Outline

Simplex Algorithm by Example

Details of the Simplex Algorithm

Finding an Initial Solution

Appendix: Cycling and Termination (non-examinable)
Simplex Algorithm: Introduction

- classical method for solving linear programs (Dantzig, 1947)
- usually fast in practice although worst-case runtime not polynomial
- iterative procedure somewhat similar to Gaussian elimination

Basic Idea:
- Each iteration corresponds to a “basic solution” of the slack form
- All non-basic variables are 0, and the basic variables are determined from the equality constraints
- Each iteration converts one slack form into an equivalent one while the objective value will not decrease
- Conversion (“pivoting”) is achieved by switching the roles of one basic and one non-basic variable

In that sense, it is a greedy algorithm.
Extended Example: Conversion into Slack Form

maximise \( 3x_1 \ + \ x_2 \ + \ 2x_3 \)

subject to

\[
\begin{align*}
3x_1 \ + \ x_2 \ + \ 3x_3 & \leq 30 \\
2x_1 \ + \ 2x_2 \ + \ 5x_3 & \leq 24 \\
4x_1 \ + \ x_2 \ + \ 2x_3 & \leq 36 \\
x_1, x_2, x_3 & \geq 0
\end{align*}
\]

Conversion into slack form

\[
\begin{align*}
z &= 3x_1 \ + \ x_2 \ + \ 2x_3 \\
x_4 &= 30 - x_1 - x_2 - 3x_3 \\
x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\
x_6 &= 36 - 4x_1 - x_2 - 2x_3
\end{align*}
\]
Extended Example: Iteration 1

\[
\begin{align*}
    z &= 3x_1 + x_2 + 2x_3 \\
    x_4 &= 30 - x_1 - x_2 - 3x_3 \\
    x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\
    x_6 &= 36 - 4x_1 - x_2 - 2x_3
\end{align*}
\]

Basic solution: \((x_1, x_2, \ldots, x_6) = (0, 0, 0, 30, 24, 36)\)

This basic solution is **feasible**

Objective value is 0.
Extended Example: Iteration 1

Increasing the value of \( x_1 \) would increase the objective value.

\[
\begin{align*}
    z &= 3x_1 + x_2 + 2x_3 \\
    x_4 &= 30 - x_1 - x_2 - 3x_3 \\
    x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\
    x_6 &= 36 - 4x_1 - x_2 - 2x_3
\end{align*}
\]

The third constraint is the tightest and limits how much we can increase \( x_1 \).

Switch roles of \( x_1 \) and \( x_6 \):
- Solving for \( x_1 \) yields:
  \[ x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}. \]
- Substitute this into \( x_1 \) in the other three equations
Extended Example: Iteration 2

Increasing the value of $x_3$ would increase the objective value.

\[
\begin{align*}
z &= 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4} \\
x_1 &= 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4} \\
x_4 &= 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} \\
x_5 &= 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2}
\end{align*}
\]

Basic solution: \((x_1, x_2, \ldots, x_6) = (9, 0, 0, 21, 6, 0)\) with objective value 27
Extended Example: Iteration 2

\[ z = 27 + \frac{x_2}{4} + \frac{x_3}{2} - \frac{3x_6}{4} \]
\[ x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4} \]
\[ x_4 = 21 - \frac{3x_2}{4} - \frac{5x_3}{2} + \frac{x_6}{4} \]
\[ x_5 = 6 - \frac{3x_2}{2} - 4x_3 + \frac{x_6}{2} \]

The third constraint is the tightest and limits how much we can increase \( x_3 \).

Switch roles of \( x_3 \) and \( x_5 \):
- Solving for \( x_3 \) yields:
  \[ x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} - \frac{x_6}{8} \]
- Substitute this into \( x_3 \) in the other three equations
Extended Example: Iteration 3

Increasing the value of $x_2$ would increase the objective value.

\[
\begin{align*}
 z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\
 x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\
 x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\
 x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16}
\end{align*}
\]

Basic solution: $\left( x_1, x_2, \ldots, x_6 \right) = \left( \frac{33}{4}, 0, \frac{3}{2}, \frac{69}{4}, 0, 0 \right)$ with objective value $\frac{111}{4} = 27.75$.
Extended Example: Iteration 3

\[
\begin{align*}
    z &= \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \\
x_1 &= \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \\
x_3 &= \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \\
x_4 &= \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16}
\end{align*}
\]

The second constraint is the tightest and limits how much we can increase \(x_2\).

Switch roles of \(x_2\) and \(x_3\):

- Solving for \(x_2\) yields:
  \[
  x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3}.
  \]
- Substitute this into \(x_2\) in the other three equations
Extended Example: Iteration 4

All coefficients are negative, and hence this basic solution is **optimal**!

\[
\begin{align*}
  z &= 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \\
  x_1 &= 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \\
  x_2 &= 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \\
  x_4 &= 18 - \frac{x_3}{2} + \frac{x_5}{2}
\end{align*}
\]

Basic solution: \((x_1, x_2, \ldots, x_6) = (8, 4, 0, 18, 0, 0)\) with objective value 28
Extended Example: Visualization of SIMPLEX

Exercise: [Ex. 6/7.6] How many basic solutions (including non-feasible ones) are there?
Extended Example: Alternative Runs (1/2)

\[
\begin{align*}
z &= 3x_1 + x_2 + 2x_3 \\
x_4 &= 30 - x_1 - x_2 - 3x_3 \\
x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\
x_6 &= 36 - 4x_1 - x_2 - 2x_3
\end{align*}
\]

Switch roles of \(x_2\) and \(x_5\)

\[
\begin{align*}
z &= 12 + 2x_1 - \frac{x_3}{2} - \frac{x_5}{2} \\
x_2 &= 12 - x_1 - \frac{5x_3}{2} - \frac{x_5}{2} \\
x_4 &= 18 - x_2 - \frac{x_3}{2} + \frac{x_5}{2} \\
x_6 &= 24 - 3x_1 + \frac{x_3}{2} + \frac{x_5}{2}
\end{align*}
\]

Switch roles of \(x_1\) and \(x_6\)

\[
\begin{align*}
z &= 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \\
x_1 &= 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \\
x_2 &= 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \\
x_4 &= 18 - \frac{x_3}{2} + \frac{x_5}{2}
\end{align*}
\]
Extended Example: Alternative Runs (2/2)

\[ z = 3x_1 + x_2 + 2x_3 \]
\[ x_4 = 30 - x_1 - x_2 - 3x_3 \]
\[ x_5 = 24 - 2x_1 - 2x_2 - 5x_3 \]
\[ x_6 = 36 - 4x_1 - x_2 - 2x_3 \]

Switch roles of \( x_3 \) and \( x_5 \)

\[ z = \frac{48}{5} + \frac{11}{5}x_1 + \frac{1}{5}x_2 - \frac{2}{5}x_5 \]
\[ x_4 = \frac{78}{5} + \frac{1}{5}x_1 + \frac{2}{5}x_2 + \frac{3}{5}x_5 \]
\[ x_3 = \frac{24}{5} - \frac{2}{5}x_1 - \frac{2}{5}x_2 - \frac{1}{5}x_5 \]
\[ x_6 = \frac{132}{5} - \frac{16}{5}x_1 - \frac{1}{5}x_2 + \frac{2}{5}x_3 \]

Switch roles of \( x_1 \) and \( x_6 \)  

\[ z = \frac{111}{4} + \frac{x_2}{16} - \frac{x_5}{8} - \frac{11x_6}{16} \]
\[ x_1 = \frac{33}{4} - \frac{x_2}{16} + \frac{x_5}{8} - \frac{5x_6}{16} \]
\[ x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8} \]
\[ x_4 = \frac{69}{4} + \frac{3x_2}{16} + \frac{5x_5}{8} - \frac{x_6}{16} \]

Switch roles of \( x_2 \) and \( x_3 \)

\[ z = 28 - \frac{x_3}{6} - \frac{x_5}{6} - \frac{2x_6}{3} \]
\[ x_1 = 8 + \frac{x_3}{6} + \frac{x_5}{6} - \frac{x_6}{3} \]
\[ x_2 = 4 - \frac{8x_3}{3} - \frac{2x_5}{3} + \frac{x_6}{3} \]
\[ x_4 = 18 - \frac{x_3}{2} + \frac{x_5}{2} \]
Outline

Simplex Algorithm by Example

Details of the Simplex Algorithm

Finding an Initial Solution

Appendix: Cycling and Termination (non-examinable)
The Pivot Step Formally

**P**IVOT\((N, B, A, b, c, v, l, e)\)

1. \(\textbf{// Compute the coefficients of the equation for new basic variable } x_e.\)
2. \(\text{let } \hat{A} \text{ be a new } m \times n \text{ matrix}\)
3. \(\hat{b}_e = b_l/a_{le}\)
4. \(\textbf{for each } j \in N - \{e\}\)
5. \(\hat{a}_{ej} = a_{lj}/a_{le}\)
6. \(\hat{a}_{el} = 1/a_{le}\)
7. \(\textbf{// Compute the coefficients of the remaining constraints.}\)
8. \(\textbf{for each } i \in B - \{l\}\)
9. \(\hat{b}_i = b_i - a_{ie}\hat{b}_e\)
10. \(\textbf{for each } j \in N - \{e\}\)
11. \(\hat{a}_{ij} = a_{ij} - a_{ie}\hat{a}_{ej}\)
12. \(\hat{a}_{il} = -a_{ie}\hat{a}_{el}\)
13. \(\textbf{// Compute the objective function.}\)
14. \(\hat{v} = v + c_e\hat{b}_e\)
15. \(\textbf{for each } j \in N - \{e\}\)
16. \(\hat{c}_j = c_j - c_e\hat{a}_{ej}\)
17. \(\hat{c}_l = -c_e\hat{a}_{el}\)
18. \(\textbf{// Compute new sets of basic and nonbasic variables.}\)
19. \(\hat{N} = N - \{e\} \cup \{l\}\)
20. \(\hat{B} = B - \{l\} \cup \{e\}\)
21. \(\textbf{return } (\hat{N}, \hat{B}, \hat{A}, \hat{b}, \hat{c}, \hat{v})\)

**Rewrite “tight” equation for entering variable** \(x_e.\)

**Substituting** \(x_e\) **into other equations.**

**Substituting** \(x_e\) **into objective function.**

**Update non-basic and basic variables**
Effect of the Pivot Step (extra material, non-examinable)

Consider a call to \texttt{Pivot}(\mathcal{N}, \mathcal{B}, \mathcal{A}, b, c, \nu, l, e) in which \(a_{le} \neq 0\). Let the values returned from the call be \((\mathcal{N}, \mathcal{B}, \hat{\mathcal{A}}, \hat{b}, \hat{c}, \hat{\nu})\), and let \(\overline{x}\) denote the basic solution after the call. Then

1. \(\overline{x}_j = 0\) for each \(j \in \mathcal{N}\).
2. \(\overline{x}_e = b_l/a_{le}\).
3. \(\overline{x}_i = b_i - a_{ie}\hat{b}_e\) for each \(i \in \hat{\mathcal{B}} \setminus \{e\}\).

**Proof:**

1. holds since the basic solution always sets all non-basic variables to zero.
2. When we set each non-basic variable to 0 in a constraint

   \[ x_i = \hat{b}_i - \sum_{j \in \hat{\mathcal{N}}} \hat{a}_{ij} x_j, \]

   we have \(\overline{x}_i = \hat{b}_i\) for each \(i \in \hat{\mathcal{B}}\). Hence \(\overline{x}_e = \hat{b}_e = b_l/a_{le}\).
3. After substituting into the other constraints, we have

   \[ \overline{x}_i = \hat{b}_i = b_i - a_{ie}\hat{b}_e. \]
Questions:

- How do we determine whether a linear program is feasible?
- What do we do if the linear program is feasible, but the initial basic solution is not feasible?
- How do we determine whether a linear program is unbounded?
- How do we choose the entering and leaving variables?

Example before was a particularly nice one!
The formal procedure SIMPLEX

SIMPLEX\((A, b, c)\)

1. \((N, B, A, b, c, v) = \text{INITIALIZE-SIMPLEX}(A, b, c)\)
2. let \(\Delta\) be a new vector of length \(m\)
3. while some index \(j \in N\) has \(c_j > 0\)
4. choose an index \(e \in N\) for which \(c_e > 0\)
5. for each index \(i \in B\)
6. if \(a_{ie} > 0\)
7. \(\Delta_i = b_i / a_{ie}\)
8. else \(\Delta_i = \infty\)
9. choose an index \(l \in B\) that minimizes \(\Delta_i\)
10. if \(\Delta_l = \infty\)
11. return "unbounded"  
12. else \((N, B, A, b, c, v) = \text{PIVOT}(N, B, A, b, c, v, l, e)\)
13. for \(i = 1\) to \(n\)
14. if \(i \in B\)
15. \(\bar{x}_i = b_i\)
16. else \(\bar{x}_i = 0\)
17. return \((\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n)\)

Returns a slack form with a feasible basic solution (if it exists)

Main Loop:
- terminates if all coefficients in objective function are non-positive
- Line 4 picks entering variable \(x_e\) with positive coefficient
- Lines 6 — 9 pick the tightest constraint, associated with \(x_l\)
- Line 11 returns "unbounded" if there are no constraints
- Line 12 calls PIVOT, switching roles of \(x_l\) and \(x_e\)

Return corresponding solution.
The formal procedure \textsc{Simplex}

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

1. \((N, B, A, b, c, v) = \text{initialize-simplex}(A, b, c)\)
2. let \(\Delta\) be a new vector of length \(m\)
3. \textbf{while} some index \(j \in N\) has \(c_j > 0\)
   4. choose an index \(e \in N\) for which \(c_e > 0\)
   5. \textbf{for} each index \(i \in B\)
   6. \phantom{5.} \textbf{if} \(a_{ie} > 0\)
      7. \phantom{5.5} \(\Delta_i = b_i/a_{ie}\)
   8. \phantom{5.} \textbf{else} \(\Delta_i = \infty\)
   9. choose an index \(l \in B\) that minimizes \(\Delta_i\)
10. \textbf{if} \(\Delta_l = \infty\)
    11. \textbf{return} “unbounded”

\textbf{Proof} is based on the following three-part loop invariant:
1. the slack form is always equivalent to the one returned by \text{initialize-simplex},
2. for each \(i \in B\), we have \(b_i \geq 0\),
3. the basic solution associated with the (current) slack form is feasible.

\begin{lemma}
Suppose the call to \text{initialize-simplex} in line 1 returns a slack form for which the basic solution is feasible. Then if \text{Simplex} returns a solution, it is a feasible solution. If \text{Simplex} returns “unbounded”, the linear program is unbounded.
\end{lemma}
Outline

Simplex Algorithm by Example

Details of the Simplex Algorithm

Finding an Initial Solution

Appendix: Cycling and Termination (non-examinable)
maximise \[ 2x_1 - x_2 \]
subject to
\[
\begin{align*}
2x_1 - x_2 & \leq 2 \\
x_1 - 5x_2 & \leq -4 \\
x_1, x_2 & \geq 0
\end{align*}
\]

Conversion into slack form

\[ z = 2x_1 - x_2 \]
\[ x_3 = 2 - 2x_1 + x_2 \]
\[ x_4 = -4 - x_1 + 5x_2 \]

Basic solution \((x_1, x_2, x_3, x_4) = (0, 0, 2, -4)\) is not feasible!
Geometric Illustration

maximise \[ 2x_1 - x_2 \]
subject to
\[ 2x_1 - x_2 \leq 2 \]
\[ x_1 - 5x_2 \leq -4 \]
\[ x_1, x_2 \geq 0 \]

Questions:
- How to determine whether there is any feasible solution?
- If there is one, how to determine an initial basic solution?
Formulating an Auxiliary Linear Program

Let $L_{aux}$ be the auxiliary LP of a linear program $L$ in standard form. Then $L$ is feasible if and only if the optimal objective value of $L_{aux}$ is 0.

Proof. Exercise!
Let us illustrate the role of $x_0$ as “distance from feasibility”.
We’ll also see that increasing $x_0$ enlarges the feasible region.
maximise $-x_0$
subject to

$2x_1 - x_2 - x_0 \leq 2$
$x_1 - 5x_2 - x_0 \leq -4$
$x_0, x_1, x_2 \geq 0$

For the animation see the full slides.
• Let us now modify the original linear program so that it is not feasible.

⇒ Hence the auxiliary linear program has only a solution for a sufficiently large $x_0 > 0$!
Geometric Illustration

maximise $-x_0$
subject to

$$
2x_1 - x_2 - x_0 \leq -2
$$

$$
-x_1 + 5x_2 - x_0 \leq 4
$$

$$
x_0, x_1, x_2 \geq 0
$$

For the animation see the full slides.
**INITIALIZE-SIMPLEX**

**INITIALIZE-SIMPLEX** \((A, b, c)\)

1. let \(k\) be the index of the minimum \(b_i\)
2. if \(b_k \geq 0\)  // is the initial basic solution feasible?
   3. return \((\{1, 2, \ldots, n\}, \{n + 1, n + 2, \ldots, n + m\}, A, b, c, 0)\)
4. form \(L_{aux}\) by adding \(-x_0\) to the left-hand side of each constraint
   and setting the objective function to \(-x_0\)
5. let \((N, B, A, b, c, v)\) be the resulting slack form for \(L_{aux}\)
6. \(l = n + k\)
7. \(// L_{aux}\) has \(n + 1\) nonbasic variables and \(m\) basic variables.
8. \((N, B, A, b, c, v) = PIVOT(N, B, A, b, c, v, l, 0)\)
9. \(//\) The basic solution is now feasible for \(L_{aux}\).
10. iterate the while loop of lines 3–12 of SIMPLEX until an optimal solution
    to \(L_{aux}\) is found
11. if the optimal solution to \(L_{aux}\) sets \(\bar{x}_0\) to 0
12.   if \(\bar{x}_0\) is basic
   13.     perform one (degenerate) pivot to make it nonbasic
   14.     from the final slack form of \(L_{aux}\), remove \(x_0\) from the constraints and
         restore the original objective function of \(L\), but replace each basic
         variable in this objective function by the right-hand side of its
         associated constraint
   15.     return the modified final slack form
16. else return “infeasible”

---

Test solution with \(N = \{1, 2, \ldots, n\}, B = \{n + 1, n + 2, \ldots, n + m\}, \bar{x}_i = b_i\) for \(i \in B, \bar{x}_i = 0\) otherwise.

\(\ell\) will be the leaving variable so that \(x_\ell\) has the most negative value.

Pivot step with \(x_\ell\) leaving and \(x_0\) entering.

This pivot step does not change the value of any variable.
Example of INITIALIZE-SIMPLEX (1/3)

maximise \[ 2x_1 - x_2 \]
subject to
\[ \begin{align*}
2x_1 & - x_2 \leq 2 \\
x_1 & - 5x_2 \leq -4 \\
x_1, x_2 & \geq 0
\end{align*} \]

Formulating the auxiliary linear program

maximise \[-x_0\]
subject to
\[ \begin{align*}
2x_1 & - x_2 - x_0 \leq 2 \\
x_1 & - 5x_2 - x_0 \leq -4 \\
x_1, x_2, x_0 & \geq 0
\end{align*} \]

Basic solution \((0, 0, 0, 2, -4)\) not feasible!

Converting into slack form

\[ \begin{align*}
z & = 2 - 2x_1 + x_2 - x_0 \\
x_3 & = 2 - 2x_1 + x_2 + x_0 \\
x_4 & = -4 - x_1 + 5x_2 + x_0
\end{align*} \]
Example of **INITIALIZE-SIMPLEX (2/3)**

\[
\begin{align*}
  z &= -x_0 \\
  x_3 &= 2 - 2x_1 + x_2 + x_0 \\
  x_4 &= -4 - x_1 + 5x_2 + x_0
\end{align*}
\]

Pivot with \(x_0\) entering and \(x_4\) leaving

\[
\begin{align*}
  z &= -4 - x_1 + 5x_2 - x_4 \\
  x_0 &= 4 + x_1 - 5x_2 + x_4 \\
  x_3 &= 6 - x_1 - 4x_2 + x_4
\end{align*}
\]

Basic solution \((4, 0, 0, 6, 0)\) is feasible!

Pivot with \(x_2\) entering and \(x_0\) leaving

\[
\begin{align*}
  z &= -x_0 \\
  x_2 &= \frac{4}{5} - \frac{x_0}{5} + \frac{x_1}{5} + \frac{x_4}{5} \\
  x_3 &= \frac{14}{5} + \frac{4x_0}{5} - \frac{9x_1}{5} + \frac{x_4}{5}
\end{align*}
\]

Optimal solution has \(x_0 = 0\), hence the initial problem was feasible!
Example of \textsc{Initialize-Simplex (3/3)}

\[
\begin{align*}
  z &= -x_0 - \frac{4}{5} x_2 + \frac{14}{5} x_3 \\
  x_2 &= \frac{4}{5} x_0 - \frac{4}{5} x_1 + \frac{14}{5} x_4 \\
  x_3 &= \frac{14}{5} + \frac{4}{5} x_0 - \frac{9}{5} x_1 + \frac{4}{5} x_4
\end{align*}
\]

Set \( x_0 = 0 \) and express objective function by non-basic variables

\[
\begin{align*}
  z &= -\frac{4}{5} + \frac{9}{5} x_1 - \frac{4}{5} x_4 \\
  x_2 &= \frac{4}{5} + \frac{4}{5} x_1 + \frac{4}{5} x_4 \\
  x_3 &= \frac{14}{5} - \frac{9}{5} x_1 + \frac{4}{5} x_4
\end{align*}
\]

Basic solution \((0, \frac{4}{5}, \frac{14}{5}, 0)\), which is feasible!

\textbf{Lemma 29.12}

If a linear program \( L \) has no feasible solution, then \textsc{Initialize-Simplex} returns “infeasible”. Otherwise, it returns a valid slack form for which the basic solution is feasible.
Fundamental Theorem of Linear Programming

Theorem 29.13 (Fundamental Theorem of Linear Programming)

For any linear program \( L \), given in standard form, either:

1. \( L \) is infeasible \( \Rightarrow \) SIMPLEX returns “infeasible”.
2. \( L \) is unbounded \( \Rightarrow \) SIMPLEX returns “unbounded”.
3. \( L \) has an optimal solution with a finite objective value \( \Rightarrow \) SIMPLEX returns an optimal solution with a finite objective value.

Small Technicality: need to equip SIMPLEX with an “anti-cycling strategy” (see extra slides)

Proof requires the concept of duality, which is not covered in this course (for details see CLRS3, Chapter 29.4)
**Workflow for Solving Linear Programs**

1. **Linear Program (in any form)**
2. **Standard Form**
3. **Slack Form**
   - **No Feasible Solution**
     - \textsc{Initialize-Simplex} terminates
   - **Feasible Basic Solution**
     - \textsc{Initialize-Simplex} followed by \textsc{Simplex}
       - LP unbounded \textsc{Simplex} terminates
       - LP bounded \textsc{Simplex} returns optimum
Linear Programming

- extremely versatile tool for modelling problems of all kinds
- basis of Integer Programming, to be discussed in later lectures

Simplex Algorithm

- In practice: usually terminates in polynomial time, i.e., $O(m + n)$
- In theory: even with anti-cycling may need exponential time

**Research Problem:** Is there a pivoting rule which makes SIMPLEX a polynomial-time algorithm?

Polynomial-Time Algorithms

- Interior-Point Methods: traverses the interior of the feasible set of solutions (not just vertices!)
1.2 Famous Failures and the Need for Alternatives

For many problems a bit beyond the scope of an undergraduate course, the downside of worst-case analysis rears its ugly head. This section reviews four famous examples in which worst-case analysis gives misleading or useless advice about how to solve a problem. These examples motivate the alternatives to worst-case analysis that are surveyed in Section 1.4 and described in detail in later chapters of the book.

1.2.1 The Simplex Method for Linear Programming

Perhaps the most famous failure of worst-case analysis concerns linear programming, the problem of optimizing a linear function subject to linear constraints (Figure 1.1). Dantzig proposed in the 1940s an algorithm for solving linear programs called the simplex method. The simplex method solves linear programs using greedy local

Source: “Beyond the Worst-Case Analysis of Algorithms” by Tim Roughgarden, 2020
Outline

Simplex Algorithm by Example

Details of the Simplex Algorithm

Finding an Initial Solution

Appendix: Cycling and Termination (non-examinable)
Degeneracy: One iteration of SIMPLEX leaves the objective value unchanged.

\[
\begin{align*}
  z &= x_1 + x_2 + x_3 \\
  x_4 &= 8 - x_1 - x_2 \\
  x_5 &= x_2 - x_3
\end{align*}
\]

Pivot with \( x_1 \) entering and \( x_4 \) leaving

\[
\begin{align*}
  z &= 8 + x_3 - x_4 \\
  x_1 &= 8 - x_2 \\
  x_5 &= x_2 - x_3
\end{align*}
\]

Cycling: If additionally slack form at two iterations are identical, SIMPLEX fails to terminate!

\[
\begin{align*}
  z &= 8 + x_2 - x_4 - x_5 \\
  x_1 &= 8 - x_2 - x_4 \\
  x_3 &= x_2 - x_5
\end{align*}
\]
Exercise: Execute one more step of the Simplex Algorithm on the tableau from the previous slide.
Termination and Running Time

**Cycling:** \textsc{Simplex} may fail to terminate.

**Anti-Cycling Strategies**

1. **Bland’s rule:** Choose entering variable with smallest index
2. **Random rule:** Choose entering variable uniformly at random
3. **Perturbation:** Perturb the input slightly so that it is impossible to have two solutions with the same objective value

Replace each \( b_i \) by \( \hat{b}_i = b_i + \epsilon_i \), where \( \epsilon_i \gg \epsilon_{i+1} \) are all small.

**Lemma 29.7**

Assuming \textsc{Initia}lize-\textsc{Simplex} returns a slack form for which the basic solution is feasible, \textsc{Simplex} either reports that the program is unbounded or returns a feasible solution in at most \( \binom{n+m}{m} \) iterations.

Every set \( B \) of basic variables uniquely determines a slack form, and there are at most \( \binom{n+m}{m} \) unique slack forms.