Recitation 7: Linear Programming and Duality

1 Review of Linear Programming

Recall that a linear program is an optimization problem where in which a target value is maximized or minimized subject to some constraints. Each constraint has the form:

\[ a_1x_1 + a_2x_2 + \ldots + a_n x_n \{\leq, \geq, =\} b \]

where the values \( a_i \) and \( b \) are real numbers the term to optimize (the objective function) has the form:

\[ c_1x_1 + c_2x_2 + \ldots + c_n x_n \]

where each \( c_i \) has real values. Our goal is to choose the \( x_i \)'s such that the term is optimized. In matrix notation this is

\[
\begin{align*}
\text{Minimize or Maximize} & \quad c^T x \\
\text{Subject To} & \quad Ax \leq b
\end{align*}
\]

Here, \( A \) is a matrix, and \( c, x, \) and \( b \) are vectors. Note that \( A \) does not have to be square. Linear programming is extremely powerful as many common optimization problems we have seen can be expressed in this form. In lecture, for example, we saw that maximum flow can be expressed as a linear program. From a theoretical standpoint, this means that we can show a problem can be solved efficiently if we can expressed it in LP form.

Some definitions from literature:

**Feasibility:** A solution \( x \in \mathbb{R}^n \) is feasible if it satisfies all the constraints.
Optimality: A solution $x \in \mathbb{R}^n$ is optimal if it is feasible and it optimizes the value $c^T x$.

There are three possible types of solutions for a linear program.

- The LP is unfeasible, meaning that there is no way to satisfy all the constraints.
- The LP is unbounded, meaning that it is possible to choose a feasible solution $x$ such that the optimal is infinite.
- The LP has a feasible and bounded optimal solution.

Standard Form: A linear program is said to be in standard form if it has the form:
\[
\begin{align*}
\text{Maximize} & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
& \quad x_i \geq 0 \; \forall x_i \in x
\end{align*}
\]

Linear programs can be solved in polynomial time. The simplex algorithm, which is the simplest method and rather efficient in practice, does not always run in polynomial time. The first true polynomial time algorithm was the ellipsoid method, though it is not efficient in practice. Since then, various algorithms known as interior point methods have been devised, and these all run polynomial time and are slightly more efficient than the ellipsoid algorithm in practice.

2 Duality

One of the most important properties of linear programs is that of duality.

Duality: Take some LP written in standard form:
\[
\begin{align*}
\text{Maximize} & \quad c^T x \\
\text{subject to} & \quad Ax \leq b \\
& \quad x \geq 0
\end{align*}
\]
This is called the *primal form* of the LP. The *dual form* of this is LP is given by:

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

The most important property of duality is that \( b^T y = c^T x \). This is proved in CLRS by the use of *weak duality*. Weak duality asserts that the primal is bounded by the dual or that the dual is bounded by the primal. The proof of strong duality is shown by using the primal to bound the solution on the dual and vice versa, thus showing strong duality. While it takes some work to prove, it has some far-reaching consequences. One important practical result is that it allows us to prove that an optimal solution can be found for a linear program. Another important result is that if the primal is unfeasible, then the dual is unbounded (and vice-versa). In addition, it provides an important theoretical tool to relate problems.

This is a summary of how to switch variables and constraints from the primal to the dual:

- Every variable in the primal corresponds to a constraint in the dual and vice-versa.

- The constraint bounds vector \( b \) become part of the new objective function, and the vectors \( c \) becomes the new bounds on the constraints.

- A constraint of the form \( \leq b \) yields a variable which must be non-negative (i.e. \( \geq 0 \)), a constraint of the form \( \geq b_i \) yields a variable which must be non-positive (i.e. \( \leq 0 \)), and an equality constraint yields a free variable (i.e. can be positive or negative).

- The above applies in reverse when going from variable to constraint. Variables that are non-negative yield lower bound constraints, non-positivity constraints yield upper bound constraints, and free variables yield and equality constraints.

Since we will be using mainly standard form in this class, taking the dual should be relatively straightforward (just follow the definition).
3 Max Flow and Duality

Let us take maximum flow as an example. Let $x_{i,j}$ be the flow between nodes $i$ and $j$. The standard LP formulation is

\[
\text{maximize } \sum_{v \in V \setminus \{s\}} x_{s,v} \\
\text{subject to } x_{i,j} \leq c_{i,j} \text{ (capacity constraints)} \\
\sum_{j} x_{i,j} - x_{j,i} = 0 \forall v - \{s, t\} \text{ (conservation constraint)} \\
x_{i,j} \geq 0 \text{ (non-negative flows)}
\]

We can write this problem in matrix form. Our variable vector $x$ consists of a variable for every pair of nodes. The matrix $A$ will consist of the capacity constraint and the conservation constraints, the constraint bound vector $b$ consists of the capacity values $c_{i,j}$ and zeroes, and the vector in the objective function $c$ consists of zeroes everywhere except for a one corresponding to the path $(s, t)$.

Though we can find the dual of this problem from this form, there is a simpler way to do it by rewriting the linear program. Let us reformulate $x$ to have values $x_p$ which correspond to a flow on each path $p$ in the set of paths $P$ from $s$ to $t$. We can now write the equivalent LP:

\[
\text{maximize } \sum_{p} x_p \\
\text{subject to } \sum_{p:(i,j) \in p} x_p \leq c_{i,j} \forall (i,j) \in E \\
x_p \geq 0
\]

Now taking the dual is much simpler. The vector $c$ is all ones, and the vector $b$ is the capacity along each edge. The matrix $A$ has a +1 for each path that uses an edge $(u,v)$. In the dual, therefore, the rows have +1 for every edge along a path. The dual is therefore
minimize \( \sum_{i,j} c_{i,j} y_{i,j} \)

subject to \( \sum_{(u,v) \in p} y_{u,v} \geq 1 \ \forall p \in P \)
\[ y_{u,v} \geq 0 \]

The variables \( y_{u,v} \) correspond to edges, and therefore the constraints say that a path from \( s \) to \( t \) must include at least one chosen edge, where the variable \( y_{u,v} \) represents whether edge \( (u, v) \) is chosen in the solution. In other words, the variables are separating \( s \) from \( t \) by at least one edge. The objective function wishes to minimize the sum of these edges chosen for each path. Not suprisingly, this corresponds to the minimum cut problem, which we saw was the dual of maximum flow previously. Since duality relates the maxima of these two problems, we have just provided another proof of the max-flow / min cut theorem.

Note that re-writing the LP allowed us to take the dual much more easily. This is good to keep in mind: your choice of variables will greatly affect how you can use the LP. For example, the second primal formulation is impractical as the number of paths is exponential, so that writing the input is intractable. However, from a theoretical standpoint it was much easier to manipulate.

4 Integer Linear Programming and Total Unimodularity

Integer linear programming (ILP) is a problem analogous to linear programming with the additional restriction that all elements of the vector \( x \) must be integers. We can define standard form for an integer linear program similarly.

**ILP Standard Form:** An integer linear program is said to be in standard form if it has the form:
Maximize \( c^T x \)  
subject to  
\[ Ax \leq b \]
\[ x_i \geq 0 \quad \forall x_i \in x \]
\[ x_i \in \mathbb{Z} \quad \forall x_i \in x \]

While LP can be solved in polynomial time, ILP is NP-hard for a general program. However, given certain conditions on the matrix \( A \), we can show that there must exist an optimal integral solution to the equivalent LP. Specifically, a linear program is guaranteed to have an optimal integral solution if the matrix \( A \) is \textit{totally unimodular}, which we define below.

**Unimodular Matrix** : A matrix \( A \) is unimodular if it is a square integer matrix having determinant +1 or −1.

**Total Unimodularity** : A matrix \( A \) is said to be \textit{totally unimodular} if every square \( m \times m \) submatrix of \( A \) is unimodular.

The three sufficiency conditions for total unimodularity are:

- Each entry of \( A \) is \{+1, −1, 0\}
- Each column contains at most two non-zero coefficients.
- The rows of \( A \) can be partitioned into two sets \( M_1, M_2 \) such that the sum of the rows of \( M_1 \) is equal to the sum of the rows of \( M_2 \).

If \( A \) is totally unimodular, we can find an optimal integral solution in polynomial time by running a polynomial time LP algorithm on the corresponding LP.

Note that being totally unimodular is not a necessary condition on \( A \) for a linear program to have an optimal integral solution — i.e., the “if” above is \textit{not} an “if and only if”.