



Algorithm Design XIII

Linear Programming II

Guoqiang Li
School of Software



SHANGHAI JIAO TONG
UNIVERSITY

Review of Previous Lecture



$$\max x_1 + 6x_2 + 13x_3$$

$$x_1 \leq 200$$

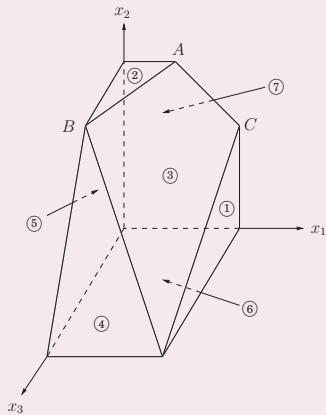
$$x_2 \leq 300$$

$$x_1 + x_2 + x_3 \leq 400$$

$$x_2 + 3x_3 \leq 600$$

$$x_1, x_2, x_3 \geq 0$$

The Example



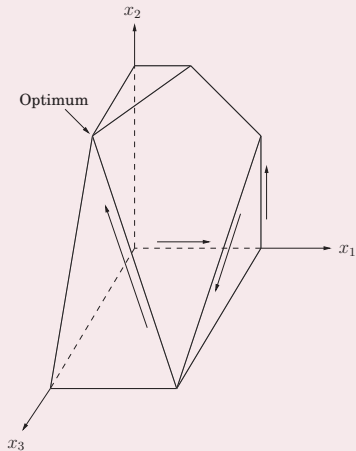
The point of final contact is the **optimal vertex**: $(0, 300, 100)$, with total **profit** \$3100.

Q: How would the **simplex** algorithm behave on this modified problem?

A possible **trajectory**

$$\frac{(0, 0, 0)}{\$0} \rightarrow \frac{(200, 0, 0)}{\$200} \rightarrow \frac{(200, 200, 0)}{\$1400} \rightarrow \frac{(200, 0, 200)}{\$2800} \rightarrow \frac{(0, 300, 100)}{\$3100}$$

The Example





Primal LP

$$\begin{aligned} \max \quad & c^T \mathbf{x} \\ & A\mathbf{x} \leq b \\ & \mathbf{x} \geq 0 \end{aligned}$$

Dual LP

$$\begin{aligned} \min \quad & \mathbf{y}^T b \\ & \mathbf{y}^T A \geq c^T \\ & \mathbf{y} \geq 0 \end{aligned}$$

Primal LP:

$$\begin{aligned} \max \quad & c_1 x_1 + \cdots + c_n x_n \\ & a_{i1} x_1 + \cdots + a_{in} x_n \leq b_i \quad \text{for } i \in I \\ & a_{i1} x_1 + \cdots + a_{in} x_n = b_i \quad \text{for } i \in E \\ & x_j \geq 0 \quad \text{for } j \in N \end{aligned}$$

Dual LP:

$$\begin{aligned} \min \quad & b_1 y_1 + \cdots + b_m y_m \\ & a_{1j} y_1 + \cdots + a_{mj} y_m \geq c_j \quad \text{for } j \in N \\ & a_{1j} y_1 + \cdots + a_{mj} y_m = c_j \quad \text{for } j \notin N \\ & y_i \geq 0 \quad \text{for } i \in I \end{aligned}$$

LP and Its Dual



SHANGHAI JIAO TONG
UNIVERSITY

$$\begin{aligned}\max \quad & x_1 + 6x_2 \\ & x_1 \leq 200 \\ & x_2 \leq 300 \\ & x_1 + x_2 \leq 400 \\ & x_1, x_2 \geq 0\end{aligned}$$

$$\begin{aligned}\min \quad & 200y_1 + 300y_2 + 400y_3 \\ & y_1 + y_3 \geq 1 \\ & y_2 + y_3 \geq 6 \\ & y_1, y_2, y_3 \geq 0\end{aligned}$$



Theorem

Assume LP problem (P) has a solution x^* and its dual problem (D) has a solution y^* .

- 1 If $x_j^* > 0$, then the j -th constraint in (D) is binding.
- 2 If the j -th constraint in (D) is not binding, then $x_j^* = 0$.
- 3 If $y_i^* > 0$, then the i -th constraint in (P) is binding.
- 4 If the i -th constraint in (P) is not binding, then $y_i^* = 0$.

Shortest Path



Shortest path problem gives a weighted, directed graph $G = (V, E)$, with weight function $w : E \rightarrow \mathbb{Q}^+$ mapping edges to real-valued weights, a source vertex s , and destination vertex t . We wish to compute the weight of a shortest path from s to t .

Shortest Path in LP



$$\max d_t$$

$$d_v \leq d_u + w(u, v) \quad (u, v) \in E$$

$$d_s = 0$$

$$d_i \geq 0 \quad i \in V$$

Q: Another formalization?

Shortest Path in LP



Let $\mathcal{S} = \{S \subseteq V : s \in S, t \notin S\}$; that is, \mathcal{S} is the set of all s - t cuts in the graph. Then we can model the shortest s - t path problem with the following integer program,

$$\min \sum_{e \in E} w_e x_e$$

$$\sum_{e \in \delta(S)} x_e \geq 1 \quad S \in \mathcal{S}$$

$$x_e \in \{0, 1\} \quad e \in E$$

where $\delta(S)$ is the set of all edges that have one endpoint in S and the other endpoint not in S .

- Can we relax the restriction $x_e \in \{0, 1\}$ to $0 \leq x_e \leq 1$?
- How about $x_e \geq 0$?

Shortest Path in LP



$$\min \sum_{e \in E} w_e x_e$$

$$\sum_{e \in \delta(S)} x_e \geq 1 \quad S \in \mathcal{S}$$

$$x_e \geq 0 \quad e \in E$$

$$\max \sum_{S \in \mathcal{S}} y_S$$

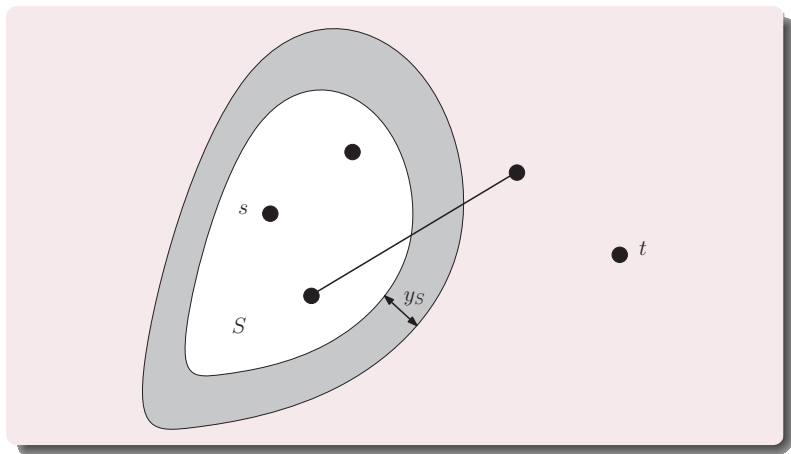
$$\sum_{S \in \mathcal{S}, e \in \delta(S)} y_S \leq w_e \quad e \in E$$

$$y_S \geq 0 \quad S \in \mathcal{S}$$

The Moat



SHANGHAI JIAO TONG
UNIVERSITY



Standard Linear Programming

Variants of Linear Programming



A general **linear program** has many degrees of freedom:

- 1 It can be either a **maximization** or a **minimization** problem.
- 2 Its constraints can be **equations** and/or **inequalities**.
- 3 The variables are often restricted to be **nonnegative**, but they can also be unrestricted in sign.

We will now show that these various LP options can all be **reduced** to one another via simple transformations.

Variants of Linear Programming



SHANGHAI JIAO TONG
UNIVERSITY

To turn a maximization problem into a minimization (or vice versa), multiply the coefficients of the objective function by -1 .

Variants of Linear Programming



To turn an **inequality constraint** like $\sum_{i=1}^n a_i x_i \leq b$ into an **equation**, introduce a new variable s and use

$$\sum_{i=1}^n a_i x_i + s = b$$
$$s \geq 0$$

This s is called the **slack variable** for the inequality.

To change an **equality constraint** into **inequalities** is easy: rewrite $ax = b$ as the equivalent pair of constraints $ax \leq b$ and $ax \geq b$.

Variants of Linear Programming



Finally, to deal with a variable x that is **unrestricted in sign**, do the following:

- Introduce two **nonnegative variables**, $x^+, x^- \geq 0$.
- Replace x , wherever it occurs in the **constraints** or the **objective function**, by $x^+ - x^-$.



We can reduce any LP into an LP of a much more constrained kind that we call the **standard form**:

- the variables are all **nonnegative**.
- the constraints are all **equations**.
- and the objective function is to be **minimized**.

$$\begin{array}{ll} \max x_1 + 6x_2 & \min -x_1 - 6x_2 \\ x_1 \leq 200 & x_1 + s_1 = 200 \\ x_2 \leq 300 & x_2 + s_2 = 300 \\ x_1 + x_2 \leq 400 & x_1 + x_2 + s_3 = 400 \\ x_1, x_2 \geq 0 & x_1, x_2, s_1, s_2, s_3 \geq 0 \end{array} \quad \Rightarrow$$

The Simplex Algorithm

General Description



SHANGHAI JIAO TONG
UNIVERSITY

Simplex

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



Definition (Vertex)

Each **vertex** is the unique point at which some subset of **hyperplanes** meet.

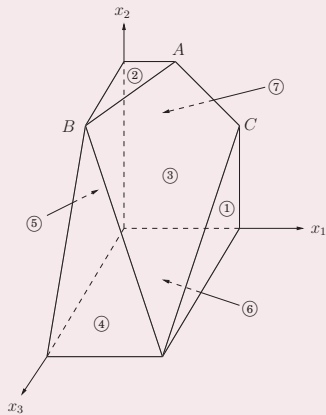
Pick a subset of the **inequalities**. If there is a **unique point** that satisfies them with equality, and this point happens to be **feasible**, then it is a vertex.

Each vertex is specified by a set of n **inequalities** (say there are n **variables**).

Definition (Neighbors)

Two vertices are **neighbors** if they have $n - 1$ defining **inequalities** in common.

The Example





Algorithm

On each iteration, **simplex** has two tasks:

- 1 **Check** whether the current vertex is **optimal** (and if so, **halt**).
- 2 **Determine** where to move **next**.

Both tasks are easy if the vertex is at the **origin**. If the vertex is elsewhere, we transform the **coordinate system** to move it to the **origin**.

The Convenience for the Origin



Suppose we have some generic LP:

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

where \mathbf{x} is the vector of variables, $\mathbf{x} = (x_1, \dots, x_n)$.

Suppose the origin is feasible. Then it is certainly a vertex, since it is the unique point at which the n inequalities

$$\{x_1 \geq 0, \dots, x_n \geq 0\}$$

are tight.

Task 1 in the Origin



Lemma

The origin is *optimal* if and only if all $c_i \leq 0$.

Proof.

If all $c_i \leq 0$, then considering the constraints $x \geq 0$, we can't hope for a better objective value.

Conversely, if some $c_i > 0$, then the origin is not optimal, since we can increase the objective function by *raising* x_i .

Task 2 in the Origin



We can move by increasing some x_i for which $c_i > 0$.

Q: How much can we increase it?

A: Until we hit some other constraint.

We release the tight constraint $x_i \geq 0$ and increase x_i until some other inequality, previously loose, now becomes tight.

We have exactly n tight inequalities, so we are at a new vertex.

An Example



$$\max 2x_1 + 5x_2$$

$$2x_1 - x_2 \leq 4$$

$$x_1 + 2x_2 \leq 9$$

$$-x_1 + x_2 \leq 3$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$



The trick is to transform u into the origin, by shifting the coordinate system from the usual (x_1, \dots, x_n) to the “local view” from u .

These local coordinates consist of distances y_1, \dots, y_n to the n hyperplanes (inequalities) that define and enclose u .

If one of these enclosing inequalities is $a_i \cdot x \leq b_i$, then the distance from a point x to that particular “wall” is

$$y_i = b_i - a_i \cdot x$$

The n equations of this type, one per wall, define the y_i ’s as linear functions of the x_i ’s, and this relationship can be inverted to express the x_i ’s as a linear function of the y_i ’s.

An Example



$$\max 2x_1 + 5x_2$$

$$2x_1 - x_2 \leq 4$$

$$x_1 + 2x_2 \leq 9$$

$$-x_1 + x_2 \leq 3$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

$$\max 15 + 7y_1 - 5y_2$$

$$y_1 + y_2 \leq 7$$

$$3y_1 - 2y_2 \leq 3$$

$$y_2 \geq 0$$

$$y_1 \geq 0$$

$$-y_1 + y_2 \leq 3$$

An Example



$$\max 15 + 7y_1 - 5y_2$$

$$y_1 + y_2 \leq 7$$

$$3y_1 - 2y_2 \leq 3$$

$$y_2 \geq 0$$

$$y_1 \geq 0$$

$$-y_1 + y_2 \leq 3$$

$$\max 22 - 7/3z_1 - 1/3z_2$$

$$-1/3z_1 + 5/3z_2 \leq 6$$

$$z_1 \geq 0$$

$$z_2 \geq 0$$

$$1/3z_1 - 2/3z_2 \leq 1$$

$$1/3z_1 + 1/3z_2 \leq 4$$

Rewriting the LP



We can rewrite the **entire LP** in terms of the **y**'s.

This doesn't fundamentally change, but expresses it in a different **coordinate frame**.

The revised "local" LP has the following three properties:

- 1 It includes the **inequalities** $\mathbf{y} \geq 0$, which are simply the transformed versions of the inequalities defining **u**.
- 2 **u** itself is the **origin** in **y**-space.
- 3 The **cost function** becomes $\max c_u + \tilde{\mathbf{c}}^T \mathbf{y}$, where c_u is the value of the **objective function** at **u** and $\tilde{\mathbf{c}}$ is a **transformed cost vector**.

Loose End

The Starting Vertex



In a general LP, the **origin** might not be **feasible** and thus not a **vertex**.

It turns out that finding a **starting vertex** can be reduced to an **LP** and solved by simplex!

Start with any linear program in **standard form**:

$$\min c^T \mathbf{x} \text{ such that } \mathbf{Ax} = \mathbf{b} \text{ and } x \geq 0.$$

We make sure that the right-hand sides of the equations are all **nonnegative**: if $b_i < 0$, multiply both sides of the i -th equation by -1 .

The Starting Vertex



Then we create a new LP as follows:

- Create m new artificial variables $z_1, \dots, z_m \geq 0$, where m is the number of equations.
- Add z_i to the left-hand side of the i -th equation.
- Let the objective, to be minimized, be $z_1 + z_2 + \dots + z_m$.

An Example



$$\begin{aligned}\min & -x_1 - 6x_2 \\ & x_1 + s_1 = 200 \\ & x_2 + s_2 = 300 \\ & x_1 + x_2 + x_3 = 400 \\ & x_1, x_2, x_3 \geq 0\end{aligned}$$

$$\begin{aligned}\min & z_1 + z_2 + z_3 \\ & x_1 + s_1 + z_1 = 200 \\ & x_2 + s_2 + z_2 = 300 \\ & x_1 + x_2 + x_3 + z_3 = 400 \\ & x_1, x_2, x_3 \geq 0 \\ & z_1, z_2, z_3 \geq 0\end{aligned}$$

The Starting Vertex



For this new LP, it's easy to come up with a **starting vertex**, namely, the one with $z_i = b_i$ for all i and all other variables **zero**.

Therefore we can solve it by simplex, to obtain the **optimum solution**.

There are two cases:

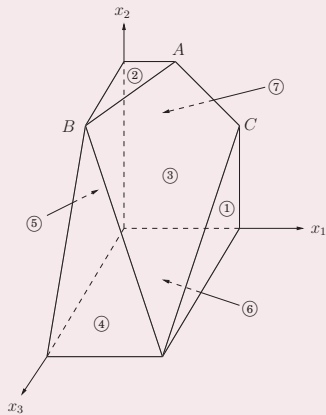
- 1 If the **optimum value** of $z_1 + \dots + z_m$ is **zero**, then all z_i 's obtained by simplex are **zero**, and hence from the optimum vertex of the new LP we get a **starting feasible vertex** of the original LP.
- 2 If the **optimum objective** turns out to be **positive**: We tried to **minimize** the sum of the z_i 's, but it cannot be **zero**. This means that the original linear program is **infeasible**.



A vertex is **degenerate** if it is the **intersection** of more than n faces of the **polyhedron**, say $n + 1$.

It means that if we choose any one of n sets of $n + 1$ inequalities and solve the corresponding system of these linear equations in n unknowns, we'll get the **same solution** in all $n + 1$ cases.

An Example





This is a **serious problem**: simplex may return a **suboptimal** degenerate vertex simply because all its neighbors are **identical** to it and thus have no better **objective**.

If we **modify simplex** so that it detects degeneracy and continues to hop from vertex to vertex despite lack of any improvement in the cost, it may **end up looping** forever.



One way to fix this is by a **perturbation**:

change each b_i by a tiny random amount to $b_i \pm \varepsilon_i$.

This doesn't change the essence of the LP, but it has the effect of differentiating between the solutions of the linear systems.



In some cases an LP is **unbounded**, in that its objective function can be made **arbitrarily large** (or **small**, if it's a minimization problem).

If this is the case, **simplex** will discover it:

- In exploring the **neighborhood**, it will notice that taking out an inequality and adding another leads to an underdetermined system of equations.
- The space of solutions contains a whole line across which the **objective** can become larger and larger, all the way to ∞ .

In this case simplex **halts** and **complains**.

An Example



$$\begin{aligned} \max & x_1 + x_2 \\ & x_1 - x_2 \geq 0 \\ & x_1, x_2 \geq 0 \end{aligned}$$

The Running Time

The Running Time of Simplex



Q: What is the running time of simplex, for a **generic linear program**:

$$\max c^T \mathbf{x} \text{ such that } \mathbf{A}\mathbf{x} \leq \mathbf{b} \text{ and } \mathbf{x} \geq \mathbf{0}$$

where there are n **variables** and \mathbf{A} contains m **inequality constraints**?

It is an **iterative algorithm** that proceeds from **vertex** to **vertex**. Let u be the current vertex.

Each of its **neighbors** shares $n - 1$ of these **inequalities**, so u can have at most $n \cdot m$ neighbors.

The Running Time of Simplex



A naive way for an iteration:

- 1 check each potential neighbor to see whether it really is a vertex of the polyhedron,
- 2 determine its cost.

Finding the cost is quick, just a dot product.

Checking whether it is a true vertex involves: solve a system of n equations and check whether the result is feasible.

By Gaussian elimination this takes $O(n^3)$ time, giving total $O(mn^4)$ per iteration.

The Running Time of Simplex



A much better way: the mn^4 can be improved to mn .

Recall the local view from vertex u . The per-iteration overhead of rewriting the LP in terms of the current local coordinates is just $O((m+n)n)$.

The local view changes only slightly between iterations, in just one of its defining inequalities.

The Running Time of Simplex



To select the **best** neighbor, we recall that the objective function is of the form

$$\max c_u + \tilde{c} \cdot \mathbf{y}$$

where c_u is the value of the **objective function** at u .

This immediately identifies a **promising direction** to move: we pick any $\tilde{c}_i > 0$.

Since the rest of the LP has now been rewritten in terms of the \mathbf{y} -coordinates, it is easy to determine how much y_i can be **increased** before some other inequality is **violated**.

The Running Time of Simplex



Q: How many iterations could there be?

A: At most $\binom{m+n}{n}$, i.e., the number of **vertices**.

It is **exponential** in n .

And in fact, there are examples of LPs for which simplex does indeed take an **exponential number** of iterations.

Simplex is an **exponential-time algorithm**.

However, such exponential examples do not occur in **practice**, and it is this fact that makes simplex so valuable and so widely used.

A Notable Result



Smoothed analysis proposed by Daniel Spielman and Shanghua Teng is a way of measuring the complexity of an algorithm. It gives a more realistic analysis of the practical performance of the algorithm. It was used to explain that the simplex algorithm runs in exponential-time in the worst-case and yet in practice it is a very efficient algorithm, which was one of the main motivations for developing smoothed analysis. The authors received the 2008 Gödel Prize and the 2009 Fulkerson Prize.