Linear Programming and Primal-Dual Schema

Chihao Zhang

BASICS, Shanghai Jiao Tong University

Oct.09, 2012
Outline

1. **Linear Programming**
   - Formulate Set Cover Problem
   - Solving Linear Programs

2. Rounding of LP
   - Set Cover

3. Primal-Dual Schema
   - Set Cover
   - Feedback Vertex Set
Example: Set Cover

**Input:** A Universe $E = \{e_1, \ldots, e_n\}$; a family of subsets $S_1, \ldots, S_m$ where each $S_j \subseteq E$; a nonnegative weight $w_j \geq 0$ for each $S_j$.

**Problem:** Find a minimum-weight collection of subsets that covers all of $E$.
Integer Program

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=1}^{m} w_j x_j \\
\text{subject to} & \quad \sum_{j : e_i \in S_j} x_j \geq 1, \quad i = 1, \ldots, n, \\
x_j & \in \{0, 1\}, \quad j = 1, \ldots, m.
\end{align*}
\]

\(x_j \in \{0, 1\}\) : indicate whether \(S_j\) is in the solution.
Linear Program Relaxation

minimize \( \sum_{j=1}^{m} w_j x_j \)

subject to \( \sum_{j: e_i \in S_j} x_j \geq 1, \quad i = 1, \ldots, n, \)

\( x_j \geq 0, \quad j = 1, \ldots, m. \)
Canonical Form

maximize $c^T x$

subject to $A x \leq b$

$x \geq 0$

$A = (a_{i,j})$ : An $m \times n$ matrix

$b = (b_1, \ldots, b_m)$ : A vector of $m$ entries

$c = (c_1, \ldots, c_n)$ : A vector of $n$ entries
Canonical Form

\[
\begin{align*}
\text{maximize} & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
\quad x & \geq 0
\end{align*}
\]

\[A = (a_{i,j}) : \text{An } m \times n \text{ matrix}\]
\[b = (b_1, \ldots, b_m) : \text{A vector of } m \text{ entries}\]
\[c = (c_1, \ldots, c_n) : \text{A vector of } n \text{ entries}\]

Every LP can be transformed to canonical form efficiently.
Algorithms to Solve LP

- Simplex Algorithm
- Ellipsoid Method
Dual of Linear Program

Consider the following linear program:

\[
\text{maximize} \quad x_1 + 6x_2 \\
\text{s.t.} \quad x_1 \leq 200 \quad (1) \\
\text{s.t.} \quad x_2 \leq 300 \quad (2) \\
\text{s.t.} \quad x_1 + x_2 \leq 400 \quad (3) \\
\text{s.t.} \quad x_1, x_2 \geq 0
\]
Dual of Linear Program

Consider the following linear program:

\[
\begin{align*}
\text{maximize} & \quad x_1 + 6x_2 \\
\text{s.t.} & \quad x_1 \leq 200 \quad (1) \\
& \quad x_2 \leq 300 \quad (2) \\
& \quad x_1 + x_2 \leq 400 \quad (3) \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]

The optimal solution is at \((x_1, x_2) = (100, 300)\), with objective value 1900.
Dual of Linear Program (cont’d)

\[(1) + 6 \times (2) : \quad x_1 + 6x_2 \leq 2000.\]
Dual of Linear Program (cont’d)

(1) + 6 \times (2) : \quad x_1 + 6x_2 \leq 2000.

0 \times (1) + 5 \times (2) + 1 \times (3) : \quad x_1 + 6x_2 \leq 1900
Dual of Linear Program (cont’d)

(1) + 6 × (2) : \( x_1 + 6x_2 \leq 2000. \)
0 × (1) + 5 × (2) + 1 × (3) : \( x_1 + 6x_2 \leq 1900 \)

<table>
<thead>
<tr>
<th>Multiplier</th>
<th>Inequality</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y_1 )</td>
<td>( x_1 \leq 200 )</td>
</tr>
<tr>
<td>( y_2 )</td>
<td>( x_2 \leq 300 )</td>
</tr>
<tr>
<td>( y_3 )</td>
<td>( x_1 + x_2 \leq 400 )</td>
</tr>
</tbody>
</table>
minimize \[ 200y_1 + 300y_2 + 400y_3 \]
\[ y_1 + y_3 \geq 1 \]
\[ y_2 + y_3 \geq 6 \]
\[ y_1, y_2, y_3 \geq 0 \]
Dual of Linear Program (cont’d)

Primal LP:

\[
\begin{align*}
\text{maximize} & \quad c^T x \\
Ax & \leq b \\
x & \geq 0
\end{align*}
\]

Dual LP:

\[
\begin{align*}
\text{minimize} & \quad y^T b \\
y^T A & \geq c^T \\
y & \geq 0
\end{align*}
\]
Dual of Linear Program (cont’d)

Primal LP:

\[
\begin{align*}
\text{maximize} & \quad c^T x \\
Ax & \leq b \\
x & \geq 0
\end{align*}
\]

Dual LP:

\[
\begin{align*}
\text{minimize} & \quad y^T b \\
y^T A & \geq c^T \\
y & \geq 0
\end{align*}
\]

weak duality property:

\[c^T x \leq y^T b\]
Dual of Linear Program (cont’d)

Primal LP:

\[
\text{maximize } \mathbf{c}^T \mathbf{x} \\
A \mathbf{x} \leq \mathbf{b} \\
\mathbf{x} \geq 0
\]

Dual LP:

\[
\text{minimize } \mathbf{y}^T \mathbf{b} \\
\mathbf{y}^T A \geq \mathbf{c}^T \\
\mathbf{y} \geq 0
\]

strong duality property:

\[
\mathbf{c}^T \mathbf{x}^* = (\mathbf{y}^*)^T \mathbf{b}
\]
Outline

1. Linear Programming
   - Formulate Set Cover Problem
   - Solving Linear Programs

2. Rounding of LP
   - Set Cover

3. Primal-Dual Schema
   - Set Cover
   - Feedback Vertex Set
A Simple Rounding Algorithm for Set Cover

Recall the linear programming relaxation for set cover:

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=1}^{m} w_j x_j \\
\text{subject to} & \quad \sum_{j : e_i \in S_j} x_j \geq 1, \quad i = 1, \ldots, n, \\
& \quad x_j \geq 0, \quad j = 1, \ldots, m.
\end{align*}
\]
A Simple Rounding Algorithm for Set Cover

Recall the linear programming relaxation for set cover:

\[
\begin{align*}
\text{minimize} & \quad \sum_{j=1}^{m} w_j x_j \\
\text{subject to} & \quad \sum_{j : e_i \in S_j} x_j \geq 1, \quad i = 1, \ldots, n, \\
& \quad x_j \geq 0, \quad j = 1, \ldots, m.
\end{align*}
\]

Consider the optimal solution of the LP. Intuitively, the set with larger \(x_j\) is more likely to be in a good solution of set cover (or the IP).
1. Let $x^*$ be the solution of LP.
2. Let $I = \{j \mid x_j^* \geq 1/f\}$, where $f = \max_{i=1,...,n} |\{j \mid e_i \in S_j\}|$.
3. Output $I$. 

Lemma $S$ is a set cover.
A Simple Rounding Algorithm for Set Cover (cont’d)

1. Let $x^*$ be the solution of LP.
2. Let $I = \{j \mid x_j^* \geq 1/f\}$, where $f = \max_{i=1,...,n} \{|\{j \mid e_i \in S_j\}|\}$.
3. Output $I$.

Lemma

$S$ is a set cover.
Analysis

Lemma

The rounding algorithm is an $f$-approximation algorithm for the set cover problem.
Analysis

Lemma

The rounding algorithm is an $f$-approximation algorithm for the set cover problem.

$$\sum_{j \in I} w_j \leq \sum_{j=1}^{m} w_j \cdot (f \cdot x^*_j)$$

$$= f \sum_{j=1}^{m} w_j x^*_j$$

$$= f \cdot \text{OPT}$$
Outline

1. Linear Programming
   - Formulate Set Cover Problem
   - Solving Linear Programs

2. Rounding of LP
   - Set Cover

3. Primal-Dual Schema
   - Set Cover
   - Feedback Vertex Set
A Primal-Dual Algorithm for Set Cover

Recall the LP relaxation for Set Cover:

\begin{align*}
\text{minimize} \quad & \sum_{j=1}^{m} w_j x_j \\
\text{subject to} \quad & \sum_{j:e_i \in S_j} x_j \geq 1, \quad i = 1, \ldots, n, \\
& x_j \geq 0, \quad j = 1, \ldots, m.
\end{align*}

and its dual:

\begin{align*}
\text{maximize} \quad & \sum_{i=1}^{n} y_i \\
\text{subject to} \quad & \sum_{i:e_i \in S_j} y_i \leq w_j, \quad j = 1, \ldots, m, \\
& y_i \geq 0, \quad i = 1, \ldots, n.
\end{align*}
Vertex Cover

- **Vertex cover problem** is the special case of set cover problem when $f = 2$. 
Vertex Cover

- **Vertex cover problem** is the special case of set cover problem when $f = 2$.
- The dual of vertex cover problem is **maximum matching problem**.
Vertex Cover

- **Vertex cover problem** is the special case of set cover problem when $f = 2$.
- The dual of vertex cover problem is **maximum matching problem**.
- The duality theorem implies

\[
\text{maximum matching} \leq \text{minimum vertex cover}
\]
Vertex Cover (cont’d)

Consider the following algorithm:

1. $M \leftarrow \emptyset$
2. $S \leftarrow \emptyset$
3. while $G$ is not empty do
   3.1 Choose an edge $e = \{u, v\} \in E(G)$ and let $M \leftarrow M \cup \{e\}$
   3.2 $S \leftarrow S \cup \{u, v\}$
   3.3 $G \leftarrow G[V \setminus \{u, v\}]$ (Remove isolated nodes)
4. return $S$
Vertex Cover (cont’d)

Consider the following algorithm:

1. \( M \leftarrow \emptyset \)
2. \( S \leftarrow \emptyset \)
3. while \( G \) is not empty do
   3.1 Choose an edge \( e = \{u, v\} \in E(G) \) and let \( M \leftarrow M \cup \{e\} \)
   3.2 \( S \leftarrow S \cup \{u, v\} \)
   3.3 \( G \leftarrow G[V \setminus \{u, v\}] \) (Remove isolated nodes)
4. return \( S \)

This is the combinatorial interpretation of a primal-dual algorithm.
The Algorithm

1. \( y \leftarrow 0 \)
2. \( I \leftarrow \emptyset \)
3. while there exists \( e_i \notin \bigcup_{j \in I} S_j \) do
   3.1 Increase the dual variable \( y_i \) until there is some \( \ell \) such that
      \[ \sum_{j : e_j \in S_\ell} y_j = w_\ell \]
   3.2 \( I \leftarrow I \cup \{\ell\} \)
4. return \( I \).
Primal-Dual Schema  Set Cover

Analysis

The primal-dual algorithm is an $f$-approximation algorithm for the set cover problem.
The primal-dual algorithm is an $f$-approximation algorithm for the set cover problem.

\[
\sum_{j \in I} w_j = \sum_{j \in I} \sum_{i : e_i \in S_j} y_i \\
= \sum_{i=1}^{n} y_i \cdot |\{j \in I \mid e_i \in S_j\}| \\
\leq f \cdot \text{OPT}
\]
Complementary Slackness

The following property is called complementary slackness.

\[ \sum_{i=1}^{n} y_i \leq \sum_{i=1}^{n} y_i \sum_{j:e_i \in S_j} x_j = \sum_{j=1}^{m} x_j \sum_{i:e_i \in S_j} y_i \leq \sum_{j=1}^{m} x_j w_j. \]
Complementary Slackness

The following property is called **complementary slackness**.

\[
\sum_{i=1}^{n} y_i \leq \sum_{i=1}^{n} y_i \sum_{j: e_i \in S_j} x_j = \sum_{j=1}^{m} x_j \sum_{i: e_i \in S_j} y_i \leq \sum_{j=1}^{m} x_j w_j.
\]

Let \( x^* \) and \( y^* \) be the optimal solution of primal and dual LP respectively, then

- \( y_i^* > 0 \implies \sum_{j: e_i \in S_j} x_j^* = 1 \)
- \( x_j^* > 0 \implies \sum_{i: e_i \in S_j} y_i^* = w_j \)
Analysis (revisited)

\[
\sum_{j \in I} w_j = \sum_{j \in I} \sum_{i : e_i \in S_j} y_i \\
= \sum_{i=1}^{n} y_i \cdot |\{j \in I \mid e_i \in S_j\}| \\
\leq f \cdot \text{OPT}
\]
Analysis (revisited)

\[
\sum_{j \in I} w_j = \sum_{j \in I} \sum_{i : e_i \in S_j} y_i \\
= \sum_{i=1}^{n} y_i \cdot \left| \{ j \in I \mid e_i \in S_j \} \right| \\
= \sum_{i=1}^{n} y_i \cdot \sum_{j : e_i \in S_j} x_j \\
\leq f \cdot \text{OPT}
\]

where \( x_j \in \{0, 1\} \) and \( x_j = 1 \) if and only if \( j \in I \).
Discussion

The primal-dual algorithm ensures

\[ x_j > 0 \implies \sum_{i : e_i \in S_j} y_i = w_j \]
Discussion

The primal-dual algorithm ensures

\[ x_j > 0 \implies \sum_{i : e_i \in S_j} y_i = w_j \]

In general, we cannot hope

\[ y_i > 0 \implies \sum_{j : e_i \in S_j} x_j = 1 \]
The primal-dual algorithm ensures

\[ x_j > 0 \implies \sum_{i:e_i \in S_j} y_i = w_j \]

In general, we cannot hope

\[ y_i > 0 \implies \sum_{j : e_i \in S_j} x_j = 1 \]

We want to show it is not too slack, i.e.

\[ y_i > 0 \implies \sum_{j : e_i \in S_j} x_j \leq \alpha \]
Feedback Vertex Set Problem

**Input:** A undirected graph $G = (V, E)$ and nonnegative weights $w_i > 0$ for $i \in V$.

**Problem:** Find a set $S \subseteq V$ of minimum weight such that $G[V \setminus S]$ is a forest.
LP formulation

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in V} w_i x_i \\
\text{subject to} & \quad \sum_{i \in C} x_i \geq 1, \quad \forall C \in \mathcal{C} \\
& \quad x_i \in \{0, 1\}, \quad \forall i \in V
\end{align*}
\]

\[x_i \in \{0, 1\} : \text{ indicate whether } v_i \text{ is in the solution.}\]
LP formulation

\[
\begin{aligned}
\text{minimize} & \quad \sum_{i \in V} w_i x_i \\
\text{subject to} & \quad \sum_{i \in C} x_i \geq 1, \quad \forall C \in C \\
& \quad x_i \geq 0, \quad \forall i \in V
\end{aligned}
\]

\(x_i \in \{0, 1\}\) : indicate whether \(v_i\) is in the solution.
Dual LP

maximize \( \sum_{C \in \mathcal{C}} y_C \)

subject to \( \sum_{C \in \mathcal{C} : i \in C} y_C \leq w_i, \quad \forall i \in V, \)

\( y_C \geq 0, \quad \forall C \in \mathcal{C} \)
The Algorithm

1. $y \leftarrow 0$
2. $S \leftarrow \emptyset$
3. while there exists a cycle $C$ in $G$ do
   3.1 Increase $y_C$ until there is some $\ell \in V$ such that
       $$\sum_{C': C' \in C: \ell \in C'} y_{C'} = w_\ell$$
   3.2 $S \leftarrow S \cup \{\ell\}$
   3.3 Remove $\ell$ from $G$
   3.4 Repeatedly remove vertices of degree one from $G$
4. return $S$. 
Analysis

\[ \sum_{i \in S} w_i = \sum_{i \in S} \sum_{C : i \in C} y_C = \sum_{C \in C} |S \cap C| y_C. \]
Analysis

\[ \sum_{i \in S} w_i = \sum_{i \in S} \sum_{C: i \in C} y_C = \sum_{C \in \mathcal{C}} |S \cap C| y_C. \]

\[ |S \cap C| \text{ may be as large as } |V|! \]
Analysis (cont’d)

Observation

For any path $P$ of vertices of degree two in graph $G$, our algorithm will choose at most one vertex from $P$. 
Analysis (cont’d)

**Observation**

*For any path P of vertices of degree two in graph G, our algorithm will choose at most one vertex from P.*

**Theorem**

*In any graph G that has no vertices of degree one, there is a cycle with at most \(2\lceil \log_2 n \rceil\) vertices of degree three or more, and it can be found in linear time.*
Algorithm (revised)

1. \( y \leftarrow 0 \)
2. \( S \leftarrow \emptyset \)
3. Repeatedly remove vertices of degree one from \( G \)
4. \textbf{while} there exists a cycle \( C \) in \( G \) \textbf{do}
   4.1 Find cycle \( C \) with at most \( 2 \lfloor \log_2 n \rfloor \) vertices of degree three or more
   4.2 Increase \( y_C \) until there is some \( \ell \in V \) such that
      \( \sum_{C' \in C : \ell \in C'} y_{C'} = w_\ell \)
   4.3 \( S \leftarrow S \cup \{ \ell \} \)
   4.4 Remove \( \ell \) from \( G \)
   4.5 Repeatedly remove vertices of degree one from \( G \)
5. \textbf{return} \( S \).
Analysis

\[ \sum_{i \in S} w_i = \sum_{C \in C} |S \cap C| y_C \leq (4 \lfloor \log_2 n \rfloor) \sum_{C \in C} y_C \leq (4 \lfloor \log_2 n \rfloor) \text{OPT}. \]