

# CS711008Z Algorithm Design and Analysis

## Lecture 8. Algorithm design technique: Linear programming

Dongbo Bu

Institute of Computing Technology  
Chinese Academy of Sciences, Beijing, China

- Some practical problems: DIET, MAXIMUM FLOW, MINIMUM COST FLOW, MULTICOMMODITYFLOW, and SAT problems
- Linear programming forms: general form, standard form, and slack form
- Intuitions of linear program
- Algorithms: SIMPLEX algorithm, INTERIOR POINT algorithm
- Smoothed complexity: why simplex algorithm usually takes polynomial time?

## Practical problem 1: DIET problem



- In 1945, G. Stigler described the diet problem in the paper *The cost of subsistence*.
- Here we use a simplified version.

A housewife wonders how much money she must spend on foods in order to get all the energy (2000 kcal), protein (55 g), and calcium (800 mg) that she needs every day.

Food	Energy	Protein	Calcium	Price
Oatmeal	110	4	2	3
Whole milk	160	8	285	9
Cherry pie	420	4	22	20
Pork with beans	260	14	80	19

Two solutions:

- 10 servings of pork with beans: 190 Cents
- 8 servings of milk + 2 servings of pie: 112 Cents.

# Linear programming formulation

A housewife wonders how much money she must spend on foods in order to get all the energy (2000 kcal), protein (55 g), and calcium (800 mg) that she needs every day.

Food	Energy	Protein	Calcium	Price	Quantity
Oatmeal	110	4	2	3	$x_1$
Whole milk	160	8	285	9	$x_2$
Cherry pie	420	4	22	20	$x_3$
Pork beans	260	14	80	19	$x_4$

# Linear programming formulation

A housewife wonders how much money she must spend on foods in order to get all the energy (2000 kcal), protein (55 g), and calcium (800 mg) that she needs every day.

Food	Energy	Protein	Calcium	Price	Quantity
Oatmeal	110	4	2	3	$x_1$
Whole milk	160	8	285	9	$x_2$
Cherry pie	420	4	22	20	$x_3$
Pork beans	260	14	80	19	$x_4$

Formalization:

$$\begin{array}{llllllll} \min & 3x_1 & + & 9x_2 & + & 20x_3 & + & 19x_4 & \text{money} \\ s.t. & 110x_1 & + & 160x_2 & + & 420x_3 & + & 260x_4 & \geq 2000 & \text{energy} \\ & 4x_1 & + & 8x_2 & + & 4x_3 & + & 14x_4 & \geq 55 & \text{protein} \\ & 2x_1 & + & 285x_2 & + & 22x_3 & + & 80x_4 & \geq 800 & \text{calcium} \\ & x_1 & , & x_2 & , & x_3 & , & x_4 & \geq 0 \end{array}$$

## Practical problem 2: MAXIMUM FLOW



# MAXIMUM FLOW problem

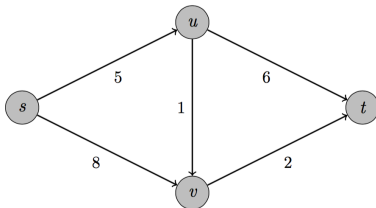
## INPUT:

A directed graph  $G = \langle V, E \rangle$ . Each edge  $e = (u, v)$  is associated with a capacity  $C(u, v)$ . Two special points: *source*  $s$  and *sink*  $t$ ;

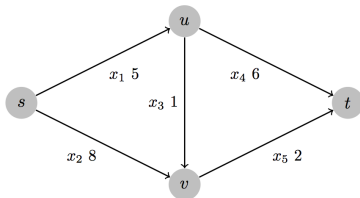
## OUTPUT:

For each edge  $e = (u, v)$ , to assign a flow  $0 \leq f(u, v) \leq C(u, v)$  such that  $\sum_{u, (s, u) \in E} f(u, v)$  is maximized.

FLOW CONSERVATION restrictions: at each node (except for  $s$  and  $t$ ), the sum of input equals the sum of output.



# Linear programming formulation



LP Formulation:

$$\begin{array}{llllll} \max & x_1 & + & x_2 & & \text{output from } s \\ \text{s.t.} & x_1 & & & - & x_3 & - & x_4 & = & 0 & \text{node } u \\ & & x_2 & + & x_3 & & & - & x_5 & = & 0 & \text{node } v \\ & & & & & 5 & \geq & x_1 & \geq & 0 & \text{edge } (s, u) \\ & & & & & \dots & & \dots & & & \end{array}$$

### Practical problem 3: MINIMUM COST FLOW problem

# MINIMUM COST FLOW problem

## INPUT:

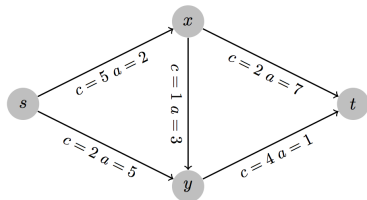
A directed graph  $G = \langle V, E \rangle$ . Each edge  $e = (u, v)$  is associated with a capacity  $C(u, v)$ , and a cost  $a(u, v)$ . If we send  $f(u, v)$  units of flow via edge  $(u, v)$ , we incur a cost of  $a(u, v)f(u, v)$ . We are also given a flow target  $d$ . Two special points: *source*  $s$  and *sink*  $t$ ;

## OUTPUT:

For each edge  $e = (u, v)$ , to assign a flow  $0 \leq f(u, v) \leq C(u, v)$  such that:

- 1 We wish to send  $d$  units of flow from  $s$  to  $t$ ;
- 2 The total cost  $\sum_{(u,v) \in E} a(u, v)f(u, v)$  is minimized.

# Linear programming formulation



LP Formulation:

$$\begin{array}{ll} \min & \sum_{(u,v) \in E} a(u,v) f(u,v) \\ \text{s.t.} & f(u,v) \leq C(u,v) \quad \text{for each } (u,v) \\ & f(u,v) \geq 0 \quad \text{for each } (u,v) \\ & \sum_{u, (u,v) \in E} f(u,v) = \sum_{w, (v,w) \in E} f(v,w) \quad \text{for each } v \in V \\ & \sum_{v, (s,v) \in E} f(s,v) = d \end{array}$$

## Practical problem 4: MULTICOMMODITYFLOW problem

# MULTICOMMODITYFLOW problem

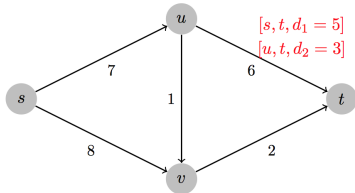
多物品流问题

## INPUT:

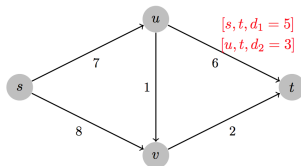
A directed graph  $G = \langle V, E \rangle$ . Each edge  $e$  has a capacity  $C_e$ . A total of  $k$  commodities, and for commodity  $i$ ,  $s_i$ ,  $t_i$ , and  $d_i$  denote the source, sink, and demand, respectively.

## OUTPUT:

A feasible flow for commodity  $i$  (denoted as  $f_i$ ) satisfying the FLOW-CONSERVATION, and CAPACITY CONSTRAINTS, i.e. the aggregate flow on edge  $e$  cannot exceed its capacity  $C_e$ .



# Linear programming formulation



LP Formulation:

$$\begin{aligned} \max & \quad 0 \\ \text{s.t.} \quad & \sum_{i=1}^k f_i(u, v) \leq c(u, v) \quad \text{for each } (u, v) \\ & f_i(u, v) \geq 0 \quad \text{for each } i, (u, v) \\ & \sum_{u, (u, v) \in E} f_i(u, v) = \sum_{w, (v, w) \in E} f_i(v, w) \quad \text{for each } i, v \in V - s, t \\ & \sum_{v, (s_i, v) \in E} f_i(s_i, v) = d_i \quad \text{for each } i \end{aligned}$$

Notes:

- 1 The unusual objective function “**max 0**” is used to express the idea that it suffices to calculate a feasible solution.
- 2 **Linear programming is the only known polynomial-time algorithm for this problem.**



## Practical problem 5: SAT problem

**INPUT:**

A set of  $m$  conjunction normal formula (CNF) clauses over  $n$  Boolean variables  $x_1, x_2, \dots, x_n$

**OUTPUT:**

Whether all clauses can be satisfied by an TRUE/FALSE assignment of the  $n$  variables.

- A SAT instance:

$$\begin{aligned}\Phi = & (x_1 \vee \neg x_2 \vee x_3) \wedge \\ & (\neg x_1 \vee x_2 \vee \neg x_3) \wedge \\ & (x_1 \vee x_2 \vee \neg x_3)\end{aligned}$$

- An assignment to make all clauses TRUE:

$$x_1 = \text{TRUE}, x_2 = \text{TRUE}, x_3 = \text{TRUE}$$

# Linear programming formulation

A SAT instance:

$$\Phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee \neg x_3)$$

LP Formulation:

$$\begin{array}{llll} \max & c_1 + & c_2 + & c_3 \\ s.t. & x_1 + (1 - x_2) + & x_3 & \geq c_1 \\ & (1 - x_1) + & x_2 + (1 - x_3) & \geq c_2 \\ & x_1 + & x_2 + (1 - x_3) & \geq c_3 \\ & x_1, & x_2, & x_3 = 0/1 \\ & c_1, & c_2, & c_3 = 0/1 \end{array}$$

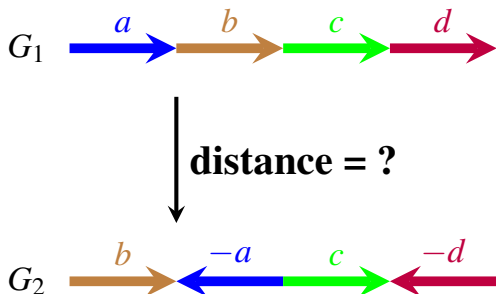
Intuitive idea:

- Constraints: The left-hand side of a constraint represents the number of satisfied literals; thus, a constraint allows  $c_i$  to be 1 if there are at least one satisfied literals.
- Objective function: The objective function denotes the number of satisfied clauses. Thus,  $\Phi$  is satisfiable iff  $c_1 + c_2 + c_3 = 3$ .

## Genome rearrangement distance problem [M. Shao, 2014]

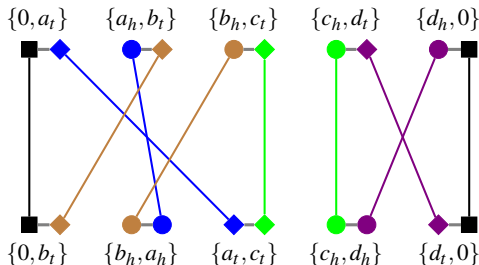
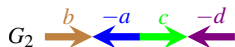
# Practical problem: genome rearrangement distance

- The minimum number of operations to transform  $G_1$  into  $G_2$

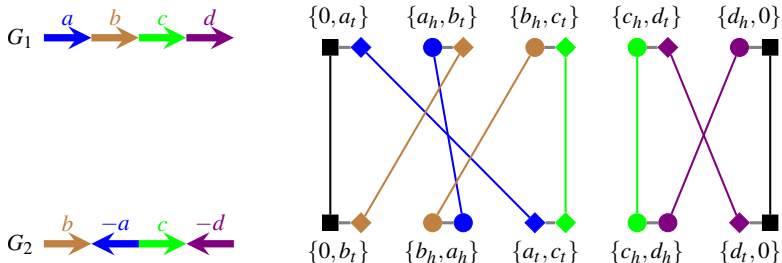


Operations: reverse a fragment of the genome;

# Adjacency graph: a more succinct formulation

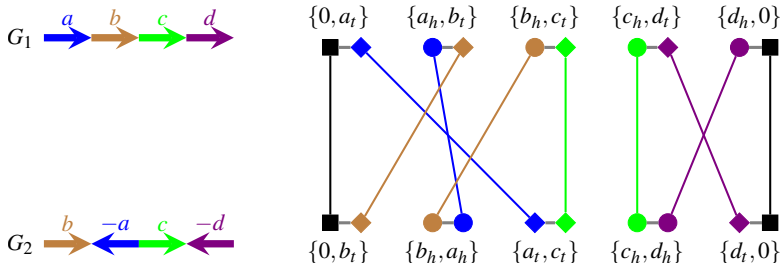


# Adjacency graph: a more succinct formulation



- DCJ distance = ( $\#$ adjacencies) - ( $\#$ cycles).

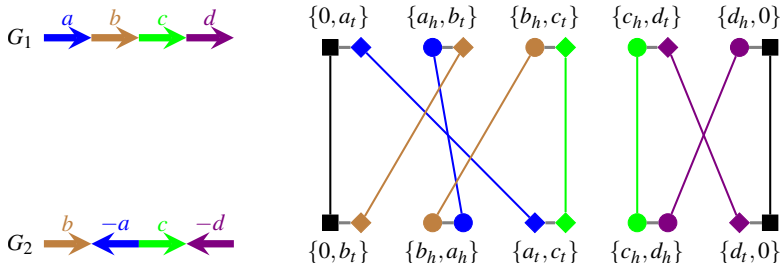
# Adjacency graph: a more succinct formulation



- DCJ distance = ( $\#$ adjacencies) - ( $\#$ cycles).
- DCJ distance = 3 in this example.



# Adjacency graph: a more succinct formulation

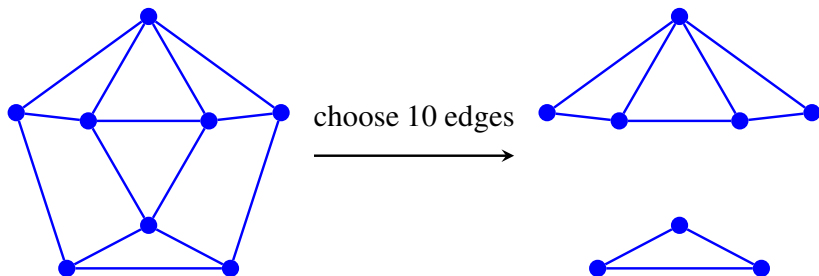


- DCJ distance =  $(\# \text{adjacencies}) - (\# \text{cycles})$ .
- DCJ distance = 3 in this example.
- To minimize DCJ distance, we need to compute a decomposition of the corresponding adjacency graph with maximized number of cycles.

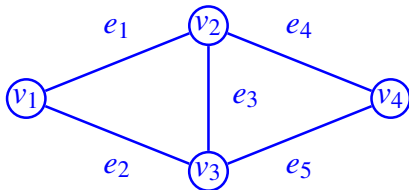
# Problem Statement

**Problem:** given an undirected graph  $G = (V, E)$ , to choose  $k$  edges and remove others, such that the number of connected components in the remaining graph is maximized.

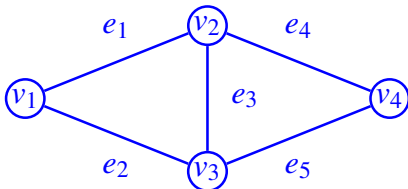
**(Formulate this problem as an ILP.)**



Consider the following example with  $k = 3$ .

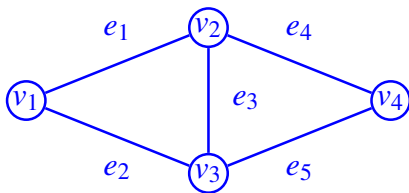


Consider the following example with  $k = 3$ .



- For each edge  $e_i$ , we use a binary variable  $x_i$  to indicate whether  $e_i$  is chosen. We use the following constraint to guarantee exactly  $k$  edges are chosen:

$$x_1 + x_2 + x_3 + x_4 + x_5 = 3$$



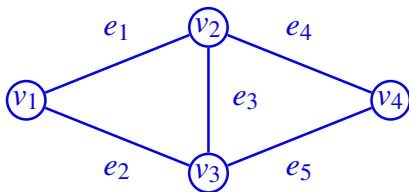
- To count the number of connected components, for vertex  $v_j$ ,  $1 \leq j \leq |V|$ , we use a variable  $y_j$  to indicate the **label** of  $v_j$ , and set **distinct** upper bounds for all the labels:

$$1 \leq y_1 \leq 1$$

$$1 \leq y_2 \leq 2$$

$$1 \leq y_3 \leq 3$$

$$1 \leq y_4 \leq 4$$



- We guarantee that if an edge is chosen, then its two adjacent vertices have the same label:

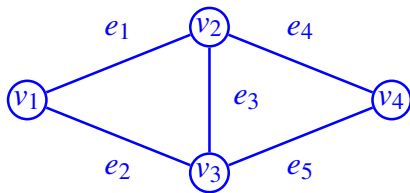
$$y_1 \leq y_2 + 1 \cdot (1 - x_1); \quad y_2 \leq y_1 + 2 \cdot (1 - x_1) \quad (\text{for } e_1)$$

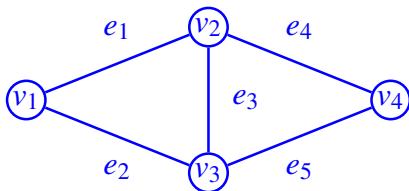
$$y_1 \leq y_3 + 1 \cdot (1 - x_2); \quad y_3 \leq y_1 + 3 \cdot (1 - x_2) \quad (\text{for } e_2)$$

$$y_2 \leq y_3 + 2 \cdot (1 - x_3); \quad y_3 \leq y_2 + 3 \cdot (1 - x_3) \quad (\text{for } e_3)$$

$$y_2 \leq y_4 + 2 \cdot (1 - x_4); \quad y_4 \leq y_2 + 4 \cdot (1 - x_4) \quad (\text{for } e_4)$$

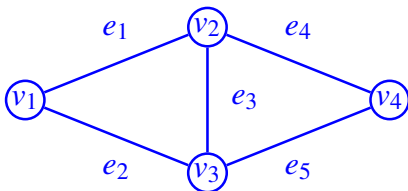
$$y_3 \leq y_4 + 3 \cdot (1 - x_5); \quad y_4 \leq y_3 + 4 \cdot (1 - x_5) \quad (\text{for } e_5)$$



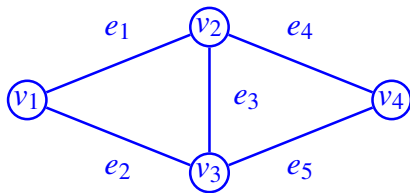


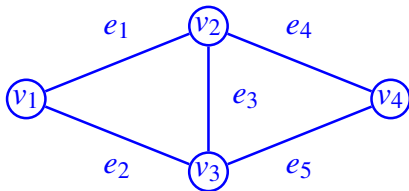
- The equality can propagate along the chosen edges. Thus, in the remaining graph all vertices in the same connected component have the same label.





- The equality can propagate along the chosen edges. Thus, in the remaining graph all vertices in the same connected component have the same label.
- Since all vertices have distinct upper bounds, in each connected component, at most one vertex can reach its upper bound. Thus, we can use the number of vertices whose upper bound is reached, to count the number of connected components.





- We use a binary variable  $z_j$  to indicate whether the label of  $v_j$  reaches its upper bound:

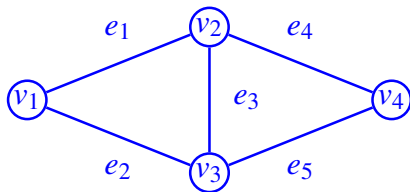
$$1 \cdot z_1 \leq y_1$$

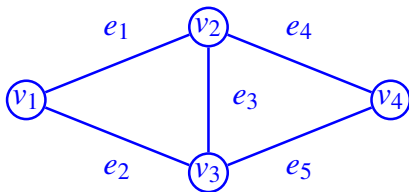
$$2 \cdot z_2 \leq y_2$$

$$3 \cdot z_3 \leq y_3$$

$$4 \cdot z_4 \leq y_4$$

We can verify that,  $z_j = 1$  only if  $y_j = j$ , i.e., the label of  $v_j$  reaches its upper bound.





- The objective function of the ILP formulation can be set to maximize the number of vertices whose upper bound can be reached:

$$\max z_1 + z_2 + z_3 + z_4$$

## A brief history of linear programming

# Concept, algorithms and analysis

- In 1939, L. Kantorovich proposed the concept of linear programming (called *extremal problem*) as mathematical formulation of practical problems in planned economy. He also proposed the *resolving multiplier* approach.
- In 1941, Hitchcock proposed the ASSIGNMENT problem.
- In 1949, G. B. Dantzig advanced this concept and proposed the *simplex* algorithm.
- In 1971, Klee and Minty gave a counter-example to show that simplex is not a polynomial-time algorithm.
- In 1975, L. V. Kantorovich, Nobel prize, application of linear programming in resource distribution;
- In 1979, L. G. Khanchian proposed a polynomial-time ellipsoid method;
- In 1984, N. Karmarkar proposed another polynomial-time interior-point method;
- In 2001, D. Spielman and S. Teng proposed smoothed complexity to prove the efficiency of simplex algorithm.



Figure: Leonid Kantorovich

L. Kantorovich was known for his theory and development of techniques for the optimal allocation of resources. He is regarded as the founder of linear programming. He was the winner of the Stalin Prize in 1949 and the Nobel Memorial Prize in Economics in 1975.



# George B. Dantzig proposed LP model in 1947



- In 1946, as mathematical adviser to the U.S. Air Force Comptroller, he was challenged by his Pentagon colleagues to see what he could do to mechanize the planning process, "to more rapidly compute a time-staged deployment, training and logistical supply program."
- In those pre-electronic computer days, mechanization meant using analog devices or punched-card machines. "Program" was a military term referring not to the instruction used by a computer to solve problems (called "codes"), but rather to plans or proposed schedules for training, logistical supply, or deployment of combat units.



Figure: Leonid G. Khanchian



Karmarkar at Bell Labs: an equation to find a new way through the maze

## Folding the Perfect Corner

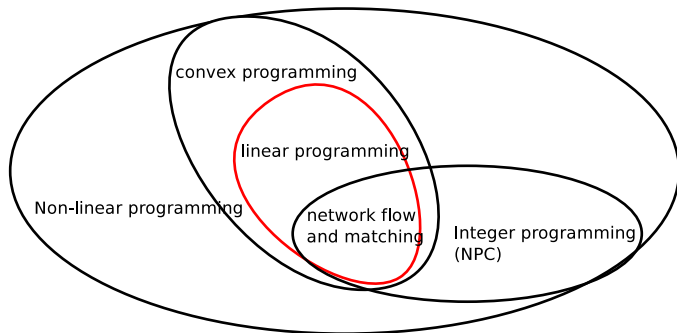
*A young Bell scientist makes a major math breakthrough*

**E**very day 1,200 American Airlines jets crisscross the U.S., Mexico, Canada and the Caribbean, stopping in 110 cities and bearing over 80,000 passengers. More than 4,000 pilots, copilots, flight personnel, maintenance workers and baggage carriers are shuffled among the flights; a total of 3.6 million gal. of high-octane fuel is burned. Nuts, bolts, altimeters, landing gears and the like must be checked at each destination. And while performing these scheduling gymnastics, the company must keep a close eye on costs, projected revenue and profits.

Like American Airlines, thousands of companies must routinely untangle the myriad variables that complicate the efficient distribu-

Indian-born mathematician at Bell Laboratories in Murray Hill, N.J., after only a years' work has cracked the puzzle of linear programming by devising a new algorithm, a step-by-step mathematical formula. He has translated the procedure into a program that should allow computers to track a greater combination of tasks than ever before and in a fraction of the time.

Unlike most advances in theoretical mathematics, Karmarkar's work will have an immediate and major impact on the real world. "Breakthrough is one of the most abused words in science," says Ronald Graham, director of mathematical sciences at Bell Labs. "But this is one situation where it is truly an-



## Notes:

- ① In convex programming, local optimum is also global optimum.
- ② NETWORK FLOW and MATCHING are special ILP problems: the special problem structure determines that an LP model can automatically generate integral solutions.

# GLPK: an efficient LP solver

- The GLPK (GNU Linear Programming Kit, <http://www.gnu.org/software/glpk/>) package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library.
- GLPK supports the GNU MathProg modeling language, which is a subset of the AMPL language.
- The GLPK package includes the following main components:
  - 1 primal and dual simplex methods
  - 2 primal-dual interior-point method
  - 3 branch-and-cut method
  - 4 translator for GNU MathProg
  - 5 application program interface (API)
  - 6 stand-alone LP/MIP solver

(See extra slides)

- The Gurobi Optimizer (<http://gurobi.com>) is a state-of-the-art solver for mathematical programming. It includes the following solvers: linear programming solver (LP solver), quadratic programming solver (QP solver), quadratically constrained programming solver (QCP solver), mixed-integer linear programming solver (MILP solver), mixed-integer quadratic programming solver (MIQP solver), and mixed-integer quadratically constrained programming solver (MIQCP solver)
- The solvers in the Gurobi Optimizer were designed from the ground up to exploit modern architectures and multi-core processors, using the most advanced implementations of the latest algorithms.

Various linear program forms: general form, standard form, and slack form.

# Form 1. General form of linear programming

- General form: mixture of linear inequalities and equalities

$$\begin{array}{llllllllll} \min & c_1x_1 & + & c_2x_2 & + & \dots & + & c_nx_n & & \\ s.t. & a_{i1}x_1 & + & a_{i2}x_2 & + & \dots & + & a_{in}x_n & \geq & b_i \quad i \in M \\ & a_{j1}x_1 & + & a_{j2}x_2 & + & \dots & + & a_{jn}x_n & = & b_j \quad j \in \overline{M} \\ & & & & & & & x_i & \geq & 0 \quad i \in N \end{array}$$



## Form 2: Standard form of linear programming

- Standard form: linear inequalities;

$$\begin{array}{llllllllll} \min & c_1x_1 & + & c_2x_2 & + & \dots & + & c_nx_n & & \text{希望都是} \leq \\ \text{s.t.} & a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & \leq & b_1 \\ & a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & \leq & b_2 \\ & \dots & & \dots & & \dots & & \dots & & \\ & a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & \leq & b_m \\ & & & & & & & x_i & \geq & 0 \text{ for } \forall i \end{array}$$

- Standard form in matrix language:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0 \end{array}$$

满秩

- Here we assume the matrix  $A$  has a full row rank. Otherwise a preprocessing step can be executed to guarantee this.

- Standard form in matrix language:

$$\begin{array}{ll}\min & c^T x \\ \text{s.t.} & Ax \leq b \\ & x \geq 0\end{array}$$

- Here  $c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}$ ,  $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ ,

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}.$$

# Transformation from general form to standard form

- Transformations:

$x_i$ 仍然可能为负

- Variables:** a free variable  $\Rightarrow$  two non-negative variables;  
 $x_i$  may or may not be positive  $\Rightarrow$  replacing  $x_i$  with  $x'_i - x''_i$   
and adding constraints:  $x'_i \geq 0; x''_i \geq 0$

- Constraints:** an equality  $\Rightarrow$  two inequalities;

$$\begin{aligned} a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n &= b_j \Rightarrow \\ a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n &\geq b_j \\ a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n &\leq b_j \end{aligned}$$

等式约束变换成  
两个不等式约束

# Form 3: Slack form of linear programming

松弛

- Slack form: linear equality;

$$\begin{array}{llllllllll} \min & c_1x_1 & + & c_2x_2 & + & \dots & + & c_nx_n & & \\ s.t. & a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1 \\ & a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2 \\ & \dots & & \dots & & \dots & & \dots & & \\ & a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & = & b_m \\ & & & & & & & x_i & \geq & 0 \text{ for } \forall i \end{array}$$

- Slack form in matrix language:

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array}$$

# Transformation from standard form to slack form

- Transformations:

- ① **Variables:** changing “inequality on partial solution  $(x_1, \dots, x_n)$ ” to “equality on full solution  $(s, x_1, \dots, x_n)$ ” by introducing a slack variable  $s$ .

$$a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n \leq b_j \Rightarrow \text{不等式约束变换成等式约束}$$
$$a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n + s = b_j$$

$s$ 为松弛变量

- ② **Constraint:**  $s \geq 0$ . ( $s$  is called a slack variable)

# Example: standard form vs. slack form

- Standard form:

$$\begin{aligned} -x_3 + 2x_4 &\leq 2 \\ 3x_3 - 2x_4 &\leq 6 \\ x_3, x_4 &\geq 0 \end{aligned}$$

- Slack form:

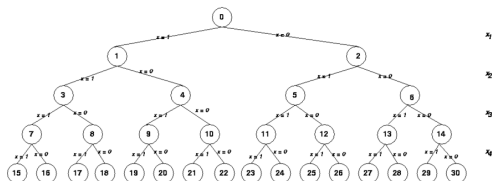
$$\begin{aligned} x_1 - x_3 + 2x_4 &= 2 \\ x_2 + 3x_3 - 2x_4 &= 6 \\ x_1, x_2, x_3, x_4 &\geq 0 \end{aligned}$$

$x_1, x_2$  为松弛变量

# Solving strategy I

$$\begin{array}{llll}
 \max & c_1 + & c_2 + & c_3 \\
 s.t. & x_1 + (1 - x_2) + & x_3 & \geq c_1 \\
 & (1 - x_1) + & x_2 + (1 - x_3) & \geq c_2 \\
 & x_1 + & x_2 + (1 - x_3) & \geq c_3 \\
 & x_1, & x_2, & x_3 = 0/1 \\
 & c_1, & c_2, & c_3 = 0/1
 \end{array}$$

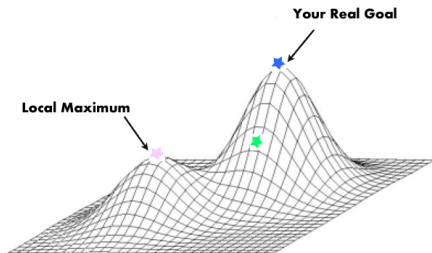
- Solution:  $X = [x_1, x_2, \dots, x_n]$ ,  $x_i = 0/1$
- Partial solution enumeration tree:



- We will talk about “intelligent enumeration”, say branch-and-bound, and backtracking, later.

# Solving strategy II

- Solution:  $x = (x_1, x_2, \dots, x_n)$
- Complete solution landscape



- Each node represents a complete solution, and two solutions are called neighbours if one solution can be obtained from another with a small change.



## Intuition of linear programming

# Two differences from linear equation formula

- Consider a LP (in slack form):

$$\begin{array}{ll}\min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0\end{array}$$

- We have already known how to solve  $Ax = b$ .
- What is the difference between LP and linear equation formula?
  - Constraints:  $x \geq 0$ ;
  - Objective function:  $\min c^T x$ ;

The effect of constraints  $x \geq 0$

# Revisiting $Ax = b$

- An example of  $Ax = b$

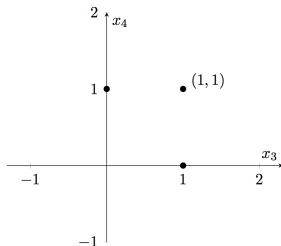
$$\begin{array}{cccccccl} x_1 & & & - & x_3 & + & 2x_4 & = & 2 \\ & x_2 & & + & 3x_3 & - & 2x_4 & = & 6 \\ 2x_1 & + & x_2 & + & x_3 & + & 2x_4 & = & 10 \end{array}$$

- By applying Gaussian elimination, we have: 高斯消元

$$\begin{array}{cccccccl} x_1 & & & - & x_3 & + & 2x_4 & = & 2 \\ & x_2 & + & 3x_3 & - & 2x_4 & = & 6 \end{array}$$

- 直觉上, **any point** in the  $(x_3, x_4)$  plane corresponds to a full solution  $(x_1, x_2, x_3, x_4)$ .

完整解



# The effect of $x \geq 0$

- An example of  $Ax = b, x \geq 0$

$$\begin{array}{rccccrcrcl} x_1 & & & - & x_3 & + & 2x_4 & = & 2 \\ & & x_2 & + & 3x_3 & - & 2x_4 & = & 6 \\ 2x_1 & + & x_2 & + & x_3 & + & 2x_4 & = & 10 \\ \textcolor{red}{x_1} & , & \textcolor{red}{x_2} & , & \textcolor{red}{x_3} & , & \textcolor{red}{x_4} & \geq & 0 \end{array}$$

- By applying Gaussian elimination, we have:

$$\begin{array}{rccccrcrcl} x_1 & & & - & x_3 & + & 2x_4 & = & 2 \\ & & x_2 & + & 3x_3 & - & 2x_4 & = & 6 \\ \textcolor{red}{x_1} & , & \textcolor{red}{x_2} & , & \textcolor{red}{x_3} & , & \textcolor{red}{x_4} & \geq & 0 \end{array}$$

- This is essentially a **linear inequality formula**:

$$\begin{array}{rccccrcrcl} & & & - & x_3 & + & 2x_4 & \leq & 2 \\ & & 3x_3 & - & 2x_4 & \leq & 6 \\ \textcolor{red}{x_3} & , & \textcolor{red}{x_4} & \geq & 0 \end{array}$$

# The effect of $x \geq 0$ cont'd

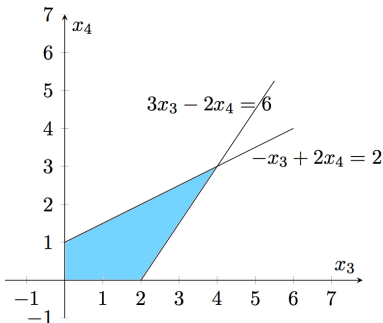
- Linear inequality formula:

$$-x_3 + 2x_4 \leq 2$$

$$3x_3 - 2x_4 \leq 6$$

$$x_3, x_4 \geq 0$$

- Any point in **the polytope** rather than **the whole plane** corresponds to a feasible solution, e.g.  $(x_3, x_4) = (1, 1)$  corresponds to  $(x_1, x_2, x_3, x_4) = (1, 5, 1, 1)$ .



多胞形：有界的多边体

## Theorem

**Any polytope**  $P \subset \mathbb{R}^{n-m}$  corresponds to the feasible region of a linear program  $Ax = b, x \geq 0$  (denoted as  $F = \{x \mid Ax = b, x \geq 0\}$ ), and vice versa.

- Basic idea: What is the effect of constraint  $x \geq 0$ ? It implies the interchangeability between **equalities on all variables** (e.g.  $x_2 + 3x_3 - 2x_4 = 6$ ) and **inequalities on partial variables** (e.g.  $3x_3 - 2x_4 \leq 6$ ).

$n$ : 有几个变量  
 $m$ : 有几个约束

把所有变量的等式  
变成部分变量的不等式

# Proof: feasible region $\Rightarrow$ polytope

- Basic idea: changing **equality** to **inequality** through Gaussian row operations followed by removing some variables.
- Consider a **feasible full solution**  $x$  of the following LP:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = & b_2 \\ & & & & \dots & & & & \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & = & b_m \\ x_1 & , & x_2 & , & \dots & , & x_n & \geq & 0 \end{array}$$

- Applying Gaussian row operations, we have:

$$\begin{array}{ccccccccc} x_1 & & & + & a'_{1,m+1}x_{m+1} & + & \dots & + & a'_{1n}x_n & = & b'_1 \\ & x_2 & & + & a'_{2,m+1}x_{m+1} & + & \dots & + & a'_{2n}x_n & = & b'_2 \\ & & \dots & & & & \dots & & & & \\ & & & x_m & + & a'_{m,m+1}x_{m+1} & + & \dots & + & a'_{mn}x_n & = & b'_m \\ x_1 & , & x_2 & , & x_m & , & x_{m+1} & , & \dots & , & x_n & \geq & 0 \end{array}$$

高斯变换后，左边为对角阵



# Proof: feasible region $\Rightarrow$ polytope cont'd

- By **removing positive variables**  $x_1, x_2, \dots, x_m$ , we have the following **linear inequalities**:

$$\begin{array}{ccccccc} a'_{1,m+1}x_{m+1} & + & \dots & + & a'_{1n}x_n & \leq & b'_1 \\ a'_{2,m+1}x_{m+1} & + & \dots & + & a'_{2n}x_n & \leq & b'_2 \\ & & & & \dots & & \\ a'_{m,m+1}x_{m+1} & + & \dots & + & a'_{mn}x_n & \leq & b'_m \\ x_{m+1} & , & \dots & , & x_n & \geq & 0 \end{array}$$

- Define a polytope  $P \subset \mathbb{R}^{n-m}$  as the intersection of  $m$  half-spaces:  
 $HS_j : a'_{j,m+1}x_{m+1} + \dots + a'_{jn}x_n \leq b'_j, 1 \leq j \leq m. \text{ (by } x_j \geq 0)$
- Thus, **any feasible full solution**  $x = (x_1, x_2, \dots, x_n) \Rightarrow$   
**partial solution**  $x_N = (x_{m+1}, \dots, x_n) \in P.$

# Proof: polytope $\Rightarrow$ feasible region

任意多边形都对应一个可行域？

- Basic idea: changing **inequality** to **equality** through introducing **slack variables**.
  - Suppose  $P$  is the intersection of  $m$  half-spaces (inequalities), say:  
 $HS_j : a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n \leq b_j \quad (1 \leq j \leq m)$
  - Introducing a non-negative slack variable  $s_j$  to each inequality, we have:  
 $a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n + s_j = b_j \quad (s_j \geq 0)$

# Proof: polytope $\Rightarrow$ feasible region cont'd

- Thus we change

每一个不等式都是一个半空间  
半空间的交 形成一个多面体

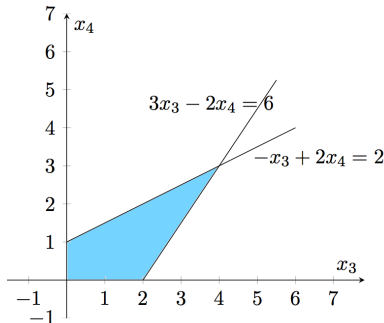
$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & \leq & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & \leq & b_2 \\ & & & & \dots & & & & \\ a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & \leq & b_m \\ x_1 & , & x_2 & , & \dots & , & x_n & \geq & 0 \end{array}$$

into

$$\begin{array}{ccccccc} s_1 & & + & a_{1,1}x_1 & + & \dots & + & a_{1n}x_n & = & b_1 \\ s_2 & & + & a_{2,1}x_1 & + & \dots & + & a_{2n}x_n & = & b_2 \\ & & \dots & & \dots & & & & & \\ & & s_m & + & a_{m,1}x_1 & + & \dots & + & a_{mn}x_n & = & b_m \\ s_1 & , & s_2 & , & s_m & , & x_1 & , & \dots & , & x_n & \geq & 0 \end{array}$$

- Thus, a **partial solution**  $(x_1, x_2, \dots, x_n) \in P \Rightarrow$  a **feasible full solution**  $(s_1, s_2, \dots, s_m, x_1, x_2, \dots, x_n) \geq 0$ .

# Notations



- Hyper plane:  $\{x \mid a_1x_1 + a_2x_2 + \dots + a_nx_n = b\}$  (linear equality constraint)

半空间 • Half space:  $\{x \mid a_1x_1 + a_2x_2 + \dots + a_nx_n \leq b\}$  (linear inequality constraint)

多面体 • Polyhedron: the intersection of several half spaces;

- Polytope: a bounded, non-empty polyhedron;

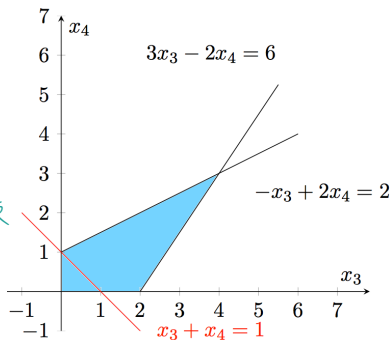
多胞形：有界

The effect of objective function  $\min c^T x$

# The effect of $\min c^T x$

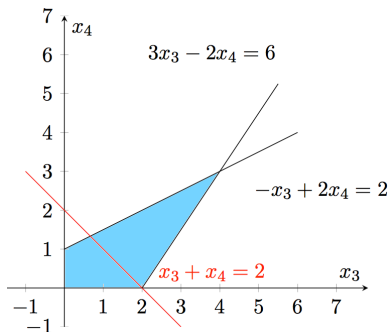
$$\begin{array}{llllll} \max & & x_3 & + & x_4 & \\ s.t. & & -x_3 & + & 2x_4 & \leq 2 \\ & & 3x_3 & - & 2x_4 & \leq 6 \\ & & x_3 & , & x_4 & \geq 0 \end{array}$$

阴影区满足上面约束



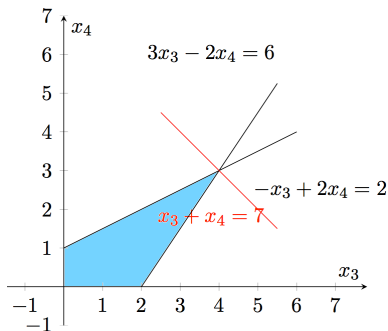
# The effect of $\min c^T x$ cont'd

$$\begin{array}{llllll} \max & & x_3 & + & x_4 & \\ s.t. & & -x_3 & + & 2x_4 & \leq 2 \\ & & 3x_3 & - & 2x_4 & \leq 6 \\ & & x_3 & , & x_4 & \geq 0 \end{array}$$



# The effect of $\min c^T x$ cont'd

$$\begin{array}{llllll} \max & & x_3 & + & x_4 & \\ s.t. & & -x_3 & + & 2x_4 & \leq 2 \\ & & 3x_3 & - & 2x_4 & \leq 6 \\ & & x_3 & , & x_4 & \geq 0 \end{array}$$



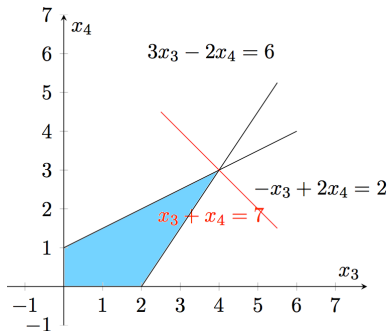
顶点

- Observation: the optimal solution can be reached at a vertex of the polytope (if the optimal objective value is finite).



# Key observations of linear program

只关心顶点  
不用关心内部点



可行域

部分变量张成的多面体

- ① What is a feasible solution? Any point within the polytope.
- ② Where is the optimal solution? A vertex of the polytope (if the optimal objective value is finite). Consequently, it is not necessary to consider the inner points. In fact, the existence of a vertex solution makes the simplex algorithm distinctive from ellipsoid method and interior point method.

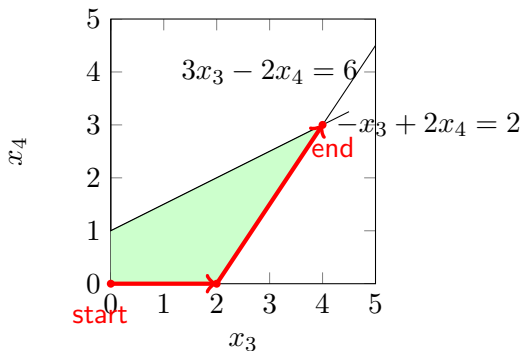
## Applying the general IMPROVEMENT strategy to LP

# The general IMPROVEMENT strategy for optimization problems

IMPROVEMENT( $f$ )

```
1:  $x = x_0$ ; //set initial solution;  
2: while TRUE do  
3:    $x = \text{IMPROVE}(x, f)$ ; //move towards optimum;  
4:   if STOPPING( $x, f$ ) then  
5:     break;  
6:   end if  
7: end while  
8: return  $x$ ;
```

# Applying the general IMPROVEMENT strategy to LP

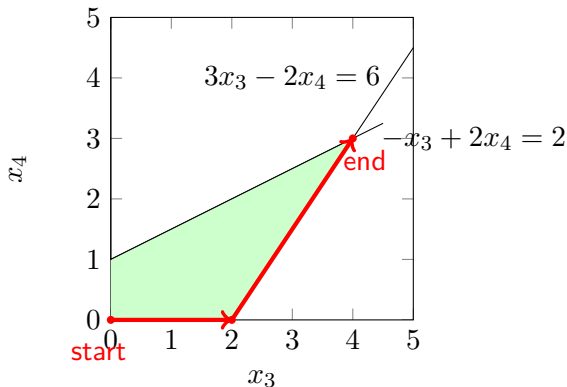


IMPROVEMENT()

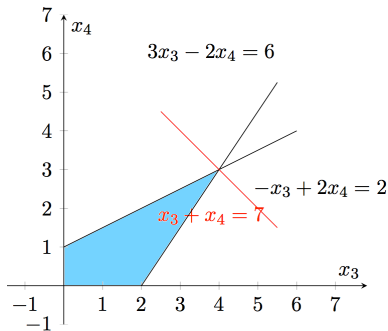
- 1:  $x = x_0$ ; //starting from a vertex;
- 2: **while** TRUE **do**
- 3:    $x = \text{IMPROVE}(x)$ ; //move to another vertex via an edge;
- 4:   **if** STOPPING( $x$ ) **then**
- 5:     **break**; //stop when  $x$  is optimal
- 6:   **end if**
- 7: **end while**

# Some questions to answer

- 1 Why does it suffice to consider vertices of the polytope only?
- 2 How to obtain a vertex?
- 3 How to implement “moving to another vertex via an edge”?
- 4 When should we stop?



Question 1: Why does it suffice to consider vertices of the polytope only?



# Optimal solution can be reached at a vertex

## Theorem

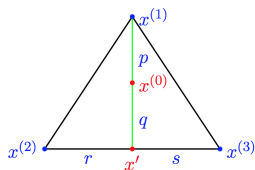
*There exists a vertex in  $P$  that takes the optimal value (if the optimal objective value is finite).*

有限的

## Proof.

- Since  $P$  is a bounded close set,  $c^T x$  reaches its optimum in  $P$ .
- Denote the optimal solution as  $x^{(0)}$ . We will show there is a vertex at least as good as  $x^{(0)}$ . Why?
  - $x^{(0)}$  can be represented as the convex combination of vertices of  $P$ , i.e.  $x^{(0)} = \lambda_1 x^{(1)} + \lambda_2 x^{(2)} + \dots + \lambda_k x^{(k)}$ , where  $\lambda_i \geq 0, \lambda_1 + \dots + \lambda_k = 1$ . (See Appendix for details.)
  - Thus  $c^T x^{(0)} = \lambda_1 c^T x^{(1)} + \lambda_2 c^T x^{(2)} + \dots + \lambda_k c^T x^{(k)}$
  - Let  $x^{(i)}$  be the vertex with the minimal objective value  $c^T x^{(i)}$ ;
  - $c^T x^{(0)} = \lambda_1 c^T x^{(1)} + \lambda_2 c^T x^{(2)} + \dots + \lambda_k c^T x^{(k)} \geq c^T x^{(i)}$ .
- Thus, vertex  $x^{(i)}$  is also an optimal solution since  $c^T x^{(i)} \leq c^T x^{(0)}$





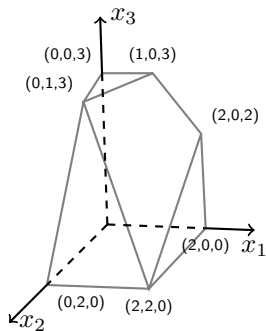
不管 $c$ 怎么取，最优解必定在三个顶点处取到

- Suppose  $x^{(0)}$  is an optimal solution.
- Connecting  $x^{(0)}$  and  $x^{(1)}$  with a line. Suppose the line intersects line segment  $(x^{(2)}, x^{(3)})$  at point  $x'$ .
- We have  $x^{(0)} = \lambda_1 x^{(1)} + (1 - \lambda_1) x'$ , where  $\lambda_1 = \frac{q}{p+q}$ .
- We also have  $x' = \lambda_2 x^{(2)} + (1 - \lambda_2) x^{(3)}$ , where  $\lambda_2 = \frac{s}{r+s}$ .
- Thus, we have  

$$x^{(0)} = \overset{w_1}{\lambda_1} x^{(1)} + (1 - \overset{w_2}{\lambda_1}) \overset{w_3}{\lambda_2} x^{(2)} + (1 - \overset{w_3}{\lambda_1}) (1 - \overset{\text{lambda都为}[0,1]}{\lambda_2}) x^{(3)}.$$
- Suppose  $c^T x^{(1)}$  is the minimum of  $c^T x^{(1)}, c^T x^{(2)}, c^T x^{(3)}$ .
- Notice that  $\lambda_1 + (1 - \lambda_1) \lambda_2 + (1 - \lambda_1) (1 - \lambda_2) = 1$ .  $w_1 + w_2 + w_3 = 1$
- We have:  $c^T x^{(1)} \leq c^T x^{(0)}$ . Thus, a vertex  $x^{(1)}$  is found not worse than  $x^{(0)}$ .



Question 2: How to obtain a vertex of the polytope?



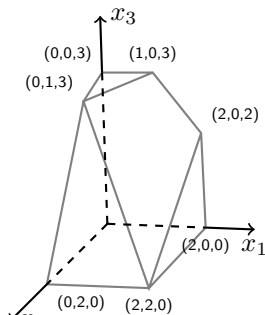
# Vertex $\Leftrightarrow$ basic feasible solution

## Theorem

*A vertex of  $P$  corresponds to a basis of matrix  $A$ .*

An example (standard form):

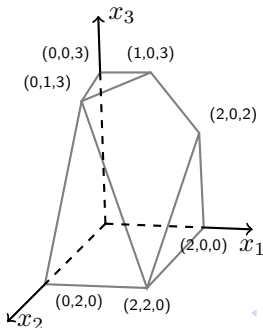
$$\begin{array}{llllllll} \min & -x_1 & - & 14x_2 & - & 6x_3 & & \\ s.t. & x_1 & + & x_2 & + & x_3 & \leq & 4 \\ & x_1 & & & & & \leq & 2 \\ & & & & & x_3 & \leq & 3 \\ & & & 3x_2 & + & x_3 & \leq & 6 \\ & x_1 & , & x_2 & , & x_3 & \geq & 0 \end{array}$$



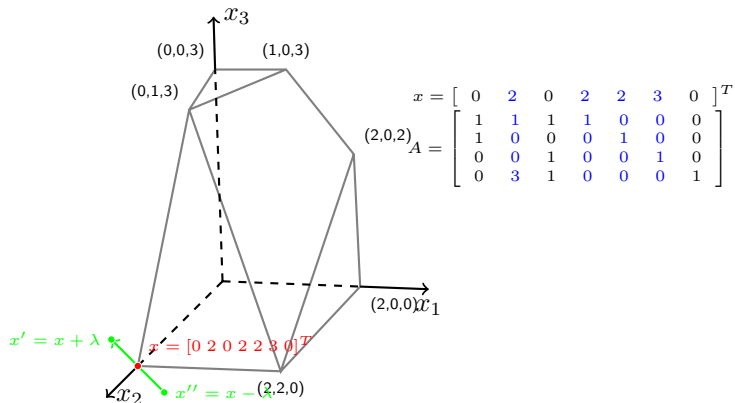
# Part 1: Vertex $\Rightarrow$ basic feasible solution

- We will first show that **any vertex** of the polytope corresponds to a basis of the matrix  $A$ .
- An example (slack form):

$$\begin{array}{llllllllllllllll} \min & -x_1 & - & 14x_2 & - & 6x_3 & & & & & & & & & & & & \\ s.t. & x_1 & + & x_2 & + & x_3 & + & x_4 & & & & & & & & & & = 4 \\ & x_1 & & & & & & & + & x_5 & & & & & & & & = 2 \\ & & & & & x_3 & & & & & + & x_6 & & & & & & = 3 \\ & & & 3x_2 & + & x_3 & & & & & & & + & x_7 & & & & = 6 \\ & x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & , & x_6 & , & x_7 & \geq 0 \end{array}$$



# Intuitive idea I



- Take the vertex  $(x_1, x_2, x_3) = (0, 2, 0)$  as an example. The corresponding full solution is  $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (0, 2, 0, 2, 2, 3, 0)$ .

# Intuitive idea II

- We will show that the column vectors corresponding to **non-zero**  $x_i$ , i.e.  $\{a_2, a_4, a_5, a_6\}$ , are linearly independent, and thus form a basis (sometimes an extension is needed). (Here  $a_i$  denotes the  $i$ -th column vector of  $A$ )
- Suppose  $\exists(\lambda_2, \lambda_4, \lambda_5, \lambda_6) \neq 0$  such that  $\lambda_2 a_2 + \lambda_4 a_4 + \lambda_5 a_5 + \lambda_6 a_6 = 0$ , i.e.  $A\lambda = 0$ , where  $\lambda = [0, \lambda_2, 0, \lambda_4, \lambda_5, \lambda_6, 0]$ .
- Then we can construct **two other points**:  $x' = x + \theta\lambda$  and  $x'' = x - \theta\lambda$ .

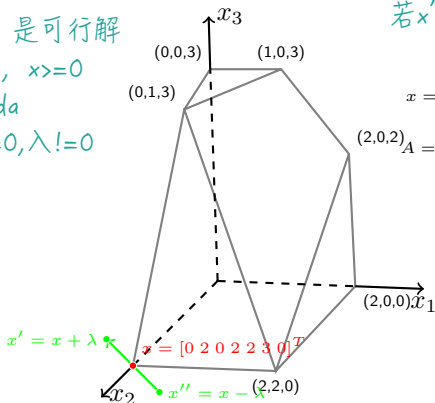
1.  $x$  是顶点, 是可行解

$$Ax = b, x \geq 0$$

2. 找到  $\lambda$

$$A\lambda = 0, \lambda \neq 0$$

若  $x' \geq 0$ , 则  $x'$  在  $P$  内



$$x = \begin{bmatrix} 0 & 2 & 0 & 2 & 2 & 3 & 0 \end{bmatrix}^T$$
$$A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- It is easy to deduce that both  $x'$  and  $x''$  lie inside  $P$  since:
  - $Ax' = Ax + 0 = b$  and  $Ax'' = Ax - 0 = b$
  - In addition, we can guarantee  $x' \geq 0$  and  $x'' \geq 0$  via setting  $\theta$  to be sufficiently small since  $x \geq 0$ , and  $\lambda_1 = \lambda_3 = \lambda_7 = 0$ .
- Contradiction: it is impossible for a vertex to be middle point of two inner points of  $P$ .

- ① Suppose  $\hat{x} = \langle x_{m+1}, \dots, x_n \rangle$  is a vertex of  $P \subset \mathbb{R}^{n-m}$ , i.e. we have  $a'_{i,m+1}x_{m+1} + \dots + a'_{i,n}x_n \leq b'_i$  for all  $1 \leq i \leq m$ .
- ② Expanding **partial** solution  $\hat{x}$  to a feasible **full** solution  $x = \langle x_1, \dots, x_m, x_{m+1}, \dots, x_n \rangle$ , where  $x_1, \dots, x_m$  are calculated according to the equality constraints of the LP model.
- ③ Considering the non-zero items  $x_j$  in  $x$ . Note that the corresponding columns  $B = \{a_j | x_j \neq 0\}$  form a basis. Why?
  - ① Suppose there exist  $d_j$  such that  $\sum_{a_j \in B} d_j a_j = 0$  ( $\langle d_j \rangle \neq 0$ ).
  - ② Since  $\sum_{a_j \in B} x_j a_j = b$  ( $x_k = 0$  for all  $a_k \notin B$ ), we can construct two **full** feasible solutions  $\langle x_i + \theta d_i \rangle$  and  $\langle x_i - \theta d_i \rangle$  since:  
 $\sum_{a_j \in B} (x_j \pm \theta d_j) a_j = b$ . (We can guarantee  $x_j \pm \theta d_j \geq 0$  through setting  $\theta$  sufficiently small.)
  - ③ Thus the corresponding two **partial** solutions are in  $P$ :  
 $x' = \langle x'_{m+1}, \dots, x'_n \rangle$ , where  $x'_j = x_j + \theta d_j$  for  $a_j \in B$ , and 0 otherwise;  
 $x'' = \langle x''_{m+1}, \dots, x''_n \rangle$ , where  $x''_j = x_j - \theta d_j$  for  $a_j \in B$ , and 0 otherwise;
  - ④ Thus  $\hat{x} = \frac{1}{2}x' + \frac{1}{2}x''$ . A contradiction. (A vertex in  $P$  cannot be represented as the convex combination of two points in  $P$ . See Appendix.)
- ④ Thus,  $x$  is a basic feasible solution corresponding to basis  $B$  since: 1)  $x$  can be represented as  $x = \begin{bmatrix} x_B \\ 0 \end{bmatrix}$ , and 2) any item  $x_j \geq 0$ .  $\square$

# Vertex $\Rightarrow$ basic feasible solution: some notations

- For a vertex  $x$  of the polytope, a basis <sup>基</sup>  $B$  can be derived via extracting the column vectors corresponding to non-zero  $x_i$ . The non-basis column vectors are denoted as  $N$ .
- Then the original LP can be represented as:

$$\begin{array}{ll} \min & c_B^T x_B + c_N^T x_N \\ \text{s.t.} & \begin{bmatrix} B & N \end{bmatrix} \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} b \end{bmatrix} \end{array}$$

- Here,  $x$  is decomposed as  $x = \begin{bmatrix} x_B \\ x_N \end{bmatrix}$ . Then we have

$x_N = 0$ , and  $x_B = B^{-1}b$  (Reason:  $Ax = b$ , i.e.  $Bx_B + Nx_N = b$ )

- The corresponding objective value is  $c^T x = c_B^T x_B + c_N^T x_N = c_B^T B^{-1}b$ .



# An example

- For a vertex  $x = [0 \ 2 \ 0 \ 2 \ 2 \ 3 \ 0]^T$ , the columns corresponding to non-zero  $x_i$  are extracted to form a basis

$$B = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 \end{bmatrix}.$$

- Let's decompose  $x = [0 \ 2 \ 0 \ 2 \ 2 \ 3 \ 0]^T$  accordingly into  $x_B = [2 \ 2 \ 2 \ 3]^T$  and  $x_N = [0 \ 0 \ 0]^T$ .
- It is easy to verify that  $x_B = B^{-1}b$ . In this example,

$$b = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 6 \end{bmatrix}.$$

## Part 2: Basic feasible solution $\Rightarrow$ vertex

- Given a basis  $B$  of matrix  $A$ , we call  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix}$  a **basic solution respect to  $B$** .
- If we further have  $x_B = B^{-1}b \geq 0$ ,  $x$  is called a **basic feasible solution respect to  $B$** .
- We will show that a **basic feasible solution  $x$  respect to  $B$**  is a vertex of the polytope  $P$ .

### Proof.

- It suffices to show that  $x$  cannot be represented as a convex 凸组合 combination of any two points in  $P$ .
- By contradiction, suppose there are two different points  $x^{(1)}$  and  $x^{(2)}$  in  $P$  such that  $x = \lambda_1 x^{(1)} + \lambda_2 x^{(2)}$ , where  $0 < \lambda_1, \lambda_2 < 1$ .
- Note that  $\lambda_1 x_N^{(1)} + \lambda_2 x_N^{(2)} = x_N = 0$ .
- So  $x_N^{(1)} = x_N^{(2)} = 0$  (by  $\lambda_1, \lambda_2 \geq 0$  and  $x_N^{(1)}, x_N^{(2)} \geq 0$ ).
- Then we have  $x_B^{(1)} = x_B^{(2)} = B^{-1}b = x_B$  (by  $Ax^{(1)} = b$  and  $Ax^{(2)} = b$ ). A contradiction.

# An example

- For matrix  $A = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$ , and  $b = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 6 \end{bmatrix}$ ,

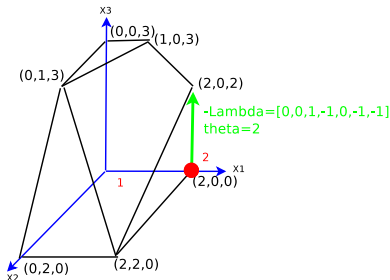
we first calculate a basis of  $A$  as  $B = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 \end{bmatrix}$ .

- The **basic feasible solution**  $x$  respect to  $B$  is

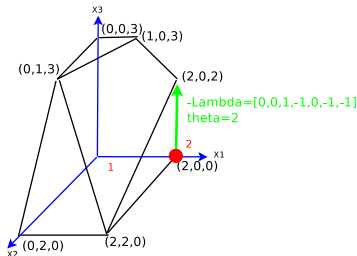
$$x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = [0 \ 2 \ 0 \ 2 \ 2 \ 3 \ 0]^T.$$

- It is easy to verify that  $(x_1, x_2, x_3) = (0, 2, 0)$  is a vertex of the polytope  $P$ .

Question 3: How to implement “moving from a vertex to another vertex via an edge”?



# Edge $\Leftrightarrow$ non-basis column vector of $A$ : an example



$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$

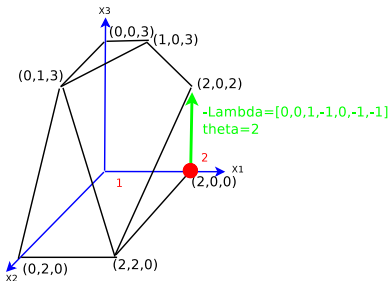
$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Take the vertex  $(x_1, x_2, x_3) = (2, 0, 0)$  as an example. The corresponding full solution is  $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (2, 0, 0, 2, 0, 3, 6)$ .
- Basis (in blue):  $B = \{a_1, a_4, a_6, a_7\}$ .
- Let's consider a **non-basis column vector**  $a_3$ .
- Since  $a_3$  can be decomposed as  $a_3 = 1a_4 + 0a_1 + 1a_6 + 1a_7$ , we have  $0a_1 + 0a_2 - 1a_3 + 1a_4 + 0a_5 + 1a_6 + 1a_7 = 0$
- We will show that the coefficients  $\lambda = [0, 0, -1, 1, 0, 1, 1]^T$  specifies the direction of **the edge in green**.
- More specifically, we can move via the edge to another vertex  $x' = x - \theta\lambda = [2, 0, 2, 0, 0, 1, 4]^T$  (by setting  $\theta = 2$ ).
- The new vertex corresponds to the basis  $B' = B - \{a_4\} \cup \{a_3\} = \{a_1, a_3, a_6, a_7\}$ .

# Edge $\Leftrightarrow$ non-basis column vector of $A$

## Theorem

Let  $x = [x_1, x_2, \dots, x_n]^T$  be a vertex corresponding to basis  $B = \{a_1, a_2, \dots, a_m\}$ . Consider a non-basis vector  $a_e \notin B$ . Suppose  $a_e$  can be decomposed as  $a_e = \lambda_1 a_1 + \lambda_2 a_2 + \dots + \lambda_m a_m$ . Let  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{x_i}{\lambda_i} = \frac{x_l}{\lambda_l}$ . Then  $x' = x - \theta \lambda$  is also a vertex corresponding to basis  $B' = B - \{a_l\} \cup \{a_e\}$ . Here  $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_m, 0, \dots, -1, \dots, 0]$ .



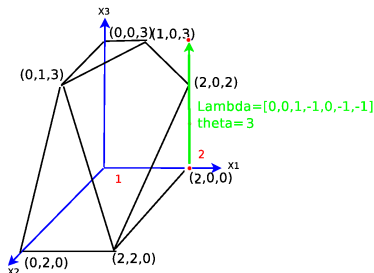
## Part 1: $x' = x - \theta\lambda$ is a feasible solution

### Proof.

- We have  $x_1a_1 + x_2a_2 + \dots + x_ma_m = b$  (Reason:  $x$  is a feasible solution).
- We also have  $\lambda_1a_1 + \lambda_2a_2 + \dots + \lambda_ma_m - a_e = 0$ .  $Ax=b$   
 $A\lambda=0$   
 $x \geq 0$
- Thus we have  
 $(x_1 - \theta\lambda_1)a_1 + \dots + (x_m - \theta\lambda_m)a_m + \dots + \theta a_e = b$ .
- To show that  $x' = x - \theta\lambda$  is also feasible, it suffices to prove  $x' \geq 0$ . There are two cases:
  - ①  $\forall i, \lambda_i \leq 0$ : for any positive  $\theta$  we still have  $x'_i = x_i - \theta\lambda_i \geq x_i \geq 0$ .
  - ②  $\exists i, \lambda_i > 0$ : we cannot set  $\theta$  too large. In fact, by setting  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{x_i}{\lambda_i} = \frac{x_l}{\lambda_l}$ , we can guarantee  $x_i - \theta\lambda_i \geq 0$ ; however, a larger  $\theta$  will cause  $(x_l - \theta\lambda_l) < 0$ . For example,  
 $x = [2, 0, 0, 2, 0, 3, 6]^T$   
 $\lambda = [0, 0, -1, 1, 0, 1, 1]^T$   
We set  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{x_i}{\lambda_i} = \frac{x_4}{\lambda_4} = 2$  and  $l = 4$ .
- Thus  $x'$  is a new feasible solution.

注意的  
取值。

# How to set $\theta$ ? Trying a larger step: $\theta = 3$



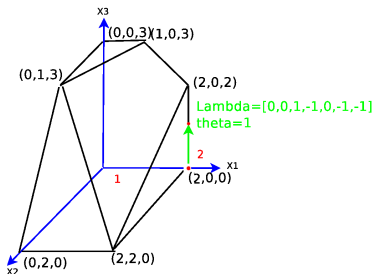
$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Vertex  $(x_1, x_2, x_3) = (2, 0, 0) \Rightarrow (x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (2, 0, 0, 2, 0, 3, 6)$ .
- Basis (in blue):  $B = \{a_1, a_4, a_6, a_7\}$ .
- Let's consider a **non-basis column vector**  $a_3$ .
- Since  $a_3$  can be decomposed as  $a_3 = 1a_4 + 0a_1 + 1a_6 + 1a_7$ , i.e.,  $0a_1 + 0a_2 - 1a_3 + 1a_4 + 0a_5 + 1a_6 + 1a_7 = 0$ .
- The coefficients  $\lambda = [0, 0, -1, 1, 0, 1, 1]^T$  corresponds to **the edge in green**.
- $x' = x - \theta\lambda = [2, 0, 3, -1, 0, 0, 3]^T$  (by setting  $\theta = 3$ ) is NOT a feasible solution.



# How to set $\theta$ ? Trying a smaller step: $\theta = 1$

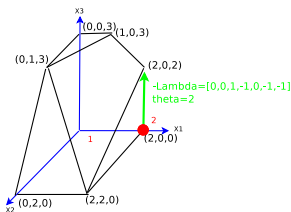


$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Vertex  $(x_1, x_2, x_3) = (2, 0, 0) \Rightarrow (x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (2, 0, 0, 2, 0, 3, 6)$ .
- Basis (in blue):  $B = \{a_1, a_4, a_6, a_7\}$ .
- Let's consider a **non-basis column vector**  $a_3$ .
- Since  $a_3$  can be decomposed as  $a_3 = 1a_4 + 0a_1 + 1a_6 + 1a_7$ , i.e.,  $0a_1 + 0a_2 - 1a_3 + 1a_4 + 0a_5 + 1a_6 + 1a_7 = 0$ .
- The coefficients  $\lambda = [0, 0, -1, 1, 0, 1, 1]^T$  corresponds to **the edge in green**.
- $x' = x - \theta\lambda = [2, 0, 1, 1, 0, 2, 5]^T$  (by setting  $\theta = 1$ ) is NOT a vertex.

## Part 2: $B' = B - \{a_l\} \cup \{a_e\}$ is a basis.



$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- To show that  $x' = x - \theta\lambda = [2, 0, 2, 0, 0, 1, 4]^T$  is also a vertex, it suffices to show that the column vectors corresponding to non-zero  $x_i$  form a basis, i.e.  $B' = B - \{a_4\} \cup \{a_3\} = \{a_1, a_3, a_6, a_7\}$  is a basis.
- Suppose  $B'$  is linear dependent, i.e. there exists  $(d_1, d_3, d_6, d_7) \neq 0$  such that  $d_1 a_1 + d_3 a_3 + d_6 a_6 + d_7 a_7 = 0$ .
- Recall that  $a_3$  can be decomposed as  $a_3 = 1a_4 + 0a_1 + 1a_6 + 1a_7$ .
- We have  $d_1 a_1 + d_3 a_4 + (d_6 + d_3) a_6 + (d_7 + d_3) a_7 = 0$ .
- Thus  $d_3 = 0$ . (Reason:  $B = \{a_1, a_4, a_6, a_7\}$  is a basis.)
- Therefore  $d_1 = d_6 = d_7 = 0$ . Contradiction.

## Part 2: $B' = B - \{a_l\} \cup \{a_e\}$ is a basis.

### Proof.

- Suppose  $B'$  is linear dependent;
- Thus, there exists  $\langle d_1, \dots, d_{l-1}, d_{l+1}, \dots, d_m, d_e \rangle \neq 0$  such that  $d_1 a_1 + \dots d_{l-1} a_{l-1} + d_{l+1} a_{l+1} + \dots + d_m a_m + d_e a_e = 0$ .
- We also have  $a_e = \lambda_1 a_1 + \dots + \lambda_l a_l + \dots + \lambda_m a_m$ .
- Substituting  $a_e$  into the above equation, we have:
- $(d_1 + d_e \lambda_1) a_1 + \dots + (d_e \lambda_l) a_l + \dots + (d_m + d_e \lambda_m) a_m = 0$
- Thus  $d_e \lambda_l = 0$ . (Reason:  $B = \{a_1, \dots, a_m\}$  is a basis.)
- Therefore  $d_e = 0$  (Reason:  $\lambda_l \neq 0$ ).
- Therefore we have  $d_i = 0$  for all  $i$  (Reason:  $d_i = d_i + d_e \lambda_i = 0$ ). A contradiction.



# Pivoting operation

- The process to change  $B$  into  $B'$  is called **“pivoting”** with  $a_e$  **“entering”** basis, and  $a_l$  **“leaving”** basis.
- The “pivoting” operation can be accomplished by Gaussian row operation.

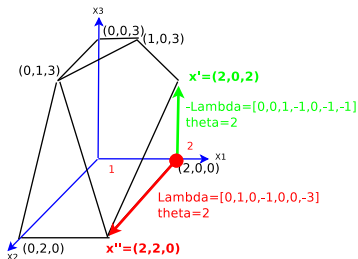
$$\left( \begin{array}{c|cccccc} 0 & \dots & 0 & \dots & c_e & \dots \\ b_1 & \dots & 0 & \dots & a_{1e} & \dots \\ b_2 & \dots & 0 & \dots & a_{2e} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_l & \dots & 1 & \dots & a_{le} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_m & \dots & 0 & \dots & a_{me} & \dots \end{array} \right) \Rightarrow \left( \begin{array}{c|cccccc} -\frac{a_{me}}{a_{le}} b_l & \dots & -\frac{c_e}{a_{le}} & \dots & 0 & \dots \\ b_1 - \frac{a_{1e}}{a_{le}} b_l & \dots & -\frac{a_{1e}}{a_{le}} & \dots & 0 & \dots \\ b_2 - \frac{a_{2e}}{a_{le}} b_l & \dots & -\frac{a_{2e}}{a_{le}} & \dots & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{1}{a_{le}} b_l & \dots & \frac{1}{a_{le}} & \dots & 1 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_m - \frac{a_{me}}{a_{le}} b_l & \dots & -\frac{a_{me}}{a_{le}} & \dots & 0 & \dots \end{array} \right)$$

- The details will be described after introducing simplex tabular.

An additional question: which edge is preferred when moving from a vertex?

# Which edge is preferred?

- Generally speaking, a vertex of  $P$  has at most  $n - m$  adjacent edges (Why?)



$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$

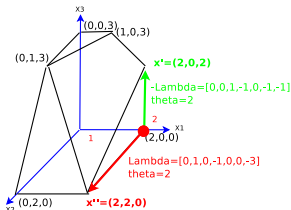
$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Here two edges adjacent to the vertex  $(x_1, x_2, x_3) = (2, 0, 0)$  are shown as example:
  - the edge in green (corresponding to  $a_3$ ) to vertex  $x'$ ;
  - the edge in red (corresponding to  $a_2$ ) to vertex  $x''$ ;
- Which edge is preferred when moving from the vertex  $(x_1, x_2, x_3) = (2, 0, 0)$ ?
- An equivalent question: which non-basis vector should be selected to enter the basis?

哪个非基  
向量应该  
被选择  
pivot



## Trial 2: pivoting in $a_3$



$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- We decompose  $a_3$  as  $a_3 = 1a_4 + 0a_1 + 1a_6 + 1a_7$ , i.e.,  $0a_1 + 0a_2 - 1a_3 + 1a_4 + 0a_5 + 1a_6 + 1a_7 = 0$ .
- The coefficient is:  $\lambda = (0, 0, -1, 1, 0, 1, 1)$ .
- By setting an appropriate  $\theta$ , we get to vertex  $x' = x - \theta\lambda$ .
- The objective value can be improved by  $c^T x' - c^T x = (c_3 - (1c_4 + 0c_1 + 1c_6 + 1c_7))\theta = -6\theta$

**Largest number rule (maximal gradient heuristic):** To make as fast improvement as possible, we select the non-basis vector  $a_e$  with the smallest  $c_e - \sum_{a_i \in B} \lambda_i c_i$  to enter the basis.



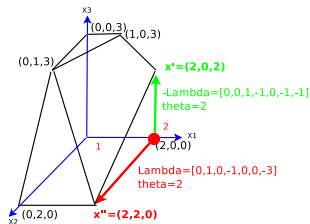
# How to choose a non-basis vector $a_e$ to enter $B$ ?

- Consider a vertex  $x = (x_1, x_2, \dots, x_n)$  corresponding to basis  $B = \{a_1, a_2, \dots, a_m\}$ .
- Suppose we choose a non-basis vector  $a_e \notin B$  to enter basis
- Since  $a_e$  is not in basis, it can be decomposed as
$$a_e = \lambda_1 a_1 + \lambda_2 a_2 + \dots + \lambda_m a_m$$
- Let  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{x_i}{\lambda_i} = \frac{x_l}{\lambda_l}$ .
- Then  $x' = x - \theta \lambda$  is also a vertex, where
$$\lambda = [\lambda_1, \lambda_2, \dots, \lambda_m, 0, \dots, -1, \dots, 0]^T. \quad x' = x - \theta \lambda$$
- Recall that the objective is to minimize  $c^T x$ . Let's see whether we can improve the objective function by moving from vertex  $x$  to  $x'$ .
- Notice  $c^T x' - c^T x = -\theta c^T \lambda = (c_e - \sum_{a_i \in B} \lambda_i c_i) \theta$ .
- Pivoting in rule:

To make as large improvement as possible, we select the non-basis vector  $a_e$  to enter the basis. Here,  $e$  is the index with the smallest  $c_e - \sum_{a_i \in B} \lambda_i c_i$ .

Question 4: When should we stop?

# Stopping criterion



$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Notice: suppose we move from vertex  $x$  to  $x' = x - \theta\lambda$ , the improvement of objective value is  

$$c^T x' - c^T x = \theta c^T \lambda = (c_e - \sum_{a_i \in B} \lambda_i c_i) \theta.$$
- We will benefit from pivoting in  $a_e$  if  $c^T x' \leq c^T x$ , i.e. 可继续下降  

$$c_e - \sum_{a_i \in B} \lambda_i c_i < 0.$$
- Thus the following stopping criteria is reasonable: 最优的情况  

$$c_e - \sum_{a_i \in B} \lambda_i c_i \geq 0 \text{ for all } e.$$
- We denote  $\bar{c}_e \triangleq c_e - \sum_{a_i \in B} \lambda_i c_i$  as **“checking number”**. 检验数
- In fact,  $\bar{c}_e$  is the  $e$ -th entry of  $\bar{c}^T = c^T - c_B^T B^{-1} A$ .

# Stopping criteria

## Theorem

Consider a LP (in slack form):

$$\begin{array}{ll}\min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0\end{array}$$

Let  $x$  be a vertex corresponding to the basis  $B$ . If  $\bar{c}^T = c^T - c_B^T B^{-1} A \geq 0$ , then  $x$  is an optimal solution.

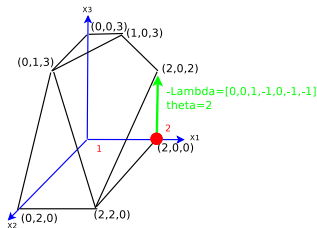
## Proof.

- Let  $y$  denote any feasible solution, i.e.  $Ay = b$  and  $y \geq 0$ .
- Then  $c^T y \geq c_B^T B^{-1} Ay = c_B^T B^{-1} b = c_B^T x_B = c^T x$ .
- In other words, any feasible solution  $y$  is not better than  $x$ .



## Simplex algorithm

# Key observations



$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$
$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- 1 What is a feasible solution? Any point in a polytope.
- 2 Where is the optimal solution? A vertex of the polytope. In other words, it is not necessary to care about the inner points.
- 3 How to obtain a vertex? Vertex corresponds to a basis of the matrix  $A$ , which can be easily calculated via Gaussian elimination.
- 4 If a vertex is not good, how to improve? Move to another vertex following an edge. The "moving" action can be accomplished via "pivoting" operation.
- 5 When shall we stop?  $\bar{c}_e \geq 0$  for all index  $e$  means that we have obtained an optimal  $x$ .

SIMPLEX( $A, b, c$ )

- 1:  $(B_I, A, b, c, z) = \text{INITIALIZE\_SIMPLEX}(A, b, c);$
- 2: //If the LP is feasible, a vertex  $x$  is returned with  $B_I$  storing the indices of vectors in the corresponding basis  $B$ ; otherwise, "infeasible" is reported.
- 3: **while** TRUE **do**
- 4:   **if** there is no index  $e$  ( $1 \leq e \leq n$ ) having  $c_e < 0$  **then**
- 5:      $x = \text{CALCULATEX}(B_I, A, b, c);$
- 6:     **return**  $(x, z);$
- 7:   **end if**;
- 8:   choose an index  $e$  having  $c_e < 0$  according to a certain rule;
- 9:   **for** each index  $i$  ( $1 \leq i \leq m$ ) **do**
- 10:     **if**  $a_{ie} > 0$  **then**
- 11:        $\theta_i = \frac{b_i}{a_{ie}};$
- 12:     **else**
- 13:        $\theta_i = \infty;$
- 14:     **end if**
- 15:   **end for**
- 16:   choose an index  $l$  that minimizes  $\theta_i$ ;
- 17:   **if**  $\theta_l = \infty$  **then**
- 18:     **return** "unbounded";
- 19:   **end if**
- 20:    $(B_I, A, b, c, z) = \text{PIVOT}(B_I, A, b, c, z, e, l);$
- 21: **end while**

CALCULATEX( $B_I, A, b, c$ )

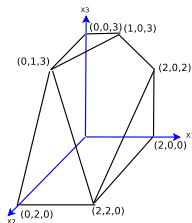
```
1: //assign non-basic variables with 0, and assign basic variables with
   corresponding  $b_i$ ;
2: for  $j = 1$  to  $n$  do
3:   if  $j \notin B_I$  then
4:      $x_j = 0$ ;
5:   else
6:     for  $i = 1$  to  $m$  do
7:       if  $a_{ij} = 1$  then
8:          $x_j = b_i$ ;
9:       end if
10:    end for
11:  end if
12: end for
13: return  $x$ ;
```



# An example

Standard form:

$$\begin{array}{llllllll} \min & -x_1 & - & 14x_2 & - & 6x_3 & & \\ s.t. & x_1 & + & x_2 & + & x_3 & \leq & 4 \\ & x_1 & & & & & \leq & 2 \\ & & & & & x_3 & \leq & 3 \\ & & & 3x_2 & + & x_3 & \leq & 6 \\ & x_1 & , & x_2 & , & x_3 & \geq & 0 \end{array}$$



Standard form:

$$\begin{array}{llllllll}
 \min & -x_1 & - & 14x_2 & - & 6x_3 & & \\
 s.t. & x_1 & + & x_2 & + & x_3 & \leq & 4 \\
 & x_1 & & & & & \leq & 2 \\
 & & & & & x_3 & \leq & 3 \\
 & & & 3x_2 & + & x_3 & \leq & 6 \\
 & x_1 & , & x_2 & , & x_3 & \geq & 0
 \end{array}$$

Slack form:

$$\begin{array}{llllllllllllll}
 \min & -x_1 & - & 14x_2 & - & 6x_3 & & & & & & & & \\
 s.t. & x_1 & + & x_2 & + & x_3 & + & x_4 & & & & & & = & 4 \\
 & x_1 & & & & & & & + & x_5 & & & & = & 2 \\
 & & & & & x_3 & & & & & + & x_6 & & = & 3 \\
 & & & 3x_2 & + & x_3 & & & & & & & + & x_7 & = & 6 \\
 & x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & , & x_6 & , & x_7 & \geq & 0
 \end{array}$$

# SIMPLEX algorithm maintains a simplex tabular

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	RHS
Basis	$\bar{c}_1=-1$	$\bar{c}_2=-14$	$\bar{c}_3=-6$	$\bar{c}_4=0$	$\bar{c}_5=0$	$\bar{c}_6=0$	$\bar{c}_7=0$	$-z = 0$
$x_4$	1	1	1	1	0	0	0	$x_{B_1} = b'_1 = 4$
$x_5$	1	0	0	0	1	0	0	$x_{B_2} = b'_2 = 2$
$x_6$	0	0	1	0	0	1	0	$x_{B_3} = b'_3 = 3$
$x_7$	0	3	1	0	0	0	1	$x_{B_4} = b'_4 = 6$

$$B^{-1}N$$

$$B^{-1}B = E$$

$$B^{-1}b$$

- Coefficient matrix:  $B^{-1}A$ . The basis forms a unit matrix, while the other part is  $B^{-1}N$ .

- The first row contains "checking number"

$$\bar{c}^T = c^T - c_B^T B^{-1}A \quad (\text{initial value: } c)$$

第一行也需要改，将单位阵那一块 ( $c_4 \sim c_7$ )

- The last column contains solution  $x_B = b' = B^{-1}b$  (initial value:  $b$ )

都改为0  
非基向量既是1

- The up-left item: objective value  $-z = c_B^T x_B = c_B^T B^{-1}b$  (initial value: 0)

# Why simplex tabular takes this form?

$$\begin{array}{c}
 \begin{array}{c|ccc|ccc}
 & \mathbf{c_B^T} & & & \mathbf{c_N^T} & & \\
 \hline
 0 & c_1 & c_2 & \cdots & c_m & \cdots & c_n \\
 \hline
 b_1 & a_{11} & a_{12} & \cdots & a_{1m} & \cdots & a_{1n} \\
 b_2 & a_{21} & a_{22} & \cdots & a_{2m} & \cdots & a_{2n} \\
 \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
 b_m & a_{m1} & a_{m2} & \cdots & a_{mm} & \cdots & a_{mn} \\
 \hline
 \mathbf{b} & \mathbf{B} & & & \mathbf{N} & & 
 \end{array}
 & \xRightarrow[\text{Row Operation}]{\mathbf{B}^{-1} \times} & 
 \begin{array}{c|ccc|ccc}
 & \mathbf{\bar{c}^T} & & & & & \\
 \hline
 & -\mathbf{c_B^T B^{-1} b} & 0 & 0 & \cdots & 0 & \mathbf{c_N^T - c_B^T B^{-1} N} \\
 \hline
 & \mathbf{B^{-1} b} & 1 & 0 & \cdots & 0 & \\
 & & 0 & 1 & \cdots & 0 & \\
 & & \vdots & \vdots & \ddots & \vdots & \\
 & & 0 & 0 & \cdots & 1 & \\
 \hline
 & \mathbf{B^{-1} B} & & & & & \mathbf{B^{-1} N}
 \end{array}
 \end{array}$$

- Coefficient matrix:  $B^{-1}A$ . **The basis always forms a unit matrix.**

Why?

- This way, for any non-basis column vector  $a_e$ , the  $e$ -th column stores the coefficients  $[\lambda_1, \lambda_2, \dots, \lambda_m]^T$ , i.e.  $a_e$  is decomposed as  $a_e = \lambda_1 a_1 + \dots + \lambda_m a_m$ .
- The “pivoting” operation is accomplished by Gaussian row operations on **all rows**, including the first row  $\bar{c}^T$ , and the column  $b'$ . Why?
  - ① The row operation make the entries in  $c_B^T$  be 0, thus the first row contains “checking number”  $\bar{c}^T = c^T - c_B^T B^{-1} A$  (initial value:  $c$ )
  - ② The up-left item shows the objective value  $-z = 0 - c_B^T B^{-1} b = -c_B^T B^{-1} b$  (initial value: 0)

# Pivoting I

$$\left( \begin{array}{c|cccccc} 0 & \dots & 0 & \dots & c_e & \dots \\ b_1 & \dots & 0 & \dots & a_{1e} & \dots \\ b_2 & \dots & 0 & \dots & a_{2e} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_l & \dots & 1 & \dots & a_{le} & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_m & \dots & 0 & \dots & a_{me} & \dots \end{array} \right) \Rightarrow \left( \begin{array}{c|cccccc} -\frac{a_{me}}{a_{le}} b_l & \dots & -\frac{c_e}{a_{le}} & \dots & 0 & \dots \\ b_1 - \frac{a_{1e}}{a_{le}} b_l & \dots & -\frac{a_{1e}}{a_{le}} & \dots & 0 & \dots \\ b_2 - \frac{a_{2e}}{a_{le}} b_l & \dots & -\frac{a_{2e}}{a_{le}} & \dots & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \frac{1}{a_{le}} b_l & \dots & \frac{1}{a_{le}} & \dots & 1 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ b_m - \frac{a_{me}}{a_{le}} b_l & \dots & -\frac{a_{me}}{a_{le}} & \dots & 0 & \dots \end{array} \right)$$

PIVOT( $B_I, A, b, c, z, e, l$ )

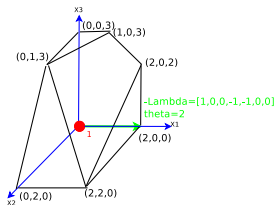
- 1: //Scaling the  $l$ -th line
- 2:  $b_l = \frac{b_l}{a_{le}};$
- 3: **for**  $j = 1$  to  $n$  **do**
- 4:      $a_{lj} = \frac{a_{lj}}{a_{le}};$
- 5: **end for**
- 6: //All other lines minus the  $l$ -th line
- 7: **for**  $i = 1$  to  $m$  but  $i \neq l$  **do**
- 8:      $b_i = b_i - a_{ie} \times b_l;$

# Pivoting II

```
9:   for  $j = 1$  to  $n$  do
10:      $a_{ij} = a_{ij} - a_{ie} \times a_{lj}$ ;
11:   end for
12: end for
13: //The first line minuses the  $l$ -th line
14:  $z = z - b_l \times c_e$ ;
15: for  $j = 1$  to  $n$  do
16:    $c_j = c_j - c_e \times a_{lj}$ ;
17: end for
18: //Calculating  $x$ 
19:  $B_I = B_I - \{l\} \cup \{e\}$ ;
20: return  $(B_I, A, b, c, z)$ ;
```

# Step 1

蓝色始终是基向量，  
非基向量用基向量表示

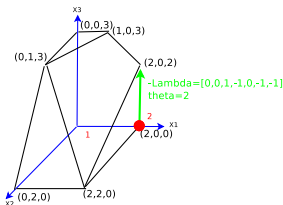


第一行也要做高斯行变化

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	RHS
Basis	$\bar{c}_1 = -1$	$\bar{c}_2 = -14$	$\bar{c}_3 = -6$	$\bar{c}_4 = 0$	$\bar{c}_5 = 0$	$\bar{c}_6 = 0$	$\bar{c}_7 = 0$	$-z = 0$
$x_4$	1	1	1	1	0	0	0	4
$x_5$	1	0	0	0	1	0	0	2
$x_6$	0	0	1	0	0	1	0	3
$x_7$	0	3	1	0	0	0	1	6

- Basis (in blue):  $B = \{a_4, a_5, a_6, a_7\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (0, 0, 0, 4, 2, 3, 6)$ . (Hint: basis variables  $x_4, x_5, x_6, x_7$  take value of  $b'_1, b'_2, b'_3, b'_4$ , respectively. )
- Pivoting: choose  $a_1$  to enter basis since  $\bar{c}_1 = -1 < 0$ ; choose  $a_5$  to exit since  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_2}{\lambda_2} = 2$ . 根据 $\theta$ 选择谁出基
- Here, the corresponding  $\lambda$  is stored in the 1-st column (Why? the basis  $B$  forms an identity matrix.)

# Step 2

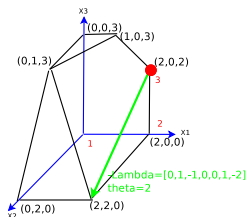


	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	RHS
Basis	$\bar{c}_1=0$	$\bar{c}_2=-14$	$\bar{c}_3=-6$	$\bar{c}_4=0$	$\bar{c}_5=1$	$\bar{c}_6=0$	$\bar{c}_7=0$	$-z=2$
$x_4$	0	1	1	1	-1	0	0	2
$x_1$	1	0	0	0	1	0	0	2
$x_6$	0	0	1	0	0	1	0	3
$x_7$	0	3	1	0	0	0	1	6

- Basis (in blue):  $B = \{a_1, a_4, a_6, a_7\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (2, 0, 0, 2, 0, 3, 6)$ . (Hint: basis variables  $x_1, x_4, x_6, x_7$  take value of  $b'_2, b'_1, b'_3, b'_4$ , respectively. )
- Pivoting: choose  $a_3$  to enter basis since  $\bar{c}_3 = -6 < 0$ ; choose  $a_4$  to exit since  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_1}{\lambda_1} = 2$ .
- Here, the corresponding  $\lambda$  is stored in the 3-rd column (Why? the basis  $B$  forms an identity matrix.)



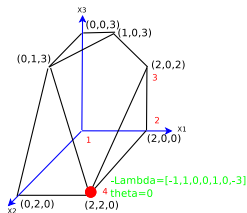
# Step 3



	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	RHS
Basis	$\bar{c}_1 = 0$	$\bar{c}_2 = -8$	$\bar{c}_3 = 0$	$\bar{c}_4 = 6$	$\bar{c}_5 = -5$	$\bar{c}_6 = 0$	$\bar{c}_7 = 0$	$-z = 14$
$x_3$	0	1	1	1	-1	0	0	2
$x_1$	1	0	0	0	1	0	0	2
$x_6$	0	-1	0	-1	1	1	0	1
$x_7$	0	2	0	-1	1	0	1	4

- Basis (in blue):  $B = \{a_1, a_3, a_6, a_7\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (2, 0, 2, 0, 0, 1, 4)$ . (Hint: basis variables  $x_3, x_1, x_6, x_7$  take value of  $b'_1, b'_2, b'_3, b'_4$ , respectively.)
- Pivoting: choose  $a_2$  to enter basis since  $\bar{c}_2 = -8 < 0$ ; choose  $a_3$  to exit since  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_1}{\lambda_1} = 2$ .
- Here, the corresponding  $\lambda$  is stored in the 2-nd column (Why? the basis  $B$  forms an identity matrix.)

## Step 4



	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	RHS
Basis	$\overline{c_1} = 0$	$\overline{c_2}=0$	$\overline{c_3}=8$	$\overline{c_4}=14$	$\overline{c_5}=-13$	$\overline{c_6}=0$	$\overline{c_7}=0$	$-z = 30$
$x_2$	0	1	1	1	-1	0	0	2
$x_1$	1	0	0	0	1	0	0	2
$x_6$	0	0	1	0	0	1	0	3
$x_7$	0	0	-2	-3	3	0	1	0

- Basis (in blue):  $B = \{a_1, a_2, a_6, a_7\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (2, 2, 0, 0, 3, 0)$ . (Hint: basis variables  $x_2, x_1, x_6, x_7$  take value of  $b'_1, b'_2, b'_3, b'_4$ , respectively. )
- Pivoting: choose  $a_5$  to enter basis since  $\bar{c}_5 = -13 < 0$ ; choose  $a_7$  to exit since  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_4}{\lambda_4} = 0$ . Note:  $\theta = 0 \Rightarrow$  same vertex (called “degeneracy”).
- Here, the corresponding  $\lambda$  is stored in the 5-th column (Why? the basis  $B$  form

# Degeneracy might lead to cycle

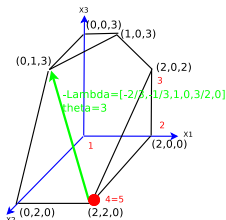
- Generally speaking, two different basis correspond to different vertices.
- However, redundant constraints can lead to degeneracy, i.e. the simplex algorithm is stuck at a vertex even after a “pivoting” operation.
- Sometimes degeneracy can lead to “cycling”: If a sequence of pivots starting from a vertex ends up at the exact same vertex, then we refer to this as cycling. If the simplex method cycles, it can cycle forever.

# How to escape from a cycle?

- Cycling is theoretically possible, but extremely rare. It is avoidable through the following three ways:
  - 1 Perturbation: Perturb the input  $A, b, c$  slightly to make any two solutions differ in objective values;
  - 2 Breaking ties lexicographically;
  - 3 Breaking ties by choosing variables with smallest index, called Bland's indexing rule:
    - choose  $a_e$  to enter:  $e = \min\{j : \bar{c}_j \leq 0, 1 \leq j \leq n\}$ .
    - choose  $a_l$  to exit: choose the smallest  $l$  to break ties.

(a demo)

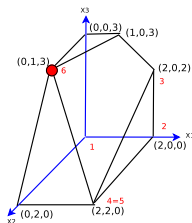
# Step 5



	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	RHS
Basis	$\bar{c}_1 = 0$	$\bar{c}_2 = 0$	$\bar{c}_3 = -2/3$	$\bar{c}_4 = 1$	$\bar{c}_5 = 0$	$\bar{c}_6 = 0$	$\bar{c}_7 = \frac{13}{3}$	$-z = 30$
$x_2$	0	1	$-\frac{1}{3}$	1	0	0	$\frac{1}{3}$	2
$x_1$	1	0	$-\frac{1}{3}$	0	0	0	$-\frac{1}{3}$	2
$x_6$	0	0	1	0	0	1	0	3
$x_5$	0	0	$-\frac{1}{3}$	-1	1	0	$\frac{1}{3}$	0

- Basis (in blue):  $B = \{a_1, a_2, a_5, a_6\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (2, 2, 0, 0, 0, 3, 0)$ . (Hint: basis variables  $x_2, x_1, x_6, x_5$  take value of  $b'_1, b'_2, b'_3, b'_4$ , respectively. )
- Pivoting: choose  $a_3$  to enter basis since  $\bar{c}_3 = -2/3 < 0$ ; choose  $a_1$  to exit since  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_2}{\lambda_2} = 3$ .
- Here, the corresponding  $\lambda$  is stored in the 3-rd column (Why? the basis  $B$  forms

# Step 6



	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$x_7$	RHS
Basis	$\bar{c}_1 = 1$	$\bar{c}_2 = 0$	$\bar{c}_3 = 0$	$\bar{c}_4 = 2$	$\bar{c}_5 = 0$	$\bar{c}_6 = 0$	$\bar{c}_7 = 4$	$-z = 32$
$x_2$	$-\frac{1}{2}$	1	0	$-\frac{1}{2}$	0	0	$\frac{1}{2}$	1
$x_3$	$-\frac{2}{3}$	0	1	$-\frac{2}{3}$	0	0	$-\frac{1}{3}$	3
$x_6$	$-\frac{2}{3}$	0	0	$-\frac{2}{3}$	0	1	$\frac{1}{3}$	0
$x_5$	1	0	0	0	1	0	0	2

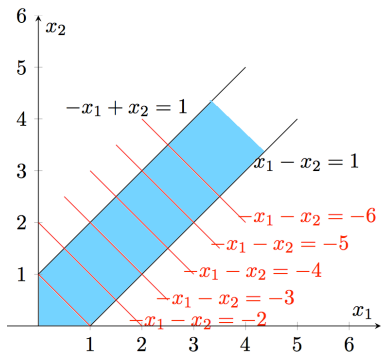
- Basis (in blue):  $B = \{a_2, a_3, a_5, a_6\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (0, 1, 3, 0, 2, 0, 0)$ . (Hint: basis variables  $x_2, x_3, x_6, x_5$  take value of  $b'_1, b'_2, b'_3, b'_4$ , respectively. )
- Pivoting: all  $\bar{c}_j \geq 0$ , thus optimal solution found.

An example with unbounded objective value

# An example with unbounded objective value

Standard form:

$$\begin{array}{llllll} \min & -x_1 & - & x_2 & & \\ \text{s.t.} & x_1 & - & x_2 & \leq & 1 \\ & -x_1 & + & x_2 & \leq & 1 \\ & x_1 & , & x_2 & \geq & 0 \end{array}$$





# An example with unbounded objective value

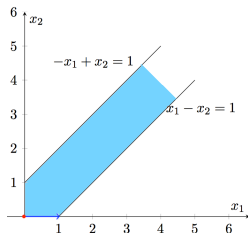
Standard form:

$$\begin{array}{llllll} \min & -x_1 & - & x_2 & & \\ s.t. & x_1 & - & x_2 & \leq & 1 \\ & -x_1 & + & x_2 & \leq & 1 \\ & x_1 & , & x_2 & \geq & 0 \end{array}$$

Slack form:

$$\begin{array}{llllllllll} \min & -x_1 & - & x_2 & & & & & & \\ s.t. & x_1 & - & x_2 & + & x_3 & & & = & 1 \\ & -x_1 & + & x_2 & & & + & x_4 & = & 1 \\ & x_1 & , & x_2 & , & x_3 & , & x_4 & \geq & 0 \end{array}$$

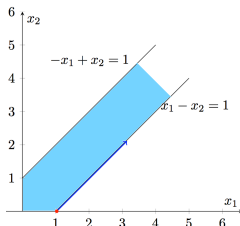
# Step 1



	$x_1$	$x_2$	$x_3$	$x_4$	RHS
Basis	$\bar{c}_1 = -1$	$\bar{c}_2 = -1$	$\bar{c}_3 = 0$	$\bar{c}_4 = 0$	$-z = 0$
$x_3$	1	-1	1	0	1
$x_4$	-1	1	0	1	1

- Basis (in blue):  $B = \{a_3, a_4\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (0, 0, 1, 1)$ .
- Pivoting: choose  $a_1$  to enter basis since  $c_1 = -1 < 0$ ; choose  $a_3$  to exit since  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{b_i}{\lambda_i} = \frac{b_1}{\lambda_1} = 1$ .
- Here, the corresponding  $\lambda$  is stored in the 1-st column (Why? the basis  $B$  forms an identity matrix.)

# Step 2



	$x_1$	$x_2$	$x_3$	$x_4$	RHS
Basis	$\bar{c}_1=0$	$\bar{c}_2=-2$	$\bar{c}_3=1$	$\bar{c}_4=0$	$-z=1$
$x_1$	1	-1	1	0	1
$x_4$	0	0	1	1	2

- Basis (in blue):  $B = \{a_1, a_4\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (1, 0, 0, 2)$ .
- Pivoting: choose  $a_2$  to enter basis since  $c_2 = -2 < 0$ ; while  $\lambda_{2i} \leq 0$  for all  $i$ , then  $\theta$  can take a value as large as possible. That is, the optimal solution of this problem is unbounded.

The last question: how to get an initial feasible solution?  
or how to solve  $Ax = b, x \geq 0$ ?

加上约束  $x \geq 0$ , 必须用线性规划

# Initial feasible solution: solving an auxiliary linear program

## Theorem

Suppose we attempt to find an initial solution to linear program  $L$ :

$$\begin{array}{llllllllll} \min & c_1x_1 & + & c_2x_2 & + & \dots & + & c_nx_n & & \\ \text{s.t.} & a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & \leq & b_1 \\ & a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & \leq & b_2 \\ & & & & & \dots & & & & \\ & a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & \leq & b_m \\ & x_1 & , & x_2 & , & \dots & , & x_n & \geq & 0 \end{array}$$

Let's construct an **auxiliary** linear program  $L_{aux}$  as follows:

$$\begin{array}{llllllllllll} \min & & & & & & & & & & x_0 & & \\ \text{s.t.} & a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & - & x_0 & \leq & b_1 \\ & a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & - & x_0 & \leq & b_2 \\ & & & & & \dots & & & & & & \\ & a_{m1}x_1 & + & a_{m2}x_2 & + & \dots & + & a_{mn}x_n & - & x_0 & \leq & b_m \\ & x_1 & , & x_2 & , & \dots & , & x_n & , & x_0 & \geq & 0 \end{array}$$

Then  $L$  is feasible i.f.f. the optimal objective value of  $L_{aux}$  is 0.

## Proof.

- Suppose that  $L$  has a feasible solution  $(x_1, x_2, \dots, x_n) = (\overline{x_1}, \overline{x_2}, \dots, \overline{x_n})$ ;
- Then expand this solution by combining with  $\overline{x_0} = 0$ , i.e.  $(x_0, x_1, x_2, \dots, x_n) = (0, \overline{x_1}, \overline{x_2}, \dots, \overline{x_n})$ ;
- The expanded solution is a feasible solution to  $L_{aux}$ ;
- And the objective value is 0; in other words, the solution is an optimal solution.
- Conversely,  $L_{aux}$  has an optimal objective value of 0 means  $L$  has a feasible solution.



Intuition: suppose  $L$  is infeasible, i.e. for any assignment  $(x_1, x_2, \dots, x_n)$ , at least one constraint is not satisfied, say  $a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n > b_i$ . Then we have  $x_0 \geq a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n - b_i > 0$ .

# How to solve $L_{aux}$ ?

- Let consider the slack form of  $L_{aux}$ :

$$\begin{array}{llllllllll} \min & & & & & & x_0 & & & & \\ s.t. & a_{11}x_1 & + \dots & + a_{1n}x_n & - x_0 & + s_1 & & & & = b_1 \\ & a_{21}x_1 & + \dots & + a_{2n}x_n & - x_0 & & + s_2 & & & = b_2 \\ & & \dots & & & & & & & \\ & a_{m1}x_1 & + \dots & + a_{mn}x_n & - x_0 & & & + s_m & & = b_m \\ & x_1 & , \dots , & x_n , & x_0 , & s_1 , & s_2 , & \dots s_m & & \geq 0 \end{array}$$

- $L_{aux}$  has an advantage that **initial feasible solution can be easily acquired.**

## Case 1: $b_i > 0$ for all $i$

- Consider the slack form of  $L_{aux}$ :

$$\begin{array}{llllllllll} \min & & & & & & & & & & \\ s.t. & a_{11}x_1 & + \dots & + a_{1n}x_n & \textcolor{red}{-x_0} & \textcolor{blue}{+s_1} & & & & = b_1 \\ & a_{21}x_1 & + \dots & + a_{2n}x_n & \textcolor{red}{-x_0} & & \textcolor{blue}{+s_2} & & & = b_2 \\ & & \dots & & & & & & & \\ & a_{m1}x_1 & + \dots & + a_{mn}x_n & \textcolor{red}{-x_0} & & & \textcolor{blue}{+s_m} & & = b_m \\ & x_1 & , \dots , & x_n , & \textcolor{red}{x_0} , & \textcolor{blue}{s_1} , & \textcolor{blue}{s_2} , & \dots \textcolor{blue}{s_m} & \geq 0 \end{array}$$

- If  $b_i > 0$  for all  $i$ :  
 $(x_1, \dots, x_n, s_1, s_2, \dots, s_m) = (0, \dots, 0, b_1, b_2, \dots, b_m)$  is an initial feasible solution.



## Case 2: there exist some negative $b_i$

- Consider the slack form of  $L_{aux}$ :

$$\begin{array}{llllllllll} \min & & & & & & x_0 & & & & \\ s.t. & a_{11}x_1 & + \dots & + a_{1n}x_n & -x_0 & +s_1 & & & & = b_1 \\ & a_{21}x_1 & + \dots & + a_{2n}x_n & -x_0 & & +s_2 & & & = b_2 \\ & & & \dots & & & & & & \\ & a_{m1}x_1 & + \dots & + a_{mn}x_n & -x_0 & & & +s_m & & = b_m \\ & x_1 & , \dots , & x_n , & x_0 , & s_1 , & s_2 , & \dots s_m & & \geq 0 \end{array}$$

- If there are some negative  $b_i$ , then  $(x_1, \dots, x_n, s_1, s_2, \dots, s_m) = (0, \dots, 0, b_1, b_2, \dots, b_m)$  is not a feasible solution.
- But a feasible solution can be easily obtained by performing **only one step of pivoting operation**: let  $l$  be the index of the minimum  $b_i$ . All other constraints minus the  $l$ -th constraint, and multiply the  $l$ -th row by -1. Then all new  $b'_i$  are now positive.

## INITIALIZESIMPLEX( $A, b, c$ )

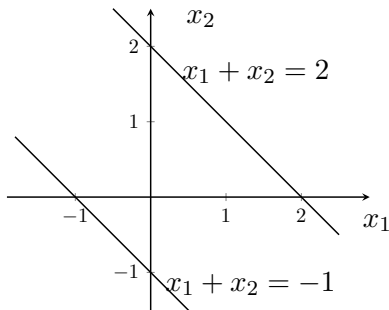
- 1: let  $l$  be the index of the minimum  $b_i$ ;
- 2: set  $B_I$  to include the indices of slack variables;
- 3: **if**  $b_l \geq 0$  **then**
- 4:     **return** ( $B_I, A, b, c, 0$ ) ;
- 5: **end if**
- 6: construct  $L_{aux}$  by adding  $-x_0$  to each constraint, and using  $x_0$  as the objective function;
- 7: let  $(A, b, c)$  be the resulting slack form for  $L_{aux}$ ;
- 8: **//perform one step of pivot to make all  $b_i$  positive;**
- 9:  $(B_I, A, b, c, z) = \text{PIVOT}(B_I, A, b, c, z, l, 0)$ ;
- 10: iterate the WHILE loop of SIMPLEX algorithm until an optimal solution to  $L_{aux}$  is found;
- 11: **if** the optimal objective value to  $L_{aux}$  is 0 **then**
- 12:     return the final slack form with  $x_0$  removed, and the original objective function of  $L$  restored;
- 13: **else**
- 14:     **return** "infeasible";
- 15: **end if**

INITIALIZESIMPLEX: an example with no feasible solution

# An example with no feasible solution

LP  $L$ :

$$\begin{array}{llllll} \min & x_1 & + & 2x_2 & & \\ s.t. & x_1 & + & x_2 & \geq & 2 \\ & x_1 & + & x_2 & \leq & -1 \\ & x_1 & , & x_2 & \geq & 0 \end{array}$$



- LP  $L$ :

$$\begin{array}{llllll} \min & x_1 & + & 2x_2 & & \\ s.t. & -x_1 & - & x_2 & \leq & -2 \\ & x_1 & + & x_2 & \leq & -1 \\ & x_1 & , & x_2 & \geq & 0 \end{array}$$

- LP  $L_{aux}$ :

$$\begin{array}{llllllll} \min & & & & x_0 & & & \\ s.t. & -x_1 & - & x_2 & -x_0 & \leq & -2 \\ & x_1 & + & x_2 & -x_0 & \leq & -1 \\ & x_1 & , & x_2 & , x_0 & \geq & 0 \end{array}$$

- LP  $L_{aux}$  (slack form):

$$\begin{array}{llllllllll} \min & & & & x_0 & & & & & \\ s.t. & -x_1 & - & x_2 & -x_0 & +x_3 & = & -2 \\ & x_1 & + & x_2 & -x_0 & & +x_4 & = & -1 \\ & x_1 & , & x_2 & , x_0 & , x_3 & , x_4 & \geq & 0 \end{array}$$

# Step 1

	$x_1$	$x_2$	$x_0$	$x_3$	$x_4$	RHS
Basis	$\overline{c_1}=0$	$\overline{c_2}=0$	$\overline{c_0}=1$	$\overline{c_3}=0$	$\overline{c_4}=0$	$L_{aux} : -z = 0$
$x_3$	-1	-1	-1	1	0	-2
$x_4$	1	1	-1	0	1	-1

- Basis (in blue):  $B = \{a_3, a_4\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (0, 0, 0, -2, -1)$  is infeasible.
- Pivoting: all rows minus the  $l$ -th row, and the  $l$ -th row multiply  $-1$ . This way, all new  $b_i$  will be positive.

# Step 2

	$x_1$	$x_2$	$x_0$	$x_3$	$x_4$	RHS
Basis	$\overline{c_1} = -1$	$\overline{c_2} = -1$	$\overline{c_0} = 0$	$\overline{c_3} = 1$	$\overline{c_4} = 0$	$L_{aux} : -z = -2$
$x_0$	1	1	1	-1	0	2
$x_4$	2	2	0	-1	1	1

- Basis (in blue):  $B = \{a_0, a_4\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (0, 0, 2, 0, 1)$  is feasible now. Thus, we can start from this initial solution to improve step by step.
- Pivoting: choose  $a_2$  to enter basis since  $\overline{c_2} = -1 < 0$ ; choose  $a_4$  to exit since  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_2}{\lambda_2} = \frac{1}{2}$ .
- Here, the corresponding  $\lambda$  is stored in the 2-nd column (Why? the basis  $B$  forms an identity matrix.)

# Step 3

	$x_1$	$x_2$	$x_0$	$x_3$	$x_4$	RHS
Basis	$\overline{c_1} = 0$	$\overline{c_2} = 0$	$\overline{c_0} = 0$	$\overline{c_3} = \frac{1}{2}$	$\overline{c_4} = \frac{1}{2}$	$L_{aux} : -z = -\frac{3}{2}$
$x_0$	0	0	1	$-\frac{3}{2}$	$-\frac{1}{2}$	$\frac{3}{2}$
$x_2$	1	1	0	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

- Basis (in blue):  $B = \{a_2, a_0\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (0, \frac{1}{2}, \frac{3}{2}, 0, 0)$  is feasible.
- Optimal solution found since  $\overline{c_j} > 0$  for all  $j$ .
- The objective value of  $L_{aux}$  is  $-\frac{3}{2}$ , indicating that the original linear program  $L$  is infeasible.

无解

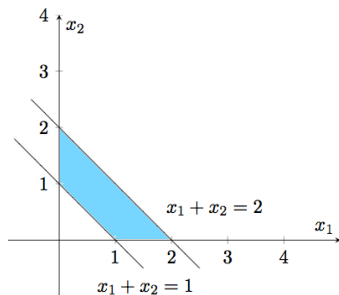


INITIALIZESIMPLEX: an example with a feasible solution

# An example with a feasible solution

LP  $L$ :

$$\begin{array}{llllll} \min & x_1 & + & 2x_2 & & \\ \text{s.t.} & x_1 & + & x_2 & \leq & 2 \\ & x_1 & + & x_2 & \geq & 1 \\ & x_1 & , & x_2 & \geq & 0 \end{array}$$



- LP  $L$ :

$$\begin{array}{llllll} \min & x_1 & + & 2x_2 & & \\ s.t. & -x_1 & - & x_2 & \leq & -1 \\ & x_1 & + & x_2 & \leq & 2 \\ & x_1 & , & x_2 & \geq & 0 \end{array}$$

- LP  $L_{aux}$ :

$$\begin{array}{llllllll} \min & & & & x_0 & & & \\ s.t. & -x_1 & - & x_2 & -x_0 & \leq & -1 \\ & x_1 & + & x_2 & -x_0 & \leq & 2 \\ & x_1 & , & x_2 & , x_0 & \geq & 0 \end{array}$$

- LP  $L_{aux}$  (slack form):

$$\begin{array}{llllllllll} \min & & & & x_0 & & & & & \\ s.t. & -x_1 & - & x_2 & -x_0 & +x_3 & = & -1 \\ & x_1 & + & x_2 & -x_0 & & +x_4 & = & 2 \\ & x_1 & , & x_2 & , x_0 & , x_3 & , x_4 & \geq & 0 \end{array}$$

# Step 1

	$x_1$	$x_2$	$x_0$	$x_3$	$x_4$	RHS
Basis	$\overline{c_1}=0$	$\overline{c_2}=0$	$\overline{c_0}=1$	$\overline{c_3}=0$	$\overline{c_4}=0$	$L_{aux} : -z = 0$
$x_3$	-1	-1	-1	1	0	-1
$x_4$	1	1	-1	0	1	2

- Basis (in blue):  $B = \{a_3, a_4\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (0, 0, 0, -1, 2)$  is infeasible.
- Pivoting: all rows minus the  $l$ -th row, and the  $l$ -th row multiply  $-1$ . This way, all new  $b_i$  will be positive.

# Step 2

	$x_1$	$x_2$	$x_0$	$x_3$	$x_4$	RHS
Basis	$\overline{c_1} = -1$	$\overline{c_2} = -1$	$\overline{c_0} = 0$	$\overline{c_3} = 1$	$\overline{c_4} = 0$	$L_{aux} : -z = -1$
$x_0$	1	1	1	-1	0	1
$x_4$	2	2	0	-1	1	3

- Basis (in blue):  $B = \{a_0, a_4\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (0, 0, 1, 0, 3)$  is feasible.
- Pivoting: choose  $a_2$  to enter basis since  $\overline{c_2} = -1 < 0$ ; choose  $a_0$  to exit since  $\theta = \min_{a_i \in B, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_1}{\lambda_1} = 1$ .
- Here, the corresponding  $\lambda$  is stored in the 1-st column (Why? the basis  $B$  forms an identity matrix.)

# Step 3

	$x_1$	$x_2$	$x_0$	$x_3$	$x_4$	RHS
Basis	$\overline{c_1}=0$	$\overline{c_2}=0$	$\overline{c_0}=0$	$\overline{c_3}=1$	$\overline{c_4}=0$	$L_{aux} : -z = 0$
$x_2$	1	1	1	-1	0	1
$x_4$	0	0	-2	1	1	1

- Basis (in blue):  $B = \{a_2, a_4\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (0, 1, 0, 0, 1)$  is feasible.
- Optimal solution found since  $\overline{c_j} > 0$  for all  $j$ .
- The optimal objective value of  $L_{aux}$  is 0, meaning that the original linear program  $L$  has a feasible solution.

# Returning an initial feasible solution to $L$ by removing $x_0$

LP  $L$  (slack form):

$$\begin{array}{llllllll} \min & x_1 & + & 2x_2 & & & & \\ s.t. & -x_1 & - & x_2 & +x_3 & & = & -1 \\ & x_1 & + & x_2 & & +x_4 & = & 2 \\ & x_1 & , & x_2 & , x_3 & , x_4 & \geq & 0 \end{array}$$

	$x_1$	$x_2$	$x_3$	$x_4$	RHS
Basis	$\overline{c}_1=1$	$\overline{c}_2=2$	$\overline{c}_3=0$	$\overline{c}_4=0$	$L: -z = 0$
$x_2$	1	1	-1	0	1
$x_4$	0	0	1	1	1

Remove  $x_0$  and perform Gaussian row operation on the first row, we obtain the initial SIMPLEX table for  $L$ :

	$x_1$	$x_2$	$x_3$	$x_4$	RHS
Basis	$\overline{c}_1=-1$	$\overline{c}_2=0$	$\overline{c}_3=3$	$\overline{c}_4=0$	$L: -z = -2$
$x_2$	1	1	-1	0	1
$x_4$	0	0	1	1	1

- Basis (in blue):  $B = \{a_2, a_4\}$
- Solution:  $x = \begin{bmatrix} B^{-1}b \\ 0 \end{bmatrix} = (0, 1, 0, 1)$  is an **initial feasible solution** to the original linear program  $L$ .

How fast is the SIMPLEX method?



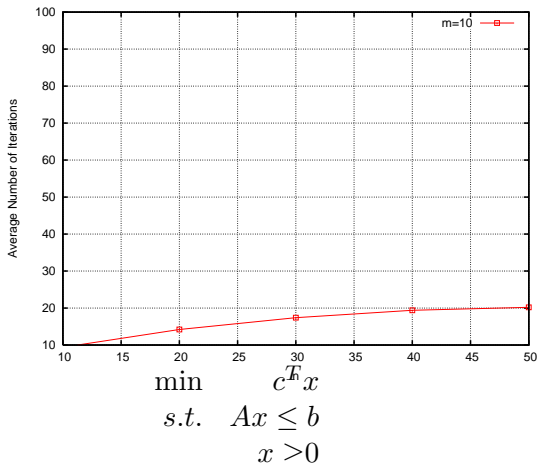
# Performance of SIMPLEX algorithm

- ① In practice, the typical number of pivoting operation is proportional to  $m$  (with the proportionality constant in the range suggested by Dantzig) and only increases very slowly with  $n$  (it is sometimes said that, for a fixed  $m$ , the number of iterations is proportional to  $\log n$ ).
- ② The complexity of simplex algorithm is expected as:  $O(m^2n)$  due to  $O(m)$  pivoting operations.
- ③ For sparse matrix:  $O(Km^\alpha nd^{0.33})$ , where  $K$  is a constant,  $1.25 \leq \alpha \leq 2.5$ ,  $d$  is the ratio of non-zero entries of matrix  $A$ .

# How does the iteration number change with $n$ ?

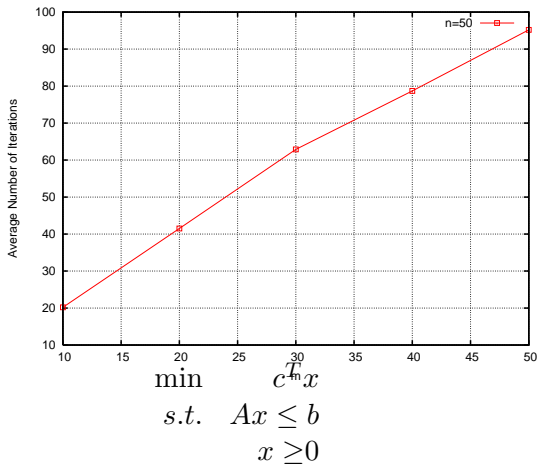
$(m + \log n) * m * n$

第一项: pivoting



Here,  $m$  denotes the number of constraints, and  $n$  denotes the number of variables.

# How does the iteration number change with $m$ ?



Here,  $m$  denotes the number of constraints, and  $n$  denotes the number of variables.

Unfortunately, SIMPLEX is not a polynomial-time algorithm

# A counter-example given by V. Klee and G. L. Minty [1972]

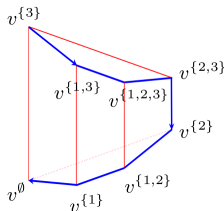
$$\begin{array}{llll} \max & & x_n & \\ s.t. & \delta & \leq & x_i \leq 1 \quad \text{for } i = 1..n \\ & \delta x_{i-1} & \leq & x_i \leq 1 - \delta x_{i-1} \quad \text{for } i = 2..n \\ & & x_i & \geq 0 \quad \text{for } i = 1..n \end{array}$$

Simplex algorithm might visit all the  $2^n$  vertices.

# Klee-Minty cube: $n = 3$ , $\delta = \frac{1}{4}$ .

$$\begin{array}{llll} \min & & x_3 & \\ s.t. & \frac{1}{4} & \leq x_1 & \leq 1 \\ & \frac{1}{4}x_1 & \leq x_2 & \leq 1 - \frac{1}{4}x_1 \\ & \frac{1}{4}x_2 & \leq x_3 & \leq 1 - \frac{1}{4}x_2 \end{array}$$

There is a path visiting  $2^n$  vertices and each edge makes  $x_3$  decrease.



Reference: A simpler and tighter redundant Klee-Minty construction (by E. Nematollahi and T. Terlaky)

## 8 Vertices of Klee-Minty cube

①  $[\frac{1}{4}, \frac{1}{16}, \frac{1}{64}]$

②  $[\frac{1}{4}, \frac{1}{16}, \frac{63}{64}]$

③  $[\frac{1}{4}, \frac{15}{16}, \frac{15}{64}]$

④  $[\frac{1}{4}, \frac{15}{16}, \frac{49}{64}]$

⑤  $[1, \frac{1}{4}, \frac{1}{16}]$

⑥  $[1, \frac{1}{4}, \frac{15}{16}]$

⑦  $[1, \frac{3}{4}, \frac{3}{16}]$

⑧  $[1, \frac{3}{4}, \frac{13}{16}]$

## Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time?



# Smoothed analysis of algorithms [Daniel A. Spielman, Shang-Hua Teng, 2001]



- Despite its favorable complexity, the performance of the ellipsoid method in practice was extremely slower than the simplex method, leading to a puzzling and deeply unsatisfying anomaly in which an exponential-time algorithm was substantially and consistently faster than a polynomial-time algorithm.
- Spielman and Teng showed that the simplex algorithm has polynomial smoothed complexity.

椭球法

# Smoothed analysis of algorithms

- Average-case analysis was first introduced to overcome the limitations of worst-case analysis, however the difficulty is saying what an average case is. The actual inputs and distribution of inputs may be different in practice from the assumptions made during the analysis.
- Smoothed analysis is a hybrid of worst-case and average-case analyses that inherits advantages of both, by measuring the expected performance of algorithms under slight random perturbations of worst-case inputs.
- The performance of an algorithm is measured in terms of both the input size, and the magnitude of the perturbations.
- If the smoothed complexity of an algorithm is low, then it is unlikely that the algorithm will take long time to solve practical instances whose data are subject to slight noises and imprecisions.

(a demo)