DS 574 Algorithmic Mechanism Design **Lecture #11**
Fall 2022 Prof. Kira Goldner

Last Time: Intro to LP Duality

Primal (P):

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \ge 0 \end{aligned}$$

Dual (D):

$$\begin{aligned} \min \quad & \mathbf{y}^T\mathbf{b} \\ \text{subject to} \quad & \mathbf{A}^T\mathbf{y} \ge \mathbf{c} \\ & \mathbf{y} \ge 0 \end{aligned}$$

The dual program is defined precisely so that weak duality holds: any feasible dual solution $y \in (D)$ provides an upper bound to any feasible primal solution $x \in (P)$ (and vice versa). The dual of the dual is the primal.

Theorem 1 (Weak Duality)**.** *If* $\mathbf{x}$ *is feasible in (P) and* $\mathbf{y}$ *is feasible in (D) then* $\mathbf{c}^T\mathbf{x} \le \mathbf{b}^T\mathbf{y}$.

Proof.

$$\mathbf{c}^T\mathbf{x} \overset{1}{\le} (\mathbf{A}^T\mathbf{y})\mathbf{x} = \mathbf{y}^T\mathbf{A}\mathbf{x} \overset{2}{\le} \mathbf{y}^T\mathbf{b} = \mathbf{b}^T\mathbf{y}.$$

Where (1) follows by the dual constraints $\mathbf{A}^T\mathbf{y} \ge \mathbf{c}$ and (2) follows by the primal constraints $\mathbf{A}\mathbf{x} \le \mathbf{b}$. □

What is not trivial (or by definition) is *strong duality*, and in fact, it is so involved that we will not even prove the hard direction: that an optimal solution always exists.

Conditions for Optimality

Strong Duality

Strong duality states that everything in fact needs to hold with equality to be optimal.

Theorem 2 (Strong Duality)**.** *A pair of solutions* $(\mathbf{x}^*, \mathbf{y}^*)$ *are optimal for the primal and dual respectively if and only if* $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^*$.

Proof. ($\Leftarrow$) The *if* direction is easy to see: we know that the dual gives an upper bound on the primal, so if these objectives are equal, then the primal objective that we are trying to maximize could not possible get any larger, as it's always *at most* the dual's objective. This is *as tight as possible.*

($\Rightarrow$) The *only if* direction is harder to prove, and we'll skip it for now. □

Complementary Slackness

We rewrite the primal and dual with each constraint separated, and then formalize another condition for optimality called *complementary slackness*, which states that for each corresponding constraint and variable, at most one can be slack in an optimal solution.

Primal (P):

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \sum_i a_{ji}x_i \le b_j \quad \forall j \quad (y_j) \\ & x_i \ge 0 \quad \forall i \end{aligned}$$

Dual (D):

$$\begin{aligned} \min \quad & \mathbf{y}^T\mathbf{b} \\ \text{subject to} \quad & \sum_i a_{ij}y_i \ge c_i \quad \forall i \quad (x_i) \\ & y_j \ge 0 \quad \forall j \end{aligned}$$

Theorem 3 (Complementary Slackness)**.** *A pair of solutions* $(\mathbf{x}^*, \mathbf{y}^*)$ *are optimal for the primal and dual respectively if and only if the following complementary slackness conditions (1) and (2) hold:*

$$\sum_i a_{ji}x_i = b_j \quad \textit{or} \quad y_j = 0 \qquad (1) \qquad\qquad \sum_i a_{ij}y_i = c_i \quad \textit{or} \quad x_i = 0. \qquad (2)$$

Proof. ($\Rightarrow$) According to complementary slackness, by rearranging our constraint, either $\sum_i a_{ji}x_i - b_j = 0$ or $y_j = 0$. This ensures that the multiplied quantity $\left(\sum_i a_{ji}x_i - b_j\right) y_j = 0$, as *one* of the two terms on the left-hand side must be 0. Then multiplying out and rearranging gives that $y_j \sum_i a_{ji}x_i = y_j b_j$. This process with all rows gives the equality from complementary slackness that $\mathbf{y}^T\mathbf{A}\mathbf{x} = \mathbf{y}^T\mathbf{b}$.

Similarly, using the condition that $\sum_i a_{ij}y_i = c_i$ or $x_i = 0$ gives that $\mathbf{c}^T\mathbf{x} = (\mathbf{A}^T\mathbf{y})\mathbf{x}$.

Then following our inequalities in the proof of weak duality, they now all hold with equality, so by Strong Duality, $(\mathbf{x}, \mathbf{y})$ are optimal solutions to the primal and dual.

$$\mathbf{c}^T\mathbf{x} \quad = \quad (\mathbf{A}^T\mathbf{y})\mathbf{x} \quad = \quad \mathbf{y}^T\mathbf{A}\mathbf{x} \quad = \quad \mathbf{y}^T\mathbf{b} \quad = \quad \mathbf{b}^T\mathbf{y}.$$

($\Leftarrow$) Similarly, if Strong Duality holds, the above inequalities hold with equality, in which case it must be that $y_j \sum_i a_{ji}x_i = y_j b_j$ for all j and $\sum_i a_{ij}y_i x_i = c_i x_i$ for all i, and hence that either $\sum_i a_{ji}x_i - b_j = 0$ or $y_j = 0$ for all j and that either $\sum_i a_{ij}y_i = c_i$ or $x_i = 0$ for all i. □

Maximizing Welfare in the Unit Demand Setting

Given n unit-demand bidders and m items, determine the allocation rule that maximizes welfare. We do this by formulating a linear program, determining our objective, decision variables, and constraints:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij}x_{ij} & & \\
\text{subject to} \quad & \sum_{i=1}^{n} x_{ij} \le 1 & \forall j \quad & \text{(items allocated at most once)} \\
& \sum_{j=1}^{m} x_{ij} \le 1 & \forall i \quad & \text{(bidders unit demand)} \\
& x_{ij} \ge 0 & \forall i,j \quad & \text{(non-negativity)}
\end{aligned}$$

We then formulate the dual:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n} u_i + \sum_{j=1}^{m} p_j & & \\
\text{subject to} \quad & u_i + p_j \ge v_{ij} & \forall (i,j) \quad & \text{(IC)} \\
& u_i, p_j \ge 0 & \forall i,j \quad & \text{(non-negativity)}
\end{aligned}$$

By rewriting our first constraint as

$$u_i \ge v_{ij} - p_j,$$

we reinterpret it as an incentive compatibility constraint. Instead of determining an allocation, we determine buyer utilities and item prices, the sum of which we minimize.

On our homework, we will use this to prove that for gross substitutes valuations, the optimal primal allocation $\mathbf{x}$ and dual solution $\mathbf{p}$ form a Walrasian equilibria, and this is precisely the valuation class for which welfare can be maximized in polynomial time.

We will need to use the following theory of when there exist polynomial-time algorithms for linear programs.

Separation Oracles

Fact 1 (Ellipsoid Algorithm)**.** Every linear program that admits a polynomial-time separation oracle can be solved in polynomial time.

Consider a linear program such that:

1. There are n decision variables.
2. There are any number of constraints, for example, exponential in n. These constraints are not provided explicitly as input.
3. There is a polynomial-time *separation oracle* for the set of constraints. By "polynomial-time," we mean running time polynomial in n and the maximum number of bits of precision required.

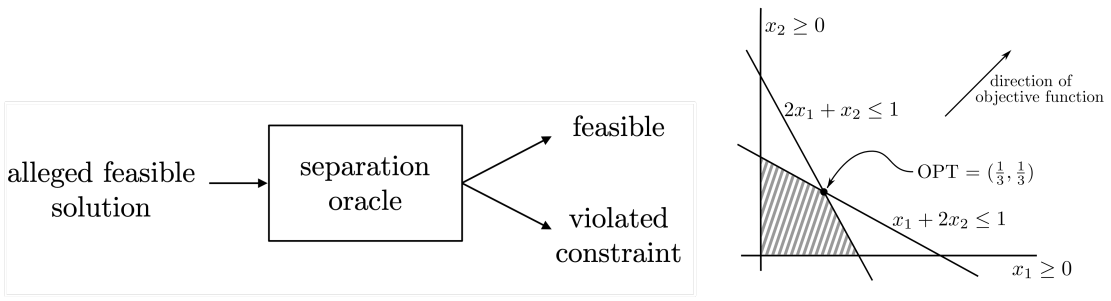

Figure 1: Left: A sketch of a separation oracle. For example, in the toy example on the right, on the alleged feasible solution $(\frac{1}{3}, \frac{1}{2})$, the separation oracle may return the violated constraint $x_1 + 2x_2 \leq 1$.

> A separation oracle (Figure 1) is a subroutine that takes as input an alleged feasible solution to the LP, and either (i) correctly declares the solution to be feasible, or (ii) correctly declares the solution to be infeasible, and more strongly provides a proof of infeasibility in the form of a constraint that the proposed solution violates.

(The ellipsoid algorithm is not actually practical, but there are other algorithms that *are* often practically useful that rely on a separation oracle, such as cutting plane methods.)