \lambda_e w_e$ . The initial weight is $w(s_i)$. The final weight is at most $w_i := w(s_i) + F^* := \sum_{e \in G_i} w_e$.

So

$$\int_{w_i}^{F^*+w_i} \frac{1}{w(R_i)} dw(R_i) > \int_{t=0}^{\frac{1}{2}} M \; dt$$

$$\log(F^* + w_i) - \log(w_i) > \frac{1}{2}M$$

i.e.

$$2\log(\frac{F^* + w_i}{w_i} = 2\log(\frac{F^*}{w_i} + 1) > M$$

e.g. $w_i = \frac{F^*}{k}$ thus $M \leq 2\log(k+1)$.

What is our approximation guarantee? Let $C$ be the set of edges given by the algorithm.

$$\sum_{e \in C} u_e = \sum_i Cap_{G_i}(R_i) \leq M \sum_i w_{G_i}(R_i)$$

$$\leq M \sum w_i + M \sum_{i: e \in R_i, e \in \delta(R_i)} \sum w_e$$

$$\leq M \sum w_i + M \sum_{e \in E} w_e$$

$$\leq 2M \sum_{e \in E} w_e = 2M(LP \; solution)$$

**Corollary 10** *There is an* $O(\log(k+1))$ *approximation algorithm for the multicut problem in general graphs.*

**Proof.** If $w_i = \frac{F^*}{k}$ then

$$\sum_{e \in C} u_e \leq 4\log(k+1) \cdot LP \; solution$$

■(This is basically tight)

**Corollary 11** max *integer flow sum is at most* max *fractional flow sum which is equal to* min *fractional multicut*

$$\leq O(\log(k)) \max \; frac \; flow \; sum$$

## 29 Open Problems

**Problem 33 (Shortest Vector)** *Given $n$ linearly independent vectors $v_1, ..., v_n \in Q^n$ find a shortest vector $v^*$ in the lattice generated by those.*

There is an exponential approximation algorithm. There is a constant lower bound $\frac{3}{2}$. QUESTION: Is there are polynomial approximation algorithm?

**Problem 34 (Directed Feedback Edge(Vertex) set)** *Given $D := (V, A)$ remove the minimum number of arcs $F \subseteq A$ such that $G - F$ is acyclic.*

- There is an $O(\log(n) \log\log(n))$ approximation algorithm.
- is there a $O \log(n)$ or a better approximation algorithm?

**Problem 35 (Directed Steiner Tree)** *$D := (V, A)$, $r \in V$ is the root , $T \subset V$ terminals, $c_a$ arc costs. Find a minimum cost arborescance rooted at $r$ containing all of $T$.*

- There is a $n^\epsilon$ approximation algorithm.
- Is there a polylog approximation algorithm?

**Problem 36 (Volume of Convex Body)** *$K$ is a convex body in $n$ dimensions. Given a separation oracle we ask: is $x \in K$ ? If yes give a separating hyper plane*

- Known can't approximate this to better than $expo(n)$ factor in polytime using a deterministic algorithm.
- Can do to within $1 + \epsilon$ with randomized algorithm

EXAM FRIDAY 10, MC 103