IIIS 2014 Spring: ATCS - Selected Topics in Optimization
Lecture date: Mar 19, 2014
Instructor: Jian Li TA: Lingxiao Huang Scribe: Fu Li
(Polishing not finished yet.)

1 Preliminary

1.1 Duality

For the primal-dual pair of LPs:

$$\begin{aligned} &\text{maximize } \langle c, x\rangle \\ &\text{s.t. } Ax \le b \end{aligned}$$

$$\begin{aligned} &\text{minimize } \langle y, b\rangle \\ &\text{s.t. } y^T A = c \\ &y \ge 0 \end{aligned}$$

Proposition 1 *(**Weak Duality**) For any feasible solution x for primal program and y for the corresponding dual program, we have $c^T x \le y^T b$.*

Proposition 2 *(**Strong Duality**) Let x^* and y^* be the optimal solution for the primal and the dual program (if exist) respectively, then $\langle c, x^*\rangle = \langle y^*, b\rangle$.*

1.2 Hitting Set Problem

Given a ground set of elements E, nonnegative costs c_e for all elements $e \in E$, and subsets $T_1, \ldots, T_p \subset E$, we want to find a minimum-cost subset $A \subset E$ so that A has a nonempty intersection with each subset T_i. We say that A hits each subset T_i.

In the former lecture, we have showed that there was a d-approximative solution by the primal-dual method, where $d = \max_i |T_i|$.

.

2 The feedback vertex set problem

2.1 Problem description

Given an undirected graph $G(V, E)$, each node i is associated by a weight w_i. Our goal is the find the set S of minimum cost such that $G(V/S)$ is acyclic.

2.2 Algorithm and Proof

The problem can be modeled by a LP problem:

$$\min \sum_{i \in V} w_i x_i$$

$$\text{s.t.} \sum_{e \in C_i} x_i \geq 1 \forall \text{ cycle } C_i,$$

$$x_i \in \{0, 1\} \quad \forall i \in V.$$

Here, x_i means whether element i is picked in S. And we relax the constraint $x_i \in \{0, 1\}$ to $x_i \geq 0$. Furthermore, any optimal solution x^* to this LP will have $x_i^* \leq 1$, for all $i \in V$.

Note that the number of circles can be exponential based on the size of graph. So the above LP may have exponential many constraints in the problem, which cannot be written down in polynomial time.

However, we can use primal-dual method to avoid the exponential restriction. If we take the dual of the resulting linear program, we obtain the following:

$$\max \sum_{C} y_c$$

$$\text{s.t.} \sum_{C: i \in C} y_c \leq w_i,$$

$$y_i \geq 0.$$

Although the number of variables in the dual is exponential, only a polynomial number of these will be considered about by our primal-dual algorithm 1 .

Our strategy is similar with hitting set algorithm. We take care that the dual solution y stays feasible all the time and increase the x values towards making the primal solution feasible. When the primal solution becomes feasible, we stop and have a solution.

And the cost our chosen solution is as follows:

$$\sum_{i \in S} w_i = \sum_{i \in S} \sum_{C: i \in C} y_C = \sum_{C} |S \cap C| y_C.$$

That is, if we show that $|S \cap C| \leq \alpha$, then we can get a the solution $\sum_{i \in S} w_i \leq \alpha \sum_C y_C \leq \alpha OPT$.

However, there may be a circle C_0 with n nodes in the graph which makes the $|S \cap C|$ quite large. Thus we need choose the circles more carefully. For doing better, remove all the nodes which are not contained in any circle (since these nodes' corresponding variables can not appear in the above linear programs. Namely we repeatedly delete all the node with degree 1 until the graph contains only vertices of degree two or higher. And notice that for each circle, choosing the vertex of least cost of it is enough for a feasible solution. Thus we can ignore other vertices of the circle; the remaining vertices we call *Interesting* vertices.

Lemma 3 *There is a circle with at most* $4 \log n$ *Interesting vertices.*

Proof: Namely from G we create a graph G' consisting of only interesting vertex. Namely G' has no degree 1 vertices, and each vertex of degree 2 has two neighbors of degree more than 2. And there is a one-to-one correspondence of cycles in G' and G

Now we use BFS to find a cycle in G'. If we revisit a node on the breadth-first search tree, then we immediately find a circle. And observe that in every 2 levels, there must be a level of the nodes with degree all larger than 2. That is, in every other level the number of explored nodes increase a factor of 2. Moreover, by depth $2\log_2 n$, we must revisit a node and namely we will have found a cycle. □

Algorithm 1: Feedback vertex algorithm

```
Initially, let S = ∅, I = {Interesting vertices};
while S is not feasible do
    Find C such that {C ∩ I} < 4 log n;
    Increasing y_c until ∃i ∈ I, Σ_{C:i∈C} y_c = w_i.;
    S = S + {i};
    Remove i from the graph.;
    Repeatedly remove the vertices of degree one from the graph.;
Return S;
```

Then

$$\begin{aligned}\sum_{i\in S} w_i &= \sum_{i\in S}\sum_{C:i\in C} y_c \\ &= \sum_C y_c |A\cap C| \\ &\le \sum_C y_c |I\cap C| \\ &\le O(\log n)\sum_C y_C.\end{aligned}$$

3 Shortest path problem

3.1 Problem Description

Given an undirected graph $G=(V,E)$, nonnegative costs $c_e \ge 0$ on all edges $e\in E$, and a parit of distinguished vertices s and t. The goal is to find the minimum-cost path from s to t.

3.2 Algorithm and Proof

Denote $\delta(S)$ by the set of all edges that have one endpoint in S and the other endpoint not in S. Let x_e represent whether e is chosen in the shortest path. The problem can be modeled as follows:

$$\min \sum_e c_e x_e$$

$$\text{s.t.} \sum_{e\in\delta(S)} x_e \geq 1 \quad \forall S \subset V, s \in S, t \notin S,$$

$$x_e \in \{0,1\}, \quad \forall e \in E$$

For a feasible solution, consider about the graph $G' = (V, \{e|x_e = 1\})$. The first constraint means that for every $s-t$ cut S in the graph G', there must be one edge in $\delta(S)$. Namely the first constraint guarantees that the minimum $s-t$ cut is at least 1. By the max-flow min-cut theorem, we know there must be $1-$unit flow from s to t, that is there is a path from s to t in G'.

Similarly with the feedback vertex problem, the number of constraints is exponential of the size of graph. However, it does not matter that we use the primal-dual method. Relax the integrality constraints to $x_e \geq 0, e \in E$ and take the dual, we obtain the following:

$$\max \sum_{S:s\in S,t\notin S} y_S$$

$$\text{s.t.} \sum_{S:e\in\delta(S),s\in S,t\notin S} y_S \leq c_e,$$

$$y_i \geq 0.$$

Algorithm 2: Shortest Path algorithm

```
Initially, let A = ∅, l = 0;
while A is not feasible do
    l = l + 1;
    Find S as the connected component of (V, A) containing s;
    Increasing y_S until ∃e_l ∈ I, Σ_{S:e∈δ(S)} y_S = c_{e_l}.;
    A = A + {e_l};
Reverse deletion: for j from l to 1 do
    if A − {e_j} feasible then
        A = A − {e_j}
Return A;
```

$$\begin{aligned}\sum_{e\in A} c_e &= \sum_{e\in A}\sum_{S:e\in\delta(S)} y_S \\ &= \sum_{S:s\in S,t\notin S} y_s |A \cap \delta(S)|\end{aligned}$$

For all $y_S \neq 0$, we know y_S can increase only when S become a connected component of the edges picked by algorithm at some time point. And after the reverse deletion, the edges of A only can be in a path. That is, by max flow min cut theorem, the $\delta(S)$, as a $s-t$ cut, contains only one of these edges. Namely $|A \cap \delta(S)| = 1$.

Note that the algorithm is just Dijkstra's algorithm.

4 The Steiner forest

Given graph $G(V, E)$ with l pairs (s_i, t_i). Each edges has a cost c_e. Find a set of edges such that s_i can connect t_i.

4.1 Algorithm and Proof

Let $\mathcal{S}_i$ be the subsets of all possible $s_i - t_i$ cut; that is $\mathcal{S}_i = \{S \subset V : |S \cap \{s_i, t_i\}| = 1\}$.

$$\min \sum_e c_e x_e$$

$$\text{s.t.} \sum_{e \in \delta(S)} x_e \geq 1, \quad \forall S \subset V : S \in \mathcal{S}_i \text{ for some i},$$

$$x_e \in \{0, 1\}, \quad \forall e \in E.$$

The constraints guarantee that for an $s_i - t_i$ cut S, there will be a chosen edge from $\delta(S)$. By a similar discussion in the Shortest Path problem, it means that there is a path connecting s_i with t_i.

Also relax the integrality constraints to $x_e \geq 0, e \in E$ and take the dual as follows:

$$\max \sum_{S \in \cup_{i \in [l]} \mathcal{S}_i} y_S$$

$$\text{s.t.} \sum_{S : e \in \delta(S)} y_S \leq c_e, \quad \forall e \in E,$$

$$y_S \geq 0, \quad S \in \cup_{i \in [l]} \mathcal{S}_i.$$

Algorithm 3: Steiner Forest algorithm

```
Initially, let A = ∅, l = 0 (l is a counter);
while A is not feasible do
    l = l + 1;
    Let C_l = {S:S is a connected component of (V, A) separating some (s_i, t_i)};
    Increasing y_S for each S in C_l by Δ_i simultaneously until ∃e_x ∉ A, Σ_{S:e_x∈δ(S)} y_S = c_{e_x};
    A = A + {e_x};
Reverse deletion: for j from l to 1 do
    if A − {e_j} feasible then
        A = A − {e_j}
Return A;
```

Then

$$\begin{aligned} \sum_{e \in A} c_e &= \sum_{e \in A} \sum_{S : e \in \delta(S)} y_S \\ &= \sum_{S \subset V} y_S |A \cap \delta(S)|. \end{aligned}$$

Concerning every iteration of the algorithm, the dual variable y_S can be increased with Δ_i only when S belongs to C_i. That is,

$$y_S = \sum_{i:S\in C_i} \Delta_i, \sum_{S\subset V} y_S = \sum_{i=1}^{l} |C_i|\Delta_i.$$

Thus,

$$\begin{aligned}\sum_{e\in A} c_e &= \sum_{S\subset V} \left(\sum_{i:S\in C_i} \Delta_i \right) |A \cap \delta(S)| \\ &= \sum_{i=1}^{l} \sum_{C\in C_i} |A \cap \delta(C)|\Delta_i.\end{aligned}$$

To get an approximating ratio about the solution , we introducing the following lemma:

Lemma 4

$$\sum_{C\in C_i} |A \cap \delta(C)| \leq 2|C_i| \quad \forall i \in [l].$$

By the Lemma, we know the cost of our solution $\sum_{e\in A} c_e$ is no larger than $\sum_{i=1}^{l} |C_i|\Delta_i = 2\sum_{S\subset V} y_S \leq 2OPT$. Namely, the above algorithm is a 2-factor approximation algorithm to the Steiner forest problem. Now we back to proof the Lemma.

Proof: For a fixed $i \in [l]$, the edge e_i is added to A. Assume at this time the set of chosen edges $e_1, \ldots, e_i$ as A_i. Let $T_i = A - A_i$. Observe that $G(V, A_i \cup T_i)$ is a forest.

Consider about the contracting way for $G = (V, A)$ such that contracting each connected component of $G = (V, A_i)$ to a new vertex. Assume the set of vertices in the new graph G' as V'. Then the new graph $G' = (V', T_i)$ is also a forest. Since from no edges in the graph $G = (V, \emptyset)$, any time the algorithm add an edge to the graph , it has at most one endpoint in any given connected component in G. Namely at any time, the graph will remain a forest.

Further, color the new vertices added after the contraction with the red color and other vertices with the blue color. Let R denote the set of red vertices and B the blue vertices v. The desired inequality reduces to prove

$$\sum_{v\in R} deg(v) \leq 2|R|.$$

Observe that no blue vertex has degree less than 2. First of all, ignore blue vertex of degree 0. Assume by contradiction that there is a blue vertex v of degree 1 and the incident edge is e. Then the edge e must be on a path from s_i to t_i for some i otherwise e will be delete from A when the algorithm do the reverse deletion. Thus v must be the endpoint of the path in the contracting graph G', namely be a red vertex. It is a contradiction. Thus no blue vertex has degree less than 2.

Furthermore we have the following:

$$\begin{aligned}\sum_{v\in R} deg(v) &= \sum_{v\in V'} deg(v) - \sum_{v\in B} deg(v)\\ &\le 2|V'| - 2 - \sum_{v\in B} deg(v) \ \text{ since } G'(V', T_i) \text{ is a forest}\\ &\le 2|V'| - 2|B|\\ &\le 2|R|.\end{aligned}$$

□

5 Facility location Problem

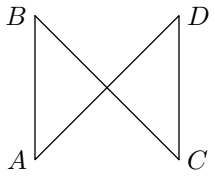
5.1 Problem description

Given a finite set $\mathcal{C}$ of customers (or clients); and a finite set $\mathcal{F}$ of potential facilities. Then there is a fixed cost $f_i \in \mathbb{R}_+$ for opening each facility $i \in \mathcal{F}$ and a service cost $c_{ij} \in \mathbb{R}_+$ for each $i \in \mathcal{F}$ and $j \in \mathcal{C}$. we look for: a subset S of facilities (called open) and an assignment $\sigma : C \to S$ of customers to open facilities, such that the sum of facility costs and service costs $\sum_{i\in S} f_i + \sum_{j\to i} c_{ij}$ is minimum.

• **Discussion**

For the above description, we can show the approximating ration is only $\Theta(\log n)$.

Here we only show why the lower bound $\Omega(\log n)$ holds. It is easy to see that the Set Covering Problem is a special case of the Facility Location Problem: Given an instance $(U; \mathcal{S}; c)$ as above, define $\mathcal{C} := U, \mathcal{F} := \mathcal{S}, f_S = c(S)$ for $S \in \mathcal{S}$, and let the service cost c_{ij} be zero for $j \in S$ and 1 for $j \in UnS$. Therefore the best we can hope for is a logarithmic approximation factor.



The follows, we only consider about the special form of the Facility location Problem: *Metric* Facility location Problem where the clients and factories are points in metric space and the assignment cost c_{ij} is the distance between factory i and client j. In particular, given clients A, C and factories B, D such that

$$c_{CB} \le c_{AB} + c_{AD} + c_{CD}.$$

In this note, we only give a 3-approximating algorithm for the Metric facility location Problem by Prime-Dual method.

5.2 LP

$$\min \sum_{i\in\mathcal{F}, j\in\mathcal{C}} c_{ij}x_{ij} + \sum_{i\in\mathcal{F}} f_i y_i$$

$$\text{s.t.} \sum_{i \in \mathcal{F}} x_{ij} \geq 1 \quad \forall j \in \mathcal{C}$$

$$y_i - x_{ij} \geq 0$$

$$y_i, x_{ij} \in [0,1] \forall i \in \mathcal{F}, j \in \mathcal{C}$$

Here, x_{ij} denote whether client j has been assigned to facility i and y_i denote whether facility i is open. The first constraint make sure that each client must be assigned to a facility and the second constraint guarantees that the facility which is assigned by a client must be open.

Also we relax the prime LP by dropping the integrality constraints and take the dual:

$$\max \sum_{j \in \mathcal{C}} \alpha_j$$

$$\text{s.t.} \sum_{j \in \mathcal{F}} \beta_{ij} \leq f_i \quad \forall i \in \mathcal{F},$$

$$\alpha_j - \beta_{ij} \leq c_{ij} \forall i \in \mathcal{F}, j \in \mathcal{C}$$

$$\alpha_i, \beta_{ij} \in [0,1]$$

Here we have an intuition for the dual. The dual variable β_{ij} is the facility j allocate the additional cost for client i. The dual variable α_j can be seen as the amount that each client j will pay. And if $\beta_{ij} > 0$, we say the client j contributes to factory i. For a fixed factory i, there is a constraint $\sum_j \beta_{ij} \leq f_i$ guarantee that the factory cannot receive the contribution from all clients exceed f_i. And for a fixed client j, there are constraints $\alpha_j - \beta_{ij} \leq c_{ij} \forall i \in \mathcal{F}$ guarantee that the client j only can pay the smallest value among the possible choice $c_{ij} + \beta_{ij}$.

The goal is to max the sum of the payment of clients under the above constraint.

Assume the optimal solution of the dual is α_j^*, β_{ij}^*. Observe that given dual variables a^*, we can get a feasible solution for β such that: $\beta_{ij} = \max\{0, \alpha^* - c_{ij}\}$.

If $\alpha_j^* > c_{ij}$ for some factories j, then $\beta_{ij} > 0$. Moreover the difficult of the approach is that the client j can arbitrary choose many facilities j where $\alpha_j = \beta_{ij} + c_{ij}$ whereas a feasible solution demands each client must choose only one factory. Thus for getting a solution, we should choose a subset of facilities to open more carefully. In the following algorithm, the reverse deletion part guarantee that the subset they open is such that no open facility is within a path of length two of any other open facility. Namely no client can contribute more than one factory in the subset.

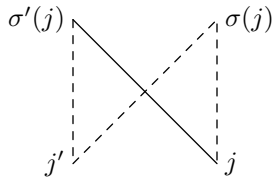
Algorithm 4: Facility location algorithm

Initially, let $T = \emptyset$,$S = \mathcal{C}$; $\sigma(j) = 0, \forall j \in \mathcal{C}$; //$\sigma$ is the assignment of customers to facilities
while $S \neq \emptyset$ **do**
    Increase α_j for all $j \in S$ uniformly and when $\alpha_i > c_{ij}$, increase all the w_{ij} uniformly to maintain feasibility until some $i \notin T$ such that $\sum_j \beta_{ij} = f_i$ or some $i \in T$ such that $\exists j \in S : \alpha_i - \beta_{ij} = c_{ij}$;
    if *some* $i \in T, j \in S$ *such that* $\alpha_i - \beta_{ij} = c_{ij}$ **then**
        $S = S - \{j\}$;
        $\sigma(j) = i$;
    if *some* $i \notin T$ *such that* $\sum_j \beta_{ij} = f_i$ **then**
        $T = T + \{i\}$;
        $S = S - \{j | \beta_{ij} > 0\}$;
        $\sigma(j) = i, \forall j : \beta_{ij} > 0$
Let $T' = \emptyset$;$\sigma' : \sigma'(j) = 0, \forall j \in \mathcal{C}$;
Reverse deletion: **while** $T \neq \emptyset$ **do**
    Pick $i \in T$; $T' = T' + \{i\}$;
    for *all* $j \in \mathcal{C}$ **do**
        //If some client j contributes to h and i, we guarantee that the assignment is from the client j to the picked factory i;
        if $\sigma(j) \in \{h \in T | \exists j \in \mathcal{C}, \beta_{ij} > 0 \wedge \beta_{hj} > 0\}$ **then**
            Let $\sigma'(j) = i$;
    // And remove all the factories h if some client j contributes to h, i;
    $T = T - \{h \in T | \exists j \in \mathcal{C}, \beta_{ij} > 0 \wedge \beta_{hj} > 0\}$;
Return T', σ';

Lemma 5 *If the client j which $\sigma(j)$ does not belong to T', then after the reverse deletion we can find a $\sigma'(j) \in T'$ such that:*

$$c_{\sigma'(j)j} \leq 3\alpha_j \quad j : \sigma(j) \notin T'.$$

Proof: The algorithm stops increasing α_j only when we find some $\sigma(j) \in T$ such that $\alpha_j = \beta_{\sigma(j)j} + c_{\sigma(j)j}$. Obviously $\sigma(j) \notin T'$ otherwise $\sigma(j) \in T'$. Then the $\sigma(j)$ only can be removed in the reverse deletion, which means there is some $j' \in \mathcal{C}$ such that $\beta_{\sigma(j)j'}, \beta_{\sigma'(j)j'} > 0$. Therefore $\alpha_{j'} = \beta_{\sigma(j)j'} + c_{\sigma(j)j'} > c_{\sigma(j)j'}, \alpha_{j'} > c_{\sigma'(j)j'}$ and also $\alpha_j = \beta_{\sigma(j)j} + c_{\sigma(j)j} \geq c_{\sigma(j)j}$.



By the metric property, we know

$$c_{\sigma(j)j} \leq c_{\sigma(j)j'} + c_{\sigma'(j)j'} + c_{\sigma'(j)j} \leq \alpha_j + 2\alpha_{j'}.$$

We would now like to show $a_{j'} \leq a_j$, therefore we can get $c_{\sigma(j)j} \leq 3a_j$.

observe that the algorithm stops incensing α_j only when the algorithm decides the value of $\sigma(j)$. Namely at that time, there are two cases: 1.$\sigma(j)$ is already in T;2.$\sigma(j)$ is added to T by $\sum_{k\in\mathcal{C}} \beta_{\sigma(j)k} = f_{\sigma(j)}$ being tight. For case 1: since $\beta_{\sigma(j)j'} > 0$, the algorithm must stop increasing $\alpha_{j'}$ before $\sigma(j)$ has been already added to T. For case 2: the $\alpha_j, \alpha_{j'}$ will stop increasing at the same time. Thus $\alpha_{j'} \leq \alpha_j$, as claimed. □

Notice that there are two assignment functions σ, σ' in our algorithm. The latter one is the assignment we finally get. The form one is decided by some tight constraints by increasing α_j, β_{ij}. That is, for some $j \in \mathcal{C}$, $\sigma(j)$ is the value that $\alpha_j = \beta_{\sigma(j)j} + c_{\sigma(j)j}$. And for each picked factory i in T, $\sum_{j:\sigma(j)=i} \beta_{ij} = f_i$.

$$
\begin{aligned}
\sum_{j\in\mathcal{C}} c_{\sigma'(j)j} + \sum_{j\in T'} f_j &= \sum_{i\in T'}(f_i + \sum_{j:\sigma'(j)=i} c_{ij}) \\
&= \sum_{i\in T'}(f_i + \sum_{j:\sigma(j)\notin T',\sigma'(j)=i} c_{ij} + \sum_{j:\sigma(j)=i} c_{ij}) \\
&= \sum_{i\in T'}\sum_{j:\sigma(j)\notin T',\sigma'(j)=i} c_{ij} + \sum_{i\in T'}\sum_{j:\sigma(j)=i}(\beta_{ij} + c_{ij}) \quad \text{(By the above discussion.)} \\
&\leq \sum_{i\in T'}\sum_{j:\sigma(j)\notin T',\sigma'(j)=i} 3\alpha_j + \sum_{i\in T'}\sum_{j:\sigma(j)=i}\alpha_j \\
&\leq 3\sum_{i\in T'}(\sum_{j:\sigma(j)\notin T',\sigma'(j)=i} a_j + \sum_{j:\sigma(j)=i} a_j) \\
&= 3\sum_{i\in T'}\alpha_i \\
&\leq 3OPT
\end{aligned}
$$

References

[1] Goemans M X, Williamson D P. The primal-dual method for approximation algorithms and its application to network design problems[J]. Approximation algorithms for NP-hard problems, 1997: 144-191.

[2] Shmoys D B, Tardos , Aardal K. Approximation algorithms for facility location problems[C]//Proceedings of the twenty-ninth annual ACM symposium on Theory of computing. ACM, 1997: 265-274.

[3] Williamson D P, Shmoys D B. The design of approximation algorithms[M]. Cambridge University Press, 2011.