Linear Programming

Linear Programming: Introduction

- A large class of optimization problems in which the constraints and optimization criterion are linear functions.
- A *Linear Programming(**LP**)* problem consists of assigning real values to variables such that these variables
  1. (**Linear constraints**) satisfy a set of *linear* equalities or inequalities, and
  2. (**Objective function**) maximize or minimize a given *linear* objective function.

Linear Programming: Introduction

- Example: A cottage industry makes two kinds of products P_1 and P_2. The daily demand for P_1 is 100 and the daily demand for P_2 is 200. The total amount of items that the industry can produce in a day is 250. The industry makes profit of $Rs. 1$ per unit item of type P_1 and $Rs. 5$ per unit item of type P_2. How many items of P_1 and P_2 should the industry produce to make maximum amount of profit?
- Let x_1 be a variable denoting the amount of P_1 items produced by the industry and x_2 the mount of P_2 items.
- The goal is to maximize the *linear objective function:*

$$1 \cdot x_1 + 5 \cdot x_2$$

under the *linear constraints*:

$$x_1 \geq 0, x_2 \geq 0, x_1 \leq 100, x_2 \leq 200, x_1 + x_2 \leq 250$$

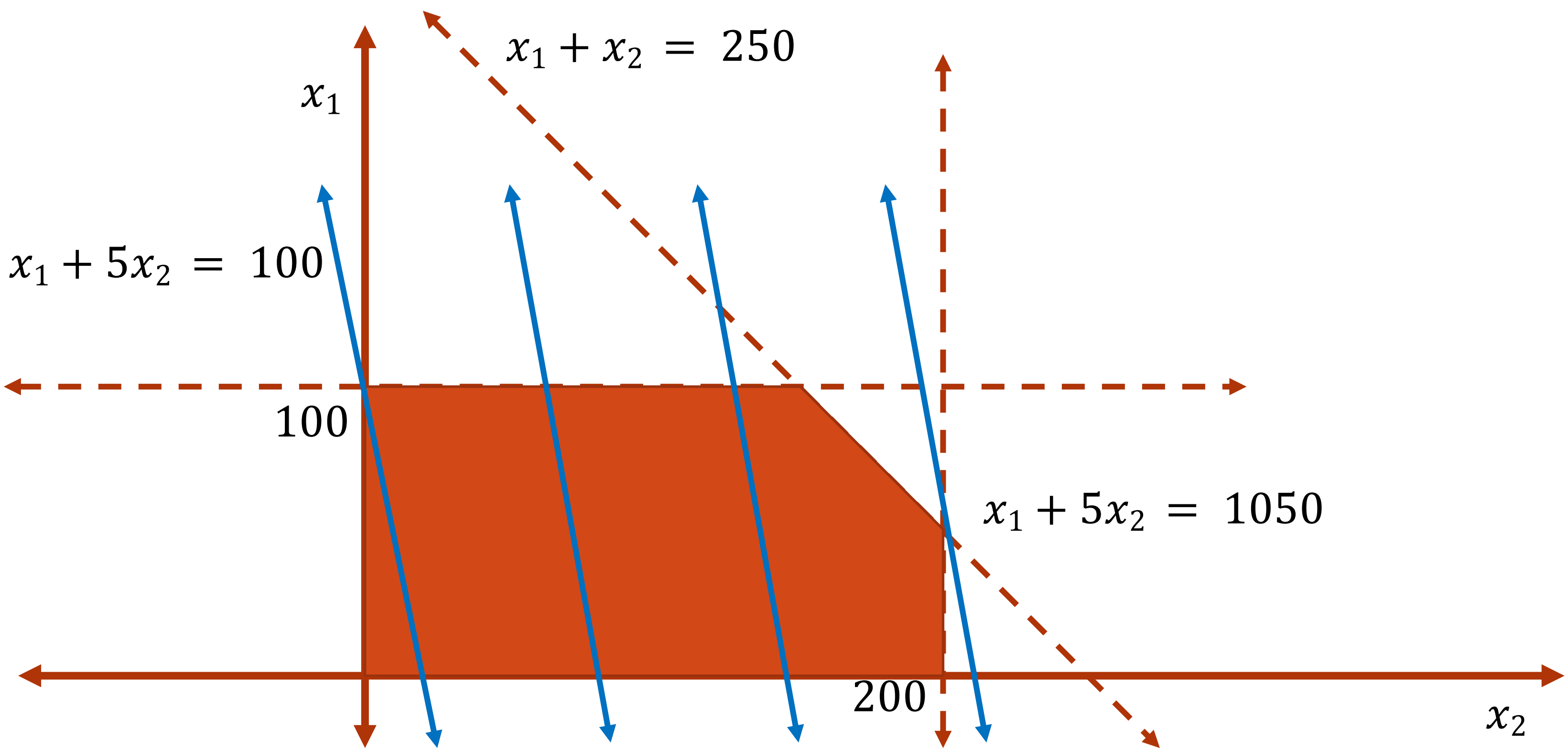
Linear Programming: Introduction

- Problem(LP): Maximize the *linear objective function:*

$$1 \cdot x_1 + 5 \cdot x_2$$

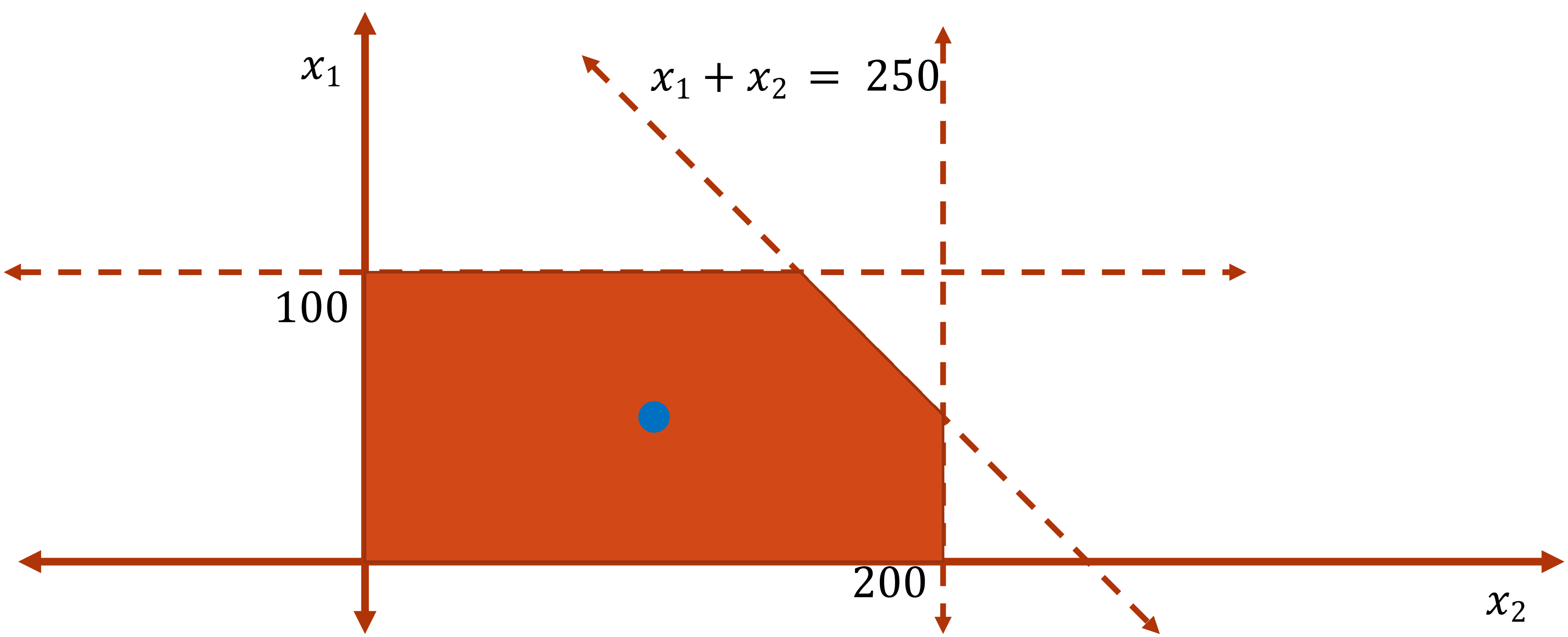
under the *linear constraints*:

$$x_1 \geq 0, x_2 \geq 0, x_1 \leq 100, x_2 \leq 200, x_1 + x_2 \leq 250$$



Linear Programming: Introduction

- Given a Linear Programming problem, we will use the following definitions:
  - <u>Feasible solution</u>: An assignment to the variables that satisfy all the linear constraints.
  - <u>Example</u>: $x_1 = 50, x_2 = 100$ is a feasible solution.



Linear Programming: Introduction

- Question: Does a Linear Programming problem always have a feasible solution?

Linear Programming: Introduction

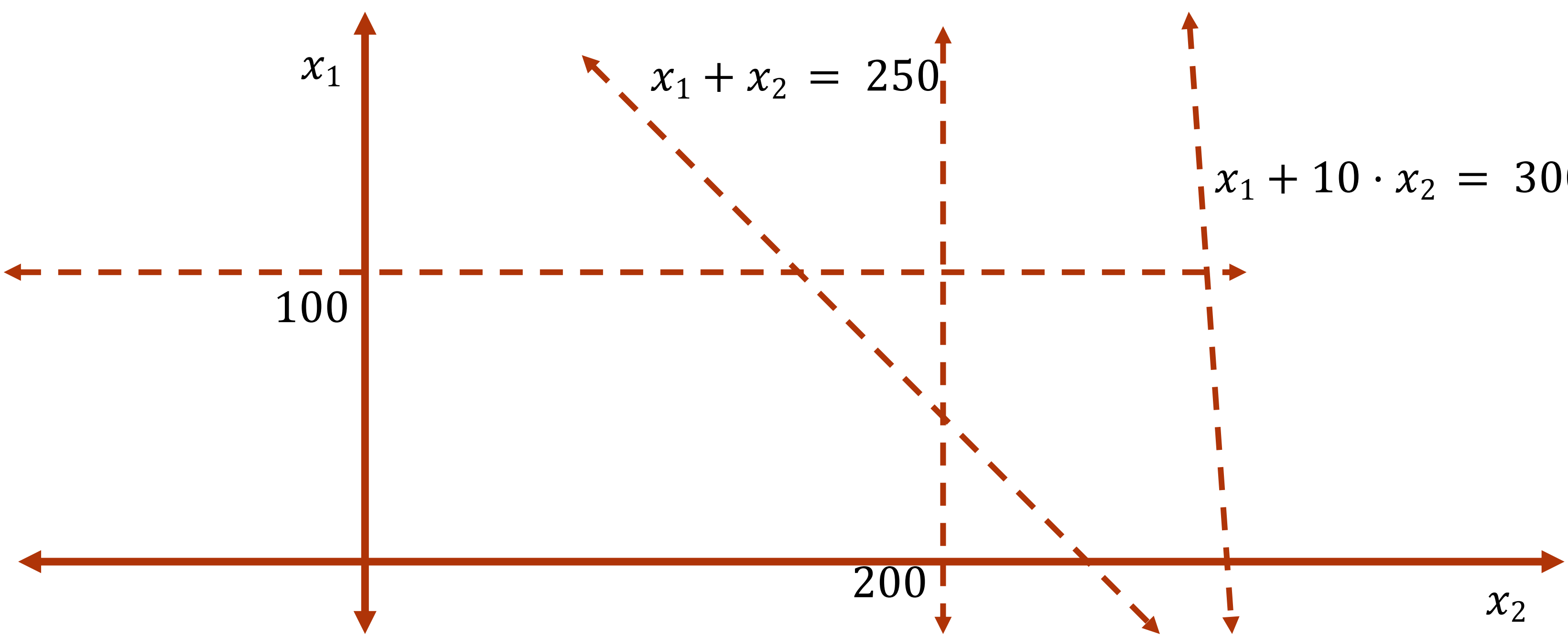
- Question: Does a Linear Programming problem always have a feasible solution?
  - Not necessarily. Suppose the linear constraints are

$$x_1 \geq 0, x_2 \geq 0, x_1 \leq 100, x_2 \leq 200,$$
$$x_1 + x_2 \leq 250, x_1 + 10 \cdot x_2 \geq 3000$$

Linear Programming: Introduction

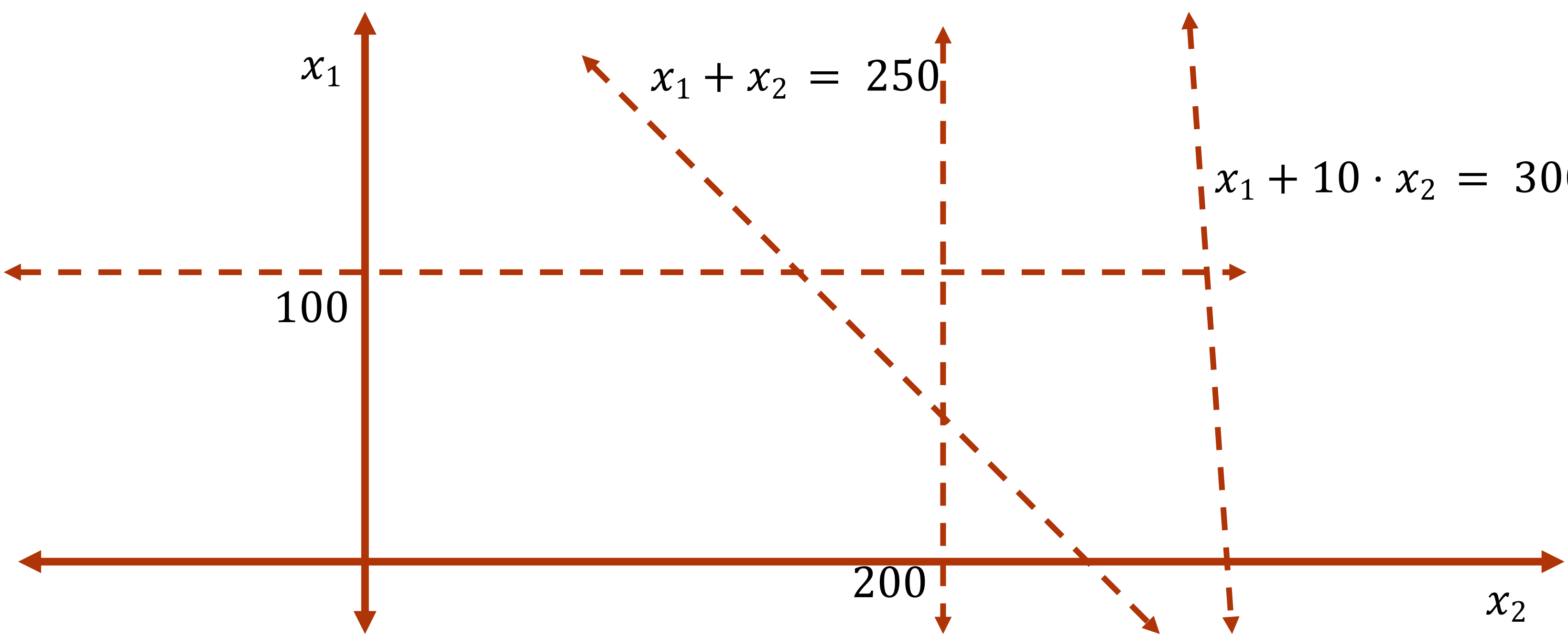
- <u>Question</u>: Does a Linear Programming problem always have a feasible solution?
  - Not necessarily. Suppose the linear constraints are

$$x_1 \geq 0, x_2 \geq 0, x_1 \leq 100, x_2 \leq 200,$$
$$x_1 + x_2 \leq 250, x_1 + 10 \cdot x_2 \geq 3000$$



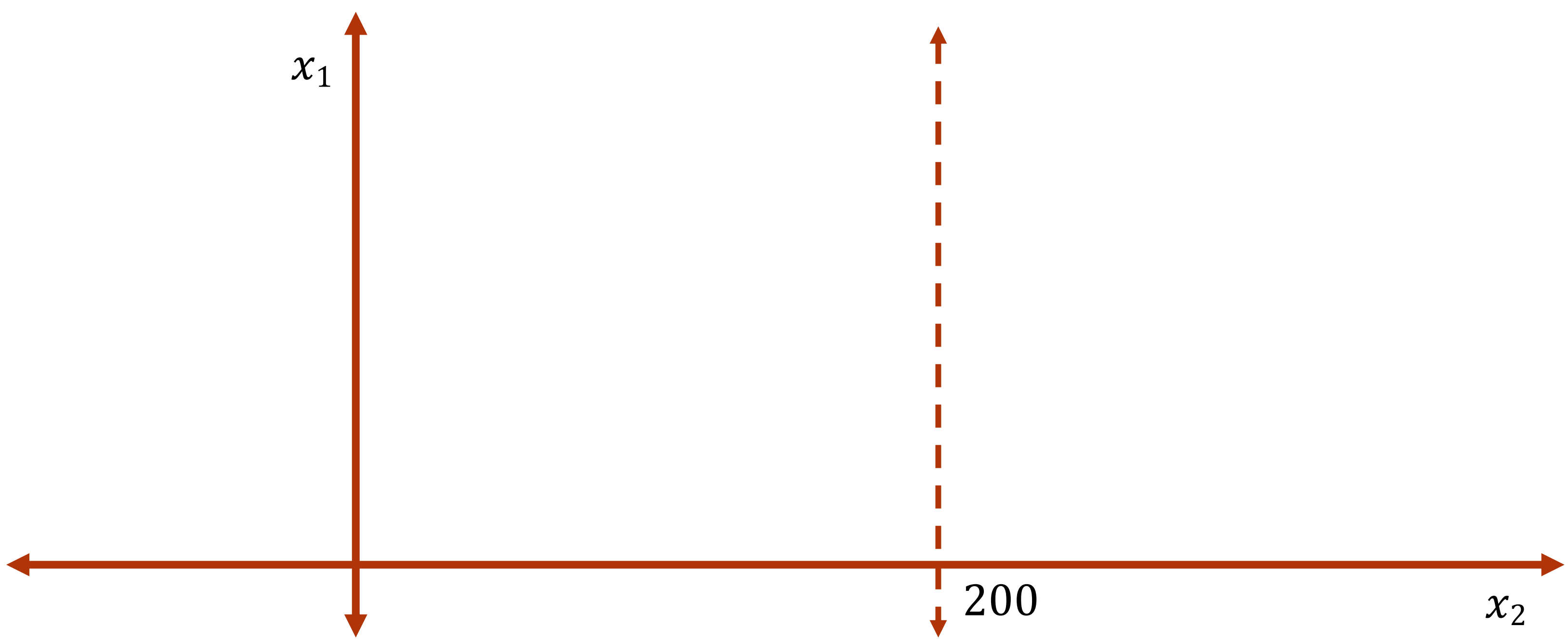
Linear Programming: Introduction

- <u>Infeasible LP</u>: A linear program is said to be infeasible if there are no feasible solutions.



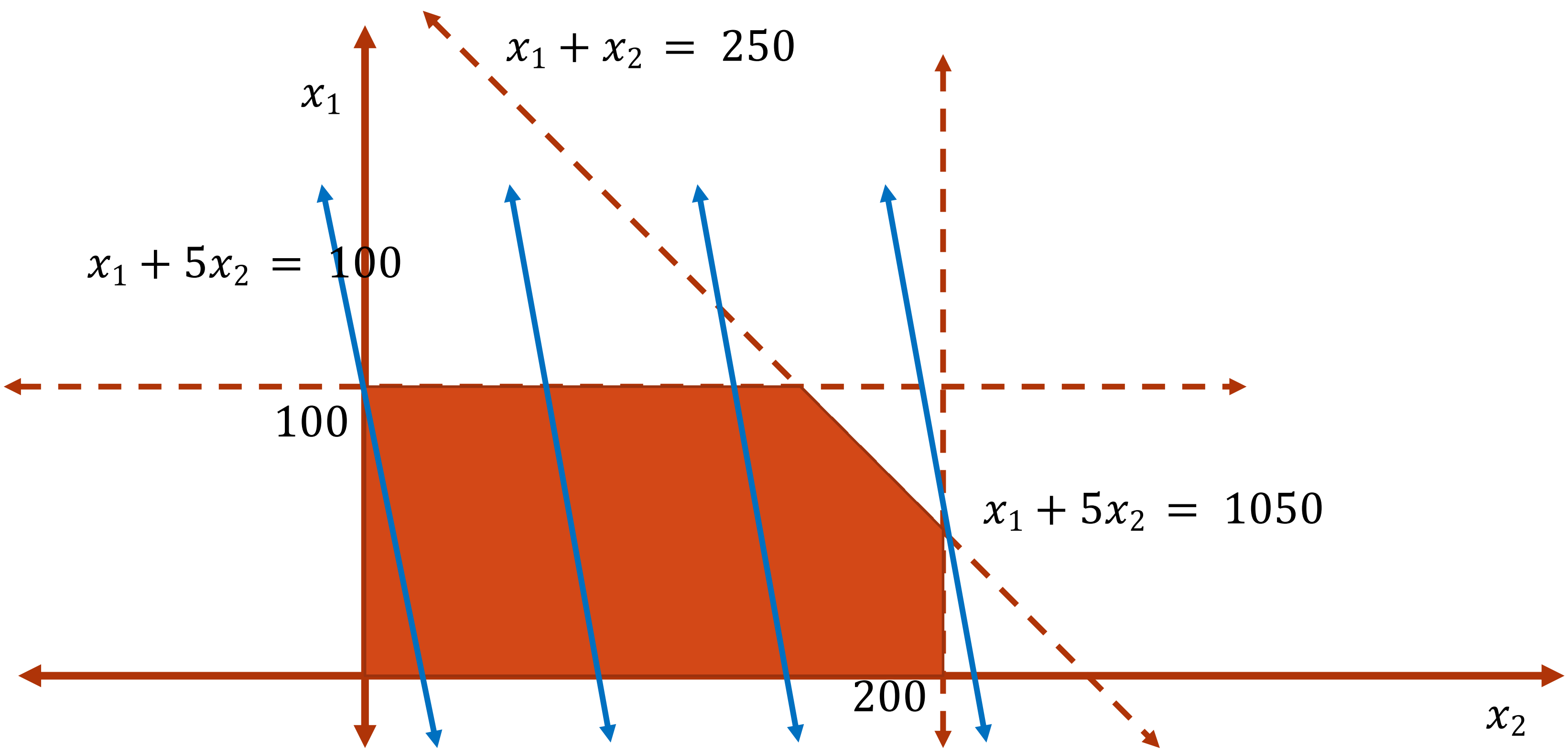
Linear Programming: Introduction

- <u>Unbounded LP</u>: A linear program is said to be unbounded if it is possible to achieve arbitrarily high values of the objective function.
  - <u>Example</u>: Maximize $(x_1 + 5 \cdot x_2)$
    subject to $x_1 \geq 0, x_2 \geq 0, x_2 \leq 200.$



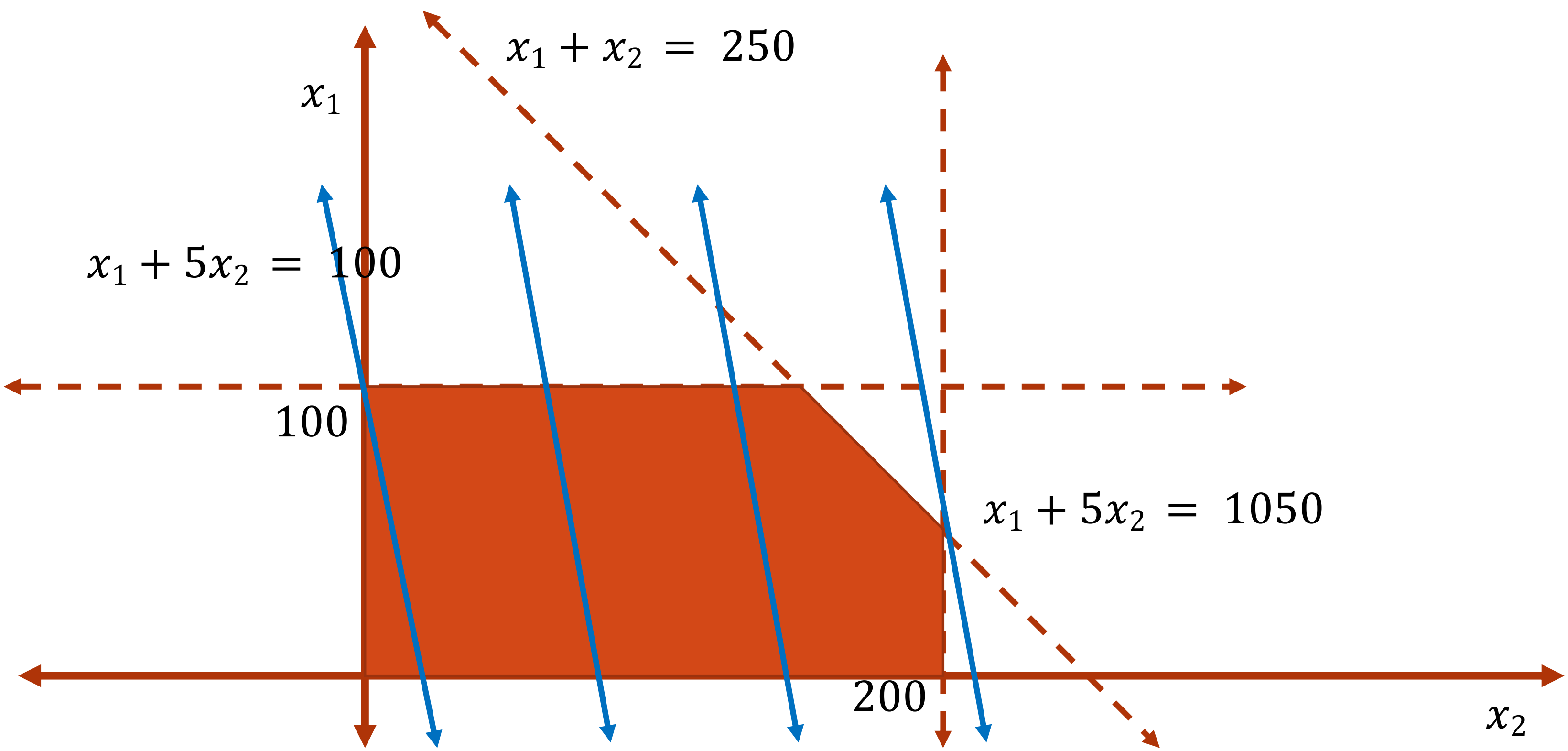
Linear Programming: Introduction

- <u>Claim</u>: For any linear program that is not infeasible and unbounded, the objective function value is maximized at one of the *vertices* of the feasible region.



Linear Programming: Introduction

- Naïve idea for solving an LP:
  - Try all possible vertex of the feasible region and return the one that maximizes the objective function.

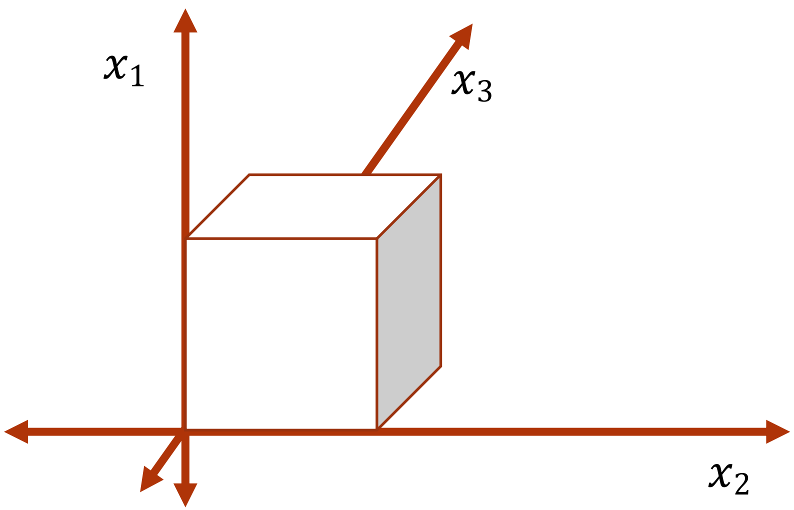


Linear Programming: Introduction

- Naïve idea for solving an LP:
  - Try all possible vertex of the feasible region and return the one that maximizes the objective function.
  - Suppose the LP has n variables and $m = O(n)$ constraints. How many vertices can the feasible region have in worst case?

Linear Programming: Introduction

- Naïve idea for solving an LP:
  - Try all possible vertex of the feasible region and return the one that maximizes the objective function.
  - Suppose the LP has n variables and $m = O(n)$ constraints. How many vertices can the feasible region have in worst case?
    - Exponentially many! Consider the LP: maximize $(x_1 + x_2 + \cdots + x_n)$ subject to $0 \le x_1, x_2, \ldots, x_n \le 1$.

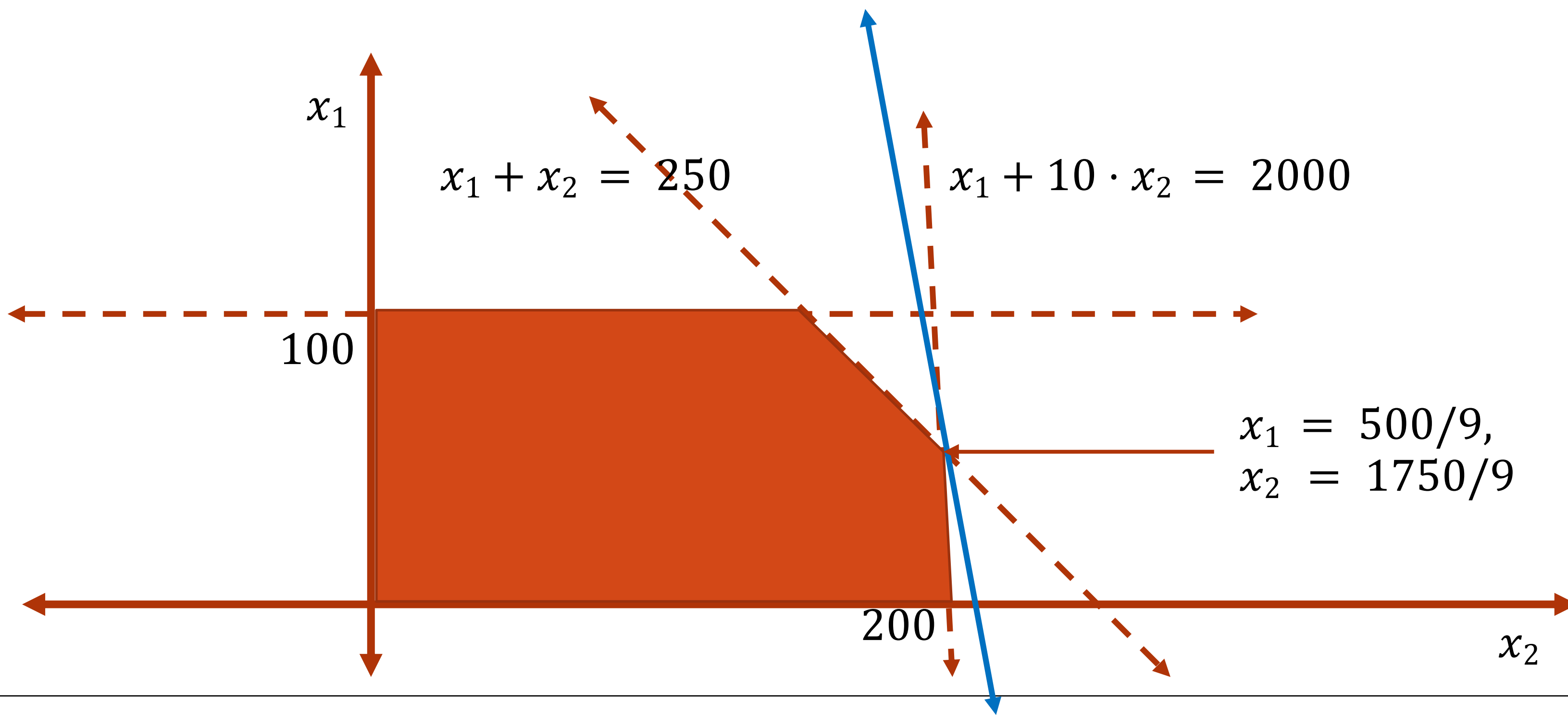


Linear Programming: Introduction

- Claim: There is an algorithm that solves any linear programming problem instance that runs in polynomial time.

Linear Programming: Introduction

- Claim: There is an algorithm that solves any linear programming problem instance that runs in polynomial time.
- The optimal solution may assign real numbers to some variables even though all of the constraints of objective function involve integers.



Linear Programming: Introduction

- Claim: There is an algorithm that solves any linear programming problem instance that runs in polynomial time.
- The optimal solution may assign real numbers to some variables even though all of the constraints of objective function involve integers.
- Suppose in addition to the linear constraint, we add another constraint that all the variables should be integers. Such linear programs are called Integer Linear Programs (ILP).
- Integer Linear Program(ILP): Consists of
  - Linear objective function
  - Linear constraints.
  - All variables should be integers.

    Decision-ILP: Given the above and an integer k, determine if there is an integer assignment to the variables such that the objective function value is at least k.

Linear Programming: Introduction

- How hard is Decision-ILP?

Linear Programming: Introduction

- How hard is Decision-ILP?
- <u>Claim</u>: Decision-ILP is **NP**-complete.
  - Proof:
    - Claim 1: Decision-ILP is in **NP**.
    - Claim 2: 3-SAT $\leq_p$ Decision-ILP

Linear Programming: Introduction

- How hard is Decision-ILP?
- Claim: Decision-ILP is **NP**-complete.
  - Proof:
    - Claim 1: Decision-ILP is in **NP**.
    - Claim 2: 3-SAT $\leq_p$ Decision-ILP
      - Proof idea: Given a 3-SAT formula, we construct an instance of Decision-ILP.
        For each clause (e.g., $(x_1 \vee x_2{}' \vee x_3)$) we create a linear constraint (e.g., $x_1 + 1 - x_2 + x_3 \geq 1$). We further consider constraints $0 \leq x_1, \ldots, x_n \leq 1$ and that all variables are integers.

Linear Programming: Introduction

- How hard is Decision-ILP?
- Claim: Decision-ILP is **NP**-complete.
  - Proof:
    - Claim 1: Decision-ILP is in **NP**.
    - Claim 2: 3-SAT $\leq_p$ Decision-ILP
      - Proof idea: Given a 3-SAT formula, we construct an instance of Decision-ILP.
        For each clause (e.g., $(x_1 \vee x_2' \vee x_3)$) we create a linear constraint (e.g., $x_1 + 1 - x_2 + x_3 \geq 1$). We further consider constraints $0 \leq x_1, \ldots, x_n \leq 1$ and that all variables are integers.
- Formulating problems as an ILP is a standard way of solving many combinatorial problems.
- Example: Maximum Independent set.
  - Consider a $0 - 1$ variable for each vertex, 1 denoting inclusion. For each edge (x, y), there is a constraint that $x + y \leq 1$.

Linear Programming

Solving problems by formulating as Linear Programs

Linear Programming: Applications

- We saw how some combinatorial problems can be formulated as an **Integer** Linear Programming (ILP) problem.
- Unfortunately, ILP is hard.
- A number of problems can be formulated as a Linear Programming problem and we know there is a polynomial time algorithm for LP.
- Some interesting applications:
  - Shortest $s - t$ path in a directed graph with non-negative weights.
  - Maximum flow in a network graph.

Linear Programming: Applications

- <u>Problem (Maximum $s - t$ flow)</u>: Given a network graph $G = (V, E)$ with special source s and sink t, find the maximum value of an $s - t$ flow in the graph.

- Let $m = |E|$. We use m variables, one for each edge.
- For an edge (u, v), we will use variable f_{uv} to denote the flow along the edge (u, v).
- We construct the following LP given G.
  - *Maximize* $\sum_{(s,v) \in E} f_{sv}$
  - Subject to,
    - $f_{uv} \leq c(u, v)$, for all (u, v) in E.
    - $\sum_{(v,u) \in E} f_{vu} = \sum_{(u,v) \in E} f_{uv}$, for all u in $V - \{s, t\}$.
    - $f_{uv} \geq 0$.

Linear Programming: Applications

- Problem (Shortest $s - t$ path): Given a weighted, directed graph $G = (V, E)$. Find the length of the shortest path from vertex s to vertex t.

- Let $n = |V|$. We use n variables, one for each vertex.
- For a vertex v, we will use variable d_v to denote the length of the shortest path from vertex s to vertex v.
- We construct the following LP given G.
  - *Maximize* d_t,
  - subject to:
    - For all edges $(u, v) \in E, d_v \leq du + w(u, v)$.
    - $d_s = 0$.

Linear Programming

Solving an LP

Linear Programming: Solving LP

- To be able to design an algorithm for solving LP problems, it will be useful if we define problems more precisely in some standard format.
- **Standard form**: A Linear Program is said to be *standard form* if the following holds:
  1. The linear objective function should be *maximized*.
  2. All variables have non-negativity constraint.
     i.e., for all i, $x_i \geq 0$.
  3. All the remaining linear constraints are of the following form:
     $\sum_{j=1}^{n} a_j \cdot x_j \leq b_j$

Linear Programming: Solving LP

- **<u>Standard form</u>**: A Linear Program is said to be *standard form* if the following holds:
  1. The linear objective function should be *maximized*.
  2. All variables have non-negativity constraint.
     i.e., for all i, $x_i \geq 0$.
  3. All the remaining linear constraints are of the following form:
     $\sum_{j=1}^{n} a_j \cdot x_j \leq b_j$

- <u>Question</u>: Is there a way to convert any LP problem to an *equivalent* standard form?
  - <u>Equivalence of LP's</u>: Two LP problems P1 and P2 are said to be equivalent if for any feasible solution for P1 with objective value z, there is a feasible solution of P2 with the same objective value and vice versa.

Linear Programming: Solving LP

- **<u>Standard form</u>**: A Linear Program is said to be *standard form* if the following holds:
  1. The linear objective function should be *maximized*.
  2. All variables have non-negativity constraint.
     i.e., for all i, $x_i \geq 0$.
  3. All the remaining linear constraints are of the following form:
     $\sum_{j=1}^{n} a_j \cdot x_j \leq b_j$

- A general LP problem might not be in standard for because it might have:
  1. Equality constraints (=) rather than inequality (≤).
  2. ≥ instead of ≤.
  3. Variables without non-negativity constraints.
  4. Minimization rather than maximization.

Linear Programming: Solving LP

- A general LP problem might not be in standard form because it might have:
  1. Equality constraints ($=$) rather than inequality ($\leq$).
     - Idea: $a = b$ can be expresses as $a \leq b$ and $a \geq b$.
  2. $\geq$ instead of $\leq$.
  3. Variables without non-negativity constraints.
  4. Minimization rather than maximization.

Linear Programming: Solving LP

- A general LP problem might not be in standard form because it might have:
  1. Equality constraints ($=$) rather than inequality ($\leq$).
     - Idea: $a = b$ can be written as $a \leq b$ and $a \geq b$.
  2. $\geq$ instead of $\leq$.
     - Idea: $a \geq b$ can be written as $-a \leq -b$.
  3. Variables without non-negativity constraints.
  4. Minimization rather than maximization.

Linear Programming: Solving LP

- A general LP problem might not be in standard form because it might have:
  1. Equality constraints ($=$) rather than inequality ($\leq$).
     - Idea: $a = b$ can be written as $a \leq b$ and $a \geq b$.
  2. $\geq$ instead of $\leq$.
     - Idea: $a \geq b$ can be written as $-a \leq -b$.
  3. Variables without non-negativity constraints.
     - Idea: Replace a variable x (that has no non-negativity constraint) with $(x' - x'')$ everywhere and put $x' \geq 0$ and $x'' \geq 0$.
  4. Minimization rather than maximization.

Linear Programming: Solving LP

- A general LP problem might not be in standard form because it might have:
  1. Equality constraints (=) rather than inequality (≤).
     - <u>Idea</u>: $a = b$ can be written as $a \le b$ and $a \ge b$.
  2. ≥ instead of ≤.
     - <u>Idea</u>: $a \ge b$ can be written as $-a \le -b$.
  3. Variables without non-negativity constraints.
     - <u>Idea</u>: Replace a variable x (that has no non-negativity constraint) with $(x' - x'')$ everywhere and put $x' \ge 0$ and $x'' \ge 0$.
  4. Minimization rather than maximization.
     - <u>Idea</u>: Replace "Minimize $\sum c_i \cdot x_i$" with "Maximize $\sum(-c_i) \cdot x_i$".
     - *In this case, equivalence of LP is in the sense that the objective values of LPs are negation of each other instead of being same. So, you can solve one to get a solution for the other.*

Linear Programming: Solving LP

- Example:
  - Minimize $-2x_1 + 3x_2$
  - subject to
    - $x_1 + x_2 = 7$
    - $x_1 - 2x_2 \leq 4$
    - $x_1 \geq 0$

Linear Programming: Solving LP

- Example: Minimize to Maximize
  - Maximize $2x_1 - 3x_2$
  - subject to
    - $x_1 + x_2 = 7$
    - $x_1 - 2x_2 \leq 4$
    - $x_1 \geq 0$

Linear Programming: Solving LP

- Example: non-negativity constraint for x_2
  - Maximize $2x_1 - 3(x_2' - x_2'')$
  - subject to
    - $x_1 + (x_2' - x_2'') = 7$
    - $x_1 - 2(x_2' - x_2'') \leq 4$
    - $x_1 \geq 0, x_2' \geq 0, x_2'' \geq 0$

Linear Programming: Solving LP

- Example: non-negativity constraint for x_2
  - Maximize $2x_1 - 3x_2' + 3x_2''$
  - subject to
    - $x_1 + x_2' - x_2'' = 7$
    - $x_1 - 2x_2' + 2x_2'' \leq 4$
    - $x_1 \geq 0, x_2' \geq 0, x_2'' \geq 0$

Linear Programming: Solving LP

- Example: renaming variables
  - Maximize $2x_1 - 3x_2 + 3x_3$
  - subject to
    - $x_1 + x_2 - x_3 = 7$
    - $x_1 - 2x_2 + 2x_3 \leq 4$
    - $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$

Linear Programming: Solving LP

- Example: Equality to inequality
  - Maximize $2x_1 - 3x_2 + 3x_3$
  - subject to
    - $x_1 + x_2 - x_3 \leq 7$
    - $-x_1 - x_2 + x_3 \leq -7$
    - $x_1 - 2x_2 + 2x_3 \leq 4$
    - $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$

Linear Programming: Solving LP

- **<u>Standard form</u>**: A Linear Program is said to be *standard form* if the following holds:
  1. The linear objective function should be *maximized*.
  2. All variables have non-negativity constraint.
     i.e., for all i, $x_i \geq 0$.
  3. All the remaining linear constraints are of the following form:
     $\sum_{j=1}^{n} a_j \cdot x_j \leq b_j$.

- It will be useful to further convert an LP in standard for to an equivalent LP in *Slack form*.
  - <u>Slack form</u>: For every inequality $\sum_j a_j x_j \leq b_j$, we introduce a *slack* variable s_j and replace$\sum_j a_j x_j \leq b_j$ with $s_j = b_j - \sum_j a_j x_j$ and $s_j \geq 0$.

Linear Programming: Solving LP

- Example:
  - Maximize $2x_1 - 3x_2 + 3x_3$
  - subject to
    - $x_1 + x_2 - x_3 \leq 7$
    - $-x_1 - x_2 + x_3 \leq -7$
    - $x_1 - 2x_2 + 2x_3 \leq 4$
    - $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$

Linear Programming: Solving LP

- Example: Standard form to slack form.
  - $z = 2x_1 - 3x_2 + 3x_3$
  - $x_4 = 7 - x_1 - x_2 + x_3$
  - $x_5 = -7 + x_1 + x_2 - x_3$
  - $x_6 = 4 - x_1 + 2x_2 - 2x_3$
  - $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0, x_6 \geq 0.$
- The variables on the LHS are called *basic variables* and those on the RHS are called *non-basic variables*.
- Basic solution: Set all non-basic variables to 0 and compute the value of the basic variables.

Linear Programming: Solving LP

- The variables on the LHS are called *basic variables* and those on the RHS are called *non-basic variables*.
- Basic solution: Set all non-basic variables to 0 and compute the value of the basic variables.
- Simplex algorithm:
  - Repeat:
    - **Pivot**: Rewrite the LP in slack form such that the objective value of the basic solution increases.

Linear Programming:

The Simplex Algorithm

Linear Programming: Solving LP

- Simplex algorithm:
  - Repeat:
    - **Pivot**: Rewrite the LP in slack form such that the objective value of the basic solution increases.
- Example:
  - $z = 3x_1 + x_2 + 2x_3$
  - $x_4 = 30 - x_1 - x_2 - 3x_3$
  - $x_5 = 24 - 2x_1 - 2x_2 - 5x_3$
  - $x_6 = 36 - 4x_1 - x_2 - 2x_3$
- Use $x_1 = (9 - x_6/4 - x_2/4 - x_3/2)$

Linear Programming: Solving LP

- Simplex algorithm:
  - Repeat:
    - **Pivot**: Rewrite the LP in slack form such that the objective value of the basic solution increases.
- Example:
  - $z = 3(9 - x_6/4 - x_2/4 - x_3/2) + x_2 + 2x_3$
  - $x_4 = 30 - (9 - x_6/4 - x_2/4 - x_3/2) - x_2 - 3x_3$
  - $x_5 = 24 - 2(9 - x_6/4 - x_2/4 - x_3/2) - 2x_2 - 5x_3$
  - $x_1 = (9 - x_6/4 - x_2/4 - x_3/2)$

Linear Programming: Solving LP

- <u>Simplex algorithm</u>:
  - Repeat:
    - **Pivot**: Rewrite the LP in slack form such that the objective value of the basic solution increases.
- Example:
  - $z = 27 + x_2/4 + x_3/2 - 3x_6/4$
  - $x_4 = 21 - 3x_2/4 - 5x_3/2 + x_6/4$
  - $x_5 = 6 - 3x_2/2 - 4x_3 + x_6/2$
  - $x_1 = 9 - x_2/4 - x_3/2 - x_6/4$
- Now x_2, x_3, and x_6 are the non-basic variables and x_1, x_4, and x_5 are the basic variables.
- The objective value of the *basic solution* is now 27.
- <u>Claim</u>: If the basic solution is feasible for the LP before pivoting, then the basic solution for the LP after pivoting is also feasible.

Linear Programming: Solving LP

- <u>Simplex algorithm</u>:
  - Repeat:
    - **Pivot**: Rewrite the LP in slack form such that the objective value of the basic solution increases.
- Example:
  - $z = 27 + x_2/4 + x_3/2 - 3x_6/4$
  - $x_4 = 21 - 3x_2/4 - 5x_3/2 + x_6/4$
  - $x_5 = 6 - 3x_2/2 - 4x_3 + x_6/2$
  - $x_1 = 9 - x_2/4 - x_3/2 - x_6/4$
- Use $x_3 = 3/2 - 3x_2/8 - x_5/4 + x_6/8$

Linear Programming: Solving LP

- <u>Simplex algorithm</u>:
  - Repeat:
    - **Pivot**: Rewrite the LP in slack form such that the objective value of the basic solution increases.
- Example:
  - $z = 111/4 + x_2/16 - x_5/8 - 11x_6/16$
  - $x_4 = 69/4 + 3x_2/16 + 5x_5/8 - x_6/16$
  - $x_1 = 33/4 - x_2/16 + x_5/8 - 5x_6/16$
  - $x_3 = 3/2 - 3x_2/8 - x_5/4 + x_6/8$
- Now x_2, x_5, and x_6 are the non-basic variables and x_1, x_3, and x_4 are the basic variables.
- The objective value of the *basic solution* is now $111/4$.

Linear Programming: Solving LP

- Simplex algorithm:
  - Repeat:
    - **Pivot**: Rewrite the LP in slack form such that the objective value of the basic solution increases.
- Example:
  - $z = 111/4 + x_2/16 - x_5/8 - 11x_6/16$
  - $x_4 = 69/4 + 3x_2/16 + 5x_5/8 - x_6/16$
  - $x_1 = 33/4 - x_2/16 + x_5/8 - 5x_6/16$
  - $x_3 = 3/2 - 3x_2/8 - x_5/4 + x_6/8$
- Use $x_2 = 4 - 8x_3/3 - 2x_5/3 + x_6/3$

Linear Programming: Solving LP

- <u>Simplex algorithm</u>:
  - Repeat:
    - **Pivot**: Rewrite the LP in slack form such that the objective value of the basic solution increases.
- Example:
  - $z = 28 - x_3/6 - x_5/6 - 2x_6/3$
  - $x_1 = 8 + x_3/6 + x_5/6 - x_6/3$
  - $x_2 = 4 - 8x_3/3 - 2x_5/3 + x_6/3$
  - $x_4 = 18 - x_3/2 + x_5/2$
- Now the basic solution is the optimal solution.
- The optimal objective value for the initial LP is 28 and the value of the variables are $x_1 = 8$, $x_2 = 4$, and $x_3 = 0$.

Linear Programming: Solving LP

- Simplex algorithm:
  - Repeat:
    - **Pivot**: Rewrite the LP in slack form such that the objective value of the basic solution increases.
- We looked at a contrived example devoid of any complications. Here are some of the complications that could arise:
  1. What if the initial basic solution is not a feasible solution?
  2. What if the LP is unbounded? How and where do we detect this?
  3. What if after a pivoting step the objective value of the basic solution does not increase? What is the running time of the Simplex algorithm?

Linear Programming: Solving LP

- Complications:
  1. What if the initial basic solution is not a feasible solution?
     - We will determine this in a preprocessing step. If the LP has a feasible solution, then we will rewrite it in a form where the basic solution is feasible.
  2. What if the LP is unbounded? How and where do we detect this?
     - We will check this while pivoting.
  3. What if after a pivoting step the objective value of the basic solution does not increase? What is the running time of the Simplex algorithm?
     - This is indeed a problem with Simplex. The algorithm might cycle without increasing the objective value. Simplex is actually not a polynomial time algorithm but it is still used in practice because it works very well on real world instances.

Linear Programming: Solving LP

- (*Complication 2*) What if the LP is unbounded? How and where do we detect this?
- Consider the following general slack LP that we obtain while running Simplex:
- $z \quad = v + c_1x_1 + c_2x_2 + \ldots + c_nx_n$
- $x_{n+1} = b_1 - a_{11}x_1 - a_{12}x_2 - \ldots - a_{1nxn}$
- $x_{n+2} = b_2 - a_{21}x_1 - a_{22}x_2 - \ldots - a_{2nxn}$
- .
- $x_{n+m} = b_m - a_{m1}x_1 - a_{m2}x_2 - \ldots - a_{mn}x_n$
- <u>Claim</u>: Suppose $c_i > 0$ and $a_{1i}, a_{2i}, a_{3i}, \ldots, a_{mi} \leq 0$. Then the LP is unbounded.

Linear Programming: Solving LP

- (*Complication 3*) What if after a pivoting step the objective value of the basic solution does not increase? What is the running time of the Simplex algorithm?
- Consider the following example:
- $z \ = \ 8 \ + x_3 - x_4$
- $x_1 = \ 8 - x_2 \ - x_4$
- $x_5 = \ x_2 - \ x_3$
- We have to pivot using $x_3 = x_2 - x_5$ but that gives us
- $z \ = \ 8 \ + x_2 - x_4 - x_5$
- $x_1 = \ 8 - x_2 \ - x_4$
- $x_3 = \ x_2 - \ x_5$
- The objective value of the basic solution does not change.

Linear Programming: Solving LP

- (*Complication 3*) What if after a pivoting step the objective value of the basic solution does not increase? What is the running time of the Simplex algorithm?
- So, the Simplex may cycle between slack forms without increasing the objective value of the basic solution.
- Claim: Each slack form is uniquely determined by the set of basic and non-basic variables.
- Question: What is the upper bound on the number of slack forms that the Simplex cycles without increasing the objective value of the basic solution?

Linear Programming: Solving LP

- (*Complication 3*) What if after a pivoting step the objective value of the basic solution does not increase? What is the running time of the Simplex algorithm?
- So, the Simplex may cycle between slack forms without increasing the objective value of the basic solution.
- Claim: Each slack form is uniquely determined by the set of basic and non-basic variables.
- Question: What is the upper bound on the number of slack forms that the Simplex cycles without increasing the objective value of the basic solution?
  - ${}^{n+m}C_m$. This is the upper bound on the number of different slack forms.

Linear Programming: Solving LP

- (*Complication 3*) What if after a pivoting step the objective value of the basic solution does not increase? What is the running time of the Simplex algorithm?
- So, the Simplex may cycle between slack forms without increasing the objective value of the basic solution.
- <u>Claim</u>: Each slack form is uniquely determined by the set of basic and non-basic variables.
- <u>Claim</u>: If the Simplex fails to terminate in ${}^{n+m}C_m$ steps, then it cycles.
- There is a way (*Bland's rule*) to choose the pivoting variables so that Simplex always terminates.

Linear Programming: Solving LP

- (*Complication 1*) What if the initial basic solution is not a feasible solution?
- We construct the following LP, L' in slack form:
- $z = -x_0$
- $x_{n+1} = b_1 - a_{11}x_1 - a_{12}x_2 - \ldots - a_{1nxn} + x_0$
- $x_{n+2} = b_2 - a_{21}x_1 - a_{22}x_2 - \ldots - a_{2n}x_n + x_0$
- .
- $x_{n+m} = b_m - a_{m1}x_1 - a_{m2}x_2 - \ldots - a_{mn}x_n + x_0$
- <u>Claim</u>: The given LP has a feasible solution if and only if the optimal objective value of L' is 0.
- So, all we need to do is to solve L'. This seems to bring us back to the original problem. However, we see that L' is a *simple* LP.

Linear Programming: Solving LP

- (*Complication 1*) What if the initial basic solution is not a feasible solution?
- We construct the following LP, L' in slack form:
- $z = -x_0$
- $x_{n+1} = b_1 - a_{11}x_1 - a_{12}x_2 - \ldots - a_{1nxn} + x_0$
- $x_{n+2} = b_2 - a_{21}x_1 - a_{22}x_2 - \ldots - a_{2nxn} + x_0$
- .
- $x_{n+m} = b_m - a_{m1}x_1 - a_{m2}x_2 - \ldots - a_{mn}x_n + x_0$
- <u>Claim</u>: The given LP has a feasible solution if and only if the optimal objective value of L' is 0.
- <u>Claim</u>: L' is feasible.
- The basic solution might not be a feasible solution since some $b_i < 0$.

Linear Programming: Solving LP

- (*Complication 1*) What if the initial basic solution is not a feasible solution?
- *L'*:
- $z = -x_0$
- $x_{n+1} = b_1 - a_{11}x_1 - a_{12}x_2 - \ldots - a_{1nxn} + x_0$
- $x_{n+2} = b_2 - a_{21}x_1 - a_{22}x_2 - \ldots - a_{2nxn} + x_0$
- .
- $x_{n+m} = b_m - a_{m1}x_1 - a_{m2}x_2 - \ldots - a_{mn}x_n + x_0$
- The basic solution might not be a feasible solution since some $b_i < 0$.
- Let b_i be the smallest among $b_1, \ldots, b_m$. We will pivot using

$$\mathrm{x}_{n+i} = b_i - a_{i1}x_1 - \ldots + x_0$$

Linear Programming: Solving LP

- (*Complication 1*) What if the initial basic solution is not a feasible solution?
- *L'*:
- $z = -x_0$
- $x_{n+1} = b_1 - a_{11}x_1 - a_{12}x_2 - \ldots - a_{1nxn} + x_0$
- $x_{n+2} = b_2 - a_{21}x_1 - a_{22}x_2 - \ldots - a_{2nxn} + x_0$
- .
- $x_{n+m} = b_m - a_{m1}x_1 - a_{m2}x_2 - \ldots - a_{mn}x_n + x_0$
- Let b_i be the smallest among $b_1, \ldots, b_m$. We will pivot using
$$x_{n+i} = b_i - a_{i1}x_1 - \ldots + x_0$$
- <u>Claim</u>: The basic solution of the LP obtained after the above pivoting is a feasible solution.

Linear Programming: Solving LP

- (*Complication 1*) What if the initial basic solution is not a feasible solution?
- Pre-processing algorithm:
  - Given L, check if all b_i's are positive. In that case return L.
  - Consider L'. Perform the pivoting using the equation with smallest b_i to obtain L''.
  - Solve L'' using Simplex and find the optimal objective value Opt.
  - If ($Opt \neq 0$), then output "LP is infeasible".
  - Otherwise, let L_S be the LP obtained at the end of the simplex. Do the following:
    - If x_0 is a basic variable in L_S, then perform a pivoting step to obtain L_S'.
    - Remove all instances of x_0 and rewrite the objective function of L in terms of non-basic variables of L_S'.

Linear Programming: Solving LP

- (*Complication 1*) What if the initial basic solution is not a feasible solution?
- Pre-processing algorithm: Example
- L:
  - $z = 2x_1 - x_2$
  - $x_3 = 2 - 2x_1 + x_2$
  - $x_4 = -4 - x_1 + 5x_2$
- L':
  - $z = - x_0$
  - $x_3 = 2 - 2x_1 + x_2 + x_0$
  - $x_4 = -4 - x_1 + 5x_2 + x_0$
- L'': After Pivot using ($x_4 = \ldots$)
  - $z = -4 - x_1 + 5x_2 - x_4$
  - $x_3 = 6 - x_1 - 4x_2 + x_4$
  - $x_0 = 4 + x_1 - 5x_2 + x_4$

Linear Programming: Solving LP

- (*Complication 1*) What if the initial basic solution is not a feasible solution?
- Pre-processing algorithm: Example
- L:
  - $z = 2x_1 - x_2$
  - $x_3 = 2 - 2x_1 + x_2$
  - $x_4 = -4 - x_1 + 5x_2$
- L_S:
  - $z = -x_0$
  - $x_2 = 4/5 - x_0/5 + x_1/5 + x_4/5$
  - $x_3 = 14/5 + 4x_0/5 - 9x_1/5 + x_4/5$
- L_S:
  - $z = 2x_1 - x_2 = 2x_1 - (4/5 + x_1/5 + x_4/5) = -4/5 + 9x_1/5 - x_4/5$
  - $x_2 = 4/5 + x_1/5 + x_4/5$
  - $x_3 = 14/5 - 9x_1/5 + x_4/5$

End