Algorithms: COMP3121/3821/9101/9801

Aleks Ignjatović

School of Computer Science and Engineering
University of New South Wales

LECTURE 9: LINEAR PROGRAMMING

Problem:

- You are given a list of food sources $f_1, f_2, \ldots, f_n$;

Linear Programming problems - Example 1

Problem:

- You are given a list of food sources $f_1, f_2, \ldots, f_n$;
- for each source f_i you are given:

Linear Programming problems - Example 1

Problem:

- You are given a list of food sources $f_1, f_2, \dots, f_n$;
- for each source f_i you are given:
  - its price per gram p_i;

Linear Programming problems - Example 1

Problem:

- You are given a list of food sources $f_1, f_2, \dots, f_n$;
- for each source f_i you are given:
  - its price per gram p_i;
  - the number of calories c_i per gram, and

Linear Programming problems - Example 1

Problem:

- You are given a list of food sources $f_1, f_2, \dots, f_n$;
- for each source f_i you are given:
  - its price per gram p_i;
  - the number of calories c_i per gram, and
  - for each of 13 vitamins $V_1, V_2, \dots, V_{13}$ you are given the content $v(i,j)$ of milligrams of vitamin V_j in one gram of food source f_i.

Linear Programming problems - Example 1

Problem:

- You are given a list of food sources $f_1, f_2, \dots, f_n$;
- for each source f_i you are given:
  - its price per gram p_i;
  - the number of calories c_i per gram, and
  - for each of 13 vitamins $V_1, V_2, \dots, V_{13}$ you are given the content $v(i, j)$ of milligrams of vitamin V_j in one gram of food source f_i.
- Your task: to find a combination of quantities of food sources such that:

Linear Programming problems - Example 1

Problem:

- You are given a list of food sources $f_1, f_2, \ldots, f_n$;
- for each source f_i you are given:
  - its price per gram p_i;
  - the number of calories c_i per gram, and
  - for each of 13 vitamins $V_1, V_2, \ldots, V_{13}$ you are given the content $v(i, j)$ of milligrams of vitamin V_j in one gram of food source f_i.
- Your task: to find a combination of quantities of food sources such that:
  - the total number of calories in all of the chosen food is equal to a recommended daily value of 2000 calories;

Linear Programming problems - Example 1

Problem:

- You are given a list of food sources $f_1, f_2, \ldots, f_n$;
- for each source f_i you are given:
  - its price per gram p_i;
  - the number of calories c_i per gram, and
  - for each of 13 vitamins $V_1, V_2, \ldots, V_{13}$ you are given the content $v(i, j)$ of milligrams of vitamin V_j in one gram of food source f_i.
- Your task: to find a combination of quantities of food sources such that:
  - the total number of calories in all of the chosen food is equal to a recommended daily value of 2000 calories;
  - the total intake of each vitamin V_j is at least the recommended daily intake of w_j milligrams for all $1 \leq j \leq 13$;

Linear Programming problems - Example 1

Problem:

- You are given a list of food sources $f_1, f_2, \ldots, f_n$;
- for each source f_i you are given:
  - its price per gram p_i;
  - the number of calories c_i per gram, and
  - for each of 13 vitamins $V_1, V_2, \ldots, V_{13}$ you are given the content $v(i,j)$ of milligrams of vitamin V_j in one gram of food source f_i.
- Your task: to find a combination of quantities of food sources such that:
  - the total number of calories in all of the chosen food is equal to a recommended daily value of 2000 calories;
  - the total intake of each vitamin V_j is at least the recommended daily intake of w_j milligrams for all $1 \leq j \leq 13$;
  - the price of all food per day is as low as possible.

Linear Programming problems - Example 1 cont.

- To obtain the corresponding constraints let us assume that we take x_i grams of each food source f_i for $1 \leq i \leq n$. Then:

Linear Programming problems - Example 1 cont.

- To obtain the corresponding constraints let us assume that we take x_i grams of each food source f_i for $1 \leq i \leq n$. Then:
  - the total number of calories must satisfy

$$\sum_{i=1}^{n} x_i c_i = 2000;$$

Linear Programming problems - Example 1 cont.

- To obtain the corresponding constraints let us assume that we take x_i grams of each food source f_i for $1 \leq i \leq n$. Then:
  - the total number of calories must satisfy

$$\sum_{i=1}^{n} x_i c_i = 2000;$$

  - for each vitamin V_j the total amount in all food must satisfy

$$\sum_{i=1}^{n} x_i v(i,j) \geq w_j \quad (1 \leq j \leq 13);$$

Linear Programming problems - Example 1 cont.

- To obtain the corresponding constraints let us assume that we take x_i grams of each food source f_i for $1 \leq i \leq n$. Then:
  - the total number of calories must satisfy
  $$\sum_{i=1}^{n} x_i c_i = 2000;$$
  - for each vitamin V_j the total amount in all food must satisfy
  $$\sum_{i=1}^{n} x_i v(i,j) \geq w_j \quad (1 \leq j \leq 13);$$
  - an implicit assumption is that all the quantities must be non-negative numbers,
  $$x_i \geq 0, \quad 1 \leq i \leq n.$$

Linear Programming problems - Example 1 cont.

- To obtain the corresponding constraints let us assume that we take x_i grams of each food source f_i for $1 \leq i \leq n$. Then:
  - the total number of calories must satisfy
  $$\sum_{i=1}^{n} x_i c_i = 2000;$$
  - for each vitamin V_j the total amount in all food must satisfy
  $$\sum_{i=1}^{n} x_i v(i,j) \geq w_j \quad (1 \leq j \leq 13);$$
  - an implicit assumption is that all the quantities must be non-negative numbers,
  $$x_i \geq 0, \quad 1 \leq i \leq n.$$
- Our goal is to minimise the objective function which is the total cost $y = \sum_{i=1}^{n} x_i p_i$.

Linear Programming problems - Example 1 cont.

- To obtain the corresponding constraints let us assume that we take x_i grams of each food source f_i for $1 \leq i \leq n$. Then:
  - the total number of calories must satisfy

$$\sum_{i=1}^{n} x_i c_i = 2000;$$

  - for each vitamin V_j the total amount in all food must satisfy

$$\sum_{i=1}^{n} x_i v(i,j) \geq w_j \quad (1 \leq j \leq 13);$$

  - an implicit assumption is that all the quantities must be non-negative numbers,

$$x_i \geq 0, \quad 1 \leq i \leq n.$$

- Our goal is to minimise the objective function which is the total cost $y = \sum_{i=1}^{n} x_i p_i$.
- Note that here all the equalities and inequalities, as well as the objective function, are **linear**. This problem is a typical example of a **Linear Programming problem**.

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.
- You can promise that you will build

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.
- You can promise that you will build
  - a certain number of bridges, each 3 billion a piece;

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.
- You can promise that you will build
  - a certain number of bridges, each 3 billion a piece;
  - a certain number of rural airports, each 2 billion a piece, and

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.
- You can promise that you will build
  - a certain number of bridges, each 3 billion a piece;
  - a certain number of rural airports, each 2 billion a piece, and
  - a certain number of olympic swimming pools each a billion a piece.

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.
- You can promise that you will build
  - a certain number of bridges, each 3 billion a piece;
  - a certain number of rural airports, each 2 billion a piece, and
  - a certain number of olympic swimming pools each a billion a piece.
- You were told by your wise advisers that

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.
- You can promise that you will build
  - a certain number of bridges, each 3 billion a piece;
  - a certain number of rural airports, each 2 billion a piece, and
  - a certain number of olympic swimming pools each a billion a piece.
- You were told by your wise advisers that
  - each bridge you promise brings you 5% of city votes, 7% of suburban votes and 9% of rural votes;

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.
- You can promise that you will build
  - a certain number of bridges, each 3 billion a piece;
  - a certain number of rural airports, each 2 billion a piece, and
  - a certain number of olympic swimming pools each a billion a piece.
- You were told by your wise advisers that
  - each bridge you promise brings you 5% of city votes, 7% of suburban votes and 9% of rural votes;
  - each rural airport you promise brings you no city votes, 2% of suburban votes and 15% of rural votes;

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.
- You can promise that you will build
  - a certain number of bridges, each 3 billion a piece;
  - a certain number of rural airports, each 2 billion a piece, and
  - a certain number of olympic swimming pools each a billion a piece.
- You were told by your wise advisers that
  - each bridge you promise brings you 5% of city votes, 7% of suburban votes and 9% of rural votes;
  - each rural airport you promise brings you no city votes, 2% of suburban votes and 15% of rural votes;
  - each olympic swimming pool promised brings you 12% of city votes, 3% of suburban votes and no rural votes.

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.
- You can promise that you will build
  - a certain number of bridges, each 3 billion a piece;
  - a certain number of rural airports, each 2 billion a piece, and
  - a certain number of olympic swimming pools each a billion a piece.
- You were told by your wise advisers that
  - each bridge you promise brings you 5% of city votes, 7% of suburban votes and 9% of rural votes;
  - each rural airport you promise brings you no city votes, 2% of suburban votes and 15% of rural votes;
  - each olympic swimming pool promised brings you 12% of city votes, 3% of suburban votes and no rural votes.
- In order to win, you have to get at least 51% of each of the city, suburban and rural votes.

Linear Programming problems - Example 2

Problem:

- Assume now that you are the (Shadow?) Treasurer and you want to make certain promises to the electorate which will ensure that your party will win in the forthcoming elections.
- You can promise that you will build
  - a certain number of bridges, each 3 billion a piece;
  - a certain number of rural airports, each 2 billion a piece, and
  - a certain number of olympic swimming pools each a billion a piece.
- You were told by your wise advisers that
  - each bridge you promise brings you 5% of city votes, 7% of suburban votes and 9% of rural votes;
  - each rural airport you promise brings you no city votes, 2% of suburban votes and 15% of rural votes;
  - each olympic swimming pool promised brings you 12% of city votes, 3% of suburban votes and no rural votes.
- In order to win, you have to get at least 51% of each of the city, suburban and rural votes.
- You wish to win the election by cleverly making a promise that **appears** that it will blow as small hole in the budget as possible.

Linear Programming problems - Example 2

- We can let the number of bridges to be built be x_b, number of airports x_a and the number of swimming pools x_p.

Linear Programming problems - Example 2

- We can let the number of bridges to be built be x_b, number of airports x_a and the number of swimming pools x_p.
- We now see that the problem amounts to minimising the objective $y = 3x_b + 2x_a + x_p$, while making sure that the following constraints are satisfied:

Linear Programming problems - Example 2

- We can let the number of bridges to be built be x_b, number of airports x_a and the number of swimming pools x_p.
- We now see that the problem amounts to minimising the objective $y = 3x_b + 2x_a + x_p$, while making sure that the following constraints are satisfied:

$$\begin{aligned}
0.05x_b + 0.12x_p &\geq 0.51 \quad \text{(securing majority of city votes)}\\
0.07x_b + 0.02x_a + 0.03x_p &\geq 0.51 \quad \text{(securing majority of suburban votes)}\\
0.09x_b + 0.15x_a &\geq 0.51 \quad \text{(securing majority of rural votes)}\\
x_b, x_a, x_p &\geq 0.
\end{aligned}$$

Linear Programming problems - Example 2

- We can let the number of bridges to be built be x_b, number of airports x_a and the number of swimming pools x_p.
- We now see that the problem amounts to minimising the objective $y = 3x_b + 2x_a + x_p$, while making sure that the following constraints are satisfied:

$$\begin{aligned}
0.05x_b \qquad\qquad\; + 0.12x_p &\geq 0.51 \quad \text{(securing majority of city votes)}\\
0.07x_b + 0.02x_a + 0.03x_p &\geq 0.51 \quad \text{(securing majority of suburban votes)}\\
0.09x_b + 0.15x_a \qquad\qquad &\geq 0.51 \quad \text{(securing majority of rural votes)}\\
x_b, x_a, x_p &\geq 0.
\end{aligned}$$

- However, there is a very significant difference with the first example:

Linear Programming problems - Example 2

- We can let the number of bridges to be built be x_b, number of airports x_a and the number of swimming pools x_p.
- We now see that the problem amounts to minimising the objective $y = 3x_b + 2x_a + x_p$, while making sure that the following constraints are satisfied:

$$\begin{aligned}
0.05x_b \phantom{{}+0.02x_a} + 0.12x_p &\geq 0.51 \quad \text{(securing majority of city votes)}\\
0.07x_b + 0.02x_a + 0.03x_p &\geq 0.51 \quad \text{(securing majority of suburban votes)}\\
0.09x_b + 0.15x_a \phantom{{}+0.03x_p} &\geq 0.51 \quad \text{(securing majority of rural votes)}\\
x_b, x_a, x_p &\geq 0.
\end{aligned}$$

- However, there is a very significant difference with the first example:
  - you can eat 1.56 grams of chocolate, but

Linear Programming problems - Example 2

- We can let the number of bridges to be built be x_b, number of airports x_a and the number of swimming pools x_p.
- We now see that the problem amounts to minimising the objective $y = 3x_b + 2x_a + x_p$, while making sure that the following constraints are satisfied:

$$\begin{aligned}
0.05x_b \qquad\qquad\;\; + 0.12x_p &\geq 0.51 \quad \text{(securing majority of city votes)}\\
0.07x_b + 0.02x_a + 0.03x_p &\geq 0.51 \quad \text{(securing majority of suburban votes)}\\
0.09x_b + 0.15x_a \qquad\qquad\;\; &\geq 0.51 \quad \text{(securing majority of rural votes)}\\
x_b, x_a, x_p &\geq 0.
\end{aligned}$$

- However, there is a very significant difference with the first example:
  - you can eat 1.56 grams of chocolate, but
  - you cannot promise to build 1.56 bridges, 2.83 airports and 0.57 swimming pools!

Linear Programming problems - Example 2

- We can let the number of bridges to be built be x_b, number of airports x_a and the number of swimming pools x_p.
- We now see that the problem amounts to minimising the objective $y = 3x_b + 2x_a + x_p$, while making sure that the following constraints are satisfied:

$$\begin{aligned}
0.05x_b \qquad\qquad\quad + 0.12x_p &\geq 0.51 \quad \text{(securing majority of city votes)}\\
0.07x_b + 0.02x_a + 0.03x_p &\geq 0.51 \quad \text{(securing majority of suburban votes)}\\
0.09x_b + 0.15x_a \qquad\qquad\quad &\geq 0.51 \quad \text{(securing majority of rural votes)}\\
x_b, x_a, x_p &\geq 0.
\end{aligned}$$

- However, there is a very significant difference with the first example:
  - you can eat 1.56 grams of chocolate, but
  - you cannot promise to build 1.56 bridges, 2.83 airports and 0.57 swimming pools!
- The second example is an example of an **Integer Linear Programming problem**, which requires all the solutions to be integers.

Linear Programming problems - Example 2

- We can let the number of bridges to be built be x_b, number of airports x_a and the number of swimming pools x_p.
- We now see that the problem amounts to minimising the objective $y = 3x_b + 2x_a + x_p$, while making sure that the following constraints are satisfied:

$$\begin{aligned}
0.05x_b \qquad\qquad\quad + 0.12x_p &\geq 0.51 \quad \text{(securing majority of city votes)}\\
0.07x_b + 0.02x_a + 0.03x_p &\geq 0.51 \quad \text{(securing majority of suburban votes)}\\
0.09x_b + 0.15x_a \qquad\qquad\quad &\geq 0.51 \quad \text{(securing majority of rural votes)}\\
x_b, x_a, x_p &\geq 0.
\end{aligned}$$

- However, there is a very significant difference with the first example:
  - you can eat 1.56 grams of chocolate, but
  - you cannot promise to build 1.56 bridges, 2.83 airports and 0.57 swimming pools!
- The second example is an example of an **Integer Linear Programming problem**, which requires all the solutions to be integers.
- Such problems are MUCH harder to solve than the “plain” Linear Programming problems whose solutions can be real numbers.

Linear Programming problems - Example 2

- We can let the number of bridges to be built be x_b, number of airports x_a and the number of swimming pools x_p.
- We now see that the problem amounts to minimising the objective $y = 3x_b + 2x_a + x_p$, while making sure that the following constraints are satisfied:

$$\begin{aligned}
0.05x_b \qquad\qquad + 0.12x_p &\geq 0.51 \quad \text{(securing majority of city votes)}\\
0.07x_b + 0.02x_a + 0.03x_p &\geq 0.51 \quad \text{(securing majority of suburban votes)}\\
0.09x_b + 0.15x_a \qquad\qquad &\geq 0.51 \quad \text{(securing majority of rural votes)}\\
x_b, x_a, x_p &\geq 0.
\end{aligned}$$

- However, there is a very significant difference with the first example:
  - you can eat 1.56 grams of chocolate, but
  - you cannot promise to build 1.56 bridges, 2.83 airports and 0.57 swimming pools!
- The second example is an example of an **Integer Linear Programming problem**, which requires all the solutions to be integers.
- Such problems are MUCH harder to solve than the "plain" Linear Programming problems whose solutions can be real numbers.
- In fact, Integer Linear Programming problems are NP hard!

Linear Programming problems

- In the **standard form** the *objective* to be maximised is given by

$$\sum_{j=1}^{n} c_j x_j$$

Linear Programming problems

- In the **standard form** the *objective* to be maximised is given by

$$\sum_{j=1}^{n} c_j x_j$$

- the *constraints* are of the form

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \qquad 1 \leq i \leq m; \tag{1}$$

$$x_j \geq 0, \qquad 1 \leq j \leq n, \tag{2}$$

Linear Programming problems

- In the **standard form** the *objective* to be maximised is given by

$$\sum_{j=1}^{n} c_j x_j$$

- the *constraints* are of the form

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \qquad 1 \leq i \leq m; \tag{1}$$

$$x_j \geq 0, \qquad 1 \leq j \leq n, \tag{2}$$

- Let the boldface $\mathbf{x}$ represent a (column) vector, $\mathbf{x} = \langle x_1 \dots x_n \rangle^{\intercal}$.

Linear Programming problems

- In the **standard form** the *objective* to be maximised is given by

$$\sum_{j=1}^{n} c_j x_j$$

- the *constraints* are of the form

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i, \qquad 1 \leq i \leq m; \tag{1}$$

$$x_j \geq 0, \qquad 1 \leq j \leq n, \tag{2}$$

- Let the boldface $\mathbf{x}$ represent a (column) vector, $\mathbf{x} = \langle x_1 \dots x_n \rangle^{\intercal}$.
- To get a more compact representation of linear programs we introduce a partial ordering on vectors $\mathbf{x} \in \mathbf{R}^n$ by $\mathbf{x} \leq \mathbf{y}$ if and only if the corresponding inequalities hold coordinate-wise, i.e., if and only if $x_j \leq y_j$ for all $1 \leq j \leq n$.

Linear Programming

- Letting $\mathbf{c} = \langle c_1 \dots c_n \rangle^\intercal \in \mathbf{R}^n$ and $\mathbf{b} = \langle b_1 \dots b_m \rangle^\intercal \in \mathbf{R}^m$, and letting A be the matrix $A = \left(a_{ij}\right)$ of size $m \times n$, we get that the above problem can be formulated simply as:
  - maximize $\mathbf{c}^\intercal \mathbf{x}$
  - subject to the following two (matrix-vector) constraints:

$$A\mathbf{x} \leq \mathbf{b}$$

and

$$\mathbf{x} \geq \mathbf{0}.$$

Linear Programming

- Letting $\mathbf{c} = \langle c_1 \ldots c_n \rangle^\intercal \in \mathbf{R}^n$ and $\mathbf{b} = \langle b_1 \ldots b_m \rangle^\intercal \in \mathbf{R}^m$, and letting A be the matrix $A = \left(a_{ij}\right)$ of size $m \times n$, we get that the above problem can be formulated simply as:
    - maximize $\mathbf{c}^\intercal \mathbf{x}$
    - subject to the following two (matrix-vector) constraints:

$$A\mathbf{x} \leq \mathbf{b}$$

    and

$$\mathbf{x} \geq \mathbf{0}.$$

- Thus, to specify a Linear Programming optimisation problem we just have to provide a triplet $(A, \mathbf{b}, \mathbf{c})$;

- Letting $\mathbf{c} = \langle c_1 \dots c_n \rangle^\intercal \in \mathbf{R}^n$ and $\mathbf{b} = \langle b_1 \dots b_m \rangle^\intercal \in \mathbf{R}^m$, and letting A be the matrix $A = \left(a_{ij}\right)$ of size $m \times n$, we get that the above problem can be formulated simply as:
  - maximize $\mathbf{c}^\intercal \mathbf{x}$
  - subject to the following two (matrix-vector) constraints:

$$A\mathbf{x} \leq \mathbf{b}$$

    and

$$\mathbf{x} \geq \mathbf{0}.$$

- Thus, to specify a Linear Programming optimisation problem we just have to provide a triplet $(A, \mathbf{b}, \mathbf{c})$;
- This is the usual form which is accepted by most standard LP solvers.

Linear Programming

- The value of the objective for any value of the variables which makes the constraints satisfied is called a *feasible solution* of the LP problem.

Linear Programming

- The value of the objective for any value of the variables which makes the constraints satisfied is called a *feasible solution* of the LP problem.
- Equality constraints of the form $\sum_{i=1}^{n} a_{ij}x_i = b_j$ can be replaced by two inequalities: $\sum_{i=1}^{n} a_{ij}x_i \geq b_j$ and $\sum_{i=1}^{n} a_{ij}x_i \leq b_j$; thus, we can assume that all constraints are inequalities.

- The value of the objective for any value of the variables which makes the constraints satisfied is called a *feasible solution* of the LP problem.
- Equality constraints of the form $\sum_{i=1}^{n} a_{ij}x_i = b_j$ can be replaced by two inequalities: $\sum_{i=1}^{n} a_{ij}x_i \geq b_j$ and $\sum_{i=1}^{n} a_{ij}x_i \leq b_j$; thus, we can assume that all constraints are inequalities.
- In general, a "natural formulation" of a problem as a Linear Program does not necessarily produce the non-negativity constrains for all of the variables.

Linear Programming

- The value of the objective for any value of the variables which makes the constraints satisfied is called a *feasible solution* of the LP problem.
- Equality constraints of the form $\sum_{i=1}^{n} a_{ij}x_i = b_j$ can be replaced by two inequalities: $\sum_{i=1}^{n} a_{ij}x_i \geq b_j$ and $\sum_{i=1}^{n} a_{ij}x_i \leq b_j$; thus, we can assume that all constraints are inequalities.
- In general, a "natural formulation" of a problem as a Linear Program does not necessarily produce the non-negativity constrains for all of the variables.
- However, in the standard form such constraints are required for all of the variables.

Linear Programming

- The value of the objective for any value of the variables which makes the constraints satisfied is called a *feasible solution* of the LP problem.
- Equality constraints of the form $\sum_{i=1}^{n} a_{ij}x_i = b_j$ can be replaced by two inequalities: $\sum_{i=1}^{n} a_{ij}x_i \geq b_j$ and $\sum_{i=1}^{n} a_{ij}x_i \leq b_j$; thus, we can assume that all constraints are inequalities.
- In general, a "natural formulation" of a problem as a Linear Program does not necessarily produce the non-negativity constrains for all of the variables.
- However, in the standard form such constraints are required for all of the variables.
- This poses no problem, because each occurrence of an unconstrained variable x_j can be replaced by the expression $x'_j - x^*_j$ where x'_j, x^*_j are new variables satisfying the constraints $x'_j \geq 0,\ x^*_j \geq 0$.

Linear Programming

- The value of the objective for any value of the variables which makes the constraints satisfied is called a *feasible solution* of the LP problem.
- Equality constraints of the form $\sum_{i=1}^{n} a_{ij}x_i = b_j$ can be replaced by two inequalities: $\sum_{i=1}^{n} a_{ij}x_i \geq b_j$ and $\sum_{i=1}^{n} a_{ij}x_i \leq b_j$; thus, we can assume that all constraints are inequalities.
- In general, a “natural formulation” of a problem as a Linear Program does not necessarily produce the non-negativity constrains for all of the variables.
- However, in the standard form such constraints are required for all of the variables.
- This poses no problem, because each occurrence of an unconstrained variable x_j can be replaced by the expression $x'_j - x^*_j$ where x'_j, x^*_j are new variables satisfying the constraints $x'_j \geq 0,\ x^*_j \geq 0$.
- Also, some problems are naturally translated into constraints of the form $|A\mathbf{x}| \leq \mathbf{b}$. This also poses no problem because we can replace such constraints with two linear constraints: $A\mathbf{x} \leq \mathbf{b}$ and $-A\mathbf{x} \leq \mathbf{b}$ because $|x| \leq y$ if and only if $x \leq y$ and $-x \leq y$.

Linear Programming - Standard Form

- Standard Form: maximize $\mathbf{c}^\intercal \mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$.

Linear Programming - Standard Form

- Standard Form: maximize $\mathbf{c}^\intercal \mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$.
- Any vector $\mathbf{x}$ which satisfies the two constraints is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\intercal \mathbf{x}$ might be.

Linear Programming - Standard Form

- Standard Form: maximize $\mathbf{c}^\intercal \mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$.
- Any vector $\mathbf{x}$ which satisfies the two constraints is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\intercal \mathbf{x}$ might be.
- As an example, let us consider the following optimisation problem:

Linear Programming - Standard Form

- Standard Form: maximize $\mathbf{c}^\intercal \mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$.
- Any vector $\mathbf{x}$ which satisfies the two constraints is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\intercal \mathbf{x}$ might be.
- As an example, let us consider the following optimisation problem:

$$\text{maximize} \quad z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 \tag{3}$$

subject to the constraints

$$x_1 + x_2 + 3x_3 \leq 30 \tag{4}$$

$$2x_1 + 2x_2 + 5x_3 \leq 24 \tag{5}$$

$$4x_1 + x_2 + 2x_3 \leq 36 \tag{6}$$

$$x_1, x_2, x_3 \geq 0 \tag{7}$$

- How large can the value of the objective $z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3$ be, without violating the constraints?

Linear Programming - Standard Form

- Standard Form: maximize $\mathbf{c}^\intercal\mathbf{x}$ subject to $A\mathbf{x} \le \mathbf{b}$ and $\mathbf{x} \ge \mathbf{0}$.
- Any vector $\mathbf{x}$ which satisfies the two constraints is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\intercal\mathbf{x}$ might be.
- As an example, let us consider the following optimisation problem:

maximize
$$z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 \tag{3}$$
subject to the constraints
$$x_1 + x_2 + 3x_3 \le 30 \tag{4}$$
$$2x_1 + 2x_2 + 5x_3 \le 24 \tag{5}$$
$$4x_1 + x_2 + 2x_3 \le 36 \tag{6}$$
$$x_1, x_2, x_3 \ge 0 \tag{7}$$

- How large can the value of the objective $z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3$ be, without violating the constraints?
- If we add inequalities (4) and (5), we get

$$3x_1 + 3x_2 + 8x_3 \le 54 \tag{8}$$

Linear Programming - Standard Form

- Standard Form: maximize $\mathbf{c}^\intercal \mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$.
- Any vector $\mathbf{x}$ which satisfies the two constraints is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\intercal \mathbf{x}$ might be.
- As an example, let us consider the following optimisation problem:

$$\text{maximize} \quad z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 \tag{3}$$

subject to the constraints

$$x_1 + x_2 + 3x_3 \leq 30 \tag{4}$$

$$2x_1 + 2x_2 + 5x_3 \leq 24 \tag{5}$$

$$4x_1 + x_2 + 2x_3 \leq 36 \tag{6}$$

$$x_1, x_2, x_3 \geq 0 \tag{7}$$

- How large can the value of the objective $z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3$ be, without violating the constraints?
- If we add inequalities (4) and (5), we get

$$3x_1 + 3x_2 + 8x_3 \leq 54 \tag{8}$$

- Since all variables are constrained to be non-negative, we are assured that

$$3x_1 + x_2 + 2x_3 \leq 3x_1 + 3x_2 + 8x_3 \leq 54$$

Linear Programming - Standard Form

- Standard Form: maximize $\mathbf{c}^\intercal\mathbf{x}$ subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$.
- Any vector $\mathbf{x}$ which satisfies the two constraints is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\intercal\mathbf{x}$ might be.
- As an example, let us consider the following optimisation problem:

$$\text{maximize} \qquad z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 \tag{3}$$

subject to the constraints

$$x_1 + x_2 + 3x_3 \leq 30 \tag{4}$$

$$2x_1 + 2x_2 + 5x_3 \leq 24 \tag{5}$$

$$4x_1 + x_2 + 2x_3 \leq 36 \tag{6}$$

$$x_1, x_2, x_3 \geq 0 \tag{7}$$

- How large can the value of the objective $z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3$ be, without violating the constraints?
- If we add inequalities (4) and (5), we get

$$3x_1 + 3x_2 + 8x_3 \leq 54 \tag{8}$$

- Since all variables are constrained to be non-negative, we are assured that

$$3x_1 + x_2 + 2x_3 \leq 3x_1 + 3x_2 + 8x_3 \leq 54$$

Linear Programming - Standard Form

$$
\begin{aligned}
\text{maximize:} \quad & z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 && (3)\\
\text{with constraints:} \quad & x_1 + x_2 + 3x_3 \leq 30 && (4)\\
& 2x_1 + 2x_2 + 5x_3 \leq 24 && (5)\\
& 4x_1 + x_2 + 2x_3 \leq 36 && (6)\\
& x_1, x_2, x_3 \geq 0 && (7)
\end{aligned}
$$

- Thus the objective $z(x_1, x_2, x_3)$ is bounded above by 54, i.e., $z(x_1, x_2, x_3) \leq 54$.

Linear Programming - Standard Form

$$
\begin{aligned}
\text{maximize:} \quad & z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 && (3)\\
\text{with constraints:} \quad & x_1 + x_2 + 3x_3 \leq 30 && (4)\\
& 2x_1 + 2x_2 + 5x_3 \leq 24 && (5)\\
& 4x_1 + x_2 + 2x_3 \leq 36 && (6)\\
& x_1, x_2, x_3 \geq 0 && (7)
\end{aligned}
$$

- Thus the objective $z(x_1, x_2, x_3)$ is bounded above by 54, i.e., $z(x_1, x_2, x_3) \leq 54$.
- Can we obtain a tighter bound? We could try to look for coefficients $y_1, y_2, y_3 \geq 0$ to be used to for a linear combination of the constraints:

$$
\begin{aligned}
y_1(x_1 + x_2 + 3x_3) &\leq 30y_1\\
y_2(2x_1 + 2x_2 + 5x_3) &\leq 24y_2\\
y_3(4x_1 + x_2 + 2x_3) &\leq 36y_3
\end{aligned}
$$

Linear Programming - Standard Form

$$\begin{aligned}
\text{maximize:} \quad & z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 & (3)\\
\text{with constraints:} \quad & x_1 + x_2 + 3x_3 \le 30 & (4)\\
& 2x_1 + 2x_2 + 5x_3 \le 24 & (5)\\
& 4x_1 + x_2 + 2x_3 \le 36 & (6)\\
& x_1, x_2, x_3 \ge 0 & (7)
\end{aligned}$$

- Thus the objective $z(x_1, x_2, x_3)$ is bounded above by 54, i.e., $z(x_1, x_2, x_3) \le 54$.
- Can we obtain a tighter bound? We could try to look for coefficients $y_1, y_2, y_3 \ge 0$ to be used to for a linear combination of the constraints:

$$\begin{aligned}
y_1(x_1 + x_2 + 3x_3) &\le 30y_1\\
y_2(2x_1 + 2x_2 + 5x_3) &\le 24y_2\\
y_3(4x_1 + x_2 + 2x_3) &\le 36y_3
\end{aligned}$$

- Then, summing up all these inequalities and factoring, we get

$$x_1(y_1 + 2y_2 + 4y_3) + x_2(y_1 + 2y_2 + y_3) + x_3(3y_1 + 5y_2 + 2y_3) \le 30y_1 + 24y_2 + 36y_3 \qquad (9)$$

Linear Programming - Standard Form

$$\begin{aligned}
\text{maximize:} \quad & z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 && (3)\\
\text{with constraints:} \quad & x_1 + x_2 + 3x_3 \leq 30 && (4)\\
& 2x_1 + 2x_2 + 5x_3 \leq 24 && (5)\\
& 4x_1 + x_2 + 2x_3 \leq 36 && (6)\\
& x_1, x_2, x_3 \geq 0 && (7)
\end{aligned}$$

- So we got

$$x_1(y_1 + 2y_2 + 4y_3) + x_2(y_1 + 2y_2 + y_3) + x_3(3y_1 + 5y_2 + 2y_3) \leq 30y_1 + 24y_2 + 36y_3 \qquad (9)$$

Linear Programming - Standard Form

$$
\begin{aligned}
\text{maximize:} \quad & z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 && (3)\\
\text{with constraints:} \quad & x_1 + x_2 + 3x_3 \le 30 && (4)\\
& 2x_1 + 2x_2 + 5x_3 \le 24 && (5)\\
& 4x_1 + x_2 + 2x_3 \le 36 && (6)\\
& x_1, x_2, x_3 \ge 0 && (7)
\end{aligned}
$$

- So we got

$$x_1(y_1 + 2y_2 + 4y_3) + x_2(y_1 + 2y_2 + y_3) + x_3(3y_1 + 5y_2 + 2y_3) \le 30y_1 + 24y_2 + 36y_3 \qquad (9)$$

- If we compare this with our objective (3) we see that if we choose y_1, y_2 and y_3 so that:

$$
\begin{aligned}
y_1 + 2y_2 + 4y_3 &\ge 3\\
y_1 + 2y_2 + y_3 &\ge 1\\
3y_1 + 5y_2 + 2y_3 &\ge 2
\end{aligned}
$$

then $\quad 3x_3 + x_2 + 2x_3 \le x_1(y_1 + 2y_2 + 4y_3) + x_2(y_1 + 2y_2 + y_3) + x_3(3y_1 + 5y_2 + 2y_3)$

Linear Programming - Standard Form

$$\begin{aligned}
&\text{maximize:} && z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 && (3)\\
&\text{with constraints:} && x_1 + x_2 + 3x_3 \leq 30 && (4)\\
& && 2x_1 + 2x_2 + 5x_3 \leq 24 && (5)\\
& && 4x_1 + x_2 + 2x_3 \leq 36 && (6)\\
& && x_1, x_2, x_3 \geq 0 && (7)
\end{aligned}$$

- So we got

$$x_1(y_1 + 2y_2 + 4y_3) + x_2(y_1 + 2y_2 + y_3) + x_3(3y_1 + 5y_2 + 2y_3) \leq 30y_1 + 24y_2 + 36y_3 \qquad (9)$$

- If we compare this with our objective (3) we see that if we choose y_1, y_2 and y_3 so that:

$$\begin{aligned}
y_1 + 2y_2 + 4y_3 &\geq 3\\
y_1 + 2y_2 + y_3 &\geq 1\\
3y_1 + 5y_2 + 2y_3 &\geq 2
\end{aligned}$$

then $\quad 3x_3 + x_2 + 2x_3 \leq x_1(y_1 + 2y_2 + 4y_3) + x_2(y_1 + 2y_2 + y_3) + x_3(3y_1 + 5y_2 + 2y_3)$

Combining this with (9) we get:

$$30y_1 + 24y_2 + 36y_3 \geq 3x_1 + x_2 + 2x_3 = z(x_1, x_2, x_3)$$

Linear Programming - Standard Form

- Consequently, in order to find as tight upper bound for our objective $z(x_1, x_2, x_3)$ of the problem P:

$$\begin{aligned}
\text{maximize:} \quad & z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 && (3)\\
\text{with constraints:} \quad & x_1 + x_2 + 3x_3 \leq 30 && (4)\\
& 2x_1 + 2x_2 + 5x_3 \leq 24 && (5)\\
& 4x_1 + x_2 + 2x_3 \leq 36 && (6)\\
& x_1, x_2, x_3 \geq 0 && (7)
\end{aligned}$$

Linear Programming - Standard Form

- Consequently, in order to find as tight upper bound for our objective $z(x_1, x_2, x_3)$ of the problem P:

$$\begin{aligned}
&\text{maximize:} && z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 && (3)\\
&\text{with constraints:} && x_1 + x_2 + 3x_3 \leq 30 && (4)\\
& && 2x_1 + 2x_2 + 5x_3 \leq 24 && (5)\\
& && 4x_1 + x_2 + 2x_3 \leq 36 && (6)\\
& && x_1, x_2, x_3 \geq 0 && (7)
\end{aligned}$$

we have to look for y_1, y_2, y_3 which

$$\begin{aligned}
&\text{minimise:} && z^*(y_1, y_2, y_3) = 30y_1 + 24y_2 + 36y_3 && (10)\\
&\text{with constraints:} && y_1 + 2y_2 + 4y_3 \geq 3 && (11)\\
& && y_1 + 2y_2 + y_3 \geq 1 && (12)\\
& && 3y_1 + 5y_2 + 2y_3 \geq 2 && (13)\\
& && y_1, y_2, y_3 \geq 0 && (14)
\end{aligned}$$

Linear Programming - Standard Form

- Consequently, in order to find as tight upper bound for our objective $z(x_1, x_2, x_3)$ of the problem P:

$$
\begin{aligned}
\text{maximize:} \quad & z(x_1, x_2, x_3) = 3x_1 + x_2 + 2x_3 && (3)\\
\text{with constraints:} \quad & x_1 + x_2 + 3x_3 \leq 30 && (4)\\
& 2x_1 + 2x_2 + 5x_3 \leq 24 && (5)\\
& 4x_1 + x_2 + 2x_3 \leq 36 && (6)\\
& x_1, x_2, x_3 \geq 0 && (7)
\end{aligned}
$$

we have to look for y_1, y_2, y_3 which

$$
\begin{aligned}
\text{minimise:} \quad & z^*(y_1, y_2, y_3) = 30y_1 + 24y_2 + 36y_3 && (10)\\
\text{with constraints:} \quad & y_1 + 2y_2 + 4y_3 \geq 3 && (11)\\
& y_1 + 2y_2 + y_3 \geq 1 && (12)\\
& 3y_1 + 5y_2 + 2y_3 \geq 2 && (13)\\
& y_1, y_2, y_3 \geq 0 && (14)
\end{aligned}
$$

- The new problem is called the *dual problem* P^* for the original problem P.

Linear Programming - Standard Form

- Let us now repeat the whole procedure with P^* in place of P, i.e., let us find the dual program $(P^*)^*$ of P^*.

Linear Programming - Standard Form

- Let us now repeat the whole procedure with P^* in place of P, i.e., let us find the dual program $(P^*)^*$ of P^*.
- We are now looking for $z_1, z_2, z_3 \geq 0$ to multiply inequalities (11)-(13) and obtain

$$\begin{aligned} z_1(y_1 + 2y_2 + 4y_3) &\geq 3z_1 \\ z_2(y_1 + 2y_2 + y_3) &\geq z_2 \\ z_3(3y_1 + 5y_2 + 2y_3) &\geq 2z_3 \end{aligned}$$

Linear Programming - Standard Form

- Let us now repeat the whole procedure with P^* in place of P, i.e., let us find the dual program $(P^*)^*$ of P^*.
- We are now looking for $z_1, z_2, z_3 \geq 0$ to multiply inequalities (11)-(13) and obtain

$$z_1(y_1 + 2y_2 + 4y_3) \geq 3z_1$$
$$z_2(y_1 + 2y_2 + y_3) \geq z_2$$
$$z_3(3y_1 + 5y_2 + 2y_3) \geq 2z_3$$

- Summing these up and factoring produces

$$y_1(z_1 + z_2 + 3z_3) + y_2(2z_1 + 2z_2 + 5z_3) + y_3(4z_1 + z_2 + 2z_3) \geq 3z_1 + z_2 + 2z_3 \tag{15}$$

Linear Programming - Standard Form

- Let us now repeat the whole procedure with P^* in place of P, i.e., let us find the dual program $(P^*)^*$ of P^*.
- We are now looking for $z_1, z_2, z_3 \geq 0$ to multiply inequalities (11)-(13) and obtain

$$
\begin{aligned}
z_1(y_1 + 2y_2 + 4y_3) &\geq 3z_1 \\
z_2(y_1 + 2y_2 + y_3) &\geq z_2 \\
z_3(3y_1 + 5y_2 + 2y_3) &\geq 2z_3
\end{aligned}
$$

- Summing these up and factoring produces

$$y_1(z_1 + z_2 + 3z_3) + y_2(2z_1 + 2z_2 + 5z_3) + y_3(4z_1 + z_2 + 2z_3) \geq 3z_1 + z_2 + 2z_3 \tag{15}$$

- If we choose multipliers z_1, z_2, z_3 so that

$$z_1 + z_2 + 3z_3 \leq 30 \tag{16}$$

$$2z_1 + 2z_2 + 5z_3 \leq 24 \tag{17}$$

$$4z_1 + z_2 + 2z_3 \leq 36 \tag{18}$$

Linear Programming - Standard Form

- Let us now repeat the whole procedure with P^* in place of P, i.e., let us find the dual program $(P^*)^*$ of P^*.
- We are now looking for $z_1, z_2, z_3 \geq 0$ to multiply inequalities (11)-(13) and obtain

$$z_1(y_1 + 2y_2 + 4y_3) \geq 3z_1$$
$$z_2(y_1 + 2y_2 + y_3) \geq z_2$$
$$z_3(3y_1 + 5y_2 + 2y_3) \geq 2z_3$$

- Summing these up and factoring produces

$$y_1(z_1 + z_2 + 3z_3) + y_2(2z_1 + 2z_2 + 5z_3) + y_3(4z_1 + z_2 + 2z_3) \geq 3z_1 + z_2 + 2z_3 \tag{15}$$

- If we choose multipliers z_1, z_2, z_3 so that

$$z_1 + z_2 + 3z_3 \leq 30 \tag{16}$$
$$2z_1 + 2z_2 + 5z_3 \leq 24 \tag{17}$$
$$4z_1 + z_2 + 2z_3 \leq 36 \tag{18}$$

we will have:
$$y_1(z_1 + z_2 + 3z_3) + y_2(2z_1 + 2z_2 + 5z_3) + y_3(4z_1 + z_1 + 2z_3) \leq 30y_1 + 24y_2 + 36y_3$$

Linear Programming - Standard Form

- Let us now repeat the whole procedure with P^* in place of P, i.e., let us find the dual program $(P^*)^*$ of P^*.
- We are now looking for $z_1, z_2, z_3 \geq 0$ to multiply inequalities (11)-(13) and obtain

$$
\begin{aligned}
z_1(y_1 + 2y_2 + 4y_3) &\geq 3z_1 \\
z_2(y_1 + 2y_2 + y_3) &\geq z_2 \\
z_3(3y_1 + 5y_2 + 2y_3) &\geq 2z_3
\end{aligned}
$$

- Summing these up and factoring produces

$$
y_1(z_1 + z_2 + 3z_3) + y_2(2z_1 + 2z_2 + 5z_3) + y_3(4z_1 + z_2 + 2z_3) \geq 3z_1 + z_2 + 2z_3 \tag{15}
$$

- If we choose multipliers z_1, z_2, z_3 so that

$$
z_1 + z_2 + 3z_3 \leq 30 \tag{16}
$$

$$
2z_1 + 2z_2 + 5z_3 \leq 24 \tag{17}
$$

$$
4z_1 + z_2 + 2z_3 \leq 36 \tag{18}
$$

  we will have:
  $$y_1(z_1 + z_2 + 3z_3) + y_2(2z_1 + 2z_2 + 5z_3) + y_3(4z_1 + z_1 + 2z_3) \leq 30y_1 + 24y_2 + 36y_3$$

- Combining this with (15) we get

$$
3z_1 + z_2 + 2z_3 \leq 30y_1 + 24y_2 + 36y_3
$$

Linear Programming - Standard Form

- Consequently, finding the dual program $(P^*)^*$ of P^* amounts to maximising the objective $3z_1 + z_2 + 2z_3$ subject to the constraints

$$\begin{aligned} z_1 + z_2 + 3z_3 &\leq 30 \\ 2z_1 + 2z_2 + 5z_3 &\leq 24 \\ 4z_1 + z_2 + 2z_3 &\leq 36 \end{aligned}$$

Linear Programming - Standard Form

- Consequently, finding the dual program $(P^*)^*$ of P^* amounts to maximising the objective $3z_1 + z_2 + 2z_3$ subject to the constraints

$$\begin{aligned} z_1 + z_2 + 3z_3 &\leq 30 \\ 2z_1 + 2z_2 + 5z_3 &\leq 24 \\ 4z_1 + z_2 + 2z_3 &\leq 36 \end{aligned}$$

- But note that, except for having different variables, $(P^*)^*$ is exactly our starting program P! Thus, the dual program $(P^*)^*$ for program P^* is just P itself, i.e., $(P^*)^* = P$.

Linear Programming - Standard Form

- Consequently, finding the dual program $(P^*)^*$ of P^* amounts to maximising the objective $3z_1 + z_2 + 2z_3$ subject to the constraints

$$\begin{aligned} z_1 + z_2 + 3z_3 &\leq 30 \\ 2z_1 + 2z_2 + 5z_3 &\leq 24 \\ 4z_1 + z_2 + 2z_3 &\leq 36 \end{aligned}$$

- But note that, except for having different variables, $(P^*)^*$ is exactly our starting program P! Thus, the dual program $(P^*)^*$ for program P^* is just P itself, i.e., $(P^*)^* = P$.

- So, at the first sight, looking for the multipliers y_1, y_2, y_3 did not help much, because it only reduced a maximisation problem to an equally hard minimisation problem.

Linear Programming - Standard Form

- Consequently, finding the dual program $(P^*)^*$ of P^* amounts to maximising the objective $3z_1 + z_2 + 2z_3$ subject to the constraints

$$\begin{aligned} z_1 + z_2 + 3z_3 &\leq 30 \\ 2z_1 + 2z_2 + 5z_3 &\leq 24 \\ 4z_1 + z_2 + 2z_3 &\leq 36 \end{aligned}$$

- But note that, except for having different variables, $(P^*)^*$ is exactly our starting program P! Thus, the dual program $(P^*)^*$ for program P^* is just P itself, i.e., $(P^*)^* = P$.
- So, at the first sight, looking for the multipliers y_1, y_2, y_3 did not help much, because it only reduced a maximisation problem to an equally hard minimisation problem.
- It is now useful to remember how we proved that the Ford - Fulkerson Max Flow algorithm in fact produces a maximal flow, by showing that it terminates only when we reach the capacity of a (minimal) cut...

Linear Programming - primal/dual problem forms

- The original, *primal* Linear Program P and its *dual* Linear Program can be easily described in the most general case:

$$
\begin{aligned}
P: \quad & \text{maximize} & z(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j, & \\
& \text{subject to the constraints} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i; & \quad 1 \leq i \leq m \\
& & x_1, \dots, x_n \geq 0; &
\end{aligned}
$$

$$
\begin{aligned}
P^*: \quad & \text{minimize} & z^*(\mathbf{y}) = \sum_{i=1}^{m} b_i y_i, & \\
& \text{subject to the constraints} & \sum_{i=1}^{m} a_{ij} y_i \geq c_j; & \quad 1 \leq j \leq n \\
& & y_1, \dots, y_m \geq 0, &
\end{aligned}
$$

Linear Programming - primal/dual problem forms

- The original, *primal* Linear Program P and its *dual* Linear Program can be easily described in the most general case:

$$
\begin{aligned}
P: \quad & \text{maximize} & & z(\mathbf{x}) = \sum_{j=1}^{n} c_j x_j, \\
& \text{subject to the constraints} & & \sum_{j=1}^{n} a_{ij} x_j \le b_i; \quad 1 \le i \le m \\
& & & x_1, \dots, x_n \ge 0;
\end{aligned}
$$

$$
\begin{aligned}
P^*: \quad & \text{minimize} & & z^*(\mathbf{y}) = \sum_{i=1}^{m} b_i y_i, \\
& \text{subject to the constraints} & & \sum_{i=1}^{m} a_{ij} y_i \ge c_j; \quad 1 \le j \le n \\
& & & y_1, \dots, y_m \ge 0,
\end{aligned}
$$

or, in matrix form,

$$
\begin{aligned}
P: \quad & \text{maximize} \quad z(\mathbf{x}) = \mathbf{c}^\intercal \mathbf{x}, \ \text{subject to the constraints} \quad A\mathbf{x} \le \mathbf{b} \ \text{and} \ \mathbf{x} \ge 0; \\
P^*: \quad & \text{minimize} \quad z^*(\mathbf{y}) = \mathbf{b}^\intercal \mathbf{y}, \ \text{subject to the constraints} \quad A^\intercal \mathbf{y} \ge \mathbf{c} \ \text{and} \ \mathbf{y} \ge 0.
\end{aligned}
$$

Weak Duality Theorem

- Recall that any vector $\mathbf{x}$ which satisfies the two constraints, $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$ is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\intercal \mathbf{x}$ might be.

Weak Duality Theorem

- Recall that any vector $\mathbf{x}$ which satisfies the two constraints, $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$ is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\intercal\mathbf{x}$ might be.
- **Theorem** If $x = \langle x_1 \dots x_n \rangle$ is any basic feasible solution for P and $y = \langle y_1 \dots y_m \rangle$ is any basic feasible solution for P^*, then:

$$z(x) = \sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{n} b_i y_i = z^*(y)$$

Weak Duality Theorem

- Recall that any vector $\mathbf{x}$ which satisfies the two constraints, $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$ is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\mathsf{T}\mathbf{x}$ might be.
- **Theorem** If $x = \langle x_1 \dots x_n \rangle$ is any basic feasible solution for P and $y = \langle y_1 \dots y_m \rangle$ is any basic feasible solution for P^*, then:

$$z(x) = \sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{n} b_i y_i = z^*(y)$$

Proof: Since x and y are basic feasible solutions for P and P^* respectively, we can use the constraint inequalities, first from P^* and then from P to obtain

$$z(x) = \sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \leq \sum_{i=1}^{n} b_i y_i = z^*(y)$$

Weak Duality Theorem

- Recall that any vector $\mathbf{x}$ which satisfies the two constraints, $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$ is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\intercal \mathbf{x}$ might be.
- **Theorem** If $x = \langle x_1 \dots x_n \rangle$ is any basic feasible solution for P and $y = \langle y_1 \dots y_m \rangle$ is any basic feasible solution for P^*, then:

$$z(x) = \sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{n} b_i y_i = z^*(y)$$

Proof: Since x and y are basic feasible solutions for P and P^* respectively, we can use the constraint inequalities, first from P^* and then from P to obtain

$$z(x) = \sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \leq \sum_{i=1}^{n} b_i y_i = z^*(y)$$

- Thus, the value of (the objective of P^* for) any feasible solution of P^* is an upper bound for the set of all values of (the objective of P for) all feasible solutions of P, and

Weak Duality Theorem

- Recall that any vector $\mathbf{x}$ which satisfies the two constraints, $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$ is called a *feasible solution*, regardless of what the corresponding objective value $\mathbf{c}^\intercal \mathbf{x}$ might be.
- **Theorem** If $x = \langle x_1 \ldots x_n \rangle$ is any basic feasible solution for P and $y = \langle y_1 \ldots y_m \rangle$ is any basic feasible solution for P^*, then:

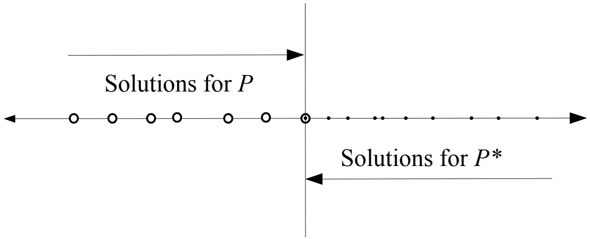
$$z(x) = \sum_{j=1}^{n} c_j x_j \leq \sum_{i=1}^{n} b_i y_i = z^*(y)$$

Proof: Since x and y are basic feasible solutions for P and P^* respectively, we can use the constraint inequalities, first from P^* and then from P to obtain

$$z(x) = \sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \leq \sum_{i=1}^{n} b_i y_i = z^*(y)$$

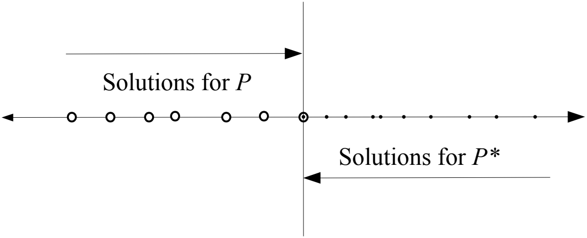
- Thus, the value of (the objective of P^* for) any feasible solution of P^* is an upper bound for the set of all values of (the objective of P for) all feasible solutions of P, and
- every feasible solution of P is a lower bound for the set of feasible solutions for P^*.

Weak Duality Theorem



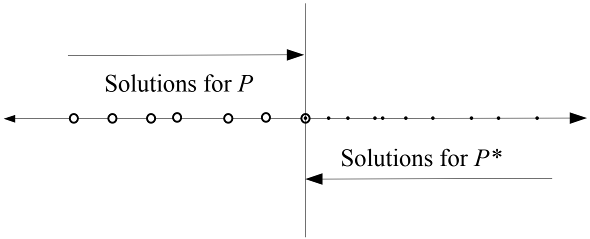
- Thus, if we find a feasible solution for P which is equal to a feasible solution to P^*, such solution must be the maximal feasible value of the objective of P and the minimal feasible value of the objective of P^*.

Weak Duality Theorem



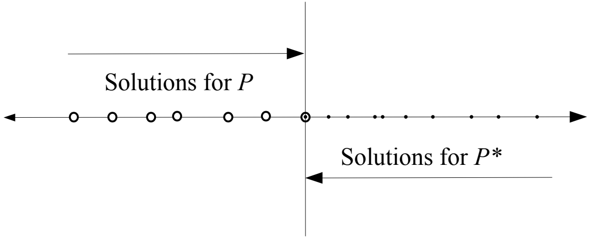
- Thus, if we find a feasible solution for P which is equal to a feasible solution to P^*, such solution must be the maximal feasible value of the objective of P and the minimal feasible value of the objective of P^*.
- If we use a search procedure to find an optimal solution for P we know when to stop: when such a value is also a feasible solution for P^*.

Weak Duality Theorem



- Thus, if we find a feasible solution for P which is equal to a feasible solution to P^*, such solution must be the maximal feasible value of the objective of P and the minimal feasible value of the objective of P^*.
- If we use a search procedure to find an optimal solution for P we know when to stop: when such a value is also a feasible solution for P^*.
- This is why the most commonly used LP solving method, the SIMPLEX method, produces optimal solution for P, because it stops at a value which is also a feasible solution to P^*.

Weak Duality Theorem



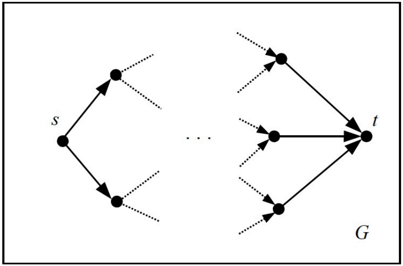
- Thus, if we find a feasible solution for P which is equal to a feasible solution to P^*, such solution must be the maximal feasible value of the objective of P and the minimal feasible value of the objective of P^*.
- If we use a search procedure to find an optimal solution for P we know when to stop: when such a value is also a feasible solution for P^*.
- This is why the most commonly used LP solving method, the SIMPLEX method, produces optimal solution for P, because it stops at a value which is also a feasible solution to P^*.
- See the Lecture Notes for the details and an example of how the SIMPLEX algorithm runs.

Examples of dual programs: Max Flow

- We would now like to formulate the Max Flow problem in a flow network as a Linear Program.

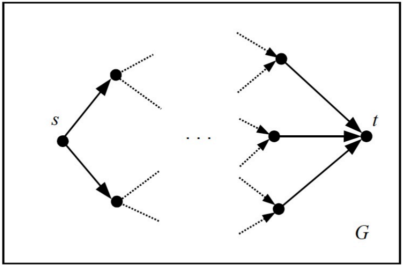
Examples of dual programs: Max Flow

- We would now like to formulate the Max Flow problem in a flow network as a Linear Program.
- Thus, assume we are given a flow network G with capacities κ_{ij} of edges $(i,j) \in G$.



Examples of dual programs: Max Flow

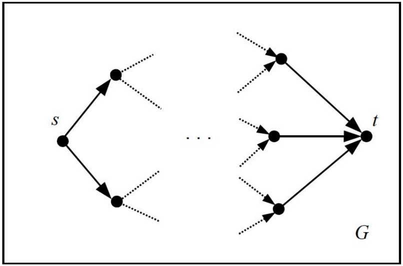
- We would now like to formulate the Max Flow problem in a flow network as a Linear Program.
- Thus, assume we are given a flow network G with capacities κ_{ij} of edges $(i, j) \in G$.



- The max flow problem seeks to maximise the total flow $\sum_{j:(s,j)\in E} f_{sj}$ through the flow network, subject to the constraints:

Examples of dual programs: Max Flow

- We would now like to formulate the Max Flow problem in a flow network as a Linear Program.
- Thus, assume we are given a flow network G with capacities κ_{ij} of edges $(i, j) \in G$.



- The max flow problem seeks to maximise the total flow $\sum_{j:(s,j)\in E} f_{sj}$ through the flow network, subject to the constraints:

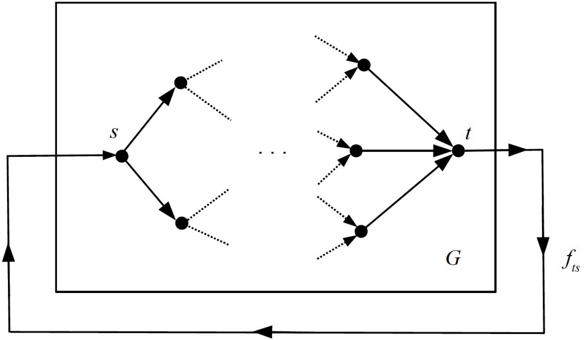
$$C^* \begin{cases} f_{ij} & \leq \kappa_{ij}; \ (i, j) \in G; \quad \text{(flow smaller than pipe's capacity)} \\ \sum_{i\,:\,(i,j)\in G} f_{ij} & = \sum_{k\,:\,(j,k)\in G} f_{jk}; \ j \in G - \{s, t\}; \quad \text{(incoming flow equals outgoing)} \\ f_{ij} & \geq 0; \ (i, j) \in G \text{ (no negative flows).} \end{cases}$$

Examples of dual programs: Max Flow

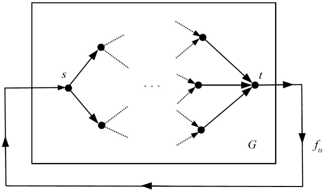
- To simplify the matters, we use a nice trick: we make the flow circular, by connecting the sink t with the source s with a pipe of infinite capacity.

Examples of dual programs: Max Flow

- To simplify the matters, we use a nice trick: we make the flow circular, by connecting the sink t with the source s with a pipe of infinite capacity.
- We now have a new graph G', $G \subset G'$, with an additional edge $(t, s) \in G'$ with capacity ∞.



Examples of dual programs: Max Flow



- We can now formulate the Max Flow problem as a Linear Program by replacing the equality in the second constraint with a single but equivalent inequality:

P: *maximize:* f_{ts}
subject to the constraints:

$$\begin{aligned} f_{ij} &\leq \kappa_{ij}; & (i,j) &\in G; \\ \sum_{i\,:\,(i,j)\in G'} f_{ij} - \sum_{k\,:\,(j,k)\in G'} f_{jk} &\leq 0; & j &\in G; \\ f_{ij} &\geq 0; & (i,j) &\in G'. \end{aligned}$$

Examples of dual programs: Max Flow

P: *maximize:* f_{ts}
subject to the constraints:

$$\begin{aligned} f_{ij} &\leq \kappa_{ij}; & (i,j) &\in G; \\ \sum_{i\,:\,(i,j)\in G'} f_{ij} - \sum_{k\,:\,(j,k)\in G'} f_{jk} &\leq 0; & j &\in G; \\ f_{ij} &\geq 0; & (i,j) &\in G'. \end{aligned}$$

- The coefficients c_{ij} of the objective of the primal P are zero for all variables f_{ij} except for f_{ts} which is equal to 1, i.e.,

$$z(\mathbf{f}) = \sum_{ij} 0 \cdot f_{ij} + 1 \cdot f_{ts} \tag{19}$$

Examples of dual programs: Max Flow

P: *maximize:* f_{ts}

subject to the constraints:

$$
\begin{aligned}
f_{ij} &\leq \kappa_{ij}; && (i,j) \in G; \\
\sum_{i\,:\,(i,j)\in G'} f_{ij} - \sum_{k\,:\,(j,k)\in G'} f_{jk} &\leq 0; && j \in G; \\
f_{ij} &\geq 0; && (i,j) \in G'.
\end{aligned}
$$

- The coefficients c_{ij} of the objective of the primal P are zero for all variables f_{ij} except for f_{ts} which is equal to 1, i.e.,

$$z(\mathbf{f}) = \sum_{ij} 0 \cdot f_{ij} + 1 \cdot f_{ts} \tag{19}$$

- To obtain the dual of P we look for coefficients d_{ij}, $(i,j) \in G$, corresponding to the first set of constraints, and coefficients p_j, $j \in G$ corresponding to the second set of constraints to use as multipliers:

$$f_{ij} d_{ij} \leq \kappa_{ij}\, d_{ij}; \qquad (i,j) \in G; \tag{20}$$

$$\sum_{i\,:\,(i,j)\in G'} f_{ij} p_j - \sum_{k\,:\,(j,k)\in G'} f_{jk} p_j \leq 0; \qquad j \in G. \tag{21}$$

Examples of dual programs: Max Flow

$$f_{ij} d_{ij} \leq \kappa_{ij}\, d_{ij}; \qquad (i,j) \in G; \qquad (20)$$

$$\sum_{i\,:\,(i,j) \in G'} f_{ij} p_j - \sum_{k\,:\,(j,k) \in G'} f_{jk} p_j \leq 0; \qquad j \in G. \qquad (21)$$

- Note the two special cases of the second inequality involving f_{ts} are

$$\sum_{i\,:\,(i,t) \in G} f_{it} p_t - f_{ts} p_t \leq 0;$$

$$f_{ts} p_s - \sum_{k\,:\,(s,k) \in G} f_{sk} p_s \leq 0.$$

Examples of dual programs: Max Flow

$$
\begin{aligned}
f_{ij} d_{ij} &\le \kappa_{ij}\, d_{ij}; && (i,j) \in G; && (20)\\
\sum_{i\,:\,(i,j)\in G'} f_{ij} p_j - \sum_{k\,:\,(j,k)\in G'} f_{jk} p_j &\le 0; && j \in G. && (21)
\end{aligned}
$$

- Note the two special cases of the second inequality involving f_{ts} are

$$
\begin{aligned}
\sum_{i\,:\,(i,t)\in G} f_{it} p_t - f_{ts} p_t &\le 0;\\
f_{ts} p_s - \sum_{k\,:\,(s,k)\in G} f_{sk} p_s &\le 0.
\end{aligned}
$$

- Summing up all inequalities in (20) and (21) and factoring out, we get

$$
\sum_{(i,j)\in G} (d_{ij} - p_i + p_j) f_{ij} + (p_s - p_t) f_{ts} \le \sum_{(i,j)\in G} \kappa_{ij} d_{ij}
$$

Examples of dual programs: Max Flow

- Summing up all inequalities in (20) and (21) and factoring out, we get

$$\sum_{(i,j)\in G}(d_{ij} - p_i + p_j)f_{ij} + (p_s - p_t)f_{ts} \le \sum_{(i,j)\in G}\kappa_{ij}d_{ij}$$

Examples of dual programs: Max Flow

- Summing up all inequalities in (20) and (21) and factoring out, we get

$$\sum_{(i,j)\in G}(d_{ij} - p_i + p_j)f_{ij} + (p_s - p_t)f_{ts} \leq \sum_{(i,j)\in G}\kappa_{ij}d_{ij}$$

- Thus, the dual objective is the right hand side of the above inequality;

Examples of dual programs: Max Flow

- Summing up all inequalities in (20) and (21) and factoring out, we get

$$\sum_{(i,j)\in G} (d_{ij} - p_i + p_j) f_{ij} + (p_s - p_t) f_{ts} \leq \sum_{(i,j)\in G} \kappa_{ij} d_{ij}$$

- Thus, the dual objective is the right hand side of the above inequality;
- As before, the dual constraints are obtained by comparing the coefficients of the left hand side with the coefficients of the objective of P, which we found to be $z(\mathbf{f}) = \sum_{ij} 0 \cdot f_{ij} + 1 \cdot f_{ts}$:

Examples of dual programs: Max Flow

- Summing up all inequalities in (20) and (21) and factoring out, we get

$$\sum_{(i,j)\in G}(d_{ij} - p_i + p_j)f_{ij} + (p_s - p_t)f_{ts} \leq \sum_{(i,j)\in G}\kappa_{ij}d_{ij}$$

- Thus, the dual objective is the right hand side of the above inequality;
- As before, the dual constraints are obtained by comparing the coefficients of the left hand side with the coefficients of the objective of P, which we found to be $z(\mathbf{f}) = \sum_{ij} 0 \cdot f_{ij} + 1 \cdot f_{ts}$:

P^* : *minimize:* $\sum_{(i,j)\in G}\kappa_{ij}d_{ij}$
subject to the constraints:

$$\begin{aligned} d_{ij} - p_i + p_j &\geq 0 \quad (i,j)\in G \\ p_s - p_t &\geq 1 \\ \\ d_{ij} &\geq 0 \quad (i,j)\in G \\ p_i &\geq 0 \quad\quad i\in G \end{aligned}$$

Examples of dual programs: Max Flow

- To sumarize:

- P: *maximize:* f_{ts}
  subject to the constraints:

$$
\begin{aligned}
& f_{ij} \leq \kappa_{ij}; && (i,j) \in G; \\
\sum_{i\,:\,(i,j)\in G'} f_{ij} - \sum_{k\,:\,(j,k)\in G'} & f_{jk} \leq 0; && j \in G; \\
& f_{ij} \geq 0; && (i,j) \in G'.
\end{aligned}
$$

- P^* : *minimize:* $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$
  subject to the constraints:

$$
\begin{array}{rll}
d_{ij} - p_i + p_j & \geq 0 & (i,j) \in G \\
p_s - p_t & \geq 1 & \\
 & & \\
d_{ij} & \geq 0 & (i,j) \in G \\
p_i & \geq 0 & i \in G
\end{array}
$$

- Note that in all constraints have coefficients ± 1;

Examples of dual programs: Max Flow

- To sumarize:
- P: *maximize:* f_{ts}
  subject to the constraints:

$$
\begin{aligned}
&f_{ij} \leq \kappa_{ij}; && (i,j) \in G;\\
\sum_{i\,:\,(i,j)\in G'} f_{ij} - \sum_{k\,:\,(j,k)\in G'} &f_{jk} \leq 0; && j \in G;\\
&f_{ij} \geq 0; && (i,j) \in G'.
\end{aligned}
$$

- P^* : *minimize:* $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$
  subject to the constraints:

$$
\begin{aligned}
d_{ij} - p_i + p_j &\geq 0 \quad (i,j) \in G\\
p_s - p_t &\geq 1\\
\\
d_{ij} &\geq 0 \quad (i,j) \in G\\
p_i &\geq 0 \quad i \in G
\end{aligned}
$$

- Note that in all constraints have coefficients ± 1;
- The SIMPLEX algorithm can be shown to return (in this particular case!) extremal values of d_{ij} and p_j which are also ± 1.

Examples of dual programs: Max Flow

- So what is the interpretation of the solution of the dual Linear Program P^*?

Examples of dual programs: Max Flow

- So what is the interpretation of the solution of the dual Linear Program P^*?
- Let $\overline{d_{ij}}$ and $\overline{p_i}$ be the values of d_{ij} and the values of p_i, respectively, for which the minimum of the objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ is achieved.

Examples of dual programs: Max Flow

- So what is the interpretation of the solution of the dual Linear Program P^*?
- Let $\overline{d_{ij}}$ and $\overline{p_i}$ be the values of d_{ij} and the values of p_i, respectively, for which the minimum of the objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ is achieved.
- Let us consider the set A of all vertices j of G for which $\overline{p}_j = 1$ and the set B of all vertices j for which $\overline{p}_j = 0$.

Examples of dual programs: Max Flow

- So what is the interpretation of the solution of the dual Linear Program P^*?
- Let $\overline{d_{ij}}$ and $\overline{p_i}$ be the values of d_{ij} and the values of p_i, respectively, for which the minimum of the objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ is achieved.
- Let us consider the set A of all vertices j of G for which $\overline{p}_j = 1$ and the set B of all vertices j for which $\overline{p}_j = 0$.
- Clearly, $A \cup B = G$ and $A \cap B = \emptyset$.

Examples of dual programs: Max Flow

- So what is the interpretation of the solution of the dual Linear Program P^*?
- Let $\overline{d_{ij}}$ and $\overline{p_i}$ be the values of d_{ij} and the values of p_i, respectively, for which the minimum of the objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ is achieved.
- Let us consider the set A of all vertices j of G for which $\overline{p}_j = 1$ and the set B of all vertices j for which $\overline{p}_j = 0$.
- Clearly, $A \cup B = G$ and $A \cap B = \emptyset$.
- The constraint $p_s - p_t \geq 1$ of P^* implies that $\overline{p}_s = 1$ and $\overline{p}_t = 0$, i.e., $s \in A$ and $t \in B$.
- Thus, A and B define a cut in the flow network.

Examples of dual programs: Max Flow

- So what is the interpretation of the solution of the dual Linear Program P^*?
- Let $\overline{d_{ij}}$ and $\overline{p_i}$ be the values of d_{ij} and the values of p_i, respectively, for which the minimum of the objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ is achieved.
- Let us consider the set A of all vertices j of G for which $\overline{p}_j = 1$ and the set B of all vertices j for which $\overline{p}_j = 0$.
- Clearly, $A \cup B = G$ and $A \cap B = \emptyset$.
- The constraint $p_s - p_t \geq 1$ of P^* implies that $\overline{p}_s = 1$ and $\overline{p}_t = 0$, i.e., $s \in A$ and $t \in B$.
- Thus, A and B define a cut in the flow network.
- Since at points $\overline{\mathbf{p}}, \overline{\mathbf{d}}$ the objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ achieves the minimal value, the constraint $d_{ij} - p_i + p_j \geq 0$ implies that $\overline{d}_{ij} = 1$ if and only if $\overline{p_i} = 1$ and $\overline{p_j} = 0$, i.e., if and only if the edge (i,j) has crossed from set A into set B.

Examples of dual programs: Max Flow

- So what is the interpretation of the solution of the dual Linear Program P^*?
- Let $\overline{d_{ij}}$ and $\overline{p_i}$ be the values of d_{ij} and the values of p_i, respectively, for which the minimum of the objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ is achieved.
- Let us consider the set A of all vertices j of G for which $\overline{p}_j = 1$ and the set B of all vertices j for which $\overline{p}_j = 0$.
- Clearly, $A \cup B = G$ and $A \cap B = \emptyset$.
- The constraint $p_s - p_t \geq 1$ of P^* implies that $\overline{p}_s = 1$ and $\overline{p}_t = 0$, i.e., $s \in A$ and $t \in B$.
- Thus, A and B define a cut in the flow network.
- Since at points $\overline{\mathbf{p}}, \overline{\mathbf{d}}$ the objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ achieves the minimal value, the constraint $d_{ij} - p_i + p_j \geq 0$ implies that $\overline{d}_{ij} = 1$ if and only if $\overline{p_i} = 1$ and $\overline{p_j} = 0$, i.e., if and only if the edge (i,j) has crossed from set A into set B.
- Thus, the minimum value of the dual objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ precisely corresponds to the capacity of the cut defined by A, B.

Examples of dual programs: Max Flow

- So what is the interpretation of the solution of the dual Linear Program P^*?
- Let $\overline{d_{ij}}$ and $\overline{p_i}$ be the values of d_{ij} and the values of p_i, respectively, for which the minimum of the objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ is achieved.
- Let us consider the set A of all vertices j of G for which $\overline{p}_j = 1$ and the set B of all vertices j for which $\overline{p}_j = 0$.
- Clearly, $A \cup B = G$ and $A \cap B = \emptyset$.
- The constraint $p_s - p_t \geq 1$ of P^* implies that $\overline{p}_s = 1$ and $\overline{p}_t = 0$, i.e., $s \in A$ and $t \in B$.
- Thus, A and B define a cut in the flow network.
- Since at points $\overline{\mathbf{p}}, \overline{\mathbf{d}}$ the objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ achieves the minimal value, the constraint $d_{ij} - p_i + p_j \geq 0$ implies that $\overline{d}_{ij} = 1$ if and only if $\overline{p_i} = 1$ and $\overline{p_j} = 0$, i.e., if and only if the edge (i, j) has crossed from set A into set B.
- Thus, the minimum value of the dual objective $\sum_{(i,j)\in G} \kappa_{ij} d_{ij}$ precisely corresponds to the capacity of the cut defined by A, B.
- Since such value of the dual problem is equal to the maximal value of the flow defined by the primal problem, we have obtained a maximal flow and a minimal cut in G!

Examples of dual programs: Max Flow

- In this particular case of the program P^*, due to the particular form of the constraints, the extremum of the objective was achieved with all the variables d_{ij} and $p_{i,j}$ taking integer values.

Examples of dual programs: Max Flow

- In this particular case of the program P^*, due to the particular form of the constraints, the extremum of the objective was achieved with all the variables d_{ij} and $p_{i,j}$ taking integer values.
- This is usually NOT the case, even if all the coefficients in the constraints are integers.

Examples of dual programs: Max Flow

- In this particular case of the program P^*, due to the particular form of the constraints, the extremum of the objective was achieved with all the variables d_{ij} and $p_{i,j}$ taking integer values.
- This is usually NOT the case, even if all the coefficients in the constraints are integers.
- If we impose an additional constraint that all the variables must take integer values (i.e., if the problem is an ILP problem), the problem is most likely intractable, i.e., it cannot be solved in polynomial time.

Examples of dual programs: Max Flow

- In this particular case of the program P^*, due to the particular form of the constraints, the extremum of the objective was achieved with all the variables d_{ij} and $p_{i,j}$ taking integer values.
- This is usually NOT the case, even if all the coefficients in the constraints are integers.
- If we impose an additional constraint that all the variables must take integer values (i.e., if the problem is an ILP problem), the problem is most likely intractable, i.e., it cannot be solved in polynomial time.
- In fact, one can show that in the general case the ILP problems are "NP hard", and what this means will be explained next week.

Examples of dual programs: Max Flow

- In this particular case of the program P^*, due to the particular form of the constraints, the extremum of the objective was achieved with all the variables d_{ij} and $p_{i,j}$ taking integer values.
- This is usually NOT the case, even if all the coefficients in the constraints are integers.
- If we impose an additional constraint that all the variables must take integer values (i.e., if the problem is an ILP problem), the problem is most likely intractable, i.e., it cannot be solved in polynomial time.
- In fact, one can show that in the general case the ILP problems are "NP hard", and what this means will be explained next week.
- Even for such problems there are quite a few solvers (mostly based on heuristic search), but of course they are not guaranteed to return true optimum value in polynomial time...