# CS 170: The Simplex Algorithm, Duality, and the Klee–Minty Cube — Lecture Notes

**Course:** CS 170 — Efficient Algorithms and Intractable Problems, Spring 2026 **Instructors:** Lijie Chen, Umesh V. Vazirani **Lectures 16 & 17**

---

## Part 1: Why Simplex?

In the previous lectures we saw that linear programming is an extraordinarily powerful framework: we can express shortest paths, max-flow, matching, and many other problems as LPs. We also saw that the optimum of an LP is always attained at a **vertex** of the feasible polyhedron (unless the LP is infeasible or unbounded).

This suggests a natural strategy: **walk along the vertices** of the polyhedron, always moving to a neighbor with a better objective value, until no improvement is possible. This is exactly what the **simplex algorithm** does. It was invented by George Dantzig in 1947 and remains one of the most practically efficient algorithms for solving LPs.

### 1.1 High-Level Idea

```
let v be any vertex of the feasible region
while there is a neighbor v' of v with better objective value:
    set v = v'
```

The key questions we need to answer are: what is a "vertex" algebraically? What is a "neighbor"? And how do we efficiently move from one to the next? We will build up to the full algorithm through two concrete examples.

---

## Part 2: A Simple Example — Two Variables

### 2.1 The Problem

A small bakery makes two products: **cookies** and **cakes**. Each batch of cookies brings 1 dollar profit; each batch of cakes brings 3 dollars profit. The bakery faces three constraints:

- At most 4 batches of cookies per day.
- At most 6 batches of cakes per day.
- Total batches (cookies + cakes) at most 8.

Letting $x_1$ = batches of cookies and $x_2$ = batches of cakes, we get:

$$\max\ x_1 + 3x_2$$

subject to:

$$x_1 \leq 4 \qquad (1)$$

$$x_2 \leq 6 \qquad (2)$$

$$x_1 + x_2 \leq 8 \qquad (3)$$
$$x_1, x_2 \geq 0 \qquad (4), (5)$$

### 2.2 The Feasible Region

Each inequality defines a half-plane. The feasible region is the intersection of these five half-planes — a convex polygon in 2D. The vertices of this polygon are the candidate optima.

**Figure 1** below shows the feasible region (shaded) and the vertices of the polygon, along with contour lines of the objective $x_1 + 3x_2 = c$ for increasing values of $c$.

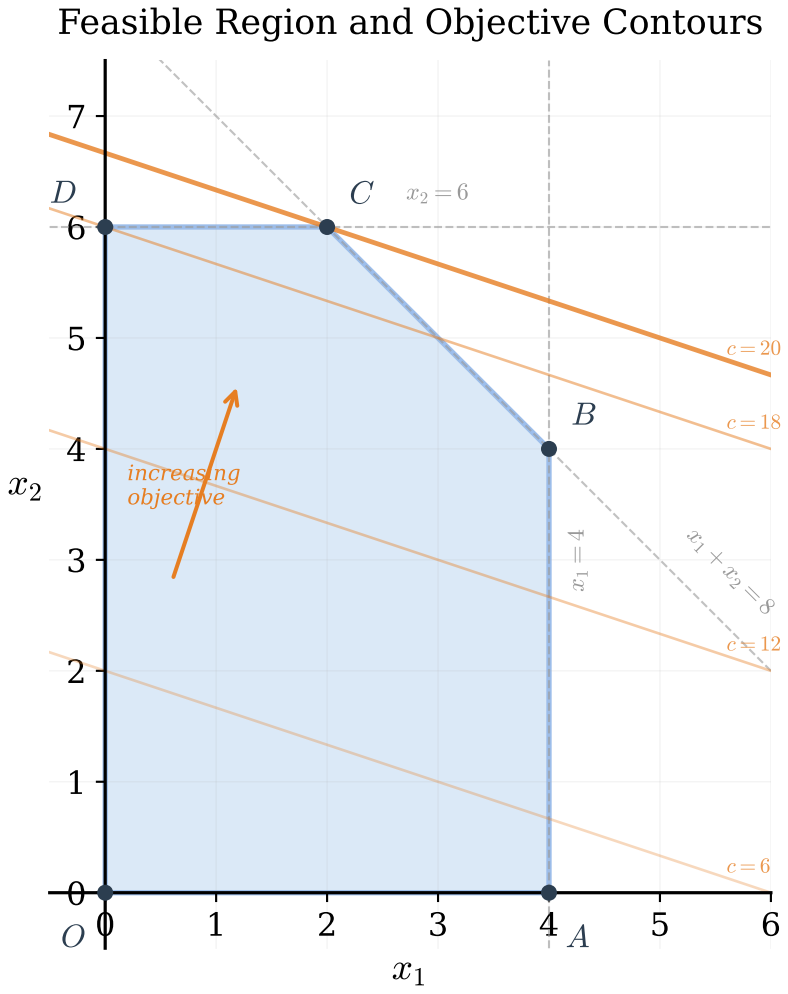


Figure 1: Feasible region for the 2-variable LP

The vertices of the feasible polygon are:

| Vertex | $(x_1, x_2)$ | Tight constraints | Objective value |
|---|---|---|---|
| O | $(0, 0)$ | (4), (5) | 0 |
| A | $(4, 0)$ | (1), (5) | 4 |
| B | $(4, 4)$ | (1), (3) | 16 |

| Vertex | $(x_1, x_2)$ | Tight constraints | Objective value |
|---|---|---|---|
| C | $(2, 6)$ | (2), (3) | 20 |
| D | $(0, 6)$ | (2), (4) | 18 |

### 2.3 Simplex Walkthrough

**Start** at the origin $O = (0, 0)$, which is defined by the tight constraints $\{(4), (5)\}$, i.e. $x_1 \geq 0$ and $x_2 \geq 0$. The objective value is 0.

**Iteration 1.** We look at the objective $x_1 + 3x_2$. Both coefficients are positive, so increasing either variable improves the objective. We pick $x_2$ (larger coefficient = steeper improvement). We release constraint (5): $x_2 \geq 0$ and increase $x_2$. As $x_2$ grows, we eventually hit constraint (2): $x_2 \leq 6$ at $x_2 = 6$. New vertex: $D = (0, 6)$, defined by $\{(2), (4)\}$. Objective value: 18.

**Iteration 2.** At vertex $D$, we look for an adjacent edge along which the objective improves. Increasing $x_1$ (while $x_2$ stays at 6) does the trick. As $x_1$ grows, we hit constraint (3): $x_1 + x_2 \leq 8$ when $x_1 = 2$. New vertex: $C = (2, 6)$, defined by $\{(2), (3)\}$. Objective value: 20.

**Iteration 3.** At vertex $C$, there is no direction we can move (increasing either local coordinate) that improves the objective. Simplex **halts** and declares $C = (2, 6)$ optimal with profit $ dollars20$.

### 2.4 Why Local Optimality Implies Global Optimality

Think of the profit line $x_1 + 3x_2 = c$ passing through the optimal vertex $C$. All of $C$'s neighbors lie on or below this line. Since the feasible region is **convex**, the entire polyhedron lies on the same side of this line. Therefore no feasible point can achieve a higher objective value.

---

## Part 3: The Simplex Algorithm in General

### 3.1 Vertices as Tight Constraints

Consider a general LP with $n$ variables and $m$ inequality constraints (plus $n$ non-negativity constraints):

$$\max \ \mathbf{c}^T\mathbf{x} \quad \text{s.t.} \quad A\mathbf{x} \leq \mathbf{b}, \ \ \mathbf{x} \geq 0.$$

The total number of inequality constraints is $m + n$. A **vertex** is a feasible point at which exactly $n$ of these inequalities are satisfied with equality (i.e., $n$ constraints are "tight"). These $n$ tight constraints give $n$ linear equations in $n$ unknowns, which (generically) determine a unique point.

**Key definition:** Each vertex is specified by a set of $n$ tight inequalities.

### 3.2 Neighbors

Two vertices are **neighbors** if they share $n - 1$ tight constraints. In other words, to go from one vertex to a neighbor, we "release" one tight constraint and "tighten" another.

In 2D ($n = 2$): each vertex is defined by 2 tight constraints; neighbors share 1 — they are connected by an **edge** of the polygon.

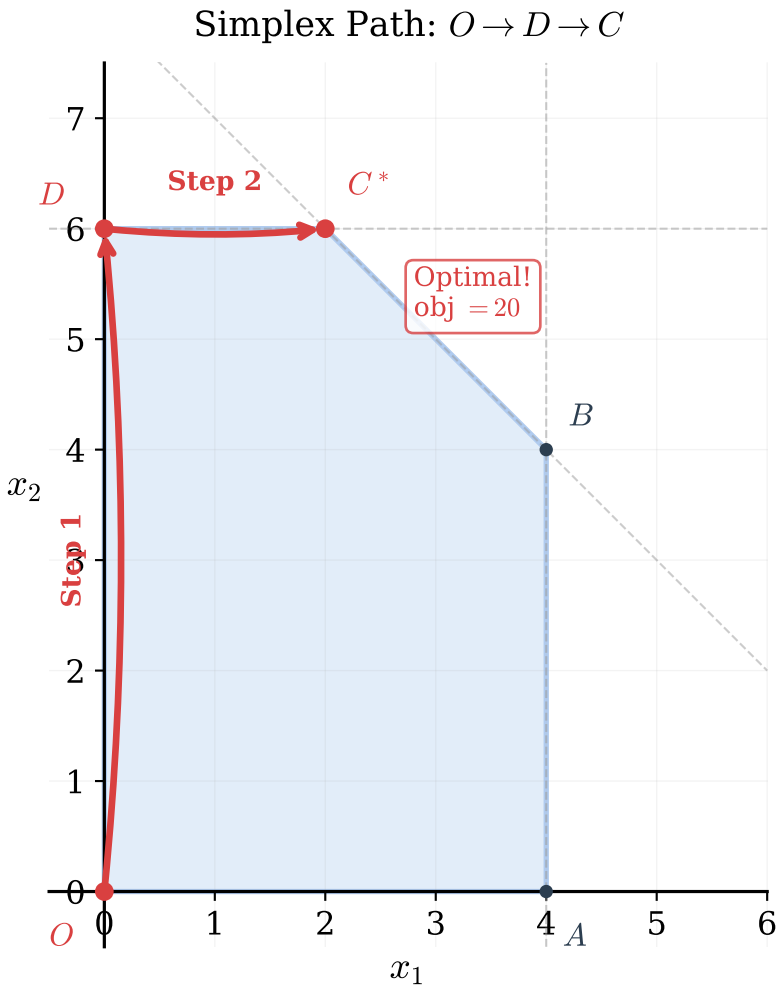


Figure 2: Simplex path on the 2D polygon

In 3D ($n = 3$): each vertex is defined by 3 tight constraints; neighbors share 2 — they are connected by an **edge** of the polyhedron.

### 3.3 The Algorithm: Origin-Centered View

Simplex is simplest when the current vertex is the **origin**. Suppose we have:

$$\max \ \mathbf{c}^T\mathbf{x} \quad \text{s.t.} \quad A\mathbf{x} \leq \mathbf{b}, \ \ \mathbf{x} \geq 0$$

and the origin is feasible (i.e. $\mathbf{b} \geq 0$). Then:

**Task 1: Is the origin optimal?** Yes, if and only if all $c_i \leq 0$. (Since $\mathbf{x} \geq 0$, no variable can be increased to improve the objective.)

**Task 2: If not, which direction to move?** Pick any $i$ with $c_i > 0$. Release the constraint $x_i \geq 0$ and increase $x_i$ while keeping all other tight constraints satisfied. Stop when a new constraint becomes tight.

### 3.4 The Coordinate-Change Trick

What if the current vertex $\mathbf{u}$ is not the origin? **Transform coordinates** so that $\mathbf{u}$ maps to the origin!

For each of the $n$ tight constraints $\mathbf{a}_i \cdot \mathbf{x} \leq b_i$, define a "distance variable":

$$y_i = b_i - \mathbf{a}_i \cdot \mathbf{x}$$

In these $\mathbf{y}$-coordinates:

1. The $n$ tight constraints become $y_i \geq 0$.
2. The current vertex $\mathbf{u}$ is the origin ($\mathbf{y} = \mathbf{0}$).
3. The objective becomes $c_{\mathbf{u}} + \bar{\mathbf{c}}^T\mathbf{y}$, where $c_{\mathbf{u}}$ is the objective value at $\mathbf{u}$ and $\bar{\mathbf{c}}$ is the transformed cost vector.

Now we're back to the origin case! Check if all $\bar{c}_i \leq 0$ (optimal) or pick some $\bar{c}_i > 0$ (move).

The figure below illustrates this on the bakery LP from Part 2. At vertex $C = (2, 6)$, the tight constraints are $x_2 \leq 6$ and $x_1 + x_2 \leq 8$. We define $y_1 = 6 - x_2$ and $y_2 = 8 - x_1 - x_2$. In local coordinates, $C$ maps to the origin and the objective becomes $20 - 2y_1 - y_2$. Since both coefficients $\bar{c}_1 = -2$ and $\bar{c}_2 = -1$ are negative, $C$ is optimal.

### 3.5 Simplex: Complete Algorithm

**Procedure** Simplex($\max \ \mathbf{c}^\top\mathbf{x}$ s.t. $A\mathbf{x} \leq \mathbf{b}, \ \mathbf{x} \geq \mathbf{0}$):

1. Let $\mathbf{u}$ be any feasible vertex.
2. **while** true:
   - Rewrite the LP in local coordinates $\mathbf{y}$ centered at $\mathbf{u}$; the objective becomes $c_{\mathbf{u}} + \bar{\mathbf{c}}^\top\mathbf{y}$.
   - **if** $\bar{c}_i \leq 0$ for all $i$: **return u** *(optimal!)*
   - Pick $i$ with $\bar{c}_i > 0$.

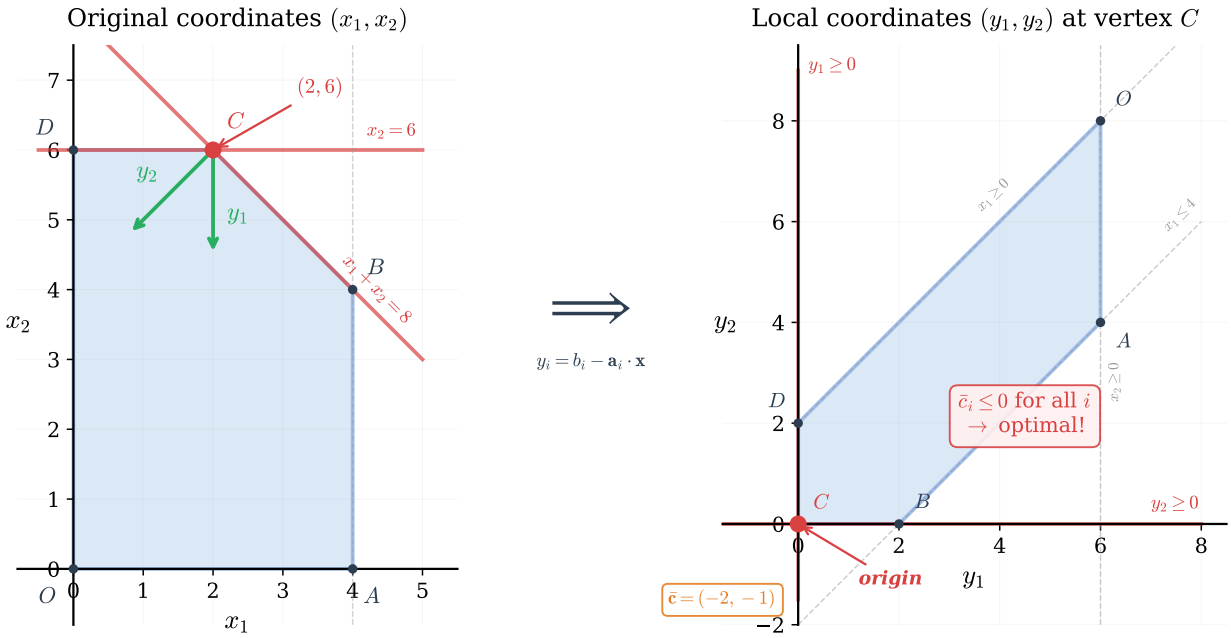


Figure 3: The coordinate-change trick: the feasible region in original coordinates (left) and local coordinates centered at vertex $C$ (right).

- Increase $y_i$ until a new constraint $\mathbf{a}_j \cdot \mathbf{x} = b_j$ becomes tight.
- $\mathbf{u} \leftarrow$ the new vertex.

**Why does it terminate?** Each step strictly improves the objective (assuming non-degeneracy). Since there are finitely many vertices, simplex must halt.

**Why is the answer correct?** When simplex halts, all local cost coefficients are $\le 0$. This means no neighboring vertex has a better objective value. By convexity, this **local** optimality implies **global** optimality.

### 3.6 An Illustrative Walkthrough (from the textbook)

Consider the LP:

$$\max \; 2x_1 + 5x_2$$

$$\begin{aligned} 2x_1 - x_2 &\le 4 \quad (1) \\ x_1 + 2x_2 &\le 9 \quad (2) \\ -x_1 + x_2 &\le 3 \quad (3) \\ x_1 \ge 0 \quad (4), \quad x_2 &\ge 0 \quad (5) \end{aligned}$$

**Start:** vertex $\{(4), (5)\}$ = origin $(0, 0)$. Objective: 0.

**Move 1:** increase $x_2$ (coefficient $5 > 0$). Release (5). First constraint hit: (3) at $x_2 = 3$. New vertex: $\{(3), (4)\} = (0, 3)$. Objective: 15.

**Move 2:** In local coordinates at $(0, 3)$, increasing $y_1 = x_1$ improves objective. Release (4). Constraint (2): $x_1 + 2(3) \leq 9 \Rightarrow x_1 \leq 3\ldots$ but also via (3): releasing and adjusting… $x_1 + 2x_2 \leq 9$ becomes tight at $x_1 = 1, x_2 = 4$ (solving (2) and (3) as equalities). New vertex: $\{(2), (3)\} = (1, 4)$. Objective: 22.

**Halt:** All local cost coefficients $\leq 0$. Optimal: $(1, 4)$ with value 22.

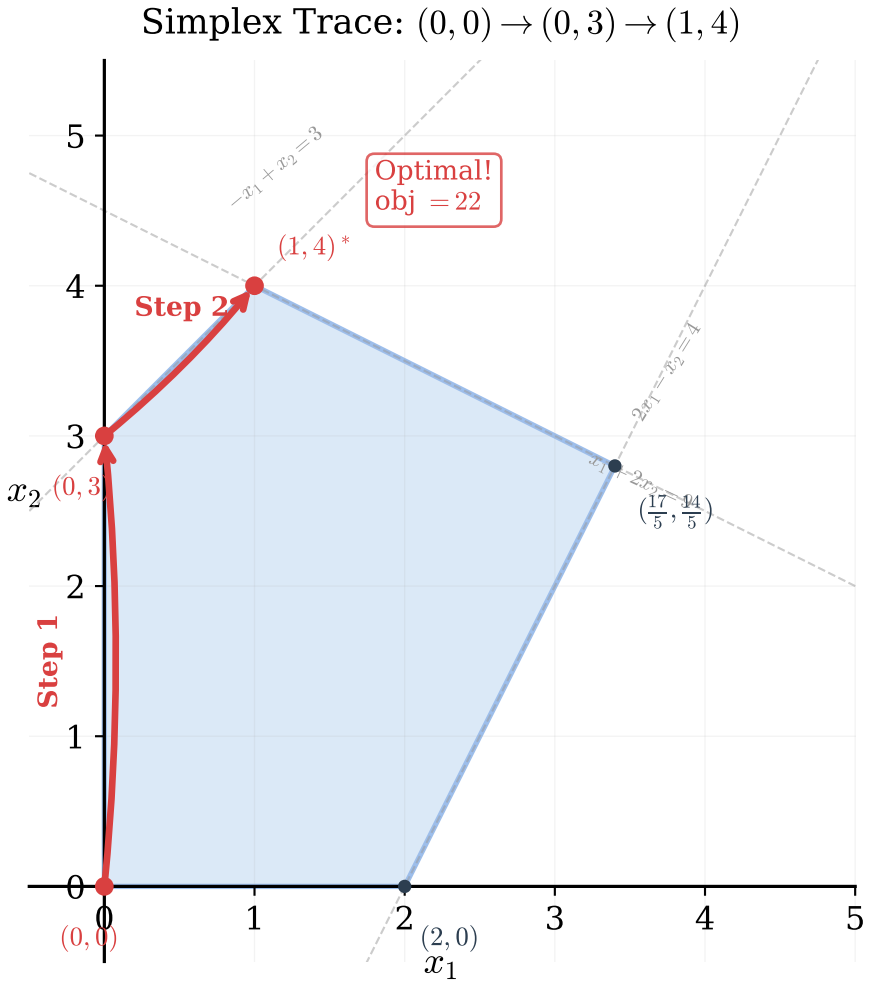


Figure 4: Simplex trace on the 2D example from the textbook

---

## Part 4: Duality — Certificates of Optimality

### 4.1 An Upper-Bounding Trick

Consider the LP from Part 2:

$$\max\ x_1 + 3x_2$$

$$x_1 \leq 4 \qquad (1)$$
$$x_2 \leq 6 \qquad (2)$$
$$x_1 + x_2 \leq 8 \qquad (3)$$
$$x_1, x_2 \geq 0$$

Simplex found the optimum $(x_1, x_2) = (2, 6)$ with value 20. Can we **certify** this is truly optimal?

Here's a clever idea: multiply constraint (2) by 2 and constraint (3) by 1:

$$2 \cdot (x_2 \leq 6) \implies 2x_2 \leq 12$$
$$1 \cdot (x_1 + x_2 \leq 8) \implies x_1 + x_2 \leq 8$$

Adding these: $x_1 + 3x_2 \leq 20$.

This tells us that no feasible solution can achieve profit exceeding 20. Since we found a feasible point achieving exactly 20, it must be optimal! The multipliers $(y_1, y_2, y_3) = (0, 2, 1)$ serve as a **certificate of optimality**.

### 4.2 Systematizing the Trick: The Dual LP

What we did can be generalized. Assign a non-negative multiplier $y_i$ to each constraint. By taking a weighted combination:

$$y_1 \cdot (x_1 \leq 4) + y_2 \cdot (x_2 \leq 6) + y_3 \cdot (x_1 + x_2 \leq 8)$$

we get:

$$(y_1 + y_3)x_1 + (y_2 + y_3)x_2 \;\leq\; 4y_1 + 6y_2 + 8y_3$$

For this to be a valid upper bound on $x_1 + 3x_2$, we need the left-hand coefficients to **dominate** the objective coefficients:

$$y_1 + y_3 \geq 1, \qquad y_2 + y_3 \geq 3$$

And we want the tightest possible upper bound, so we **minimize** the right-hand side. This gives a brand-new LP:

$$\min \; 4y_1 + 6y_2 + 8y_3$$

$$y_1 + y_3 \geq 1$$
$$y_2 + y_3 \geq 3$$
$$y_1, y_2, y_3 \geq 0$$

This is the **dual** LP. The original LP is called the **primal**.

### 4.3 The General Dual

For a general LP in standard form:

$$\textbf{Primal:} \quad \max \ \mathbf{c}^T\mathbf{x} \quad \text{s.t.} \quad A\mathbf{x} \leq \mathbf{b}, \ \ \mathbf{x} \geq 0$$

the dual is:

$$\textbf{Dual:} \quad \min \ \mathbf{y}^T\mathbf{b} \quad \text{s.t.} \quad \mathbf{y}^T A \geq \mathbf{c}^T, \ \ \mathbf{y} \geq 0$$

The matrix $A$ that defines primal constraints by its **rows** defines dual constraints by its **columns**. There is a beautiful symmetry: one primal constraint per dual variable, and one dual constraint per primal variable.

### 4.4 Weak and Strong Duality

Recall the two key duality theorems from previous lectures. **Weak duality** says that every dual feasible $\mathbf{y}$ gives an upper bound on the primal: $\mathbf{c}^T\mathbf{x} \leq \mathbf{y}^T\mathbf{b}$. **Strong duality** says that when a bounded optimum exists, this bound is tight: Primal OPT = Dual OPT. Together, they mean that finding a primal-dual pair with equal objective values certifies optimality for both — which is exactly what we did in our example above.

---

## Part 5: Proving Simplex Correct via Duality

### 5.1 The Correctness Argument

We can now prove that simplex returns the global optimum.

**Claim:** When simplex halts at a vertex $\mathbf{x}^*$, the LP is optimally solved.

*Proof.* Recall how the coordinate rewrite works. At vertex $\mathbf{x}^*$, suppose the $n$ tight constraints are $\mathbf{a}_1 \cdot \mathbf{x} \leq b_1, \ldots, \mathbf{a}_n \cdot \mathbf{x} \leq b_n$. We introduce local variables $y_i = b_i - \mathbf{a}_i \cdot \mathbf{x} \geq 0$, so that $\mathbf{x}^*$ maps to $\mathbf{y} = \mathbf{0}$. In these coordinates, the objective becomes:

$$\mathbf{c}^T\mathbf{x} \ = \ c^* + \bar{c}_1 y_1 + \cdots + \bar{c}_n y_n$$

where $c^* = \mathbf{c}^T\mathbf{x}^*$ is the objective value at $\mathbf{x}^*$ and the $\bar{c}_i$ are the transformed cost coefficients. When simplex halts, all $\bar{c}_i \leq 0$.

Now we construct a dual certificate. Define $y_i^* = -\bar{c}_i \geq 0$ for the $n$ tight constraints, and $y_i^* = 0$ for all non-tight constraints. We need to check dual feasibility: $\mathbf{y}^{*T} A \geq \mathbf{c}^T$, i.e., for each variable $x_j$ we need $\sum_i y_i^* a_{ij} \geq c_j$.

Since the $n$ tight constraints at $\mathbf{x}^*$ define the local coordinates, we can write $\mathbf{x}$ in terms of $\mathbf{y}$ by inverting $y_i = b_i - \mathbf{a}_i \cdot \mathbf{x}$. Let $S$ be the $n \times n$ matrix with rows $\mathbf{a}_1, \ldots, \mathbf{a}_n$ (the tight constraints). Then $\mathbf{y} = \mathbf{b}_S - S\mathbf{x}$, so $\mathbf{x} = S^{-1}(\mathbf{b}_S - \mathbf{y})$. Substituting into the objective:

$$\mathbf{c}^T\mathbf{x} = \mathbf{c}^T S^{-1}\mathbf{b}_S - \mathbf{c}^T S^{-1}\mathbf{y} = c^* + \bar{\mathbf{c}}^T\mathbf{y}$$

so the transformed cost vector is $\bar{\mathbf{c}} = -S^{-T}\mathbf{c}$, i.e., $\mathbf{c} = -S^T\bar{\mathbf{c}}$. Our dual multipliers for the tight constraints are $\mathbf{y}_S^* = -\bar{\mathbf{c}}$, so:

$$(\mathbf{y}_S^*)^T S = (-\bar{\mathbf{c}})^T S = \mathbf{c}^T S^{-1} S = \mathbf{c}^T$$

This says that the contribution from the tight constraints alone already matches $\mathbf{c}^T$ exactly. Since the remaining (non-tight) constraints have $y_i^* = 0$ and contribute non-negatively, we get $\mathbf{y}^{*T} A \geq \mathbf{c}^T$. So $\mathbf{y}^*$ is dual feasible.

Finally, for the dual objective: since $y_i^* = 0$ for every non-tight constraint and $\mathbf{a}_i \cdot \mathbf{x}^* = b_i$ for every tight constraint:

$$\mathbf{y}^{*T}\mathbf{b} = \sum_{\text{tight } i} y_i^* \cdot b_i = (-\bar{\mathbf{c}})^T \mathbf{b}_S = \mathbf{c}^T S^{-1} \mathbf{b}_S = \mathbf{c}^T \mathbf{x}^* = c^*$$

So the dual objective equals the primal objective at $\mathbf{x}^*$. By weak duality, no feasible solution can do better than $c^*$. Therefore $\mathbf{x}^*$ is optimal. ■

**Concrete check.** In the bakery example, simplex halts at $C = (2, 6)$ with transformed costs $\bar{c}_1 = -2, \bar{c}_2 = -1$. The tight constraints are (2): $x_2 \leq 6$ and (3): $x_1 + x_2 \leq 8$. Setting $y_2^* = 2, y_3^* = 1$ (and $y_1^* = 0$) gives dual objective $0 \cdot 4 + 2 \cdot 6 + 1 \cdot 8 = 20 = c^*$. This is exactly the certificate we found in Part 4.

---

## Part 6: Loose Ends

### 6.1 Finding a Starting Vertex

We assumed simplex starts at a feasible vertex — but how do we find one? In our examples the origin worked because all $b_i \geq 0$. In general, this won't hold.

**The Big-M / Two-Phase method:** Convert the LP to standard form $\min \mathbf{c}^T\mathbf{x}$ s.t. $A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq 0$ (make all $b_i \geq 0$ by flipping signs). Create an **auxiliary LP** with artificial variables $z_1, \ldots, z_m \geq 0$:

$$\min\ z_1 + z_2 + \cdots + z_m \quad \text{s.t.} \quad A\mathbf{x} + I\mathbf{z} = \mathbf{b},\ \ \mathbf{x}, \mathbf{z} \geq 0$$

This auxiliary LP has an obvious starting vertex: $\mathbf{x} = 0, \mathbf{z} = \mathbf{b}$. Solve it with simplex. If the optimum is 0, we've found a feasible vertex for the original LP. If not, the original LP is **infeasible**.

### 6.2 Degeneracy

A vertex is **degenerate** if more than $n$ constraints are tight at that point. This can cause simplex to cycle — it moves to "neighboring" vertices with the same objective value, potentially forever. In practice, a small random **perturbation** of the $b_i$'s resolves degeneracy.

### 6.3 Matrix-Vector Notation Recap

For reference, the standard LP forms:

**Inequality form:** max $\mathbf{c}^T\mathbf{x}$ s.t. $A\mathbf{x} \leq \mathbf{b},\ \mathbf{x} \geq 0$

**Standard form** (with slack variables $\mathbf{s}$): max $\mathbf{c}^T\mathbf{x}$ s.t. $A\mathbf{x} + \mathbf{s} = \mathbf{b},\ \mathbf{x}, \mathbf{s} \geq 0$

Each row of $A$ is one constraint; the slack variable $s_i = b_i - \mathbf{a}_i \cdot \mathbf{x}$ measures how far constraint $i$ is from being tight.

---

## Part 7: Running Time of Simplex

### 7.1 Cost Per Iteration

Consider a generic LP with $n$ variables and $m$ inequality constraints:

$$\max\ \mathbf{c}^T\mathbf{x} \quad \text{s.t.} \quad A\mathbf{x} \leq \mathbf{b},\ \ \mathbf{x} \geq 0$$

At each iteration, simplex sits at a vertex $\mathbf{u}$ defined by $n$ tight constraints. The key operations are:

1. **Rewrite the LP in local coordinates.** This involves a change of basis — essentially solving an $n \times n$ linear system. However, since the basis changes by only **one constraint** per iteration, the update can be done incrementally in $O((m+n) \cdot n)$ time.
2. **Choose a direction to move.** Inspect the transformed cost coefficients $\bar{c}_i$ — pick any positive one. This is $O(n)$.
3. **Determine the step length.** Check which constraint becomes tight first — a ratio test over $O(m)$ constraints.

Total: $O(mn)$ per iteration using the incremental update.

### 7.2 How Many Iterations?

Each vertex is specified by choosing $n$ tight constraints out of $m+n$ total, so there are at most $\binom{m+n}{n}$ vertices. This is an **exponential** upper bound in general.

The natural question: can simplex actually visit exponentially many vertices? The answer, disturbingly, is **yes**.

---

## Part 8: The Klee–Minty Example

### 8.1 Background

In 1972, Victor Klee and George Minty constructed a family of LPs on which the simplex algorithm (with the standard "largest coefficient" pivot rule) visits **every single vertex** of the feasible polyhedron before reaching the optimum. Since the polyhedra have $2^n$ vertices, simplex takes $2^n - 1$ steps.

### 8.2 The Construction

The **Klee–Minty cube** in $n$ dimensions is defined as:

$$\max \sum_{j=1}^{n} 2^{n-j} x_j$$

subject to:

$$\sum_{j=1}^{i-1} 2^{i-j+1} x_j + x_i \leq 5^i \qquad \text{for } i = 1, \ldots, n$$

$$x_j \geq 0 \qquad \text{for } j = 1, \ldots, n$$

### 8.3 The $n = 2$ Case

For $n = 2$, the LP is:

$$\max \; 2x_1 + x_2$$

$$x_1 \leq 5 \qquad (1)$$
$$4x_1 + x_2 \leq 25 \qquad (2)$$
$$x_1, x_2 \geq 0$$

The feasible region is a quadrilateral with 4 vertices:

| Vertex | $(x_1, x_2)$ | Objective |
|---|---|---|
| $O$ | $(0, 0)$ | 0 |
| $A$ | $(5, 0)$ | 10 |
| $B$ | $(5, 5)$ | 15 |
| $C$ | $(0, 25)$ | 25 |

The optimum is at $C = (0, 25)$. But simplex (using the "largest coefficient" rule, which picks $x_1$ first since $2 > 1$) takes the path:

$$O \to A \to B \to C$$

visiting all $2^2 = 4$ vertices. The direct path $O \to C$ would have sufficed, but the pivot rule doesn't find it.

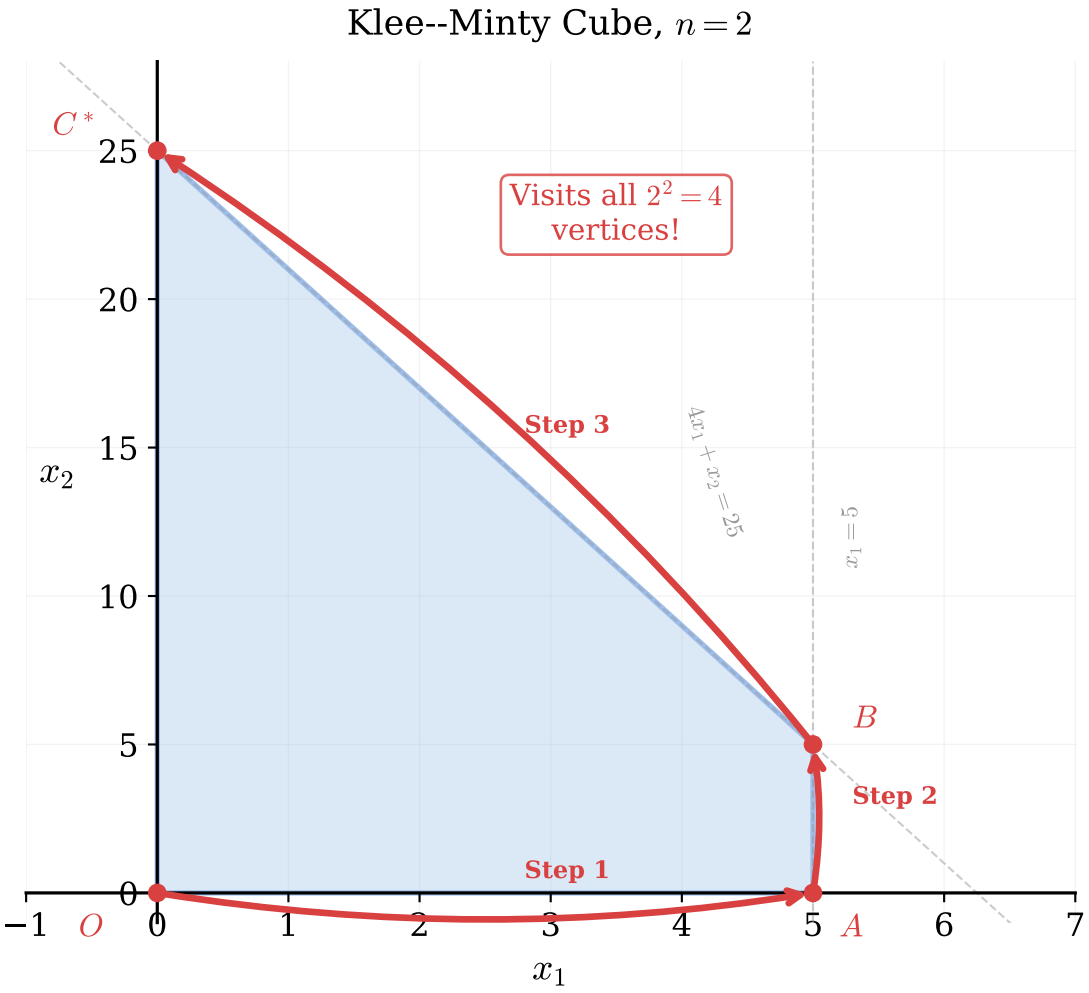


Figure 5: Klee-Minty cube n=2

### 8.4 The $n = 3$ Case

For $n = 3$:

$$\max \; 4x_1 + 2x_2 + x_3$$

$$x_1 \leq 5 \qquad (1)$$
$$4x_1 + x_2 \leq 25 \qquad (2)$$
$$8x_1 + 4x_2 + x_3 \leq 125 \qquad (3)$$
$$x_1, x_2, x_3 \geq 0$$

This has $2^3 = 8$ vertices, and simplex visits all of them in a winding path through the "squeezed cube":

$$(0,0,0) \to (5,0,0) \to (5,5,0) \to (0,25,0) \to (0,25,25) \to (5,5,105) \to (5,0,85) \to (0,0,125)$$

The geometry of the Klee–Minty cube is a regular cube that has been **skewed** so that the vertices are visited in a Hamiltonian path — the simplex algorithm must traverse all $2^n$ vertices to reach the optimum at the far corner.

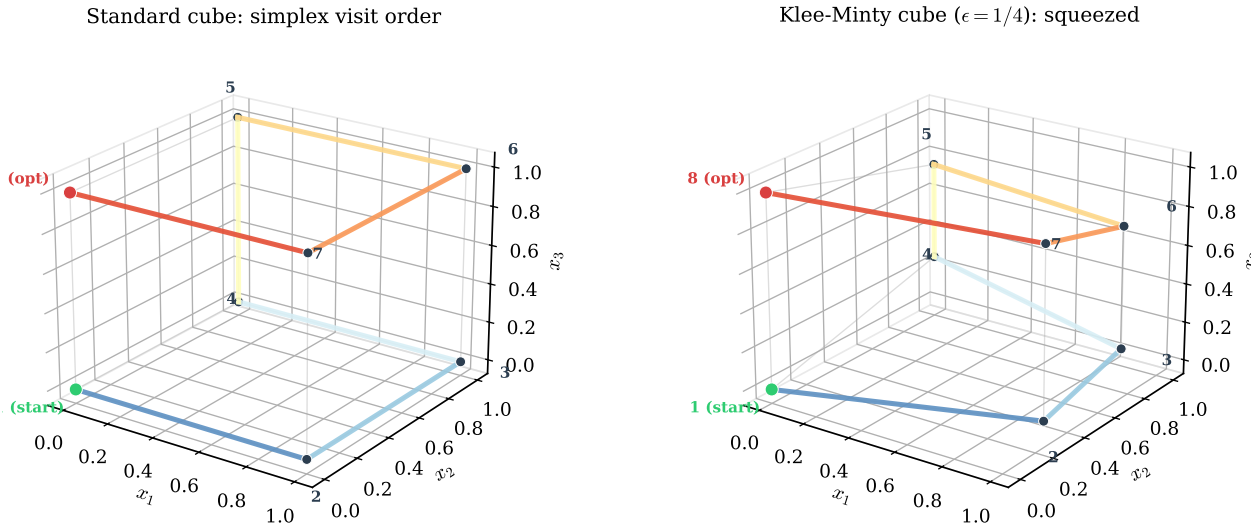


Figure 6: Klee-Minty cube n=3

### 8.5 Why This Happens

The key insight is in the objective function. The coefficients $2^{n-1}, 2^{n-2}, \ldots, 1$ are chosen so that the "greedy" pivot rule (pick the variable with the largest reduced cost) always makes the **locally attractive but globally suboptimal** choice. Let us trace this in detail on the $n = 2$ case.

**Step-by-step trace for $n = 2$: max $2x_1 + x_2$.**

**At $O = (0, 0)$.** The reduced costs are $\bar{c}_1 = 2$ and $\bar{c}_2 = 1$. The largest-coefficient rule picks $x_1$ (since $2 > 1$). Increasing $x_1$, we hit constraint (1): $x_1 \leq 5$ at $x_1 = 5$. Move to $A = (5, 0)$, objective $= 10$.

**At $A = (5, 0)$.** Constraint (1) is now tight, so we can no longer increase $x_1$. We rewrite in local coordinates. The only improving direction is to increase $x_2$. Increasing $x_2$, we hit constraint (2): $4(5) + x_2 \leq 25$, so $x_2 \leq 5$. Move to $B = (5, 5)$, objective $= 15$.

**At $B = (5, 5)$.** Both (1) and (2) are tight. In local coordinates, the only way to improve is to **decrease** $x_1$ (releasing constraint (1)) while increasing $x_2$ along the edge defined by (2): $4x_1 + x_2 = 25$, so $x_2 = 25 - 4x_1$. The objective along this edge is $2x_1 + (25 - 4x_1) = 25 - 2x_1$, which increases as $x_1$ decreases! We decrease $x_1$ until we hit the next constraint: $x_1 \geq 0$ at $x_1 = 0$, giving $x_2 = 25$. Move to $C = (0, 25)$, objective $= 25$.

**At $C = (0, 25)$.** All reduced costs are $\leq 0$. **Halt** — optimal.

**The trap.** Notice what happened: simplex first rushed to maximize $x_1$ (the variable with the larger coefficient), only to discover that this was a dead end — the real profit comes from $x_2$, which has a smaller coefficient but much more room to grow (up to 25 vs. 5). The constraints are designed so that maximizing $x_1$ first always leads simplex on a detour.

**The general pattern.** In the $n$-dimensional Klee–Minty cube, the same phenomenon cascades. The objective $2^{n-1}x_1 + 2^{n-2}x_2 + \cdots + x_n$ tempts simplex into increasing $x_1$ first (largest coefficient). But the constraint structure forces it to eventually undo this choice and work on $x_2$, then undo again for $x_3$, and so on.

**The Gray code connection.** Each vertex of the Klee–Minty cube can be described by a binary string $b_1 b_2 \cdots b_n$, where $b_i = 0$ means "$x_i$ is at its lower bound" and $b_i = 1$ means "$x_i$ is at its upper bound." Since neighboring vertices differ in exactly one tight constraint, each simplex step flips exactly one bit. The sequence of vertices visited is therefore:

| Step | $b_1 b_2$ | Vertex | Bit flipped |
|---|---|---|---|
| 1 | 00 | $O = (0, 0)$ | — |
| 2 | 10 | $A = (5, 0)$ | $b_1$ |
| 3 | 11 | $B = (5, 5)$ | $b_2$ |
| 4 | 01 | $C = (0, 25)$ | $b_1$ |

This is exactly the **2-bit Gray code**: $00 \to 10 \to 11 \to 01$. For $n = 3$, the same pattern gives the 3-bit Gray code $000 \to 100 \to 110 \to 010 \to 011 \to 111 \to 101 \to 001$, visiting all $2^3 = 8$ vertices. In general, the $n$-bit Gray code visits all $2^n$ binary strings by flipping one bit at a time — and this is precisely the path simplex is forced to take, resulting in $2^n - 1$ steps.

### 8.6 Implications

The Klee–Minty example proves:

**Theorem (Klee–Minty, 1972).** *For the simplex algorithm with the "largest coefficient" pivot rule, there exist LPs with $n$ variables on which simplex performs $2^n - 1$ iterations.*

This means simplex is an **exponential-time** algorithm in the worst case. Similar constructions exist for other standard pivot rules (Bland's rule, steepest-edge, etc.).

---

## Part 9: LP in Polynomial Time

### 9.1 The Paradox

The Klee–Minty result created a paradox: simplex works beautifully in practice (typically $O(m)$ to $O(m \cdot n)$ iterations), but is exponential in the worst case. For decades, people wondered: can LP be solved in polynomial time at all?

### 9.2 The Ellipsoid Method (1979)

The answer came in 1979 when Leonid Khachiyan introduced the **ellipsoid algorithm**. Instead of walking along vertices, it confines the solution to smaller and smaller ellipsoids. This runs in **polynomial time** — $O(n^4 L)$ where $L$ is the input size — but is slow in practice.

### 9.3 Interior Point Methods (1984)

In 1984, Narendra Karmarkar (a Berkeley graduate student!) introduced the **interior point method**. Instead of hopping along the surface of the polyhedron like simplex, it cuts a clever path through the **interior**. It is both polynomial-time in theory and fast in practice.

The resulting competition between simplex and interior point methods led to the development of highly optimized LP solvers. Today, the best solvers often use both methods: interior point to get close to the optimum, then simplex to identify the exact vertex.

### 9.4 Summary of Running Times

| Algorithm | Worst-case iterations | Per iteration | Practical performance |
|---|---|---|---|
| Simplex | $2^n$ (exponential) | $O(mn)$ | Excellent |
| Ellipsoid | $O(n^2 L)$ (polynomial) | $O(n^2)$ | Poor |
| Interior Point | $O(\sqrt{n}L)$ (polynomial) | $O(n^3)$ | Excellent |

---

## Bonus: Smoothed Analysis — Why Simplex Works in Practice

The Klee–Minty result leaves us with a puzzle: simplex is exponential in the worst case, yet blazingly fast on virtually every real-world instance. For decades this gap had no satisfying theoretical explanation. **Worst-case** analysis says simplex is slow; **average-case** analysis (over random LPs) says it is fast — but real-world LPs are not random.

### The Smoothed Complexity Framework

In 2001, Daniel Spielman and Shang-Hua Teng introduced **smoothed analysis**, a framework that interpolates between worst-case and average-case. The idea is simple and elegant:

1. An adversary chooses a worst-case LP instance (constraints $A\mathbf{x} \leq \mathbf{b}$, objective $\mathbf{c}$).
2. Each entry of $A$ is then **perturbed** by a small random Gaussian noise of magnitude $\sigma$.
3. We measure the **expected** running time over the random perturbation.

If an algorithm has low smoothed complexity, it means that exponential behavior is **fragile**: the slightest perturbation destroys it. This captures the intuition that worst-case instances like Klee–Minty are "knife-edge" constructions that never arise in practice, where data always has some small amount of noise or imprecision.

### The Spielman–Teng Theorem

**Theorem (Spielman–Teng, 2004).** *There exists a variant of the simplex method (using the shadow vertex pivot rule) whose smoothed complexity is polynomial in $n$, $m$, and $1/\sigma$.*

More precisely, after Gaussian perturbation of magnitude $\sigma$, the expected number of simplex iterations is at most $\text{poly}(n, m, 1/\sigma)$.

### Why Klee–Minty Breaks Under Perturbation

This result directly explains the Klee–Minty paradox. The exponential path through the squeezed cube depends on the constraints being in **exact** alignment: the coefficient $2^{i-j+1}$ in constraint $i$ must be precisely tuned to create the cascading detour. Even a tiny random perturbation of these coefficients — say, changing $4x_1 + x_2 \leq 25$ to $4.01x_1 + 0.99x_2 \leq 25.02$ — breaks the delicate structure that forces simplex through all $2^n$ vertices. After perturbation, simplex typically finds a short path to the optimum.

### Significance

Smoothed analysis resolved one of the most long-standing mysteries in optimization. It earned Spielman and Teng the **Gödel Prize** in 2008 (one of the highest honors in theoretical computer science) and has since been applied to explain the practical efficiency of many other algorithms with poor worst-case bounds, including the $k$-means algorithm, online learning algorithms, and local search heuristics.

---

## Part 10: Summary

1. **The simplex algorithm** walks along vertices of the feasible polyhedron, improving the objective at each step, and halts when no neighbor offers improvement.
2. **Duality** provides a principled framework for certifying optimality: every LP has a dual, and by strong duality, the primal and dual optima coincide.
3. **Simplex correctness** follows from duality: when simplex halts, the termination condition (all reduced costs $\leq 0$) implicitly produces a dual certificate.
4. **Per-iteration cost** of simplex is $O(mn)$, which is efficient.

5. **Klee–Minty example** shows simplex can take $2^n$ iterations in the worst case — an exponential blowup caused by a skewed cube that forces the greedy pivot rule through every vertex.
6. **Polynomial-time algorithms** for LP exist (ellipsoid, interior point), resolving the theoretical question. But simplex remains dominant in practice.

This tension — an exponential worst case paired with superb practical performance — is one of the great mysteries of optimization, and an active area of research to this day.