CPS 223

Linear Programming Duality, Reductions, and Bipartite Matching

Yu Cheng
Linear Programming Duality
Example linear program

- We make reproductions of two paintings

\[
\begin{align*}
\text{maximize} & \quad 3x + 2y \\
\text{subject to} & \quad 4x + 2y \leq 16 \\
& \quad x + 2y \leq 8 \\
& \quad x + y \leq 5 \\
& \quad x \geq 0 \\
& \quad y \geq 0
\end{align*}
\]

- Painting 1 sells for $30, painting 2 sells for $20
- Painting 1 requires 4 units of blue, 1 green, 1 red
- Painting 2 requires 2 blue, 2 green, 1 red
- We have 16 units blue, 8 green, 5 red
Solving the linear program graphically

maximize \(3x + 2y\)

subject to

\(4x + 2y \leq 16\)

\(x + 2y \leq 8\)

\(x + y \leq 5\)

\(x \geq 0\)

\(y \geq 0\)

optimal solution: \(x=3, y=2\)
Proving optimality

\[
\begin{align*}
\text{maximize} & \quad 3x + 2y \\
\text{subject to} & \quad 4x + 2y \leq 16 \\
& \quad x + 2y \leq 8 \\
& \quad x + y \leq 5 \\
& \quad x \geq 0 \\
& \quad y \geq 0
\end{align*}
\]

Recall: optimal solution: \( x=3, y=2 \)

Solution value = 9+4 = 13

How do we prove this is optimal (without the picture)?
Proving optimality...

**maximize** $3x + 2y$

**subject to**

$4x + 2y \leq 16$

$x + 2y \leq 8$

$x + y \leq 5$

$x \geq 0$

$y \geq 0$

We can rewrite the blue constraint as

$2x + y \leq 8$

If we add the red constraint $x + y \leq 5$ we get

$3x + 2y \leq 13$

Matching upper bound!

(Really, we added .5 times the blue constraint to 1 times the red constraint)
Linear combinations of constraints

**maximize** 3x + 2y  
**subject to**  
4x + 2y ≤ 16  
x + 2y ≤ 8  
x + y ≤ 5  
x ≥ 0  
y ≥ 0  

\[ b(4x + 2y \leq 16) + g(x + 2y \leq 8) + r(x + y \leq 5) \]
\[ = \]
\[ (4b + g + r)x + (2b + 2g + r)y \leq 16b + 8g + 5r \]

4b + g + r must be at least 3  
2b + 2g + r must be at least 2  
Given this, minimize 16b + 8g + 5r
Using LP for getting the best upper bound on an LP

**maximize** $3x + 2y$

**subject to**

$4x + 2y \leq 16$

$x + 2y \leq 8$

$x + y \leq 5$

$x \geq 0$

$y \geq 0$

**minimize** $16b + 8g + 5r$

**subject to**

$4b + g + r \geq 3$

$2b + 2g + r \geq 2$

$b \geq 0$

$g \geq 0$

$r \geq 0$

the dual of the original program

- **Duality theorem:** any linear program has the same optimal value as its dual!
Another View

• Painting 1: 4 blue, 1 green, 1 red, sells for $30
• Painting 2: 2 blue, 2 green, 1 red, sells for $20
• We have 16 units blue, 8 green, 5 red

• Suppose Vince wants to buy paints from us.
• Pay $b$ for a unit of blue, $g$ for green, $r$ for red.
• We can choose to sell the paints, or produce paintings and sell the paintings, or both.

\[
\begin{align*}
b & \geq 0 \\
g & \geq 0 \\
r & \geq 0 \\
4b + g + r & \geq 3 \\
2b + 2g + r & \geq 2
\end{align*}
\]
Another View

• Vince pays $(16b + 8g + 5r)$ in total.

• We have 16 units blue, 8 green, 5 red

  • Suppose Vince wants to buy paints from us.
  • Pay $b$ for a unit of blue, $g$ for green, $r$ for red.
  • We can choose to sell the paints, or produce paintings and sell the paintings, or both.

\[
\begin{align*}
  b &\geq 0 \\
  g &\geq 0 \\
  r &\geq 0 \\
  4b + g + r &\geq 3 \\
  2b + 2g + r &\geq 2
\end{align*}
\]
Using LP for getting the best upper bound on an LP

**maximize** $3x + 2y$

subject to

$4x + 2y \leq 16$

$x + 2y \leq 8$

$x + y \leq 5$

$x \geq 0$

$y \geq 0$

**minimize** $16b + 8g + 5r$

subject to

$4b + g + r \geq 3$

$2b + 2g + r \geq 2$

$b \geq 0$

$g \geq 0$

$r \geq 0$

primal
dual
Duality

• Weak duality:
  Optimal value of primal ≥ Optimal value of dual
  (when primal LP is max and dual LP is min)

• We can make $13 if we produce paintings
  Vince should pay at least as much

• Strong Duality
  Optimal value of primal = Optimal value of dual
  Vince is a good negotiator
Using LP for getting the best upper bound on an LP

\[
\begin{align*}
& \text{maximize } 3x + 2y \\
& \text{subject to } \\
& 4x + 2y \leq 16 \\
& x + 2y \leq 8 \\
& x + y \leq 5 \\
& x \geq 0 \\
& y \geq 0 \\
& \text{primal}
\end{align*}
\]

\[
\begin{align*}
& \text{minimize } 16b + 8g + 5r \\
& \text{subject to } \\
& 4b + g + r \geq 3 \\
& 2b + 2g + r \geq 2 \\
& b \geq 0 \\
& g \geq 0 \\
& r \geq 0 \\
& \text{dual}
\end{align*}
\]
Reductions
NP ("nondeterministic polynomial time")

• Recall: decision problems require a yes or no answer

• NP: the class of all decision problems such that if the answer is yes, there is a simple proof of that

• E.g., “does there exist a set cover of size k?”

• If yes, then just show which subsets to choose!

• Technically:
  – The proof must have polynomial length
  – The correctness of the proof must be verifiable in polynomial time
“Easy to verify” problems: NP

• All decision problems such that we can verify the correctness of a solution in polynomial time.
NP-hardness

- A problem is **NP-hard** if it is at least as hard as all problems in NP
- So, trying to find a polynomial-time algorithm for it is like trying to prove P=NP
- Set cover is NP-hard

Typical way to prove problem Q is NP-hard:
- Take a known NP-hard problem Q’
- Reduce it to your problem Q
  - (in polynomial time)

E.g., (M)IP is NP-hard, because we have already reduced set cover to it
  - (M)IP is more general than set cover, so it can’t be easier
Reductions

- Sometimes you can reformulate problem A in terms of problem B (i.e., reduce A to B)
  - E.g., we have seen how to formulate several problems as linear programs or integer programs

- In this case problem A is at most as hard as problem B
  - Since LP is in P, all problems that we can formulate using LP are in P
    - Caveat: only true if the linear program itself can be created in polynomial time!
Independent Set

• In the below graph, does there exist a subset of vertices, of size 4, such that there is no edge between members of the subset?
Independent Set

• In the below graph, does there exist a subset of vertices, of size 4, such that there is no edge between members of the subset?

• General problem (decision variant): given a graph and a number k, are there k vertices with no edges between them?

• NP-complete
Set Cover (a computational problem)

• We are given:
  – A finite set \( S = \{1, \ldots, n\} \)
  – A collection of subsets of \( S \): \( S_1, S_2, \ldots, S_m \)

• We are asked:
  – Find a subset \( T \) of \( \{1, \ldots, m\} \) such that \( \bigcup_{j \in T} S_j = S \)
  – Minimize \( |T| \)

• Decision variant of the problem:
  – we are additionally given a target size \( k \), and
  – asked whether a \( T \) of size at most \( k \) will suffice

• One instance of the set cover problem:
\( S = \{1, \ldots, 6\} \), \( S_1 = \{1,2,4\} \), \( S_2 = \{3,4,5\} \), \( S_3 = \{1,3,6\} \), \( S_4 = \{2,3,5\} \), \( S_5 = \{4,5,6\} \), \( S_6 = \{1,3\} \)
Visualizing Set Cover

- \( S = \{1, \ldots, 6\} \), \( S_1 = \{1,2,4\} \), \( S_2 = \{3,4,5\} \), \( S_3 = \{1,3,6\} \), \( S_4 = \{2,3,5\} \), \( S_5 = \{4,5,6\} \), \( S_6 = \{1,3\} \)
Reducing independent set to set cover

- In set cover instance (decision variant),
  - let $S = \{1,2,3,4,5,6,7,8,9\}$ (set of edges),
  - for each vertex let there be a subset with the vertex’s adjacent edges: \{1,4\}, \{1,2,5\}, \{2,3\}, \{4,6,7\}, \{3,6,8,9\}, \{9\}, \{5,7,8\}
  - target size = #vertices - $k = 7 - 4 = 3$

- Claim: answer to both instances is the same (why??)
Reducing independent set to set cover

• In set cover instance (decision variant),
  – let $S = \{1,2,3,4,5,6,7,8,9\}$ (set of edges),
  – for each vertex let there be a subset with the vertex’s adjacent edges: $\{1,4\}$, $\{1,2,5\}$, $\{2,3\}$, $\{4,6,7\}$, $\{3,6,8,9\}$, $\{9\}$, $\{5,7,8\}$
  – target size = #vertices - $k = 7 - 4 = 3$

Claim: answer to both instances is the same (why??)

• So which of the two problems is harder?
Reductions:

To show problem Q is easy:

\[ Q \xrightarrow{reduce} \text{Problem known to be easy (e.g., LP)} \]

To show problem Q is (NP-)hard:

\[ \text{Problem known to be (NP-)hard (e.g., set cover, (M)IP)} \xrightarrow{reduce} Q \]
Polynomial time reductions

- Reduce A to B: a polynomial time algorithm that maps instances of A to instances of problem B, such that the answers are the same.

\[ A \leq_p B \]: B is at least as hard as A.

If you can solve B (in poly time) then you can solve A.
Weighted Bipartite Matching
Weighted bipartite matching

- Match each node on the left with one node on the right (can only use each node once)
- Minimize total cost (weights on the chosen edges)
Weighted bipartite matching…

• minimize $c_{ij} x_{ij}$
• subject to
• for every $i$, $\sum_j x_{ij} = 1$
• for every $j$, $\sum_i x_{ij} = 1$
• for every $i, j$, $x_{ij} \geq 0$

• Theorem [Birkhoff-von Neumann]: this linear program always has an optimal solution consisting of just integers
  – and typical LP solving algorithms will return such a solution

• So weighted bipartite matching is in P
Weighted bipartite matching

- Match each node on the left with one node on the right (can only use each node once)
- Minimize total cost (weights on the chosen edges)