Primal Dual Algorithms (Lecture 14)

Last lecture we saw how to use both primal & dual LPs to round a primal solution (after having solved the primal/dual LP pair).

Today: create solutions to the P/D pair by hand, such that

1. Primal and dual are feasible
2. Primal is integer solution
3. cost of primal ≤ p. cost of dual

\[ \text{for minimization (primal) problem} \]

⇒ primal is \( p \)-approximation.

\[ \text{why? cost(primal) ≤ p. cost(dual) ≤ p. cost(optimal primal) ≤ p. cost(optimal integer)} \]

\[ \text{weak duality} \]

Let's warm up with a simpler problem.

Vertex cover. Pick set of vertices that hit every edge. \( G = (V,E) \).

HW1 gave greedy algo.

Pick an edge uncovered until now. Pick both endpoints, repeat.

But what about the case when vertices are weighted? Want to pick least-weight solution that hits all edges.

Let's give a primal-dual solution for this.

If \( 2 \)-approx: opt value ≥ \( \alpha \) matching in graph and we give matching \( \epsilon \)

\[ \text{2ln(2)cost ≤ 2 matching} \]

(\( \alpha \) \)}
\[ \begin{align*}
\text{LP:} & \quad \min \sum_{e} c_e x_e \\
& \quad s.t. \quad \sum_{e \in E} x_e t_e = 1, \forall e \in E \\
& \quad x_e \geq 0.
\end{align*} \]

dual \[ \begin{align*}
\text{max} \sum_{e} y_e \\
& \quad s.t. \quad \sum_{e \in E} y_e = c_e \forall e \in E \\
& \quad y_e \geq 0.
\end{align*} \]

Note: if \( c_e = 1 \) then any matching \( M \) in a feasible solution to dual (\( \sum y_e = 1 \Rightarrow c_e \geq 1 \)) and we have a solution (both endpoints of this maximal matching) of cost \( \leq 2M \).

So we are also beig primal-dual.

More generally, for costs: start with \( x = 0, y = 0 \).

- Pick an edge \( e \), raise \( y_e \) until some dual constraint becomes tight
- Pick all vertices whose dual constraint is tight
- Set \( x_e = 1 \)

Until solution is feasible

\[ \begin{align*}
\text{Primal} = \sum_{e \in E} c_e x_e &= \sum_{e \in E} \sum_{v \in G} y_e \\
&= \sum_{v \in G} y_e \sum_{e \in E} 1 \\
&\leq 2 \sum_{e \in E} y_e = 2 \cdot \text{dual}.
\end{align*} \]

\[ \Rightarrow 2 \text{approx.} \]

What's happening?

1. hand crafting dual solution, raising "money", and trying to maintain some kind of complementary slackness (if primal var > 0 \( \Rightarrow \) dual constraint is tight).

Here we could raise single dual var at a time.

For other problems, my help to raise many vars. simultaneously.
(2) each edge offers money to nodes. "Tight" vertices can open.

But each edge offers money to both its endpoints, so double dipping
that's the factor of 2.

Often the normal dual edges will have this double-dipping flavour, which is
where the approx. arises.

**How to facility location**

\[
\begin{align*}
\min & \quad \sum_i f_i y_i + \sum_{ij} d_{ij} x_{ij} \\
\text{s.t.} & \quad x_{ij} \leq y_i + v_{ij} \\
& \quad \sum_j x_{ij} \geq 1 \quad \forall i, c \\
& \quad x, y \geq 0.
\end{align*}
\]

**Interpretation:** \(d_{ij} = \text{money that client } i \text{ is offering to be connected.}\)

out of that some fraction \((d_{ij})\) is to be connected to facility @j

and \(v_{ij}\) is to open the facility.

The total offering \(\sum_j v_{ij}\) should never be more than the cost of the location \(i\)

else we can open it.

What is the next we can raise this way? That's the dual.

So now we want to

(a) make this money, and \(f\) simultaneously!
(b) build a solution

Here's how we will do it...
Each client grows a \( \eta \) (simultaneously)

\[
\text{contributing: } P_i j = (\eta - \eta_{ij})^+ = \max(\eta - \eta_{ij}, 0).
\]

- When \( \sum_j P_i j = \eta_i \), "tentatively open" facility at \( i \).
- If \( j \)oucher tentatively open facility (i.e. \( \eta_j > \eta_{ij} \) for some such \( i \)), stop growing \( \eta \).

Stop when all \( \eta \) stop growing.

Immediate problem:

- Many facilities may be tentatively open. 😐
  - E.g. single client, but facilities at dist 1 F cost 1.
  - All will tentatively open at same time. ("double dipping" \( \Rightarrow \) "many times dipping")
  - 🎁 Use the metric properly to close many of these.

**Clean-up Phase**

Consider tentatively opened facilities (call it \( F' \)).

- Add edge between \( i \) and \( i' \) \( \in F' \) if \( j \) client \( j \) that has \( P_{ij} > 0 \) \( P_{ij'} > 0 \)
  - Non-zero contribution to both \( i' \).

Pick a maximal independent set of \( F' \), call it \( F \).

Great! Removed double-dipping by construction, but what about cost of solution? Not too bad, we claim...
What's the cost?

Let's assign clients to open facilities in \( F \).

(a) if \( P_{ij} > 0 \) then \( j \) "contributes" to facility \( i \) if \( F \Rightarrow \) assign \( j \) to \( N(i) \).

Note: by construction of \( F \) from \( F' \), each client contributes to only one \( i \in F \).

(b) if \( j \) not assigned above, and if \( \delta_{ij} = \delta_{ij} \) for some \( i \in F \)

then again assign \( j \) to \( N(i) \).

For remaining clients \( j \), neither facility \( i \) st. \( j \) contributed to them

nor \( i \) st. \( \delta_{ij} = \delta_{ij} \)

are open.

All these lonely clients.

Lemma (3-hop): For a lonely client \( j \), \( i \) open facility at distance at most \( 3 \delta_{ij} \) in \( F \). (proof soon)

Given this lemma, let's prove 3-approx:

- **Cost of non-lonely clients + open facilities**

\[
\sum_{i \in F} \left( f_i + \sum_{j \in N(i)} \delta_{ij} \right) = \sum_{i \in F} \left( \sum_{j \in N(i)} \delta_{ij} + \sum_{j \in N(i)} \delta_{ij} \right) = \sum_{i \in F} \sum_{j \in N(i)} \delta_{ij} = \sum_{j \text{ non-lonely}} \delta_{ij}
\]

- **Cost of lonely clients**

\[
\sum_{j \text{ non-lonely}} \delta_{ij}, F) \leq 3 \sum_{j \text{ lonely}} \delta_{ij} \Rightarrow \text{total cost} \leq 3 \sum_{j \text{ lonely}} \delta_{ij} \leq 3 \cdot \text{OPT}
\]
Proof of 3-hop Lemma

Suppose facility i is isolated. Consider the dual increase procedure.

We stopped raising \( d_j \) because of some facility \( i' \) that was tentatively opened (in \( F' \)).

Now \( i \) must be closed in \( F \) (because \( i \) is isolated).

So \( i \) closed because \( i' \) contributed to \( i \) and \( i' \) both and \( i' \) open in \( F \).

\[
d_{ij} \leq d_j
\]

\[
d(j,F) = d(j,i) + d(i,j') + d(j',i') \leq d_j + d_{ij'} + d_{ij'}. \]

Now we claim that \( d_{ij'} \leq d_j \)

Since we raise all duals simultaneously, and \( d_j \) stopped raising \( b/c \) of \( i', d_j \) would stop \( b/c \) of \( i \) if it had not stopped earlier (had tentatively opened at time \( d_j \) if not before)

\[
\Rightarrow d_{ij'} \leq d_j
\]

This completes the proof of 3-hop Lemma
Finishes for location proof.

\[ \text{primal} \]

\[ \text{cost of solution } \leq 3 \sum_i d_{ij} = 3 \text{ dual solution} \leq 3 \cdot \text{OPT} \]

\[ \text{weak duality} \]

Aside: can use Slater's slightly stronger guarantee —

\[ \sum_i f_i + \sum_j d_{ij} \leq 3 \sum_j d_{ij} \]

In other words

\[ \sum_j d_{ij} \leq 3 \left( \sum_j d_{ij} - \sum_i f_i \right) \]

Why is this interesting?

- Can give an \( O(1) \)-apx for \( k \)-median this way [Jain, Vazirani 99]

- Method of Lagrangian-Multiplier preserving algo.

  Some other time, or see the CWS IOJ book for details.

- Allows relatively constrained problems to Lagrangian version

\[
\begin{align*}
\text{min } & \sum_i d_{ij} x_{ij} \\
\text{subject to } & \sum_i x_{ij} = 1, \ x_{ij} \leq y_i \\
\text{and } & \sum_i y_i \leq K
\end{align*}
\]

\[ \text{\textbf{facility location!}} \]
Recap:

Primal-dual technique:
- Solve primal and dual in coordinated ways, basically construct primal solution, and
  - raise dual "proof" that optimal solution must cost that much
  - or just view dual as money that pays for primal.

. Can also view as approximate complementary slackness.

Several problems:
- Steiner tree/forest
- Facility location and k-median
- Network design problems
all have primal-dual algorithms.
(Actually can also get P-D algo for problems in P.)

Do not use LP solver
  => can be faster than LP rounding methods.
  X

Next week, some hardness,
then we'll be back the week after with some more algo.