CS711008Z Algorithm Design and Analysis

Lecture 8. Algorithm design technique: Linear programming [1]

Dongbo Bu

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

[1] The slides are made based on Chapter 29 of Introduction to algorithms, Combinatorial optimization algorithm and complexity by C. H. Papadimitriou and K. Steiglitz.

Outline

- 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

DIET problem

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:

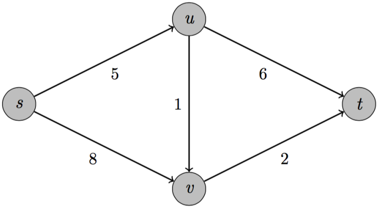
$$
\begin{array}{lrcrcrcrll}
\min & 3x_1 & + & 9x_2 & + & 20x_3 & + & 19x_4 & & \text{money} \\
s.t. & 110x_1 & + & 160x_2 & + & 420x_3 & + & 260x_4 & \geq 2000 & \text{energy} \\
 & 4x_1 & + & 8x_2 & + & 4x_3 & + & 14x_4 & \geq 55 & \text{protein} \\
 & 2x_1 & + & 285x_2 & + & 22x_3 & + & 80x_4 & \geq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

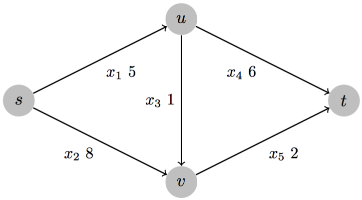
Practical problem 2: MAXIMUM FLOW

MAXIMUM FLOW problem

INPUT:
A directed graph $G =< V, E >$. 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.





LP Formulation:

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

Practical problem 3: Minimum Cost Flow problem

INPUT:
A directed graph $G =< V, E >$. 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.

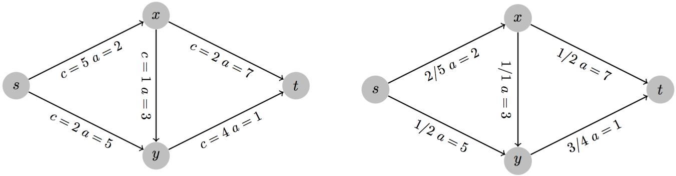
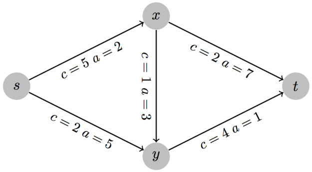


Figure: Cost$=2 \times 2 + 1 \times 7 + 1 \times 3 + 1 \times 5 + 3 \times 1 = 18$

Linear programming formulation

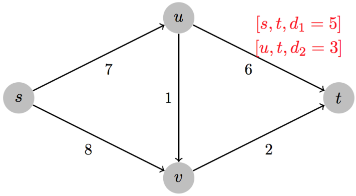


LP Formulation:

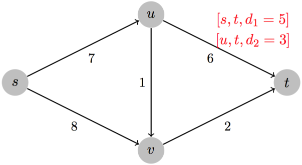
$$
\begin{array}{llcll}
\min & \sum_{(u,v)\in E} a(u,v)f(u,v) & & & \\
s.t. & f(u,v) & \leq & C(u,v) & \text{for each } (u,v) \\
 & f(u,v) & \geq & 0 & \text{for each } (u,v) \\
 & \sum_{u,(u,v)\in E} f(u,v) & = & \sum_{w,(v,w)\in E} f(v,w) & \text{for each } v \in V \\
 & \sum_{v,(s,v)\in E} f(s,v)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

Practical problem 4: MultiCommodityFlow problem

INPUT:
A directed graph $G =< V, E >$. 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:

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

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

SAT problem

INPUT:
A set of m conjunction normal formula (CNF) clauses over n Boolean variables $x_1, x_2, ..., x_n$
OUTPUT:
Whether all clauses can be satisfied by an `TRUE/FALSE` assignment of the n variables.

- A SAT instance:

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

- An assignment to make all clauses `TRUE`:

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

Linear programming formulation

A SAT instance:

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

LP Formulation:

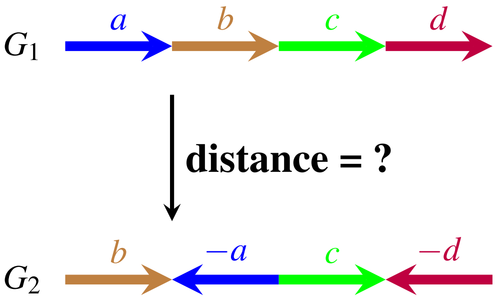
$$\begin{array}{rrrrl}\max & c_1 + & c_2 + & c_3 & \\ s.t. & x_1 + & (1-x_2) + & x_3 & \geq c_1 \\ & (1-x_1) + & x_2 + & (1-x_3) & \geq c_2 \\ & x_1 + & x_2 + & (1-x_3) & \geq c_3 \\ & x_1, & x_2, & x_3 & = 0/1 \\ & c_1, & c_2, & c_3 & = 0/1 \end{array}$$

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 liters.
- Objective function: The objective function denotes the number of satisfied clauses. Thus, Φ is satisfiable iff $c_1 + c_2 + c_3 = 3$.

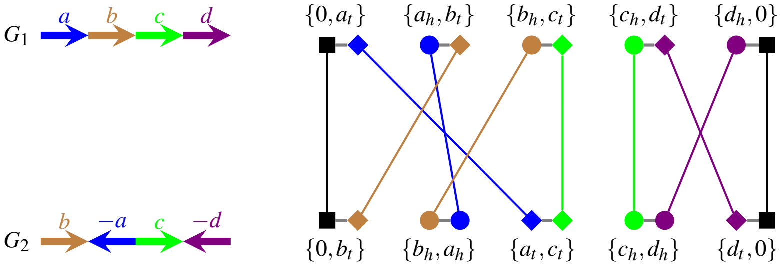
Genome rearrangement distance problem [M. Shao, 2014]

- The minimum number of operations to transform G_1 into G_2

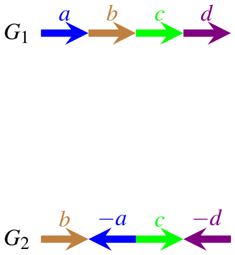


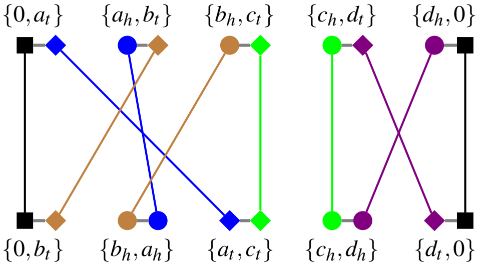
Operations: reverse a fragment of the genome;

Adjacency graph: a more succinct formulation



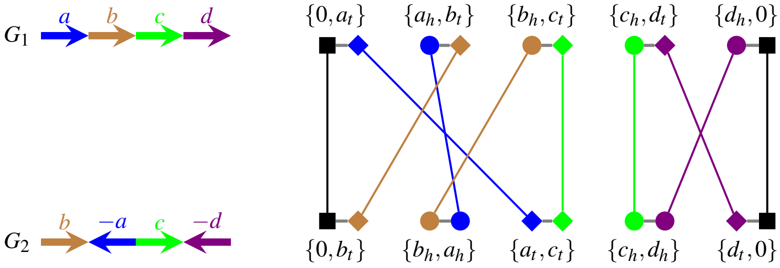
Adjacency graph: a more succinct formulation





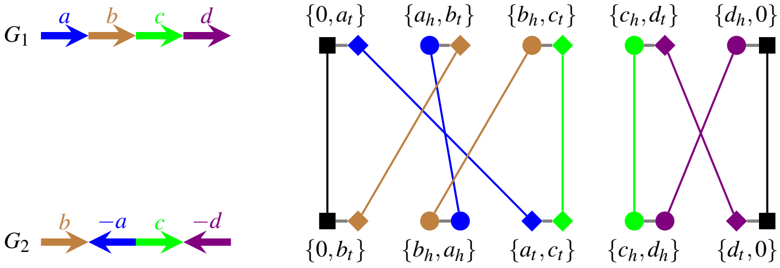
- DCJ distance = (#adjacencies) − (#cycles).

Adjacency graph: a more succinct formulation



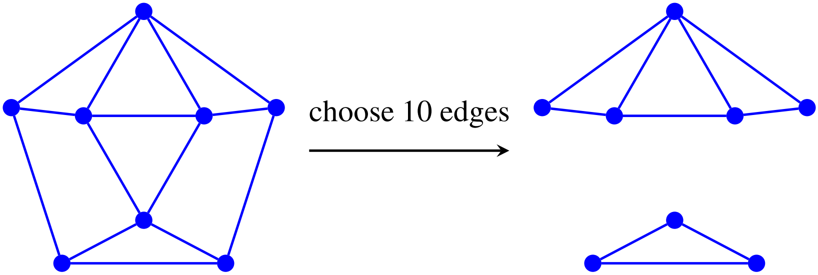
- DCJ distance $=$ (#adjacencies) $-$ (#cycles).
- DCJ distance $= 3$ in this example.

Adjacency graph: a more succinct formulation

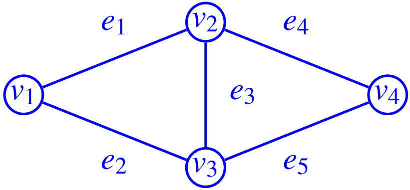


- DCJ distance = (#adjacencies) − (#cycles).
- DCJ distance = 3 in this example.
- To minimize DCJ distance, we need to compute a decomposition of the corresponding adjacency graph with maximized number of cycles.

Problem: 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**.)

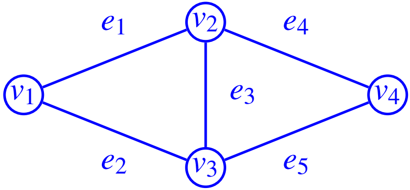


Consider the following example with $k = 3$.



ILP Formulation

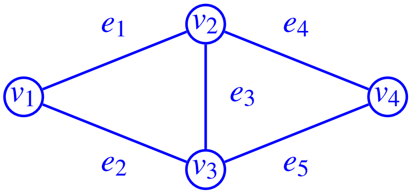
Consider the following example with $k = 3$.



- 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:

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

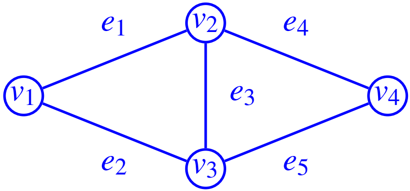
ILP Formulation



- 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:

$$1 \leq y_1 \leq 1$$
$$1 \leq y_2 \leq 2$$
$$1 \leq y_3 \leq 3$$
$$1 \leq y_4 \leq 4$$

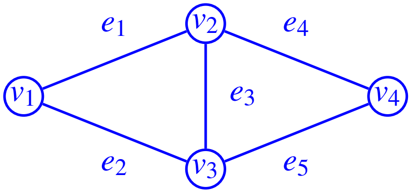
ILP Formulation



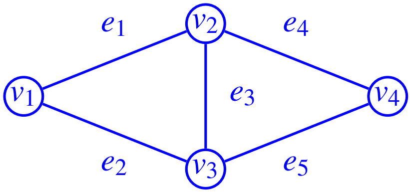
- We guarantee that if an edge is chosen, then its two adjacent vertices have the same label:

$$y_1 \le y_2 + 1 \cdot (1 - x_1);\ \ y_2 \le y_1 + 2 \cdot (1 - x_1) \qquad \text{(for } e_1\text{)}$$
$$y_1 \le y_3 + 1 \cdot (1 - x_2);\ \ y_3 \le y_1 + 3 \cdot (1 - x_2) \qquad \text{(for } e_2\text{)}$$
$$y_2 \le y_3 + 2 \cdot (1 - x_3);\ \ y_3 \le y_2 + 3 \cdot (1 - x_3) \qquad \text{(for } e_3\text{)}$$
$$y_2 \le y_4 + 2 \cdot (1 - x_4);\ \ y_4 \le y_2 + 4 \cdot (1 - x_4) \qquad \text{(for } e_4\text{)}$$
$$y_3 \le y_4 + 3 \cdot (1 - x_5);\ \ y_4 \le y_3 + 4 \cdot (1 - x_5) \qquad \text{(for } e_5\text{)}$$

ILP Formulation

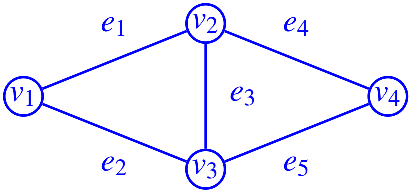


ILP Formulation



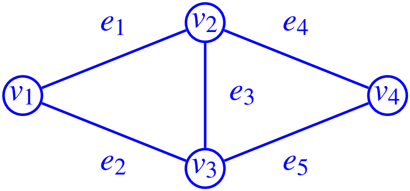
- The equality can propagate along the chosen edges. Thus, in the remaining graph all vertices in the same connected component have the same label.

ILP Formulation

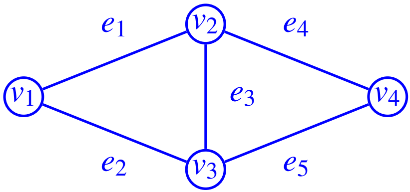


- The equality can propagate along the chosen edges. Thus, in the remaining graph all vertices in the same connected component have the same label.
- Since all vertices have distinct upper bounds, in each connected component, at most one vertex can reach its upper bound. Thus, we can use the number of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

ILP Formulation



ILP Formulation

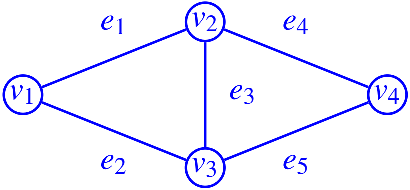


- We use a binary variable z_j to indicate whether the label of v_j reaches its upper bound:

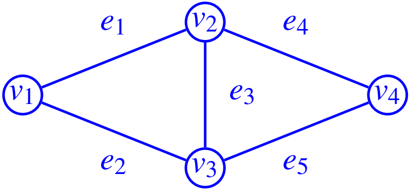
$$1 \cdot z_1 \le y_1$$
$$2 \cdot z_2 \le y_2$$
$$3 \cdot z_3 \le y_3$$
$$4 \cdot z_4 \le y_4$$

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

ILP Formulation



ILP Formulation



- 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

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.

Algorithms and analysis

- In 1949, G. B. Dantzig 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 and T. C. Koopmans, 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.

LP is in P I

Figure: Leonid G. Khanchian

LP is in P II

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

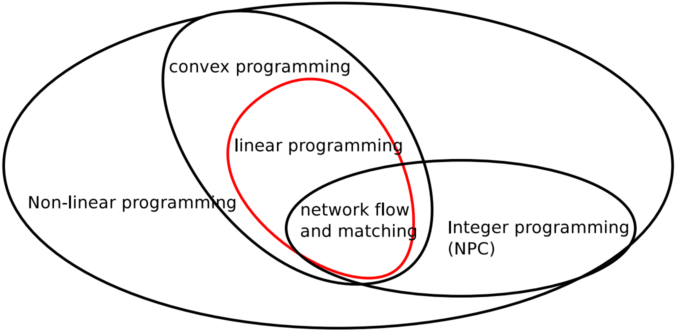
Every day 1,200 American Airlines jets crisscross the U.S., Mexico, Canada and the Caribbean, stopping in 110 cities and bearing over 80,000 passengers. More than 4,000 pilots, copilots, flight personnel, maintenance workers and baggage carriers are shuffled among the flights; a total of 3.6 million 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-

NLP, Convex Programming, LP, Network flow, and ILP.



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.

- General form: mixture of linear inequalities and equalities

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

Form 2: Standard form of linear programming

- Standard form: linear inequalities;

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

- Standard form in matrix language:

$$\begin{array}{rr}
\min & \mathbf{c^T x} \\
s.t. & \mathbf{Ax} \le \mathbf{b} \\
 & \mathbf{x} \ge \mathbf{0}
\end{array}$$

- Standard form in matrix language:

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathbf{T}}\mathbf{x} \\ s.t. \quad & \mathbf{A}\mathbf{x} \le \mathbf{b} \\ & \mathbf{x} \ge \mathbf{0} \end{aligned}$$

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

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

Transformation from general form to standard form

- Transformations:
  1. **Variables:** a free variable $\Rightarrow$ two non-negativeive variables; x_i may or may not be positive $\Rightarrow$ replacing x_i with $x_i' - x_i''$ and adding constraints: $x_i' \geq 0; x_i'' \geq 0$
  2. **Constraints:** an equality $\Rightarrow$ two inequalities;
     $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n = b_j \Rightarrow$
     $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n \geq b_j$
     $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n \leq b_j$

Form 3: Slack form of linear programming

- Slack form: linear equality;

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

- Slack form in matrix language:

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

Transformation from standard form to slack form

- Transformations:
  1. **Variables:** changing "inequality on partial solution $(x_1, ..., x_n)$" to "equality on full solution $(s, x_1, ..., x_n)$" by introducing a slack variable s.
     $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n \leq b_j \Rightarrow$
     $a_{j1}x_1 + a_{j2}x_2 + ... + a_{jn}x_n + s = b_j$
  2. **Constraint:** $s \geq 0$. (s is called a slack variable)

Example: standard form vs. slack form

- Standard form:

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

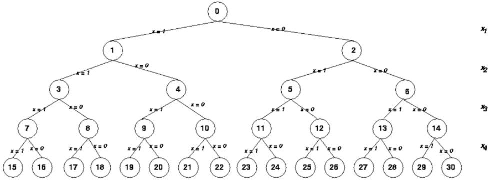
- Slack form:

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

Solving strategy I

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

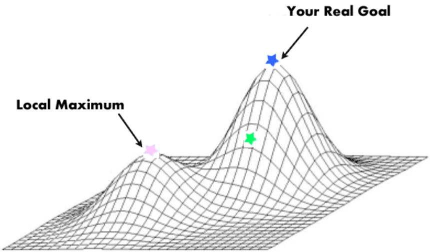
- Solution: $X = [x_1, x_2, ..., x_n]$, $x_i = 0/1$
- Partial solution enumeration tree:



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

Solving strategy II

- Solution: $X = [x_1, x_2, ..., x_n]$
- 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):

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

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

The effect of constraints $\mathbf{x} \geq \mathbf{0}$

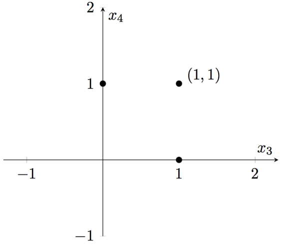
- An example of $\mathbf{A}\mathbf{x} = \mathbf{b}$

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

- By applying Gaussian elimination, we have:

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

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



The effect of $\mathbf{x} \geq \mathbf{0}$

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

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

- By applying Gaussian elimination, we have:

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

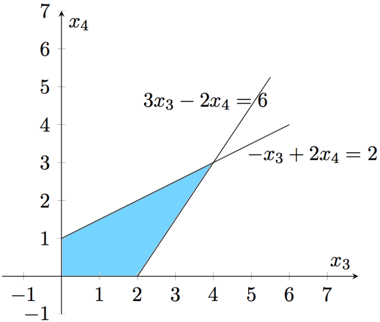
- This is essentially a **linear inequality formula**:

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

- Linear inequality formua:

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

- 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 $(x_1, x_2, x_3, x_4) = (1, 5, 1, 1)$.



Polytope $\Leftrightarrow$ feasible region

Theorem

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

- Basic idea: What is the effect of constraint $\mathbf{x} \geq \mathbf{0}$? 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. $3x_3 - 2x_4 \leq 6$).

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 $\mathbf{x}$** of the following LP:

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

- Applying Gaussian row operations, we have:

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

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

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

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

Proof: polytope $\Rightarrow$ feasible region

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

Proof: polytope ⇒ feasible region cont'd

- Thus we change

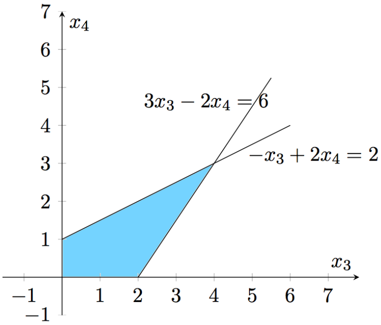
$$
\begin{array}{ccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & ... & + & a_{1n}x_n & \leq & b_1 \\
a_{21}x_1 & + & a_{22}x_2 & + & ... & + & a_{2n}x_n & \leq & b_2 \\
 & & & & ... & & & & \\
a_{m1}x_1 & + & a_{m2}x_2 & + & ... & + & a_{mn}x_n & \leq & b_m \\
x_1 & , & x_2 & , & ... & , & x_n & \geq & 0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

into

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

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

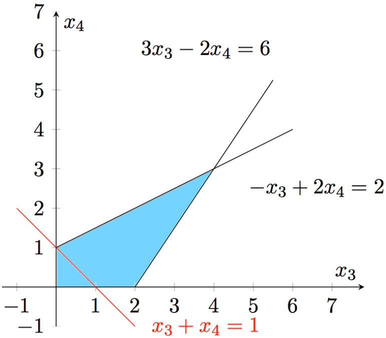
Notations



- Hyper plane: $\{\mathbf{x} : a_1x_1 + a_2x_2 + ... + a_nx_n = b\}$ (linear equality constraint)
- Half space: $\{\mathbf{x} : a_1x_1 + a_2x_2 + ... + a_nx_n \leq b\}$ (linear inequality constraint)
- Polyhedron: the intersection of several half spaces;
- Polytope: a bounded, non-empty polyhedron;

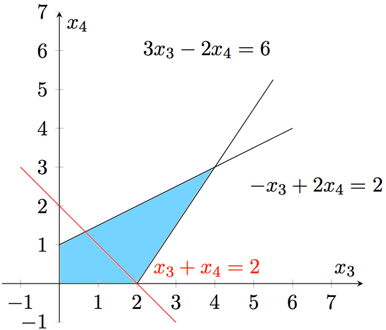
The effect of objective function $\min \mathbf{c}^{\mathbf{T}}\mathbf{x}$

$$
\begin{array}{lrcrcr}
\max & x_3 & + & x_4 & & \\
s.t. & -x_3 & + & 2x_4 & \le & 2 \\
 & 3x_3 & - & 2x_4 & \le & 6 \\
 & x_3 & , & x_4 & \ge & 0
\end{array}
$$

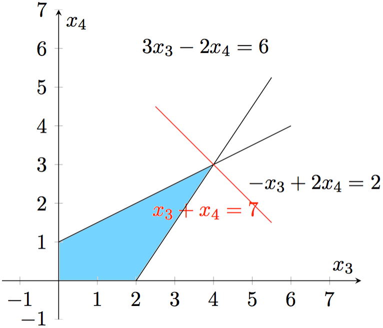


The effect of $\min \mathbf{c}^T\mathbf{x}$ cont'd

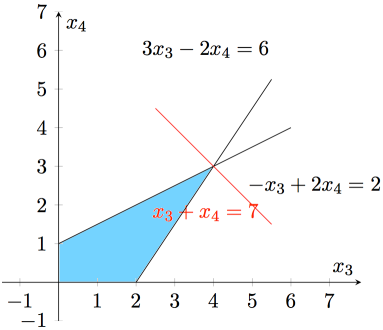
$$\begin{array}{lrcrcr}
\max & x_3 & + & x_4 & & \\
s.t. & -x_3 & + & 2x_4 & \leq & 2 \\
 & 3x_3 & - & 2x_4 & \leq & 6 \\
 & x_3 & , & x_4 & \geq & 0
\end{array}$$



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



- Observation: the optimal solution can be reached at a vertex of the polytope.



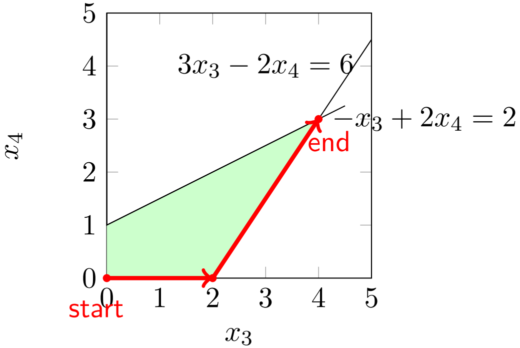
1. What is a feasible solution? Any point within the polytope.
2. Where is the optimal solution? A vertex of the polytope. Consequently, it is not necessary to consider the inner points.

Applying the general IMPROVEMENT strategy to LP

The general IMPROVEMENT strategy for optimization problems

IMPROVEMENT(f)

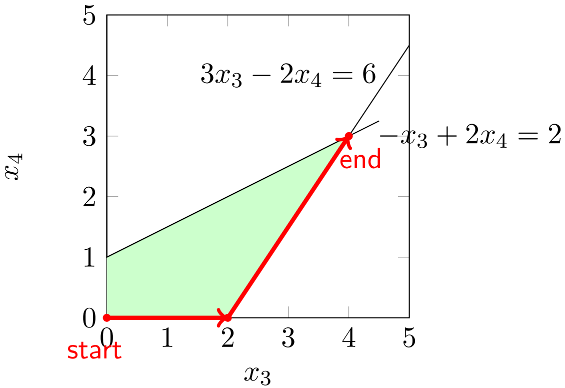
```
x = x0; //set initial solution;
while TRUE do
    x = IMPROVE(x, f); //move towards optimum;
    if STOPPING(x, f) then
        break;
    end if
end while
return x;
```



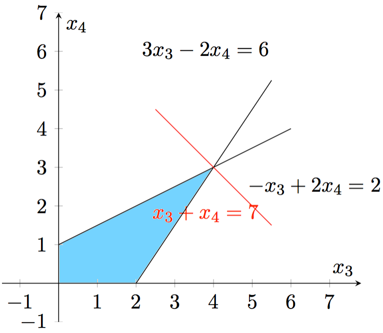
IMPROVEMENT()

$\mathbf{x} = \mathbf{x_0}$; //starting from a vertex;
while TRUE **do**
    $\mathbf{x} =$IMPROVE($\mathbf{x}$); //move to another vertex via an edge;
    if STOPPING($\mathbf{x}$) **then**
        break; //stop when $\mathbf{x}$ is optimal
    end if
end while

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

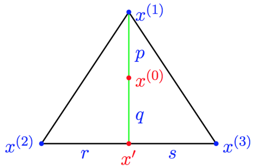
Theorem

There exists a vertex in $\mathbf{P}$ that takes the optimal value.

Proof.

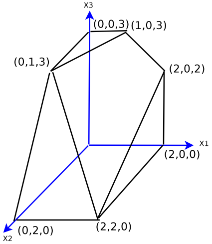
- Since $\mathbf{P}$ is a bounded close set, $\mathbf{c^T x}$ reaches its optimum in $\mathbf{P}$.
- Denote the optimal solution as $\mathbf{x}^{(0)}$. We will show there is a vertex at least as good as $\mathbf{x^{(0)}}$. Why?
  - $\mathbf{x^{(0)}}$ can be represented as the convex combination of vertices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

□



- Suppose $\mathbf{x}^{(0)}$ is an optimal solution.
- Connecting $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$ with a line. Suppose the line intersects line segment $(\mathbf{x}^{(2)}, \mathbf{x}^{(3)})$ at point $\mathbf{x}'$.
- We have $\mathbf{x}^{(0)} = \lambda_1 \mathbf{x}^{(1)} + (1-\lambda_1)\mathbf{x}'$, where $\lambda_1 = \frac{p}{p+q}$.
- We also have $\mathbf{x}' = \lambda_2 \mathbf{x}^{(2)} + (1-\lambda_2)\mathbf{x}^{(3)}$, where $\lambda_2 = \frac{r}{r+s}$.
- Thus, we have $\mathbf{x}^{(0)} = \lambda_1 \mathbf{x}^{(1)} + (1-\lambda_1)\lambda_2 \mathbf{x}^{(2)} + (1-\lambda_1)(1-\lambda_2)\mathbf{x}^{(3)}$.
- Suppose $\mathbf{c^T x^{(1)}}$ is the minimum of $\mathbf{c^T x^{(1)}}, \mathbf{c^T x^{(2)}}, \mathbf{c^T x^{(3)}}$.
- Notice that $\lambda_1 + (1-\lambda_1)\lambda_2 + (1-\lambda_1)(1-\lambda_2) = 1$.
- We have: $\mathbf{c^T x^{(1)}} \le \mathbf{c^T x^{(0)}}$. Thus, a vertex $\mathbf{x}^{(1)}$ is found not worse than $\mathbf{x}^{(0)}$.

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



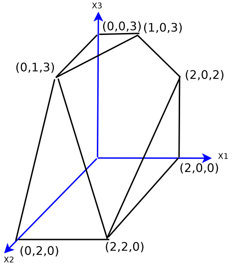
Vertex $\Leftrightarrow$ basic feasible solution

Theorem

A vertex of $\mathbf{P}$ *corresponds to a basis of matrix* $\mathbf{A}$.

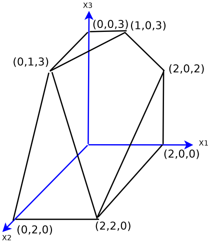
An example (standard form):

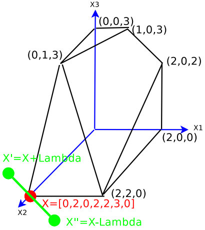
$$\begin{array}{rrrrrrcl}
\min & -x_1 & - & 14x_2 & - & 6x_3 & & \\
s.t. & x_1 & + & x_2 & + & x_3 & \leq & 4 \\
 & x_1 & & & & & \leq & 2 \\
 & & & & & x_3 & \leq & 3 \\
 & & & 3x_2 & + & x_3 & \leq & 6 \\
 & x_1 & , & x_2 & , & x_3 & \geq & 0
\end{array}$$



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

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





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

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

- Take the vertex $(x_1, x_2, x_3) = (0, 2, 0)$ as an example. The corresponding full solution is $(x_1, x_2, x_3, x_4, x_5, x_6, x_7) = (0, 2, 0, 2, 2, 3, 0)$.
- We will show that the column vectors corresponding to **non-zero** x_i, i.e. $\{\mathbf{a_2}, \mathbf{a_4}, \mathbf{a_5}, \mathbf{a_6}\}$, are linearly independent, and thus form a basis (sometimes an extension is needed).
- Suppose $\exists(\lambda_2, \lambda_4, \lambda_5, \lambda_6) \neq 0$ such that $\lambda_2\mathbf{a_2} + \lambda_4\mathbf{a_4} + \lambda_5\mathbf{a_5} + \lambda_6\mathbf{a_6} = 0$, i.e. $\mathbf{A}\lambda = \mathbf{0}$, where $\lambda = [0, \lambda_2, 0, \lambda_4, \lambda_5, \lambda_6, 0]$.
- Then we can construct **two other points: $\mathbf{x}' = \mathbf{x} + \theta\lambda$ and $\mathbf{x}'' = \mathbf{x} - \theta\lambda$.**
- It is easy to deduce that both $\mathbf{x}'$ and $\mathbf{x}''$ lie inside P since:
  - $\mathbf{Ax}' = \mathbf{Ax} + \mathbf{0} = \mathbf{b}$ and $\mathbf{Ax}'' = \mathbf{Ax} - \mathbf{0} = \mathbf{b}$
  - In addition, we can guarantee $\mathbf{x}' \geq \mathbf{0}$ and $\mathbf{x}'' \geq \mathbf{0}$ via setting θ to be sufficiently small since $\mathbf{x} \geq \mathbf{0}$, and $\lambda_1 = \lambda_3 = \lambda_7 = 0$.
- Contradiction: it is impossible for a vertex to be middle point of two inner points of $\mathbf{P}$.

Proof

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

Vertex $\Rightarrow$ basic feasible solution: some notations

- For a vertex $\mathbf{x}$ of the polytope, a basis $\mathbf{B}$ can be derived via extracting the column vectors corresponding to non-zero x_i. The non-basis column vectors are denoted as $\mathbf{N}$.
- Then the original LP can be represented as:

$$
\begin{aligned}
\min \quad & \boxed{\mathbf{c_B^T x_B} +} \boxed{\mathbf{c_N^T x_N}} \\
s.t. \quad & \begin{bmatrix} \boxed{\quad \mathbf{B} \quad} & \boxed{\quad \mathbf{N} \quad} \end{bmatrix} \begin{bmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \end{bmatrix}
\end{aligned}
$$

- Here, $\mathbf{x}$ is decomposed as $\mathbf{x} = \begin{bmatrix} \mathbf{x_B} \\ \mathbf{x_N} \end{bmatrix}$. Then we have $\mathbf{x_N} = \mathbf{0}$, and $\mathbf{x_B} = \mathbf{B}^{-1}\mathbf{b}$ (Reason: $\mathbf{Ax} = \mathbf{b}$, i.e. $\mathbf{Bx_B} + \mathbf{Nx_N} = \mathbf{b}$)
- The corresponding objective value is $\mathbf{c^T x} = \mathbf{c_B^T x_B} + \mathbf{c_N^T x_N} = \mathbf{c_B^T B^{-1} b}$.

- For a vertex $\mathbf{x} = \begin{bmatrix} 0 & 2 & 0 & 2 & 2 & 3 & 0 \end{bmatrix}^T$, the columns corresponding to non-zero x_i are extracted to form a basis
$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 \end{bmatrix}.$$
- Let's decompose $\mathbf{x} = \begin{bmatrix} 0 & 2 & 0 & 2 & 2 & 3 & 0 \end{bmatrix}^T$ accordingly into $\mathbf{x_B} = [2\ 2\ 2\ 3]^T$ and $\mathbf{x_N} = [0\ 0\ 0]^T$.
- It is easy to verify that $\mathbf{x_B} = \mathbf{B^{-1}b}$. In this example,
$$\mathbf{b} = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 6 \end{bmatrix}.$$

Part 2: Basic feasible solution $\Rightarrow$ vertex

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

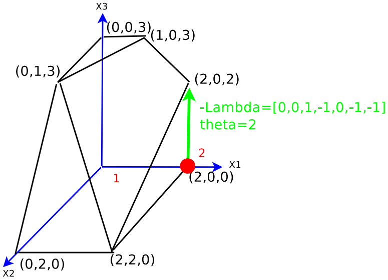
Proof.

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

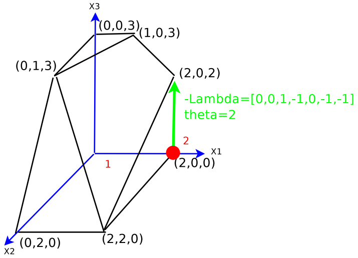
An example

- For matrix $\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} 4 \\ 2 \\ 3 \\ 6 \end{bmatrix}$,
  we first calculate a basis of $\mathbf{A}$ as $\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 3 & 0 & 0 & 0 \end{bmatrix}$.
- The **basic feasible solution $\mathbf{x}$ respect to $\mathbf{B}$** is
  $\mathbf{x} = \begin{bmatrix} \mathbf{B}^{-1}\mathbf{b} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} 0 & 2 & 0 & 2 & 2 & 3 & 0 \end{bmatrix}^T$.
- It is easy to verify that $(x_1, x_2, x_3) = (0, 2, 0)$ is a vertex of the polytope $\mathbf{P}$.

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



Edge $\Leftrightarrow$ non-basis column vector of $\mathbf{A}$: an example



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

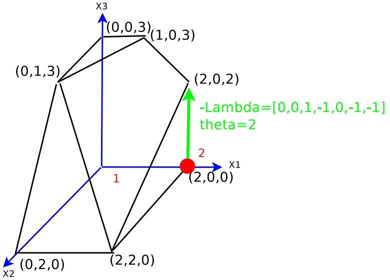
$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 3 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- 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): $\mathbf{B} = \{\mathbf{a_1}, \mathbf{a_4}, \mathbf{a_6}, \mathbf{a_7}\}$.
- Let's consider a non-basis column vector $\mathbf{a_3}$.
- Since $\mathbf{a_3}$ can be decomposed as $\mathbf{a_3} = \mathbf{1a_4} + \mathbf{0a_1} + \mathbf{1a_6} + \mathbf{1a_7}$, we have $\mathbf{0a_1} + \mathbf{0a_2} - \mathbf{1a_3} + \mathbf{1a_4} + \mathbf{0a_5} + \mathbf{1a_6} + \mathbf{1a_7} = \mathbf{0}$
- 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 $\mathbf{x}' = \mathbf{x} - \theta\lambda = [2, 0, 2, 0, 0, 1, 4]^T$ (by setting $\theta = 2$).
- The new vertex corresponds to the basis $\mathbf{B}' = \mathbf{B} - \{\mathbf{a_4}\} \cup \{\mathbf{a_3}\} = \{\mathbf{a_1}, \mathbf{a_3}, \mathbf{a_6}, \mathbf{a_7}\}$.

Edge $\Leftrightarrow$ non-basis column vector of $\mathbf{A}$

Theorem

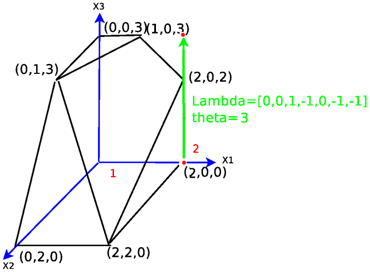
Let $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ *be a vertex corresponding to basis* $\mathbf{B} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_m\}$. *Consider a non-basis vector* $\mathbf{a}_e \notin \mathbf{B}$. *Suppose* $\mathbf{a}_e$ *can be decomposed as* $\mathbf{a}_e = \lambda_1 \mathbf{a}_1 + \lambda_2 \mathbf{a}_2 + ... + \lambda_m \mathbf{a}_m$. *Let* $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{x_i}{\lambda_i} = \frac{x_l}{\lambda_l}$. *Then* $\mathbf{x}' = \mathbf{x} - \theta\lambda$ *is also a vertex corresponding to basis* $\mathbf{B}' = \mathbf{B} - \{\mathbf{a}_l\} \cup \{\mathbf{a}_e\}$. *Here* $\lambda = [\lambda_1, \lambda_2, ..., \lambda_m, 0, ..., -1, ..., 0]^T$.



Proof.

- We have $x_1\mathbf{a}_1 + x_2\mathbf{a}_2 + ... + x_m\mathbf{a}_m = \mathbf{b}$ (Reason: $\mathbf{x}$ is a feasible solution).
- We also have $\lambda_1\mathbf{a}_1 + \lambda_2\mathbf{a}_2 + ... + \lambda_m\mathbf{a}_m - \mathbf{a}_e = 0$.
- Thus we have $(x_1 - \theta\lambda_1)\mathbf{a}_1 + ... + (x_m - \theta\lambda_m)\mathbf{a}_m + ... + \theta\mathbf{a}_e = \mathbf{b}$.
- To show that $\mathbf{x}' = \mathbf{x} - \theta\lambda$ is also feasible, it suffices to prove $\mathbf{x}' \geq \mathbf{0}$. There are two cases:
  1. $\forall i, \lambda_i \leq 0$**:** for any positive θ we still have $x'_i = x_i - \theta\lambda_i \geq x_i \geq 0$.
  2. $\exists i, \lambda_i > 0$**:** we cannot set θ too large. In fact, by setting $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{x_i}{\lambda_i} = \frac{x_l}{\lambda_l}$, we can guarantee $x_i - \theta\lambda_i \geq 0$; however, a larger θ will cause $(x_l - \theta\lambda_l) < 0$. For example,
     $\mathbf{x} = [2, 0, 0, 2, 0, 3, 6]^T$
     $\lambda = [0, 0, -1, 1, 0, 1, 1]^T$
     We set $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{x_i}{\lambda_i} = \frac{x_4}{\lambda_4} = 2$ and $l = 4$.
- Thus $\mathbf{x}'$ is a new feasible solution.

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

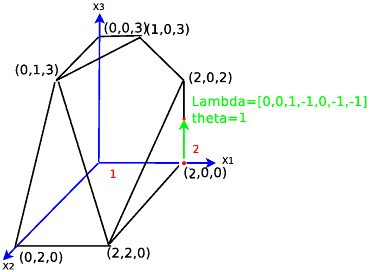


$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$

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

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

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

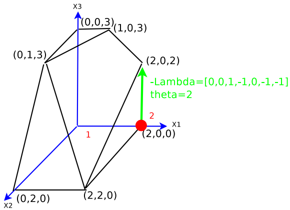


$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$

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

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

Part 2: $\mathbf{B}' = \mathbf{B} - \{\mathbf{a}_l\} \cup \{\mathbf{a}_e\}$ 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 $\mathbf{x}' = \mathbf{x} - \theta\lambda = [2, 0, 2, 0, 0, 1, 4]^T$ is also a vertex, it suffices to show that the column vectors corresponding to non-zero x_i form a basis, i.e. $\mathbf{B}' = \mathbf{B} - \{\mathbf{a_4}\} \cup \{\mathbf{a_3}\} = \{\mathbf{a_1}, \mathbf{a_3}, \mathbf{a_6}, \mathbf{a_7}\}$ is a basis.
- Suppose $\mathbf{B}'$ is linear dependent, i.e. there exists $(d_1, d_3, d_6, d_7) \neq \mathbf{0}$ such that $d_1\mathbf{a}_1 + d_3\mathbf{a}_3 + d_6\mathbf{a}_6 + d_7\mathbf{a}_7 = 0$.
- Recall that $\mathbf{a}_3$ can be decomposed as $\mathbf{a}_3 = 1\mathbf{a}_4 + 0\mathbf{a}_1 + 1\mathbf{a}_6 + 1\mathbf{a}_7$.
- We have $d_1\mathbf{a}_1 + d_3\mathbf{a}_4 + (d_6 + d_3)\mathbf{a}_6 + (d_7 + d_3)\mathbf{a}_7 = 0$.
- Thus $d_3 = 0$. (Reason: $\mathbf{B} = \{\mathbf{a_1}, \mathbf{a_4}, \mathbf{a_6}, \mathbf{a_7}\}$ is a basis.)
- Therefore $d_1 = d_6 = d_7 = 0$. Contradiction.

Part 2: $\mathbf{B}' = \mathbf{B} - \{\mathbf{a}_l\} \cup \{\mathbf{a}_e\}$ is a basis.

Proof.

- Suppose $\mathbf{B}'$ is linear dependent;
- Thus, there exists $< d_1, ..., d_{l-1}, d_{l+1}, ..., d_m, d_j > \neq \mathbf{0}$ such that $d_1\mathbf{a}_1 + ...d_{l-1}\mathbf{a}_{l-1} + d_{l+1}\mathbf{a}_{l+1} + ... + d_m\mathbf{a}_m + d_e\mathbf{a}_e = 0$.
- We also have $\mathbf{a}_e = \lambda_1\mathbf{a}_1 + ... + \lambda_l\mathbf{a}_l + ... + \lambda_m\mathbf{a}_m$.
- Substituting $\mathbf{a}_e$ into the above equation, we have:
- $(d_1 + d_e\lambda_1)\mathbf{a}_1 + ... + (d_e\lambda_l)\mathbf{a}_l + ... + (d_m + d_e\lambda_m)\mathbf{a}_m = 0$
- Thus $d_e\lambda_l = 0$. (Reason: $\mathbf{B} = \{\mathbf{a_1}, ..., \mathbf{a_m}\}$ is a basis.)
- Therefore $d_e = 0$ (Reason: $\lambda_l > 0$).
- Therefore we have $d_i = 0$ for all i (Reason: $d_i = d_i + d_e\lambda_i = 0$). A contradiction.

$\square$

Pivoting operation

- The process to change $\mathbf{B}$ into $\mathbf{B}'$ is called **"pivoting"** with $\mathbf{a}_e$ **"entering"** basis, and $\mathbf{a}_l$ **"leaving"** basis.
- The "pivoting" operation can be accomplished by Gaussian row operation.

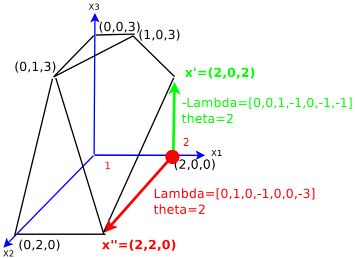
$$
\left(\begin{array}{c|ccccc}
0 & \ldots & 0 & \cdots & c_e & \cdots \\
\hline
b_1 & \ldots & 0 & \cdots & a_{1e} & \cdots \\
b_2 & \ldots & 0 & \cdots & a_{2e} & \cdots \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
b_l & \ldots & 1 & \cdots & a_{le} & \cdots \\
\cdots & \cdots & \cdots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

- 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?

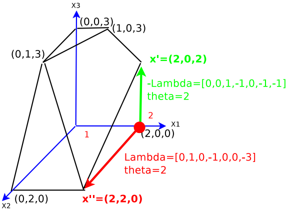
- Generally speaking, a vertex of P has at most $n - m$ adjacent edges (Why?)



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

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

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

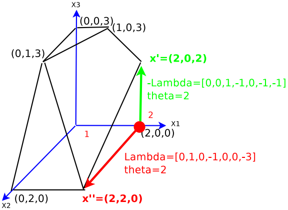


$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 2 & 0 & 3 & 6 \end{bmatrix}^T$$

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

- We decompose $\mathbf{a_2}$ as $\mathbf{a_2} = \mathbf{1a_4} + \mathbf{0a_1} + \mathbf{0a_6} + \mathbf{3a_7}$, i.e. $\mathbf{0a_1} - \mathbf{a_2} + \mathbf{0a_3} + \mathbf{1a_4} + \mathbf{0a_3} + \mathbf{0a_6} + \mathbf{3a_7} = 0$.
- The coefficient is: $\lambda = [0, -1, 0, 1, 0, 0, 3]^T$.
- By setting an appropriate θ, we get to vertex $\mathbf{x}'' = \mathbf{x} - \theta\lambda$.
- The objective value can be improved by $\mathbf{c^T x}'' - \mathbf{c^T x} = (c_2 - (1c_4 + 0c_1 + 0c_6 + 3c_7))\theta = -14\theta$

Trial 2: pivoting in $\mathbf{a_3}$



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

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

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

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

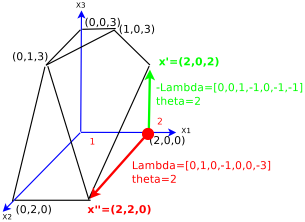
How to choose a non-basis vector $\mathbf{a}_e$ to enter $\mathbf{B}$?

- Consider a vertex $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ corresponding to basis $\mathbf{B} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_m\}$.
- Suppose we choose a non-basis vector $\mathbf{a}_e \notin \mathbf{B}$ to enter basis
- Since $\mathbf{a}_e$ is not in basis, it can be decomposed as $\mathbf{a}_e = \lambda_1 \mathbf{a}_1 + \lambda_2 \mathbf{a}_2 + ... + \lambda_m \mathbf{a}_m$
- Let $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{x_i}{\lambda_i} = \frac{x_l}{\lambda_l}$.
- Then $\mathbf{x}' = \mathbf{x} - \theta\lambda$ is also a vertex, where $\lambda = [\lambda_1, \lambda_2, .., \lambda_m, 0, ..., -1, ...0]^T$.
- Recall that the objective is to minimize $\mathbf{c}^\mathbf{T}\mathbf{x}$. Let's see whether we can improve the objective function by moving from vertex $\mathbf{x}$ to $\mathbf{x}'$.
- Notice $\mathbf{c}^\mathbf{T}\mathbf{x}' - \mathbf{c}^\mathbf{T}\mathbf{x} = \theta \mathbf{c}^\mathbf{T}\lambda = (c_e - \sum_{\mathbf{a}_i \in \mathbf{B}} \lambda_i c_i)\theta$.
- Pivoting in rule:

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

Question 4: When should we stop?

Stopping criteria



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

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

- Notice: suppose we move from vertex $\mathbf{x}$ to $\mathbf{x}' = \mathbf{x} - \theta\lambda$, the improvement of objective value is $\mathbf{c}^{\mathbf{T}}\mathbf{x}' - \mathbf{c}^{\mathbf{T}}\mathbf{x} = \theta\mathbf{c}^{\mathbf{T}}\lambda = (c_e - \sum_{\mathbf{a}_i \in \mathbf{B}} \lambda_i c_i)\theta$.
- We will benefit from pivoting in $\mathbf{a_e}$ if $\mathbf{c}^{\mathbf{T}}\mathbf{x}' \leq \mathbf{c}^{\mathbf{T}}\mathbf{x}$, i.e. $c_e - \sum_{\mathbf{a}_i \in \mathbf{B}} \lambda_i c_i < 0$.
- Thus the following stopping criteria is reasonable: $c_e - \sum_{\mathbf{a}_i \in \mathbf{B}} \lambda_i c_i \geq 0$ for all e.
- We denote $\overline{c_e} \triangleq c_e - \sum_{\mathbf{a}_i \in \mathbf{B}} \lambda_i c_i$ as **"checking number"**.
- In fact, $\overline{c_e}$ is the e-th entry of $\overline{\mathbf{c}}^{\mathbf{T}} = \mathbf{c}^{\mathbf{T}} - \mathbf{c}_{\mathbf{B}}^{\mathbf{T}}\mathbf{B}^{-1}\mathbf{A}$.

Stopping criteria

Theorem

Consider a LP (in slack form):

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

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

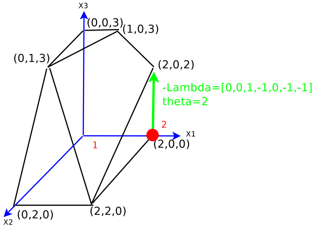
Proof.

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

$\square$

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 obtains a vertex? Vertex corresponds to a basis of the matrix $\mathbf{A}$, 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? $\overline{c_e} \geq 0$ for all index e means that we have obtained an optimal $\mathbf{x}$.

$\textsc{Simplex}(\mathbf{A}, \mathbf{b}, \mathbf{c})$

$(B_I, \mathbf{A}, \mathbf{b}, \mathbf{c}, z) = \textsc{InitializeSimplex}(\mathbf{A}, \mathbf{b}, \mathbf{c})$;
//If the LP is feasible, a vertex $\mathbf{x}$ is returned with B_I storing the indices of vectors in the corresponding basis $\mathbf{B}$; otherwise, "infeasible" is reported.
while TRUE **do**
    if there is no index e $(1 \le e \le n)$ having $c_e < 0$ **then**
        $\mathbf{x} = \textsc{CalculateX}(B_I, \mathbf{A}, \mathbf{b}, \mathbf{c})$;
        return $(\mathbf{x}, z)$;
    end if;
    choose an index e having $c_e < 0$ according to a certain rule;
    for each index i $(1 \le i \le m)$ **do**
        if $a_{ie} > 0$ **then**
            $\Delta_i = \frac{b_i}{a_{ie}}$;
        else
            $\Delta_i = \infty$;
        end if
    end for
    choose an index l that minimizes Δ_i;
    if $\Delta_l = \infty$ **then**
        return "unbounded";
    end if
    $(B_I, \mathbf{A}, \mathbf{b}, \mathbf{c}, z) = \textsc{Pivot}(B_I, \mathbf{A}, \mathbf{b}, \mathbf{c}, z, e, l)$;
end while

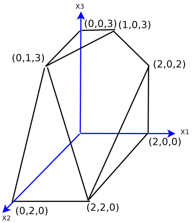
CALCULATEX(B_I, $\mathbf{A}$, $\mathbf{b}$, $\mathbf{c}$)

//assign non-basic variables with 0, and assign basic variables with corresponding b_i;
for $j = 1$ to n **do**
    if $j \notin B_I$ **then**
        $x_j = 0$;
    else
        for $i = 1$ to m **do**
            if $a_{ij} = 1$ **then**
                $x_j = b_i$;
            end if
        end for
    end if
end for
return $\mathbf{x}$;

An example

Standard form:

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



Standard form:

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

Slack form:

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

SIMPLEX algorithm maintains a simplex tabular

	x_1	x_2	x_3	x_4	x_5	x_6	x_7
-z= 0	$\overline{c_1}$=-1	$\overline{c_2}$=-14	$\overline{c_3}$=-6	$\overline{c_4}$=0	$\overline{c_5}$=0	$\overline{c_6}$=0	$\overline{c_7}$=0
$\mathbf{x_{B1}} = b'_1$=4	1	1	1	1	0	0	0
$\mathbf{x_{B2}} = b'_2$=2	1	0	0	0	1	0	0
$\mathbf{x_{B3}} = b'_3$=3	0	0	1	0	0	1	0
$\mathbf{x_{B4}} = b'_4$=6	0	3	1	0	0	0	1

- Coefficient matrix: $\mathbf{B^{-1}A}$. The basis forms a unit matrix, while the other part is $\mathbf{B^{-1}N}$.
- The first row contains "checking number" $\overline{\mathbf{c}}^{\mathbf{T}} = \mathbf{c^T} - \mathbf{c_B^T B^{-1} A}$ (initial value: $\mathbf{c}$)
- The first column contains solution $\mathbf{x_B} = \mathbf{b'} = \mathbf{B^{-1}b}$ (initial value: $\mathbf{b}$)
- The up-left item: objective value $-z = \mathbf{c_B^T x_B} = \mathbf{c_B^T B^{-1} b}$ (initial value: 0)

Why simplex tabular takes this form?

$$
\begin{array}{cc}
 & \begin{array}{ccccccc} & & \mathbf{c}_\mathbf{B}^\mathbf{T} & & & \mathbf{c}_\mathbf{N}^\mathbf{T} & \end{array} \\
 & \left(\begin{array}{c|cccc|cc}
0 & c_1 & c_2 & \cdots & c_m & \cdots & c_n \\ \hline
b_1 & a_{11} & a_{12} & \cdots & a_{1m} & \cdots & a_{1n} \\
b_2 & a_{21} & a_{22} & \cdots & a_{2m} & \cdots & a_{2n} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
b_m & a_{m1} & a_{m2} & \cdots & a_{mm} & \cdots & a_{mn}
\end{array}\right) \\
 & \begin{array}{ccc} \mathbf{b} & \mathbf{B} & \mathbf{N} \end{array}
\end{array}
\overset{\text{Row Operation}}{\Longrightarrow}
\underset{\mathbf{B}^{-1}\times}{\Longrightarrow}
\begin{array}{c}
\overline{\mathbf{c}}^\mathbf{T} \\
\left(\begin{array}{c|cccc|c}
-\mathbf{c}_\mathbf{B}^\mathbf{T}\mathbf{B}^{-1}\mathbf{b} & 0 & 0 & \cdots & 0 & \mathbf{c}_\mathbf{N}^\mathbf{T} - \mathbf{c}_\mathbf{B}^\mathbf{T}\mathbf{B}^{-1}\mathbf{N} \\ \hline
 & 1 & 0 & \cdots & 0 & \\
\mathbf{B}^{-1}\mathbf{b} & 0 & 1 & \cdots & 0 & \mathbf{B}^{-1}\mathbf{N} \\
 & \vdots & \vdots & \ddots & \vdots & \\
 & 0 & 0 & \cdots & 1 &
\end{array}\right) \\
\mathbf{B}^{-1}\mathbf{B}
\end{array}
$$

- Coefficient matrix: $\mathbf{B}^{-1}\mathbf{A}$. **The basis always forms a unit matrix.** Why?
  - This way, for any non-basis column vector $\mathbf{a}_e$, the e-th column stores the coefficients $[\lambda_1, \lambda_2, ..., \lambda_m]^T$, i.e. $\mathbf{a}_e$ is decomposed as $\mathbf{a}_e = \lambda_1\mathbf{a}_1 + ... + \lambda_m\mathbf{a}_m$.
- The "pivoting" operation is accomplished by Gaussian row operations on **all rows**, including the first row $\overline{\mathbf{c}^\mathbf{T}}$, and the column $\mathbf{b}'$. Why?
  1. The row operation make the entries in $\mathbf{c}_\mathbf{B}^\mathbf{T}$ be 0, thus the first row contains "checking number" $\overline{\mathbf{c}}^\mathbf{T} = \mathbf{c}^\mathbf{T} - \mathbf{c}_\mathbf{B}^\mathbf{T}\mathbf{B}^{-1}\mathbf{A}$ (initial value: $\mathbf{c}$)
  2. The up-left item shows the objective value $-z = 0 - \mathbf{c}_\mathbf{B}^\mathbf{T}\mathbf{B}^{-1}\mathbf{b} = -\mathbf{c}_\mathbf{B}^\mathbf{T}\mathbf{B}^{-1}\mathbf{b}$ (initial value: 0)

Pivotting I

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

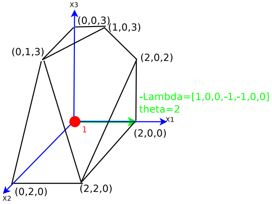
$\textsc{Pivot}(B_I, \mathbf{A}, \mathbf{b}, \mathbf{c}, z, e, l)$

```
//Scaling the l-th line
b_l = b_l / a_le;
for j = 1 to n do
    a_lj = a_lj / a_le;
end for
//All other lines minus the l-th line
for i = 1 to m but i ≠ l do
    b_i = b_i − a_ie × b_l;
```

Pivotting II

```
    for j = 1 to n do
        a_ij = a_ij − a_ie × a_lj;
    end for
end for
//The first line minuses the l-th line
z = z − b_l × c_e;
for j = 1 to n do
    c_j = c_j − c_e × a_lj;
end for
//Calculating x
B_I = B_I − {l} ∪ {e};
return (B_I, A, b, c, z);
```

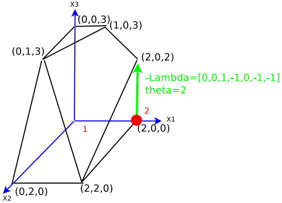
Step 1



	x_1	x_2	x_3	x_4	x_5	x_6	x_7
-z= 0	$\overline{c_1}$= -1	$\overline{c_2}$=-14	$\overline{c_3}$=-6	$\overline{c_4}$=0	$\overline{c_5}$=0	$\overline{c_6}$=0	$\overline{c_7}$=0
$\mathbf{x_{B1}} = b'_1$=4	1	1	1	1	0	0	0
$\mathbf{x_{B2}} = b'_2$=2	1	0	0	0	1	0	0
$\mathbf{x_{B3}} = b'_3$=3	0	0	1	0	0	1	0
$\mathbf{x_{B4}} = b'_4$=6	0	3	1	0	0	0	1

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_4}, \mathbf{a_5}, \mathbf{a_6}, \mathbf{a_7}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [0,0,0,4,2,3,6]^T$. (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 $\mathbf{a_1}$ to enter basis since $\overline{c_1} = -1 < 0$; choose $\mathbf{a_5}$ to exit since $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_2}{\lambda_2} = 2$.
- Here, the corresponding λ is stored in the 1-st column (Why? the basis $\mathbf{B}$ forms an identity matrix.)

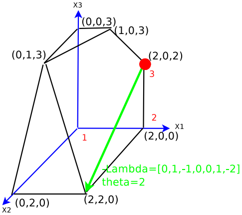
Step 2



	x_1	x_2	x_3	x_4	x_5	x_6	x_7
-z= 2	$\overline{c_1}$= 0	$\overline{c_2}$=-14	$\overline{c_3}$= -6	$\overline{c_4}$=0	$\overline{c_5}$=1	$\overline{c_6}$=0	$\overline{c_7}$=0
$\mathbf{x_{B1}} = b'_1$=2	0	1	1	1	-1	0	0
$\mathbf{x_{B2}} = b'_2$=2	1	0	0	0	1	0	0
$\mathbf{x_{B3}} = b'_3$=3	0	0	1	0	0	1	0
$\mathbf{x_{B4}} = b'_4$=6	0	3	1	0	0	0	1

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_1}, \mathbf{a_4}, \mathbf{a_6}, \mathbf{a_7}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [2, 0, 0, 2, 0, 3, 6]^T$. (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 $\mathbf{a_3}$ to enter basis since $\overline{c_3} = -6 < 0$; choose $\mathbf{a_4}$ to exit since $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_1}{\lambda_1} = 2$.
- Here, the corresponding λ is stored in the 3-rd column (Why? the basis $\mathbf{B}$ forms an identity matrix.)

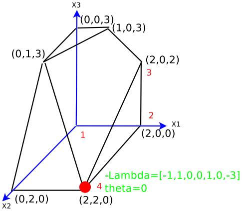
Step 3



	x_1	x_2	x_3	x_4	x_5	x_6	x_7
-z= 14	$\overline{c_1}$= 0	$\overline{c_2}$= -8	$\overline{c_3}$=0	$\overline{c_4}$=6	$\overline{c_5}$=-5	$\overline{c_6}$=0	$\overline{c_7}$=0
$\mathbf{x_{B1}} = b'_1$=2	0	1	1	1	-1	0	0
$\mathbf{x_{B2}} = b'_2$=2	1	0	0	0	1	0	0
$\mathbf{x_{B3}} = b'_3$=1	0	-1	0	-1	1	1	0
$\mathbf{x_{B4}} = b'_4$=4	0	2	0	-1	1	0	1

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_1}, \mathbf{a_3}, \mathbf{a_6}, \mathbf{a_7}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [2, 0, 2, 0, 0, 1, 4]^T$. (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 $\mathbf{a_2}$ to enter basis since $\overline{c_2} = -8 < 0$; choose $\mathbf{a_3}$ to exit since $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_1}{\lambda_1} = 2$.
- Here, the corresponding λ is stored in the 2-nd column (Why? the basis $\mathbf{B}$ forms an identity matrix.)

Step 4



	x_1	x_2	x_3	x_4	x_5	x_6	x_7
-z= 30	$\overline{c_1}$= 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
$\mathbf{x_{B1}} = b'_1$=2	0	1	1	1	-1	0	0
$\mathbf{x_{B2}} = b'_2$=2	1	0	0	0	1	0	0
$\mathbf{x_{B3}} = b'_3$=3	0	0	1	0	0	1	0
$\mathbf{x_{B4}} = b'_4$=0	0	0	-2	-3	3	0	1

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_6}, \mathbf{a_7}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B}^{-1}\mathbf{b} \\ \mathbf{0} \end{bmatrix} = [2, 2, 0, 0, 0, 3, 0]^T$. (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 $\mathbf{a_5}$ to enter basis since $\overline{c_5} = -13 < 0$; choose $\mathbf{a_7}$ to exit since $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_4}{\lambda_4} = 0$. Note: $\theta = 0 \Rightarrow$ same vertex (called **“degeneracy”**).
- Here, the corresponding λ is stored in the 5-th column (Why? the basis $\mathbf{B}$ forms

Degeneracy might lead to cycle

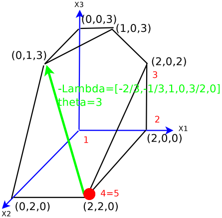
- Generally speaking, two different basis correspond to different vertices.
- However, redundant constraints can lead to degeneracy, i.e. the simplex algorithm is stuck at a vertex even after a "pivoting" operation.
- Sometimes degeneracy can lead to "cycling": If a sequence of pivots starting from a 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 $\mathbf{A}, \mathbf{b}, \mathbf{c}$ 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 $\mathbf{a}_e$ to enter: $e = \min\{j : \overline{c_j} \leq 0, 1 \leq j \leq n\}$.
     - choose $\mathbf{a}_l$ to exit: choose the smallest l to break ties.

(see extra slides for an example)

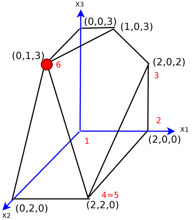
Step 5



	x_1	x_2	x_3	x_4	x_5	x_6	x_7
-z= 30	$\overline{c_1}$= 0	$\overline{c_2}$=0	$\overline{c_3}$= -$\frac{2}{3}$	$\overline{c_4}$=1	$\overline{c_5}$=0	$\overline{c_6}$=0	$\overline{c_7}$=$\frac{13}{3}$
$\mathbf{x_{B1}} = b'_1$=2	0	1	$\frac{1}{3}$	1	0	0	$\frac{1}{3}$
$\mathbf{x_{B2}} = b'_2$=2	1	0	$\frac{2}{3}$	0	0	0	-$\frac{1}{3}$
$\mathbf{x_{B3}} = b'_3$=3	0	0	1	0	0	1	0
$\mathbf{x_{B4}} = b'_4$=0	0	0	-$\frac{2}{3}$	-1	1	0	$\frac{1}{3}$

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_5}, \mathbf{a_6}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [2, 2, 0, 0, 0, 3, 0]^T$. (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 $\mathbf{a_3}$ to enter basis since $\overline{c_3} = -2/3 < 0$; choose $\mathbf{a_1}$ to exit since $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_2}{\lambda_2} = 3$.
- Here, the corresponding λ is stored in the 3-rd column (Why? the basis $\mathbf{B}$ forms an identity matrix.)

Step 6



	x_1	x_2	x_3	x_4	x_5	x_6	x_7
-z= 32	$\overline{c_1}$= 1	$\overline{c_2}$=0	$\overline{c_3}$= 0	$\overline{c_4}$=2	$\overline{c_5}$=0	$\overline{c_6}$=0	$\overline{c_7}$=4
$\mathbf{x_{B1}} = b'_1$=1	-$\frac{1}{2}$	1	0	-$\frac{1}{2}$	0	0	$\frac{1}{2}$
$\mathbf{x_{B2}} = b'_2$=3	$\frac{3}{2}$	0	1	$\frac{3}{2}$	0	0	-$\frac{1}{2}$
$\mathbf{x_{B3}} = b'_3$=0	-$\frac{3}{2}$	0	0	-$\frac{3}{2}$	0	1	$\frac{1}{2}$
$\mathbf{x_{B4}} = b'_4$=2	1	0	0	0	1	0	0

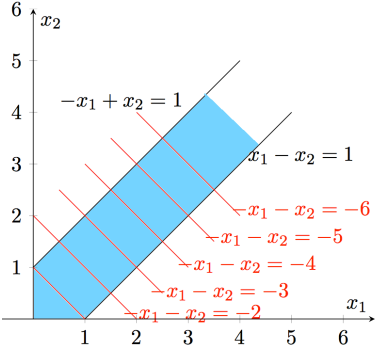
- Basis (in blue): $\mathbf{B = \{a_2, a_3, a_5, a_6\}}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [0, 1, 3, 0, 2, 0, 0]^T$. (Hint: basis variables x_2, x_3, x_6, x_5 take value of b'_1, b'_2, b'_3, b'_4, respectively.)
- Pivoting: all $\overline{c_j} \geq 0$, thus optimal solution found.

An example with unbounded objective value

An example with unbounded objective value

Standard form:

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



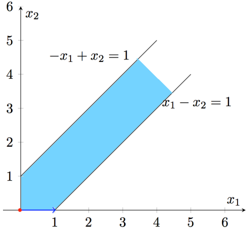
An example with unbounded objective value

Standard form:

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

Slack form:

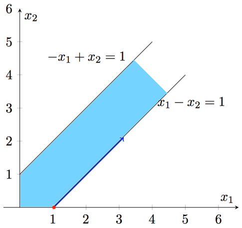
$$\begin{array}{rrcrcrcrcl} \min & -x_1 & - & x_2 & & & & & & \\ s.t. & x_1 & - & x_2 & + & x_3 & & & = & 1 \\ & -x_1 & + & x_2 & & & + & x_4 & = & 1 \\ & x_1 & , & x_2 & , & x_3 & , & x_4 & \geq & 0 \end{array}$$



	x_1	x_2	x_3	x_4
-z= 0	$\overline{c_1}$= -1	$\overline{c_2}$=-1	$\overline{c_3}$= 0	$\overline{c_4}$=0
$\mathbf{x_{B_1}} = b'_1$=1	1	-1	1	0
$\mathbf{x_{B_2}} = b'_2$=1	-1	1	0	1

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

Step 2



	x_1	x_2	x_3	x_4
-z= 1	$\overline{c_1}$=0	$\overline{c_2}$= -2	$\overline{c_3}$= 1	$\overline{c_4}$=0
$\mathbf{x_{B_1}} = b'_1$=1	1	-1	1	0
$\mathbf{x_{B_2}} = b'_2$=1	0	0	1	1

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_1}, \mathbf{a_4}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [1, 0, 0, 2]^T$.
- Pivoting: choose $\mathbf{a_2}$ to enter basis since $c_2 = -2 < 0$; while $\lambda_{2i} \leq 0$ for all i, then θ 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 $\mathbf{A}\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$?

Theorem

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

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

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

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

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

Proof.

- Suppose that L has a feasible solution $(x_1, x_2, ..., x_n) = (\overline{x_1}, \overline{x_2}, ..., \overline{x_n})$;
- Then expand this solution by combining with $\overline{x_0} = 0$, i.e. $(x_0, x_1, x_2, ..., x_n) = (0, \overline{x_1}, \overline{x_2}, ..., \overline{x_n})$;
- 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.

$\square$

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

How to solve L_{aux}?

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

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

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

- Consider the slack form of L_{aux}:

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

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

- Consider the slack form of L_{aux}:

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

- If there are some negative b_i, then $(x_1, \ldots, x_n, s_1, s_2, \ldots, s_m) = (0, \ldots, 0, b_1, b_2, \ldots, b_m)$ 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$(\mathbf{A}, \mathbf{b}, \mathbf{c})$

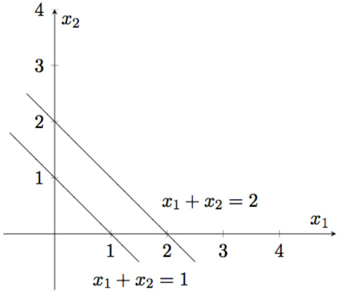
let l be the index of the minimum b_i;
set B_I to include the indices of slack variables;
if $b_l \geq 0$ **then**
    return $(B_I, \mathbf{A}, \mathbf{b}, \mathbf{c}, 0)$;
end if
construct L_{aux} by adding $-x_0$ to each constraint, and using x_0 as the objective function;
let $(\mathbf{A}, \mathbf{b},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

InitializeSimplex: an example with no feasible solution

An example with no feasible solution

LP L:

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



- LP L:

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

- LP L_{aux}:

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

- LP L_{aux} (slack form):

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

	x_1	x_2	x_0	x_3	x_4
L_{aux} : -z= 0	$\overline{c_1}$=0	$\overline{c_2}$=0	$\overline{c_0}$= 1	$\overline{c_3}$= 0	$\overline{c_4}$=0
$\mathbf{x_{B_1}} = b'_1$=-2	-1	-1	-1	1	0
$\mathbf{x_{B_2}} = b'_2$=1	1	1	-1	0	1

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_3}, \mathbf{a_4}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B}^{-1}\mathbf{b} \\ \mathbf{0} \end{bmatrix} = [0, 0, 0, -2, 1]^T$ 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
L_{aux} : -z= -2	$\overline{c_1}$= -1	$\overline{c_2}$=-1	$\overline{c_0}$= 0	$\overline{c_3}$= 1	$\overline{c_4}$=1
$\mathbf{x_{B_1}}$ $= b'_1$=2	1	1	1	-1	0
$\mathbf{x_{B_2}}$ $= b'_2$=3	2	2	0	-1	1

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_0}, \mathbf{a_4}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [0, 0, 2, 0, 3]^T$ is feasible.
- Pivoting: choose $\mathbf{a_2}$ to enter basis since $\overline{c_2} = -1 < 0$; choose $\mathbf{a_4}$ to exit since $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_2}{\lambda_2} = \frac{2}{3}$.
- Here, the corresponding λ is stored in the 2-nd column (Why? the basis $\mathbf{B}$ forms an identity matrix.)

	x_1	x_2	x_0	x_3	x_4
L_{aux} : -z$= -\frac{1}{2}$	$\overline{c_1}$= 0	$\overline{c_2}$= 0	$\overline{c_0}$= 0	$\overline{c_3} = \frac{1}{2}$	$\overline{c_4} = \frac{1}{2}$
$\mathbf{x_{B_1}} = b'_1 = \frac{1}{2}$	0	0	1	$-\frac{1}{2}$	$-\frac{1}{2}$
$\mathbf{x_{B_2}} = b'_2 = \frac{3}{2}$	1	1	0	$-\frac{1}{2}$	$\frac{1}{2}$

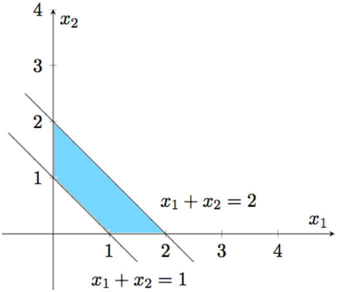
- Basis (in blue): $\mathbf{B} = \{\mathbf{a_2}, \mathbf{a_0}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [0, \frac{3}{2}, \frac{1}{2}, 0, 0]^T$ is feasible.
- Optimal solution found since $\overline{c_j} > 0$ for all j.
- The objective value of L_{aux}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

InitializeSimplex: an example with a feasible solution

An example with a feasible solution

LP L:

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



Constructing L_{aux}

- LP L:

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

- LP L_{aux}:

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

- LP L_{aux} (slack form):

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

Step 1

	x_1	x_2	x_0	x_3	x_4
L_{aux} : -z= 0	$\overline{c_1}$=0	$\overline{c_2}$=0	$\overline{c_0}$= 1	$\overline{c_3}$= 0	$\overline{c_4}$=0
$\mathbf{x_{B_1}} = b'_1$=-1	-1	-1	-1	1	0
$\mathbf{x_{B_2}} = b'_2$=2	1	1	-1	0	1

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_3}, \mathbf{a_4}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [0, 0, 0, -1, 2]^T$ 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
L_{aux} : -z= -1	$\overline{c_1}$=-1	$\overline{c_2}$= -1	$\overline{c_0}$= 0	$\overline{c_3}$= 1	$\overline{c_4}$=0
$\mathbf{x_{B_1}} = b'_1$=1	1	1	1	-1	0
$\mathbf{x_{B_2}} = b'_2$=3	2	2	0	-1	1

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_0}, \mathbf{a_4}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [0, 0, 1, 0, 3]^T$ is feasible.
- Pivoting: choose $\mathbf{a_2}$ to enter basis since $\overline{c_2} = -1 < 0$; choose $\mathbf{a_0}$ to exit since $\theta = \min_{\mathbf{a}_i \in \mathbf{B}, \lambda_i > 0} \frac{b'_i}{\lambda_i} = \frac{b'_1}{\lambda_1} = 1$.
- Here, the corresponding λ is stored in the 1-st column (Why? the basis $\mathbf{B}$ forms an identity matrix.)

Step 3

	x_1	x_2	x_0	x_3	x_4
L_{aux} : -z= 0	$\overline{c_1}$=0	$\overline{c_2}$=0	$\overline{c_0}$= 0	$\overline{c_3}$= 1	$\overline{c_4}$=0
$\mathbf{x_{B_1}} = b'_1$=1	1	1	1	-1	0
$\mathbf{x_{B_2}} = b'_2$=1	0	0	-2	1	1

- Basis (in blue): $\mathbf{B} = \{\mathbf{a_2}, \mathbf{a_4}\}$
- Solution: $\mathbf{x} = \begin{bmatrix} \mathbf{B^{-1}b} \\ \mathbf{0} \end{bmatrix} = [0, 1, 0, 0, 1]^T$ 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.

LP L (slack form):

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

	x_1	x_2	x_3	x_4
$L:$ -z= 0	$\overline{c_1}$=1	$\overline{c_2}$=2	$\overline{c_3}$= 0	$\overline{c_4}$=0
$\mathbf{x_{B_1}} = b'_1$=1	1	1	-1	0
$\mathbf{x_{B_2}} = b'_2$=1	0	0	1	1

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

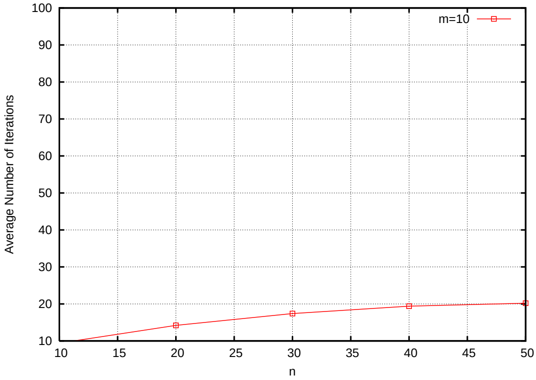
	x_1	x_2	x_3	x_4
$L:$ -z= -2	$\overline{c_1}$=-1	$\overline{c_2}$=0	$\overline{c_3}$= 3	$\overline{c_4}$=0
$\mathbf{x_{B_1}} = b'_1$=1	1	1	-1	0
$\mathbf{x_{B_2}} = b'_2$=1	0	0	1	1

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

How fast is the SIMPLEX method?

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: $O(m^2 n)$ due to $O(m)$ pivoting operations.
3. For sparse matrix: $O(K m^{\alpha} n d^{0.33})$, where K is a constant, $1.25 \leq \alpha \leq 2.5$, d is the ratio of non-zero entries of matrix $\mathbf{A}$.

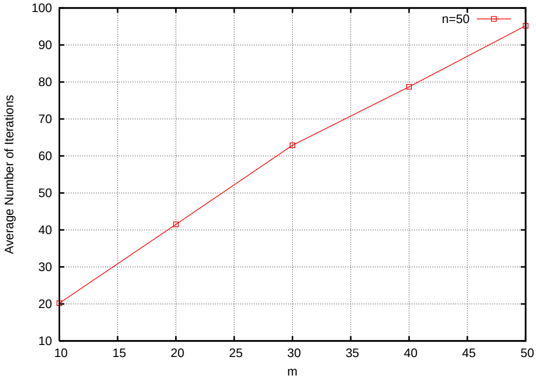
How does the iteration number change with n?



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

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

How does the iteration number change with m?



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

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]

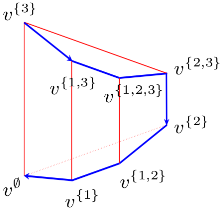
$$\begin{array}{rrcccl}
\max & & & x_n & & & \\
s.t. & \delta & \leq & x_i & \leq & 1 & \text{for } i = 1..n \\
 & \delta x_{i-1} & \leq & x_i & \leq & 1 - \delta x_{i-1} & \text{for } i = 2..n \\
 & & & x_i & \geq & 0 & \text{for } i = 1..n
\end{array}$$

Simplex algorithm might visit all the 2^n vertices.

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

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

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. $[1, \frac{3}{4}, \frac{13}{16}]$

(See extra slides)

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

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

Spielman and Teng showed that the simplex algorithm has polynomial smoothed complexity.

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.

(see a demo)

Appendix

Appendix: preliminary knowledge of optimization

- Convex combination: $\lambda x_1 + (1-\lambda)x_2$. $0 \le \lambda \le 1$
- Convex set: S is a convex set iff for any two elements $s_1, s_2 \in S$, the combination of s_1 and s_2 is still in S.
- Convex function: $f(x)$ is a convex function iff for any $x_1, x_2, 0 \le \lambda \le 1$, $f(\lambda x_1 + (1-\lambda)x_2) \ge \lambda f(x_1) + (1-\lambda) f(x_2)$.

Note: a local optimum is also a global optimum for a convex optimization problem.
(see an extra slide)

Appendix: some facts of polytope

- Fact 1: A vertex cannot be represented as a convex combination of two points in $\mathbf{P}$.
- Fact 2: Any point in P is a convex combination of the vertices of $\mathbf{P}$.

A note on "$\geq$" v.s. "$>$"

Note:

- A constraint should be "$\geq$" rather than "$>$". Why? See an example:
- The LP model

$$\begin{array}{rr} \max & x \\ s.t. & x \leq 1 \end{array}$$

  has optimal solutions.
- However, the LP model

$$\begin{array}{rr} \max & x \\ s.t. & x < 1 \end{array}$$

  doesn't have optimal solutions.