### NEW CS 473: Theory II, Fall 2015

## **Linear Programming II**

Lecture 19 October 29, 2015

# 19.1: The Simplex Algorithm in Detail

```
Simplex(\hat{L} a LP)
        Transform L into slack form.
        Let L be the resulting slack form.
        L' \leftarrow \mathsf{Feasible}(L)
        x \leftarrow \mathsf{LPStartSolution}(L')
        x' \leftarrow \mathsf{SimplexInner}(L', x)
        z \leftarrow objective function value of x'
        if z > 0 then
              return "No solution"
        x'' \leftarrow \mathsf{SimplexInner}(L, x')
        return x''
```

- **SimplexInner**: solves a LP if the trivial solution of assigning zero to all the nonbasic variables is feasible.
- ②  $L' = \mathsf{Feasible}(L)$  returns a new LP with feasible solution.
- ① Done by adding new variable  $x_0$  to each equality.
- lacksquare Set target function in L' to  $\min x_0$
- ullet original LP L feasible  $\iff$  LP L' has feasible solution with  $x_0=0.$
- **o** Apply **SimplexInner** to L' and solution computed (for L') by **LPStartSolution**(L').
- $\bigcirc$  If  $x_0 = 0$  then have a feasible solution to L.
- $\odot$  Use solution in **SimplexInner** on L.
- need to describe SimplexInner: solve LP in slack form given a feasible solution (all nonbasic vars assigned value 0).

- **SimplexInner**: solves a LP if the trivial solution of assigning zero to all the nonbasic variables is feasible.
- ② L' = Feasible(L) returns a new LP with feasible solution.
- ullet Done by adding new variable  $x_0$  to each equality.
- $ext{ } ext{ }$
- ullet original LP L feasible  $\iff$  LP L' has feasible solution with  $x_0=0.$
- lacktriangledown Apply SimplexInner to L' and solution computed (for L') by LPStartSolution(L').
- $\bigcirc$  If  $x_0 = 0$  then have a feasible solution to L.
- $\odot$  Use solution in **SimplexInner** on L.
- need to describe SimplexInner: solve LP in slack form given a feasible solution (all nonbasic vars assigned value 0).

- SimplexInner: solves a LP if the trivial solution of assigning zero to all the nonbasic variables is feasible.
- ② L' = Feasible(L) returns a new LP with feasible solution.
- **③** Done by adding new variable  $x_0$  to each equality.
- Set target function in L' to  $\min x_0$ .
- ullet original LP L feasible  $\iff$  LP L' has feasible solution with  $x_0=0.$
- **1** Apply SimplexInner to L' and solution computed (for L') by LPStartSolution(L').
- lacksquare Use solution in **SimplexInner** on L.
- o need to describe SimplexInner: solve LP in slack form given a feasible solution (all nonbasic vars assigned value 0).

- SimplexInner: solves a LP if the trivial solution of assigning zero to all the nonbasic variables is feasible.
- ② L' = Feasible(L) returns a new LP with feasible solution.
- **③** Done by adding new variable  $x_0$  to each equality.
- Set target function in L' to  $\min x_0$ .
- ullet original  $\operatorname{LP} L$  feasible  $\iff \operatorname{LP} L'$  has feasible solution with  $x_0=0$ .
- lacktriangledown Apply SimplexInner to L' and solution computed (for L') by LPStartSolution(L').
- $\odot$  Use solution in **SimplexInner** on L.
- need to describe SimplexInner: solve LP in slack form given a feasible solution (all nonbasic vars assigned value 0).

- SimplexInner: solves a LP if the trivial solution of assigning zero to all the nonbasic variables is feasible.
- ② L' = Feasible(L) returns a new LP with feasible solution.
- **③** Done by adding new variable  $x_0$  to each equality.
- Set target function in L' to  $\min x_0$ .
- ullet original  $\begin{subarray}{c} \end{subarray} LP\ L'$  has feasible solution with  $x_0=0.$
- Apply SimplexInner to L' and solution computed (for L') by LPStartSolution(L').
- lacksquare Use solution in **SimplexInner** on L.
- o need to describe SimplexInner: solve LP in slack form given a feasible solution (all nonbasic vars assigned value 0).

- SimplexInner: solves a LP if the trivial solution of assigning zero to all the nonbasic variables is feasible.
- ② L' = Feasible(L) returns a new LP with feasible solution.
- **③** Done by adding new variable  $x_0$  to each equality.
- Set target function in L' to  $\min x_0$ .
- lacktriangledown original  $f LP\ L$  feasible  $\iff f LP\ L'$  has feasible solution with  $x_0=0.$
- **Output** Apply SimplexInner to L' and solution computed (for L') by LPStartSolution(L').
- **1** If  $x_0 = 0$  then have a feasible solution to L.
- lacksquare Use solution in **SimplexInner** on  $oldsymbol{L}$ .
- need to describe SimplexInner: solve LP in slack form given a feasible solution (all nonbasic vars assigned value 0).

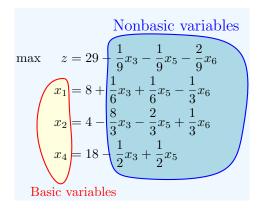
- SimplexInner: solves a LP if the trivial solution of assigning zero to all the nonbasic variables is feasible.
- ② L' = Feasible(L) returns a new LP with feasible solution.
- **③** Done by adding new variable  $x_0$  to each equality.
- **9** Set target function in L' to  $\min x_0$ .
- lacktriangledown original  $f LP\ L$  feasible  $\iff f LP\ L'$  has feasible solution with  $x_0=0.$
- **Output** Apply SimplexInner to L' and solution computed (for L') by LPStartSolution(L').
- If  $x_0 = 0$  then have a feasible solution to L.
- **1** Use solution in **SimplexInner** on L.
- need to describe **SimplexInner**: solve LP in slack form given a feasible solution (all nonbasic vars assigned value 0).

```
B - Set of indices of basic variables
N - Set of indices of nonbasic variables
n = |N| - number of original variables
b, c - two vectors of constants
m = |B| - number of basic variables (i.e., number
of inequalities)
A = \{a_{ij}\} - The matrix of coefficients
N \cup B = \{1, \dots, n+m\}
\boldsymbol{v} - objective function constant.
LP in slack form is specified by a tuple (N, B, A, b, c, v).
```

### The corresponding LP

$$egin{array}{ll} \max & z=v+\sum_{j\in N}c_jx_j, \ & ext{s.t.} & x_i=b_i-\sum_{j\in N}a_{ij}x_j ext{ for } i\in B, \ & x_i\geq 0, & orall i=1,\dots,n+m. \end{array}$$

### Reminder - basic/nonbasic



#### Description **SimplexInner** algorithm:

- **1** LP is in slack form.
- 2 Trivial solution x= au (i.e., all nonbasic variables zero), is feasible.
- $\odot$  objective value for this solution is v.
- **©** Reminder: Objective function is  $z = v + \sum_{j \in N} c_j x_j$ .
- x<sub>e</sub>: nonbasic variable with positive coefficient in objective function.
- ullet Formally: e is one of the indices of  $\left\{ j \; \middle| \; c_j > 0, j \in N 
  ight\}$ .
- $oldsymbol{0}$   $x_e$  is the **entering variable** (enters set of basic variables).
- $\odot$  If increase value  $x_e$  (from current value of 0 in au)...
- one of basic variables is going to vanish (i.e., become zero).

#### Description **SimplexInner** algorithm:

- 1 LP is in slack form.
- 2 Trivial solution  $x = \tau$  (i.e., all nonbasic variables zero), is feasible.
- $\odot$  objective value for this solution is v.
- ① Reminder: Objective function is  $z = v + \sum_{j \in N} c_j x_j$ .
- $oldsymbol{x}_e$ : nonbasic variable with positive coefficient in objective function.
- lacksquare Formally: e is one of the indices of  $\Big\{ j \ \Big| \ c_j > 0, j \in N \, \Big\}.$
- $oldsymbol{0}$   $x_e$  is the **entering variable** (enters set of basic variables).
- lacksquare If increase value  $x_e$  (from current value of 0 in au)...
- one of basic variables is going to vanish (i.e., become zero).

#### Description **SimplexInner** algorithm:

- LP is in slack form.
- 2 Trivial solution  $x = \tau$  (i.e., all nonbasic variables zero), is feasible.
- $oldsymbol{\circ}$  objective value for this solution is  $oldsymbol{v}$ .
- **1** Reminder: Objective function is  $z = v + \sum_{j \in N} c_j x_j$ .
- $oldsymbol{x}_e$ : nonbasic variable with positive coefficient in objective function.
- $exttt{ igstar}$  Formally: e is one of the indices of  $\Big\{ j \ \Big| \ c_j > 0, j \in N \, \Big\}.$
- $oldsymbol{0}$   $x_e$  is the **entering variable** (enters set of basic variables).
- ullet If increase value  $x_e$  (from current value of 0 in au)...
- one of basic variables is going to vanish (i.e., become zero).

#### Description **SimplexInner** algorithm:

- LP is in slack form.
- ② Trivial solution  $x = \tau$  (i.e., all nonbasic variables zero), is feasible.
- $oldsymbol{\circ}$  objective value for this solution is  $oldsymbol{v}$ .
- **1** Reminder: Objective function is  $z = v + \sum_{j \in N} c_j x_j$ .
- $oldsymbol{x}_e$ : nonbasic variable with positive coefficient in objective function.
- $lackbox{0}$  Formally: e is one of the indices of  $\Big\{ j \ \Big| \ c_j > 0, j \in N \Big\}.$
- $oldsymbol{0}$   $x_e$  is the **entering variable** (enters set of basic variables).
- @ If increase value  $x_e$  (from current value of 0 in au)...
- one of basic variables is going to vanish (i.e., become zero).

#### Description **SimplexInner** algorithm:

- LP is in slack form.
- ② Trivial solution  $x = \tau$  (i.e., all nonbasic variables zero), is feasible.
- ullet objective value for this solution is  $oldsymbol{v}$ .
- **1** Reminder: Objective function is  $z = v + \sum_{j \in N} c_j x_j$ .
- $oldsymbol{x}_e$ : nonbasic variable with positive coefficient in objective function.
- $lackbox{0}$  Formally: e is one of the indices of  $\Big\{ j \ \Big| \ c_j > 0, j \in N \Big\}.$
- **0**  $x_e$  is the **entering variable** (enters set of basic variables).
- @ If increase value  $x_e$  (from current value of 0 in au)...
- one of basic variables is going to vanish (i.e., become zero).

#### Description **SimplexInner** algorithm:

- LP is in slack form.
- 2 Trivial solution  $x = \tau$  (i.e., all nonbasic variables zero), is feasible.
- ullet objective value for this solution is  $oldsymbol{v}$ .
- **9** Reminder: Objective function is  $z = v + \sum_{j \in N} c_j x_j$ .
- $oldsymbol{x}_e$ : nonbasic variable with positive coefficient in objective function.
- lacktriangledown Formally: e is one of the indices of  $\Big\{ j \ \Big| \ c_j > 0, j \in N \Big\}.$
- **0**  $x_e$  is the **entering variable** (enters set of basic variables).
- lacktriangledown If increase value  $x_e$  (from current value of 0 in au)...
- one of basic variables is going to vanish (i.e., become zero).

- $oldsymbol{1}{2} x_e$ : entering variable
- ②  $x_l$ : leaving variable vanishing basic variable
- $ext{ on increase value of } x_e ext{ till } x_l ext{ becomes zero}$
- ullet How do we now which variable is  $x_l$ ?
- ${ to}$  set all nonbasic to  ${ to}$  zero, except  $x_e$
- $lacksquare{0}$  Require:  $orall i \in B$   $x_i = b_i a_{ie} x_e \geq 0$ .
- $lacksquare x_e \leq (b_i/a_{ie})$
- @ If more than one achieves  $\min_i b_i/a_{ie}$ , just pick one.

- $oldsymbol{1}{2} x_e$ : entering variable
- ②  $x_l$ : leaving variable vanishing basic variable.
- $ext{ on }$  increase value of  $x_e$  till  $x_l$  becomes zero
- ullet How do we now which variable is  $x_l$ ?
- ${ to}$  set all nonbasic to  ${ to}$  zero, except  $x_e$
- $lacksquare{0}$  Require:  $orall i \in B$   $x_i = b_i a_{ie}x_e \geq 0$ .
- $lacksquare x_e \leq (b_i/a_{ie})$
- @ If more than one achieves  $\min_i b_i/a_{ie}$ , just pick one.

- **1**  $x_e$ : entering variable
- ②  $x_l$ : leaving variable vanishing basic variable.
- ullet increase value of  $x_e$  till  $x_l$  becomes zero.
- ${ to}$  set all nonbasic to  ${ to}$  zero, except  $x_e$
- $lacksquare{0}$  Require:  $orall i \in B$   $x_i = b_i a_{ie}x_e \geq 0$ .
- $lacksquare x_e \leq (b_i/a_{ie})$
- @ If more than one achieves  $\min_i b_i/a_{ie}$ , just pick one.

- $oldsymbol{1}{2} x_e$ : entering variable
- ②  $x_l$ : leaving variable vanishing basic variable.
- $oldsymbol{0}$  increase value of  $x_e$  till  $x_l$  becomes zero.
- How do we now which variable is  $x_l$ ?
- ullet set all nonbasic to 0 zero, except  $x_e$
- $lacksquare{0}$  Require:  $orall i \in B$   $x_i = b_i a_{ie}x_e \geq 0$ .
- $lacksquare x_e \leq (b_i/a_{ie})$
- @ If more than one achieves  $\min_i b_i/a_{ie}$ , just pick one.

- **1**  $x_e$ : entering variable
- ②  $x_l$ : leaving variable vanishing basic variable.
- ullet increase value of  $x_e$  till  $x_l$  becomes zero.
- ullet How do we now which variable is  $x_l$ ?
- $oldsymbol{\circ}$  set all nonbasic to 0 zero, except  $x_e$
- $m{0}$  Require:  $orall i \in B$   $x_i = b_i a_{ie}x_e \geq 0$ .

- @ If more than one achieves  $\min_i b_i/a_{ie}$ , just pick one.

- **1**  $x_e$ : entering variable
- ②  $x_l$ : leaving variable vanishing basic variable.
- lacktriangle increase value of  $x_e$  till  $x_l$  becomes zero.
- ullet How do we now which variable is  $x_l$ ?
- $oldsymbol{\circ}$  set all nonbasic to 0 zero, except  $x_e$
- $ext{ @ Require: } orall i \in B \quad x_i = b_i a_{ie} x_e \geq 0.$

- $_{0}$  If more than one achieves  $\min_{i}b_{i}/a_{ie}$ , just pick one

- **1**  $x_e$ : entering variable
- ②  $x_l$ : leaving variable vanishing basic variable.
- $oldsymbol{0}$  increase value of  $x_e$  till  $x_l$  becomes zero.
- How do we now which variable is  $x_l$ ?
- $oldsymbol{\mathfrak{o}}$  set all nonbasic to  $oldsymbol{\mathfrak{o}}$  zero, except  $oldsymbol{x}_e$
- $lackbox{0}$  Require:  $orall i \in B$   $x_i = b_i a_{ie}x_e \geq 0$ .

- $_{0}$  If more than one achieves  $\min_{i}b_{i}/a_{ie}$ , just pick one

- $oldsymbol{0}$   $x_e$ : entering variable
- ②  $x_l$ : leaving variable vanishing basic variable.
- ullet increase value of  $x_e$  till  $x_l$  becomes zero.
- How do we now which variable is  $x_l$ ?
- $oldsymbol{\mathfrak{o}}$  set all nonbasic to  $oldsymbol{\mathfrak{o}}$  zero, except  $oldsymbol{x}_e$
- $lackbox{0}$  Require:  $orall i \in B$   $x_i = b_i a_{ie} x_e \geq 0$ .

- @ If more than one achieves  $\min_i b_i/a_{ie}$ , just pick one.

10 / 16

- **1**  $x_e$ : entering variable
- ②  $x_l$ : leaving variable vanishing basic variable.
- $oldsymbol{0}$  increase value of  $x_e$  till  $x_l$  becomes zero.
- How do we now which variable is  $x_l$ ?
- $oldsymbol{\mathfrak{g}}$  set all nonbasic to  $oldsymbol{0}$  zero, except  $oldsymbol{x}_e$
- $lackbox{0}$  Require:  $orall i \in B$   $x_i = b_i a_{ie}x_e \geq 0$ .

- $_{0}$  If more than one achieves  $\min_{i}b_{i}/a_{ie}$ , just pick one.

- $oldsymbol{0}$   $x_e$ : entering variable
- ②  $x_l$ : leaving variable vanishing basic variable.
- ullet increase value of  $x_e$  till  $x_l$  becomes zero.
- How do we now which variable is  $x_l$ ?
- $oldsymbol{\mathfrak{g}}$  set all nonbasic to  $oldsymbol{0}$  zero, except  $oldsymbol{x}_e$
- $lackbox{0}$  Require:  $orall i \in B$   $x_i = b_i a_{ie}x_e \geq 0$ .

- lacktriangle If more than one achieves  $\min_i b_i/a_{ie}$ , just pick one.

- **1** Determined  $x_e$  and  $x_l$ .
- ② Rewrite equation for  $x_l$  in LP.
  - (Every basic variable has an equation in the LP!)
  - $x_l = b_l \sum_{j \in N} a_{lj} x_j \ \implies x_e = rac{b_l}{a_{le}} \sum_{j \in N \cup \{l\}} rac{a_{lj}}{a_{le}} x_j, \qquad ext{where } a_{ll} = 1.$
- $exttt{ iny Cleanup: remove all appearances (on right) in LP of } x_e.$
- ullet Substituting  $x_e$  into the other equalities, using above.
- On Alternatively, do Gaussian elimination remove any appearance of  $x_e$  on right side LP (including objective).

  Transfer  $x_t$  on the left side, to the right side.

- **1** Determined  $x_e$  and  $x_l$ .
- ② Rewrite equation for  $x_l$  in LP.
  - (Every basic variable has an equation in the LP!)

② 
$$x_l = b_l - \sum_{j \in N} a_{lj} x_j$$
  $\implies x_e = \frac{b_l}{a_{le}} - \sum_{j \in N \cup \{l\}} \frac{a_{lj}}{a_{le}} x_j, \qquad ext{where } a_{ll} = 1.$ 

- $exttt{ iny Cleanup: remove all appearances (on right) in LP of } x_e.$
- ullet Substituting  $x_e$  into the other equalities, using above.
- Alternatively, do Gaussian elimination remove any appearance of  $x_e$  on right side LP (including objective). Transfer  $x_t$  on the left side, to the right side.

- **1** Determined  $x_e$  and  $x_l$ .
- 2 Rewrite equation for  $x_l$  in LP.
  - (Every basic variable has an equation in the LP!)

$$egin{aligned} oldsymbol{x}_l &= b_l - \sum_{j \in N} a_{lj} x_j \ &\Longrightarrow \quad x_e &= rac{b_l}{a_{le}} - \sum_{j \in N \cup \{l\}} rac{a_{lj}}{a_{le}} x_j, \qquad ext{where } a_{ll} = 1. \end{aligned}$$

- Oleanup: remove all appearances (on right) in LP of  $x_e$ .
- ① Substituting  $x_e$  into the other equalities, using above.
- Alternatively, do Gaussian elimination remove any appearance of  $x_e$  on right side LP (including objective). Transfer  $x_t$  on the left side, to the right side.

- **1** Determined  $x_e$  and  $x_l$ .
- ② Rewrite equation for  $x_l$  in LP.
  - (Every basic variable has an equation in the LP!)
  - $a_l = b_l \sum_{j \in N} a_{lj} x_j$

$$\implies \quad x_e = rac{b_l}{a_{le}} - \sum_{j \in N \cup \{l\}} rac{a_{lj}}{a_{le}} x_j, \qquad ext{where } a_{ll} = 1.$$

- $exttt{ iny Cleanup: remove all appearances (on right) in LP of <math>x_e.$
- ① Substituting  $x_e$  into the other equalities, using above.
- ① Alternatively, do Gaussian elimination remove any appearance of  $x_e$  on right side LP (including objective).

  Transfer  $x_t$  on the left side, to the right side.

- **1** Determined  $x_e$  and  $x_l$ .
- ② Rewrite equation for  $x_l$  in LP.
  - (Every basic variable has an equation in the LP!)
  - $egin{aligned} oldsymbol{a} & x_l = b_l \sum_{j \in N} a_{lj} x_j \ & \Longrightarrow & x_e = rac{b_l}{a_{le}} \sum_{j \in N \cup \{l\}} rac{a_{lj}}{a_{le}} x_j, \qquad ext{where } a_{ll} = 1. \end{aligned}$
- $exttt{ iny Cleanup: remove all appearances (on right) in LP of <math>x_e.$
- ① Substituting  $x_e$  into the other equalities, using above.
- ① Alternatively, do Gaussian elimination remove any appearance of  $x_e$  on right side LP (including objective).

  Transfer  $x_t$  on the left side, to the right side.

- **1** Determined  $x_e$  and  $x_l$ .
- 2 Rewrite equation for  $x_l$  in LP.
  - (Every basic variable has an equation in the LP!)
  - $egin{aligned} oldsymbol{a} & x_l = b_l \sum_{j \in N} a_{lj} x_j \ & \Longrightarrow & x_e = rac{b_l}{a_{le}} \sum_{j \in N \cup \{l\}} rac{a_{lj}}{a_{le}} x_j, \qquad ext{where } a_{ll} = 1. \end{aligned}$
- **3** Cleanup: remove all appearances (on right) in LP of  $x_e$ .
- ① Substituting  $x_e$  into the other equalities, using above.
- ① Alternatively, do Gaussian elimination remove any appearance of  $x_e$  on right side LP (including objective).

  Transfer  $x_t$  on the left side, to the right side.

# Pivoting on x<sub>e</sub>...

- **1** Determined  $x_e$  and  $x_l$ .
- 2 Rewrite equation for  $x_l$  in LP.
  - (Every basic variable has an equation in the LP!)
  - $egin{aligned} oldsymbol{a} & x_l = b_l \sum_{j \in N} a_{lj} x_j \ & \Longrightarrow & x_e = rac{b_l}{a_{le}} \sum_{j \in N \cup \{l\}} rac{a_{lj}}{a_{le}} x_j, \qquad ext{where } a_{ll} = 1. \end{aligned}$
- **3** Cleanup: remove all appearances (on right) in LP of  $x_e$ .
- lacksquare Substituting  $x_e$  into the other equalities, using above.
- ullet Alternatively, do Gaussian elimination remove any appearance of  $x_e$  on right side LP (including objective). Transfer  $x_l$  on the left side, to the right side.

# Pivoting on x<sub>e</sub>...

- **1** Determined  $x_e$  and  $x_l$ .
- 2 Rewrite equation for  $x_l$  in LP.
  - (Every basic variable has an equation in the LP!)
  - $egin{aligned} oldsymbol{a} & x_l = b_l \sum_{j \in N} a_{lj} x_j \ & \Longrightarrow & x_e = rac{b_l}{a_{le}} \sum_{j \in N \cup \{l\}} rac{a_{lj}}{a_{le}} x_j, \qquad ext{where } a_{ll} = 1. \end{aligned}$
- ullet Cleanup: remove all appearances (on right) in  $\operatorname{LP}$  of  $x_e$ .
- lacksquare Substituting  $x_e$  into the other equalities, using above.
- **3** Alternatively, do Gaussian elimination remove any appearance of  $x_e$  on right side LP (including objective). Transfer  $x_l$  on the left side, to the right side.

- End of this process: have new equivalent LP.
- ② basic variables:  $B' = (B \setminus \{l\}) \cup \{e\}$
- 3 non-basic variables:  $N' = (N \setminus \{e\}) \cup \{l\}$ .
- End of this **pivoting** stage:LP objective function value increased
- Made progress.
- LP is completely defined by which variables are basic, and which are non-basic.
- Pivoting never returns to a combination (of basic/non-basic variable) already visited.
- ...because improve objective in each pivoting step.
- ① Can do at most  $\binom{n+m}{n} \leq \left(\frac{n+m}{n} \cdot e\right)^n$ .
- $oldsymbol{0}$  examples where  $oldsymbol{2}^n$  pivoting steps are needed.

- End of this process: have new equivalent LP.
- $oldsymbol{0}$  basic variables:  $B' = (B \setminus \{l\}) \cup \{e\}$
- non-basic variables:  $N' = (N \setminus \{e\}) \cup \{l\}.$
- End of this **pivoting** stage:LP objective function value increased
- Made progress.
- LP is completely defined by which variables are basic, and which are non-basic.
- Pivoting never returns to a combination (of basic/non-basic variable) already visited.
- ...because improve objective in each pivoting step.
- ① Can do at most  $\binom{n+m}{n} \le \left(\frac{n+m}{n} \cdot e\right)^n$ .
- ${ t @}$  examples where  $2^n$  pivoting steps are needed.

- End of this process: have new equivalent LP.
- $oldsymbol{0}$  basic variables:  $B' = (B \setminus \{l\}) \cup \{e\}$
- lacksquare non-basic variables:  $N' = (N \setminus \{e\}) \cup \{l\}$ .
- End of this **pivoting** stage:LP objective function value increased
- Made progress.
- LP is completely defined by which variables are basic, and which are non-basic.
- Pivoting never returns to a combination (of basic/non-basic variable) already visited.
- ...because improve objective in each pivoting step.
- ① Can do at most  $\binom{n+m}{n} \leq \left(\frac{n+m}{n} \cdot e\right)^n$ .
- $oldsymbol{0}$  examples where  $oldsymbol{2}^n$  pivoting steps are needed.

- End of this process: have new equivalent LP.
- $oldsymbol{0}$  basic variables:  $B' = (B \setminus \{l\}) \cup \{e\}$
- lacksquare non-basic variables:  $N' = (N \setminus \{e\}) \cup \{l\}$ .
- End of this **pivoting** stage:
  LP objective function value increased.
- Made progress.
- LP is completely defined by which variables are basic, and which are non-basic.
- Pivoting never returns to a combination (of basic/non-basic variable) already visited.
- ...because improve objective in each pivoting step.
- ① Can do at most  $\binom{n+m}{n} \leq \left(\frac{n+m}{n} \cdot e\right)^n$ .
- $\bigcirc$  examples where  $2^n$  pivoting steps are needed.

- End of this process: have new equivalent LP.
- $oldsymbol{0}$  basic variables:  $B' = (B \setminus \{l\}) \cup \{e\}$
- lacksquare non-basic variables:  $N' = (N \setminus \{e\}) \cup \{l\}$ .
- End of this **pivoting** stage:
  LP objective function value increased.
- Made progress.
- LP is completely defined by which variables are basic, and which are non-basic.
- Pivoting never returns to a combination (of basic/non-basic variable) already visited.
- ...because improve objective in each pivoting step.
- ① Can do at most  $\binom{n+m}{n} \leq \left(\frac{n+m}{n} \cdot e\right)^n$ .
- $\bigcirc$  examples where  $2^n$  pivoting steps are needed.

- End of this process: have new equivalent LP.
- $oldsymbol{0}$  basic variables:  $B' = (B \setminus \{l\}) \cup \{e\}$
- lacksquare non-basic variables:  $N' = (N \setminus \{e\}) \cup \{l\}$ .
- End of this **pivoting** stage:
  LP objective function value increased.
- Made progress.
- LP is completely defined by which variables are basic, and which are non-basic.
- Pivoting never returns to a combination (of basic/non-basic variable) already visited.
- ...because improve objective in each pivoting step.
- ① Can do at most  $\binom{n+m}{n} \le \left(\frac{n+m}{n} \cdot e\right)^n$ .
- $ext{@}$  examples where  $2^n$  pivoting steps are needed.

- End of this process: have new equivalent LP.
- $oldsymbol{2}$  basic variables:  $B' = (B \setminus \{l\}) \cup \{e\}$
- lacksquare non-basic variables:  $N' = (N \setminus \{e\}) \cup \{l\}$ .
- End of this **pivoting** stage:
  LP objective function value increased.
- Made progress.
- LP is completely defined by which variables are basic, and which are non-basic.
- Pivoting never returns to a combination (of basic/non-basic variable) already visited.
- ...because improve objective in each pivoting step.
- ① Can do at most  $\binom{n+m}{n} \le \left(\frac{n+m}{n} \cdot e\right)^n$ .
- $\bigcirc$  examples where  $2^n$  pivoting steps are needed.

- End of this process: have new equivalent LP.
- $oldsymbol{0}$  basic variables:  $B' = (B \setminus \{l\}) \cup \{e\}$
- lacksquare non-basic variables:  $N' = (N \setminus \{e\}) \cup \{l\}$ .
- End of this **pivoting** stage:
  LP objective function value increased.
- Made progress.
- LP is completely defined by which variables are basic, and which are non-basic.
- Pivoting never returns to a combination (of basic/non-basic variable) already visited.
- ...because improve objective in each pivoting step.
- ① Can do at most  $\binom{n+m}{n} \leq \left(\frac{n+m}{n} \cdot e\right)^n$ .
- ullet examples where  $2^n$  pivoting steps are needed

- End of this process: have new equivalent LP.
- $oldsymbol{0}$  basic variables:  $B' = (B \setminus \{l\}) \cup \{e\}$
- lacksquare non-basic variables:  $N' = (N \setminus \{e\}) \cup \{l\}$ .
- End of this **pivoting** stage:
  LP objective function value increased.
- Made progress.
- LP is completely defined by which variables are basic, and which are non-basic.
- Pivoting never returns to a combination (of basic/non-basic variable) already visited.
- ...because improve objective in each pivoting step.
- ${ t @}$  examples where  $2^n$  pivoting steps are needed.

- End of this process: have new equivalent LP.
- $oldsymbol{2}$  basic variables:  $B' = (B \setminus \{l\}) \cup \{e\}$
- lacksquare non-basic variables:  $N' = (N \setminus \{e\}) \cup \{l\}$ .
- End of this **pivoting** stage:
   LP objective function value increased.
- Made progress.
- LP is completely defined by which variables are basic, and which are non-basic.
- Pivoting never returns to a combination (of basic/non-basic variable) already visited.
- ...because improve objective in each pivoting step.
- $\mathbf{0}$  examples where  $\mathbf{2}^n$  pivoting steps are needed.

## Simplex algorithm summary...

- **①** Each pivoting step takes polynomial time in n and m.
- Running time of Simplex is exponential in the worst case
- In practice, Simplex is extremely fast

## Simplex algorithm summary...

- Each pivoting step takes polynomial time in n and m.
- Running time of Simplex is exponential in the worst case.
- In practice, Simplex is extremely fast

## Simplex algorithm summary...

- **Q** Each pivoting step takes polynomial time in n and m.
- Running time of Simplex is exponential in the worst case.
- In practice, Simplex is extremely fast.

- **1** Simplex might get stuck if one of the  $b_i$ s is zero.
- More than > m hyperplanes (i.e., equalities) passes through the same point.
- Result: might not be able to make any progress at all in a pivoting step.
- Solution I: add tiny random noise to each coefficient. Can be done symbolically. Intuitively, the degeneracy, being a local phenomena on the polytope disappears with high probability.

- **1** Simplex might get stuck if one of the  $b_i$ s is zero.
- ullet More than >m hyperplanes (i.e., equalities) passes through the same point.
- Result: might not be able to make any progress at all in a pivoting step.
- Solution I: add tiny random noise to each coefficient. Can be done symbolically. Intuitively, the degeneracy, being a local phenomena on the polytope disappears with high probability.

- **1** Simplex might get stuck if one of the  $b_i$ s is zero.
- ullet More than >m hyperplanes (i.e., equalities) passes through the same point.
- Result: might not be able to make any progress at all in a pivoting step.
- Solution I: add tiny random noise to each coefficient. Can be done symbolically. Intuitively, the degeneracy, being a local phenomena on the polytope disappears with high probability.

- **1** Simplex might get stuck if one of the  $b_i$ s is zero.
- ullet More than >m hyperplanes (i.e., equalities) passes through the same point.
- Result: might not be able to make any progress at all in a pivoting step.
- Solution I: add tiny random noise to each coefficient. Can be done symbolically. Intuitively, the degeneracy, being a local phenomena on the polytope disappears with high probability.

- **1** Simplex might get stuck if one of the  $b_i$ s is zero.
- ullet More than >m hyperplanes (i.e., equalities) passes through the same point.
- Result: might not be able to make any progress at all in a pivoting step.
- Solution I: add tiny random noise to each coefficient. Can be done symbolically.
  - Intuitively, the degeneracy, being a local phenomena on the polytope disappears with high probability.

- **1** Simplex might get stuck if one of the  $b_i$ s is zero.
- ullet More than >m hyperplanes (i.e., equalities) passes through the same point.
- Result: might not be able to make any progress at all in a pivoting step.
- Solution I: add tiny random noise to each coefficient. Can be done symbolically. Intuitively, the degeneracy, being a local phenomena on the polytope disappears with high probability.

## Degeneracies – cycling

- Might get into cycling: a sequence of pivoting operations that do not improve the objective function, and the bases you get are cyclic (i.e., infinite loop).
- ② Solution II: Bland's rule. Always choose the lowest index variable for entering and leaving out of the possible candidates. (Not prove why this work - but it does.)

## Degeneracies – cycling

- Might get into cycling: a sequence of pivoting operations that do not improve the objective function, and the bases you get are cyclic (i.e., infinite loop).
- Solution II: Bland's rule.
  Always choose the lowest index variable for entering and leaving out of the possible candidates.
  (Not prove why this work but it does.)

19.2.1: Correctness of linear programming

## Definition

A solution to an LP is a **basic solution** if it the result of setting all the nonbasic variables to zero.

Simplex algorithm deals only with basic solutions.

#### **Theorem**

For an arbitrary linear program, the following statements are true:

- (A) If there is no optimal solution, the problem is either infeasible or unbounded.
- (B) If a feasible solution exists, then a basic feasible solution exists.
- (C) If an optimal solution exists, then a basic optimal solution exists.

## **Definition**

A solution to an LP is a **basic solution** if it the result of setting all the nonbasic variables to zero.

**Simplex** algorithm deals only with basic solutions.

#### Theorem

For an arbitrary linear program, the following statements are true:

- (A) If there is no optimal solution, the problem is either infeasible or unbounded.
- (B) If a feasible solution exists, then a basic feasible solution exists.
- (C) If an optimal solution exists, then a basic optimal solution exists.

## **Definition**

A solution to an LP is a **basic solution** if it the result of setting all the nonbasic variables to zero.

**Simplex** algorithm deals only with basic solutions.

#### **Theorem**

For an arbitrary linear program, the following statements are true:

- (A) If there is no optimal solution, the problem is either infeasible or unbounded.
- (B) If a feasible solution exists, then a basic feasible solution exists.
- (C) If an optimal solution exists, then a basic optimal solution exists.

### **Definition**

A solution to an LP is a **basic solution** if it the result of setting all the nonbasic variables to zero.

**Simplex** algorithm deals only with basic solutions.

#### **Theorem**

For an arbitrary linear program, the following statements are true:

- (A) If there is no optimal solution, the problem is either infeasible or unbounded.
- (B) If a feasible solution exists, then a basic feasible solution exists.
- (C) If an optimal solution exists, then a basic optimal solution exists.

- Simplex has exponential running time in the worst case.
- ellipsoid method is weakly polynomial.
  It is polynomial in the number of bits of the input.
- Mhachian in 1979 came up with it. Useless in practice
- In 1984, Karmakar came up with a different method, called the interior-point method.
- Solution Also weakly polynomial. Quite useful in practice.
- Result in arm race between the interior-point method and the simplex method.
- BIG OPEN QUESTION: Is there strongly polynomial time algorithm for linear programming?

- Simplex has exponential running time in the worst case.
- ellipsoid method is weakly polynomial.
  It is polynomial in the number of bits of the input
- Mhachian in 1979 came up with it. Useless in practice.
- In 1984, Karmakar came up with a different method, called the interior-point method.
- Solution Also weakly polynomial. Quite useful in practice.
- Result in arm race between the interior-point method and the simplex method.
- OBIG OPEN QUESTION: Is there strongly polynomial time algorithm for linear programming?

- Simplex has exponential running time in the worst case.
- ellipsoid method is weakly polynomial. It is polynomial in the number of bits of the input.
- Khachian in 1979 came up with it. Useless in practice.
- In 1984, Karmakar came up with a different method, called the interior-point method.
- Salso weakly polynomial. Quite useful in practice.
- Result in arm race between the interior-point method and the simplex method.
- OBIG OPEN QUESTION: Is there strongly polynomial time algorithm for linear programming?

- Simplex has exponential running time in the worst case.
- **ellipsoid method** is *weakly* polynomial. It is polynomial in the number of bits of the input.
- 3 Khachian in 1979 came up with it. Useless in practice.
- In 1984, Karmakar came up with a different method, called the interior-point method.
- Solution Also weakly polynomial. Quite useful in practice.
- Result in arm race between the interior-point method and the simplex method.
- OBIG OPEN QUESTION: Is there strongly polynomial time algorithm for linear programming?

- Simplex has exponential running time in the worst case.
- ellipsoid method is weakly polynomial. It is polynomial in the number of bits of the input.
- Khachian in 1979 came up with it. Useless in practice.
- In 1984, Karmakar came up with a different method, called the interior-point method.
- Solution Also weakly polynomial. Quite useful in practice.
- Result in arm race between the interior-point method and the simplex method.
- OBIG OPEN QUESTION: Is there strongly polynomial time algorithm for linear programming?

- Simplex has exponential running time in the worst case.
- ellipsoid method is weakly polynomial. It is polynomial in the number of bits of the input.
- Khachian in 1979 came up with it. Useless in practice.
- In 1984, Karmakar came up with a different method, called the interior-point method.
- Solution Also weakly polynomial. Quite useful in practice.
- Result in arm race between the interior-point method and the simplex method.
- OBIG OPEN QUESTION: Is there strongly polynomial time algorithm for linear programming?

- Simplex has exponential running time in the worst case.
- ellipsoid method is weakly polynomial. It is polynomial in the number of bits of the input.
- Khachian in 1979 came up with it. Useless in practice.
- In 1984, Karmakar came up with a different method, called the interior-point method.
- Solution Also weakly polynomial. Quite useful in practice.
- Result in arm race between the interior-point method and the simplex method.
- BIG OPEN QUESTION: Is there strongly polynomial time algorithm for linear programming?

- Simplex has exponential running time in the worst case.
- ellipsoid method is weakly polynomial. It is polynomial in the number of bits of the input.
- Khachian in 1979 came up with it. Useless in practice.
- In 1984, Karmakar came up with a different method, called the interior-point method.
- Also weakly polynomial. Quite useful in practice.
- Result in arm race between the interior-point method and the simplex method.
- BIG OPEN QUESTION: Is there strongly polynomial time algorithm for linear programming?

# Solving LPs without ever getting into a loop - symbolic perturbations

Details in the class notes.

Sariel (UIUC) New CS473 22 Fall 2015 22 / 16

Sariel (UIUC) New CS473 23 Fall 2015 23 / 16

Sariel (UIUC) New CS473 24 Fall 2015 24 / 16