COMPSCI 532: Design and Analysis of Algorithms	October 28, 2015
Lecture 15	
Lecturer: Debmalya Panigrahi	*Scribe: Allen Xiao*

1 Overview

In this lecture, we introduce the primal-dual method, an algorithmic pattern widespread in combinatorial optimization. Many familiar algorithms can be interpreted as primal-dual algorithms, including Kruskal's algorithm, Dijkstra's algorithm, and the Hungarian algorithm.

2 Primal-Dual Method

Primal-dual algorithms solve linear programming problems without solving the LP directly, in contrast to the LP rounding schemes we saw before and similar to dual-fitting algorithms. Instead, primal-dual uses the LP and complementary slackness to guide the construction of a feasible primal solution (often alongside some dual whose cost bounds the primal). First, we remind the reader of complementary slackness:

Theorem 1 (Complementary Slackness)**.** *Let x^* and y^* be the optimal primal and For each primal variable x_i and its corresponding dual constraint, either $x_i^* = 0$ or y^* is tight on the i-th dual constraint. Similarly for the j-th primal constraint and dual variable y_j.*

The pattern for primal-dual is roughly:

1. Initialize infeasible primal $P = 0$ and feasible dual $D = 0$.
2. Select some subset of dual variables to be *active*.
3. Increase the value of the active duals until a dual constraint becomes tight.
4. Complementary slackness allows the primal variables corresponding to the new tight dual constraints to be nonzero. Update the primal somehow, depending on the problem.
5. Redefine active duals somehow, depending on the problem, and repeat until the primal is feasible.

3 Primal-Dual Minimum Spanning Tree (Kruskal's)

Recall the formulation of the minimum spanning tree problem as an LP (there are multiple, but let's see one).

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in S} x_e \geq 1 \quad \forall S \subseteq V \\ & x_e \geq 0 \qquad \forall e \in E \end{aligned}$$

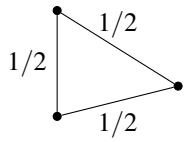


Figure 1: The fractional optimal solution to a triangle with unit cost edges. Any integer solution must use 2 edges (total cost 2), while this fractional solution has total cost $3/2$.

As it turns out, this linear program has an integrality gap; we can construct instances where the integer optimal solution is arbitrarily close to 2 times the fractional optimal. A simple example with non-unit gap is a triangle with uniform cost on edges.

A stronger formulation does not have this integrality gap. Instead of placing our constraints over cuts, we will use *partitions* on the vertices. Let $\pi = \{\pi_1, \pi_2, \ldots, \pi_k\}$ be a partitioning of the vertices into k clusters.

Definition 1. *Let X be a set. A **partition** of X is a collection of clusters $\pi = (\pi_1, \pi_2, \ldots)$, where each cluster is a subset of elements of X satisfying:*

1. *No π_i is empty.*
2. *The union of all $\pi_i \in \pi$ forms X.*
3. *All π_i are mutually disjoint.*

Since we are partitioning V, the number of clusters in the partition is always finite. For any partition, the MST must span the clusters π_i (otherwise, we can find an empty cut in the graph).

The constraint we use for the linear program will therefore be: for any partition π of V, there must be enough edges between clusters to span the clusters. For a partition with k clusters, there must be at least $k-1$ edges to span. We will use the notation $|\pi|$ for the number of clusters in π.

$$\sum_{e \in (\pi_1, \ldots, \pi_k)} x_e \geq k - 1$$

Let Π be the set of partitions of V, and let the notation $e \in \pi$ say that e spans two clusters in partition π. The complete program we will use is:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \pi} x_e \geq |\pi| - 1 \quad \forall \pi \in \Pi \\ & x_e \geq 0 \qquad\qquad\quad \forall e \in E \end{aligned}$$

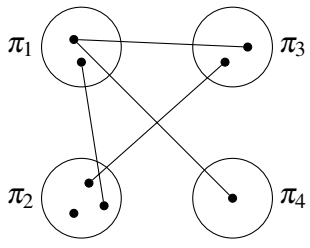


Figure 2: A partition π of vertices. The edges of the partition are those between clusters.

The dual is:

$$\begin{aligned} \max \quad & \sum_{\pi \in \Pi} (|\pi| - 1) y_\pi \\ \text{s.t.} \quad & \sum_{\pi : e \in \pi} y_\pi \leq c_e \qquad \forall e \in E \\ & y_\pi \geq 0 \qquad \forall \pi \in \Pi \end{aligned}$$

Fitting the primal-dual method to this primal-dual pair:

1. Initially, all $x_e = 0$ and all $y_\pi = 0$.
2. The active dual will be a single partition, the partition of connected components of the current solution. Initially, this is the singleton partition (each vertex in its own set).
3. Increase the active duals uniformly until a constraint becomes tight. Since the singleton partition "participates" in every edge's constraint, the first dual constraint to become tight will be the one for the minimum-cost edge.
4. For each edge e whose dual constraint became tight, set $x_e = 1$ (add e to the MST).
5. The tight constraints make it infeasible to increase the current active dual any longer. The new partition of connected components merges the clusters surrounding e, and no longer contributes to e's constraint as we expect.

Algorithm 1 (Primal-Dual Algorithm for MST)

$t \leftarrow 0$
$F \leftarrow 0$
while F is not a spanning tree **do**
    $\pi \leftarrow$ COMPONENTS(F)
    $y_\pi \leftarrow \min_{(u,v) \in \pi} (c_{uv} - t)$
    $e \leftarrow \operatorname{argmin}_{(u,v) \in \pi} (c_{uv} - t)$
    $F \leftarrow F \cup \{e\}$
    $t \leftarrow t + y_\pi$
end while

(a) Active partition $\pi^{(1)}$.

(b) Active partition $\pi^{(2)}$.

Figure 3: Active partition after e_2 becomes tight (effectively, e_2 is contracted). e_1 is still an edge of the partition, so next iteration the value of its dual constraint will be $y_{\pi^{(1)}} + y_{\pi^{(2)}}$. In contrast, the constraint on e_2 is tight and receives no more dual contributions after this iteration.

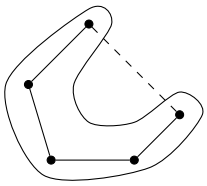


Figure 4: Proving that F must be acyclic. If there was somehow a cycle in F, the last edge added (dashed in figure) must have been contained in the connected component of the other edges. This is a contradiction, since edges within connected components do not have their dual values change after the component forms (and therefore cannot be picked).

The active dual each iteration is exactly "the partition of connected components of selected edges". Since the active dual only adds to constraints for edges between its clusters, the edge added ($x_e = 1$) will be "the minimum cost edge not already in a connected component", the same choice as Kruskal's.

3.1 Proof of optimality

To conclude, we will prove that this primal-dual algorithm is produces an optimal solution. We will use strong duality to show that the constructed primal solution is optimal. That is, we want to show that for the final primal solution x and the dual y:

$$\sum_{e \in E} c_e x_e = \sum_{\pi \in \Pi} (|\pi| - 1) y_\pi$$

Let F be the set of added edges ($x_e = 1$) in the final output. We can write the primal objective value as:

$$\begin{aligned} \sum_{e \in E} c_e x_e &= \sum_{e \in F} c_e \\ &= \sum_{e \in F} \sum_{\pi : e \in \pi} y_\pi \\ &= \sum_{\pi \in \Pi} |\pi \cap F| y_\pi \end{aligned}$$

Thus, if we can show that $|\pi \cap F| = |\pi| - 1$ for these dual variables, then the rest of the strong duality argument follows.

Lemma 2. *For partitions π where $y_\pi > 0$, $|\pi \cap F| = |\pi| - 1$.*

Proof. For any $y_\pi > 0$, it was updated once from $y_\pi = 0$ when it became the active dual. At that point, π formed the connected components of the intermediate F. If F is acyclic, then the number of its edges on the graph of connected components should be exactly $|\pi| - 1$.

Assume for contradiction that F has a cycle. One edge was added last, but at that poin the rest of the edges formed a connected component, and therefore at edge was not a spanning edge of the active dual (partition). F must therefore span the components of π, so with $|\pi|$ components there are $|\pi| - 1$ edges in $\pi \cap F$.

□

Finally, optimality follows from strong duality:

$$P = \sum_{e \in E} c_e x_e = \sum_{\pi \in \Pi} y_\pi |\pi \cap F| = \sum_{\pi \in \Pi} y_\pi (|\pi| - 1) = D$$

4 Primal-Dual Steiner Forest Approximation

Let $\mathscr{S}$ be the set of *separating cuts* of G, where for at least one terminal pair i, s_i is in one partition and t_i in the other.

$$\mathscr{S} = \{S \subseteq V \mid \exists i \quad |S \cap \{s_i, t_i\}| = 1\}$$

The following is a linear program for Steiner forest:

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in S} x_e \geq 1 \quad \forall S \in \mathscr{S} \\ & x_e \geq 0 \qquad \forall e \in E \end{aligned}$$

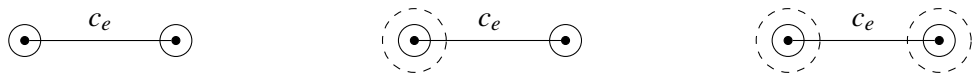
Its dual is:

$$\begin{aligned} \max \quad & \sum_{S \in \mathscr{S}} y_S \\ \text{s.t.} \quad & \sum_{S : e \in (S, \bar{S})} y_S \leq c_e \quad \forall e \in E \\ & y_S \geq 0 \qquad \forall S \in \mathscr{S} \end{aligned}$$

Intuitively, connectivity requirements don't cross non-separating cuts. The optimal Steiner forest would not change if these edges were removed.

The following algorithm was the first constant approximation for Steiner forest and a result of Goemans and Williamson [GW95]. There are now multiple duals maintained as active – the connected components of the constructed primal-so-far each have their own variable. Otherwise, it follows the same *ball-growing process* as Kruskal's.

1. Initialize all $x_e = 0$ and all $y_S = 0$.
2. The active duals we use will be all connected components which are separating cuts (members of $\mathscr{S}$). Initially, these are the singleton separating sets.
3. Increase the active duals uniformly until a constraint becomes tight. In contrast to Kruskal's, it is possible for an edge to have 0, 1, or 2 active S in its constraint (since active S do not intersect). When we increase the active y_S at a constant rate, the constraint's gap decreases at 0, 1, or 2 times that rate respectively.



(a) No active duals. (b) 1 active dual. (c) 2 active duals.

Instead of using the minimum over $c_e - \sum_{S : e \in \delta(S)} y_S$, we can find the next tight constraint by taking:

$$\min_{(u,v) \in E} \left(\frac{c_{uv} - \sum_{S : (u,v) \in S} y_S}{|A \cap \{S_u, S_v\}|} \right)$$

4. Set the $x_e = 1$ for the edge whose constraint became tight.

5. Choose as the next active duals the separating sets among the connected components of $x_e = 1$.

This primal-dual procedure will form the first phase of the algorithm, which we call forward add. There is an additional phase after forward add called reverse delete, whose purpose and effects we will discuss later in the analysis.

Algorithm 2 (Goemans-Williamson 1995)

(FORWARD ADD)
$F \leftarrow \emptyset$
while F is not feasible **do**
  $A \leftarrow$ COMPONENTS$(F) \cap \mathcal{S}$
  $\varepsilon \leftarrow \min_{(u,v)\in E} \left(\frac{c_{uv} - \sum_{S:(u,v)\in S} y_S}{|A \cap \{S_u, S_v\}|} \right)$
  $e \leftarrow \operatorname{argmin}_{(u,v)\in E} \left(\frac{c_{uv} - \sum_{S:(u,v)\in S} y_S}{|A \cap \{S_u, S_v\}|} \right)$
  $F \leftarrow F \cup \{e\}$
  for $S \in A$ **do**
    $y_S \leftarrow y_S + \varepsilon$
  end for
end while

(REVERSE DELETE)
for $e \in F$, in reverse order of addition to F **do**
  if $F \setminus \{e\}$ is feasible **then**
    $F \leftarrow F \setminus \{e\}$
  end if
end for

4.1 Analysis

The first phase of the algorithm is nearly the same as Kruskal's. Without the second phase, however, the approximation could be poor even for 1 terminal pair.

The addition of the second phase allows us to prove a 2-approximation. Let F be the set of edges output by the algorithm as the Steiner forest. Let ΔP be the total change in primal value on F during a single iteration of forward add (similarly for the dual, ΔD). In this fixed iteration, let F_i be the subset of F selected

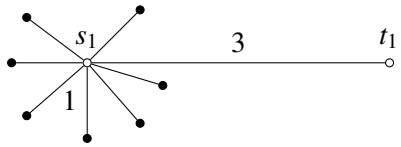


Figure 6: Since all the cost 1 edges become tight first, they are added to the solution before the length 3 edge (although they are "unnecessary" for connecting any pairs). Without reverse delete, the forest has cost $O(n)$ while the optimal has cost 3.

so far, and A be the set of active components.

$$\begin{aligned}\Delta P &= \varepsilon \cdot \sum_{S \in A} |\delta(S) \cap F| \\ \Delta D &= \varepsilon \cdot |A|\end{aligned}$$

Here, $\delta(S)$ is the set of edges in the cut $(S, \overline{S})$.

Lemma 3. *Within a single iteration of forward add,*

$$\sum_{S \in A} |\delta(S) \cap F| \leq 2|A|$$

Proof. $G \cap F_i$ is the subgraph induced by the edges selected so far. Let G' be the graph where each connected component of $G \cap F_i$ is contracted in $G \cap F$. Each vertex $v_S \in G'$ corresponds to a connected component $S \subseteq V$. First, notice that the left quantity is precisely:

$$\sum_{S \in A} |\delta(S) \cap F| = \sum_{S \in A} \deg_{G'}(v_S)$$

In other words, "the sum of degree of active v_S in G'". We will categorize the vertices of $v_S \in G'$ this way:

1. The set of v_S where S is an active component (these are the components A).
2. The set of v_S where S is an inactive component. (we will call these components I).

The total number of connected components is $|A| + |I|$.

F is treelike and spans the component, therefore there are $|A| + |I| - 1$ edges in G'. Summing over the degree of all vertices in G' gives:

$$\sum_{S \in A} \deg_{G'}(v_S) + \sum_{S \in I} \deg_{G'}(v_S) \leq 2(|A| + |I| - 1)$$

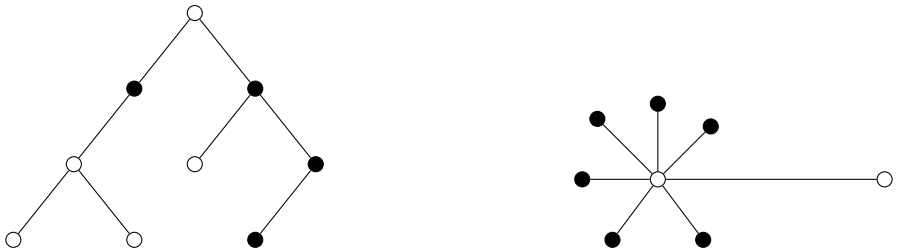
Rearranging:

$$\begin{aligned}\sum_{S \in A} \deg_{G'}(v_S) &\leq 2(|A| + |I| - 1) - \sum_{S \in I} \deg_{G'}(v_S) \\ &= 2(|A| - 1) + 2|I| - \sum_{S \in I} \deg_{G'}(v_S) \\ &\leq 2|A| + \left(2|I| - \sum_{S \in I} \deg_{G'}(v_S)\right)\end{aligned}$$

If the rightmost term is ≤ 0, the statement of the lemma is proved. In other words, we would like to show that:

$$2|I| \leq \sum_{S \in I} \deg_{G'}(v_S)$$

Claim 4. *For an inactive component $S \in I$, $\deg_{G'}(v_S) \geq 2$*



(a) (b)

Figure 7: (a) G' is the graph formed by contracting (in F) the connected components of the partial solution F_i. The set of active components A is given by the hollow nodes. Some components, although connected in F_i, are no longer active (solid nodes). G' edges to inactive leaf nodes do not contribute to terminal connectivity, and would have been removed by reverse delete. In this figure, the bottom-right two edges would have been removed by reverse delete. (b) Therefore, reverse delete fixes the problem posed by the "unnecessary" short edges in the earlier example.

In other words, inactive components are never "leaves" of F. This follows from the effects of reverse delete. If an inactive component S is a leaf of F, removing the leaf edge from F does not destroy feasibility of F (since it disconnects S and S was not active). This edge would have been removed reverse delete when it was examineed.

Plugging in the claim, the lemma is proved.

$$\begin{aligned}\sum_{S\in A} |\delta(S)\cap F| &= \sum_{S\in A} \deg_{G'}(v_S) \\ &\leq 2|A| + \left(2|I| - \sum_{S\in I} \deg_{G'}(v_S)\right) \\ &\leq 2|A|\end{aligned}$$

□

This immediately gives us the per-iteration primal-dual ratio:

$$\Delta P \leq 2\Delta D$$

Summing over iterations (and since the dual solution is kept feasible):

$$\text{ALGO} = P \leq 2D \leq 2D^* = 2P^* \leq 2\text{OPT}$$

Thus the algorithm is 2-approximate.

5 Summary

In this lecture, we introduced the primal-dual method and analyzed a ball-growing process behind both Kruskal's MST algorithm and the 2-approximation for Steiner forest by Goemans and Williamson. Additionally, the names "forward add" and "reverse delete" are references to two variations of Kruskal's MST algorithm:

1. Beginning with an empty graph and adding minimum cost edges..
2. Beginning with the entire graph and removing edges which don't help connectivity.

References

[GW95] Michel X Goemans and David P Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.