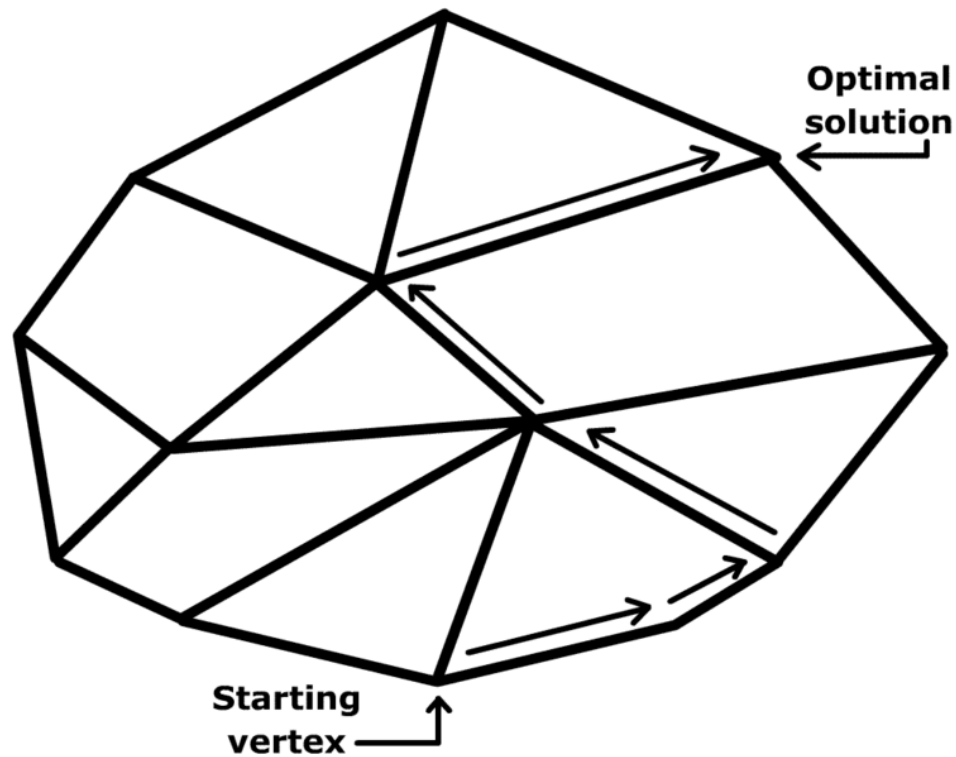


Algorithms and Analysis

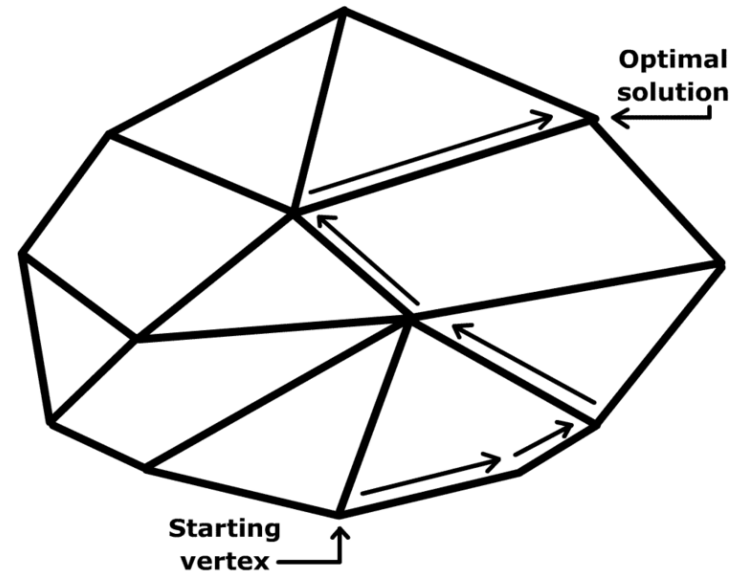
Lesson 28: *Solving Linear Programs*



linear programming, simplex methods, iterative search

Outline

1. **Recap**
2. Basic Feasible Solutions
3. Simplex Method
4. Classic LP Problems



Recap

- Linear programs are problems that can be formulated as follows

$$\min_{\mathbf{x}} \mathbf{c} \cdot \mathbf{x}$$

subject to

$$\mathbf{A}^{\leq} \mathbf{x} \leq \mathbf{b}^{\leq}, \quad \mathbf{A}^{\geq} \mathbf{x} \geq \mathbf{b}^{\geq}, \quad \mathbf{A}^{\doteq} \mathbf{x} = \mathbf{b}^{\doteq}, \quad \mathbf{x} \geq \mathbf{0}$$

- Where $\mathbf{x} = (x_1, x_2, \dots, x_n)$
- \mathbf{A}^* are matrices and we interpret the inequalities to mean

$$\forall k \quad \sum_{j=1}^n A_{kj}^{\leq} x_j \leq b_k^{\leq}$$

Recap

- Linear programs are problems that can be formulated as follows

$$\min_{\mathbf{x}} \mathbf{c} \cdot \mathbf{x}$$

subject to

$$\mathbf{A}^{\leq} \mathbf{x} \leq \mathbf{b}^{\leq}, \quad \mathbf{A}^{\geq} \mathbf{x} \geq \mathbf{b}^{\geq}, \quad \mathbf{A}^{\doteq} \mathbf{x} = \mathbf{b}^{\doteq}, \quad \mathbf{x} \geq \mathbf{0}$$

- Where $\mathbf{x} = (x_1, x_2, \dots, x_n)$
- \mathbf{A}^* are matrices and we interpret the inequalities to mean

$$\forall k \quad \sum_{j=1}^n A_{kj}^{\leq} x_j \leq b_k^{\leq}$$

Recap

- Linear programs are problems that can be formulated as follows

$$\min_{\mathbf{x}} \mathbf{c} \cdot \mathbf{x}$$

subject to

$$\mathbf{A}^{\leq} \mathbf{x} \leq \mathbf{b}^{\leq}, \quad \mathbf{A}^{\geq} \mathbf{x} \geq \mathbf{b}^{\geq}, \quad \mathbf{A}^{\doteq} \mathbf{x} = \mathbf{b}^{\doteq}, \quad \mathbf{x} \geq \mathbf{0}$$

- Where $\mathbf{x} = (x_1, x_2, \dots, x_n)$
- \mathbf{A}^* are matrices and we interpret the inequalities to mean

$$\forall k \quad \sum_{j=1}^n A_{kj}^{\leq} x_j \leq b_k^{\leq}$$

Optima and Vertices

- Because the objective function is linear ($c \cdot x$) there is a direction where the objective is always improving
- Thus, the optima cannot lie in the interior of the search space
- When we meet a constraint that limits the direction we can move, but we can still move along the constraint
- We then meet another constraint which restricts the direction we can move by two degrees of freedom
- Eventually, we will reach n constraints which defines a vertex of the feasible region and is optimal

Optima and Vertices

- Because the objective function is linear ($c \cdot x$) there is a direction where the objective is always improving
- Thus, the optima cannot lie in the interior of the search space
- When we meet a constraint that limits the direction we can move, but we can still move along the constraint
- We then meet another constraint which restricts the direction we can move by two degrees of freedom
- Eventually, we will reach n constraints which defines a vertex of the feasible region and is optimal

Optima and Vertices

- Because the objective function is linear ($c \cdot x$) there is a direction where the objective is always improving
- Thus, the optima cannot lie in the interior of the search space
- When we meet a constraint that limits the direction we can move, but we can still move along the constraint
- We then meet another constraint which restricts the direction we can move by two degrees of freedom
- Eventually, we will reach n constraints which defines a vertex of the feasible region and is optimal

Optima and Vertices

- Because the objective function is linear ($c \cdot x$) there is a direction where the objective is always improving
- Thus, the optima cannot lie in the interior of the search space
- When we meet a constraint that limits the direction we can move, but we can still move along the constraint
- We then meet another constraint which restricts the direction we can move by two degrees of freedom
- Eventually, we will reach n constraints which defines a vertex of the feasible region and is optimal

Optima and Vertices

- Because the objective function is linear ($c \cdot x$) there is a direction where the objective is always improving
- Thus, the optima cannot lie in the interior of the search space
- When we meet a constraint that limits the direction we can move, but we can still move along the constraint
- We then meet another constraint which restricts the direction we can move by two degrees of freedom
- Eventually, we will reach n constraints which defines a vertex of the feasible region and is optimal

Transforming Linear Programs

- We can always transform an inequality constraint into an equality constraint by adding slack variables
- E.g.

$$a_1 \cdot x \geq 0 \quad \Rightarrow \quad a_1 \cdot x - z_1 = 0 \quad z_1 \geq 0$$

$$a_2 \cdot x \leq 0 \quad \Rightarrow \quad a_2 \cdot x + z_2 = 0 \quad z_2 \geq 0$$

z_1 (the excess) and z_2 (the deficit) are known as slack variables

- A linear program with just equality constraints and non-negativity constraints is said to be in normal form

Transforming Linear Programs

- We can always transform an inequality constraint into an equality constraint by adding slack variables
- E.g.

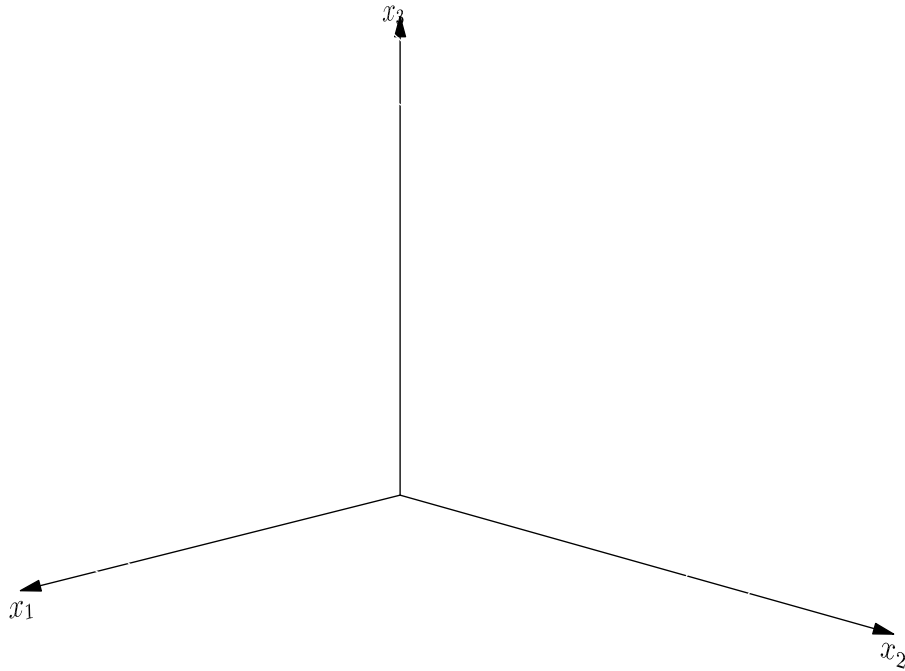
$$a_1 \cdot x \geq 0 \quad \Rightarrow \quad a_1 \cdot x - z_1 = 0 \quad z_1 \geq 0$$

$$a_2 \cdot x \leq 0 \quad \Rightarrow \quad a_2 \cdot x + z_2 = 0 \quad z_2 \geq 0$$

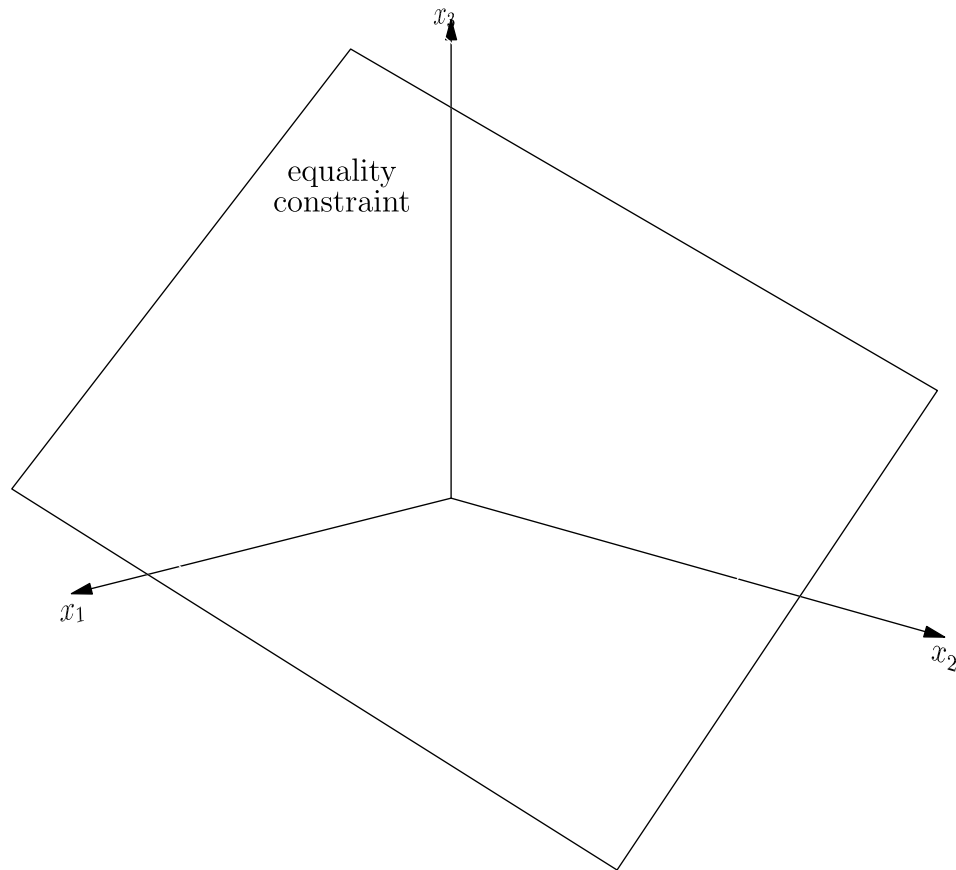
z_1 (the excess) and z_2 (the deficit) are known as slack variables

- A linear program with just equality constraints and non-negativity constraints is said to be in normal form

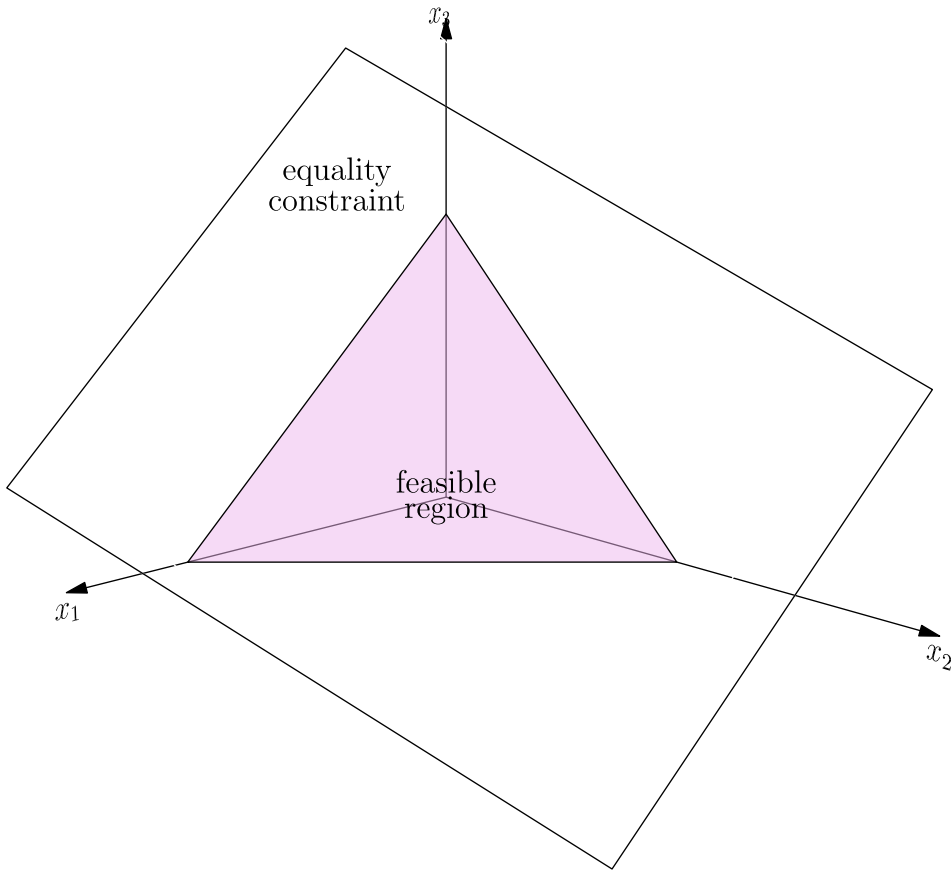
Solving Linear Programming



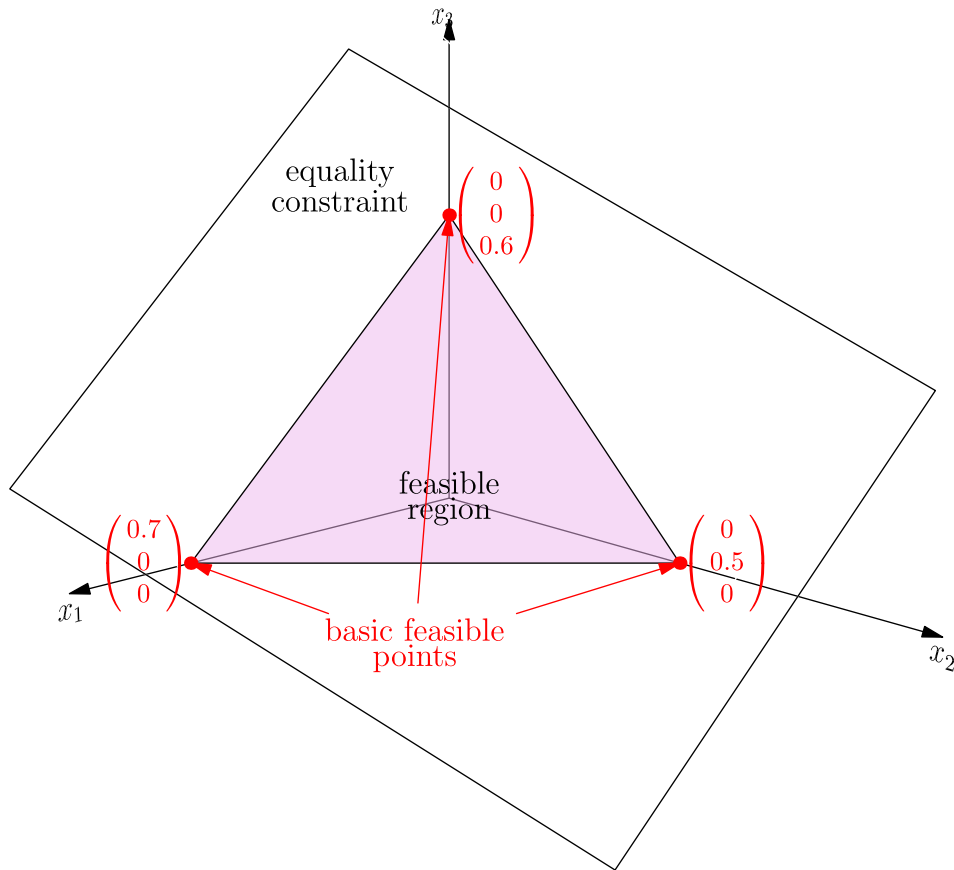
Solving Linear Programming



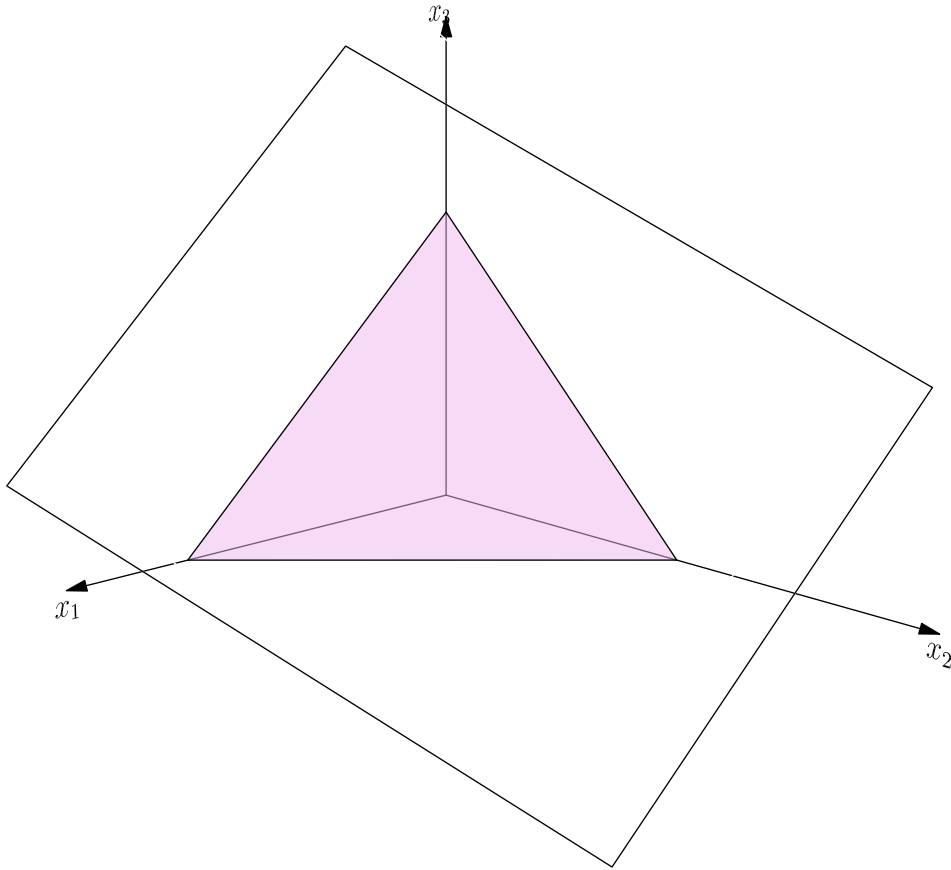
Solving Linear Programming



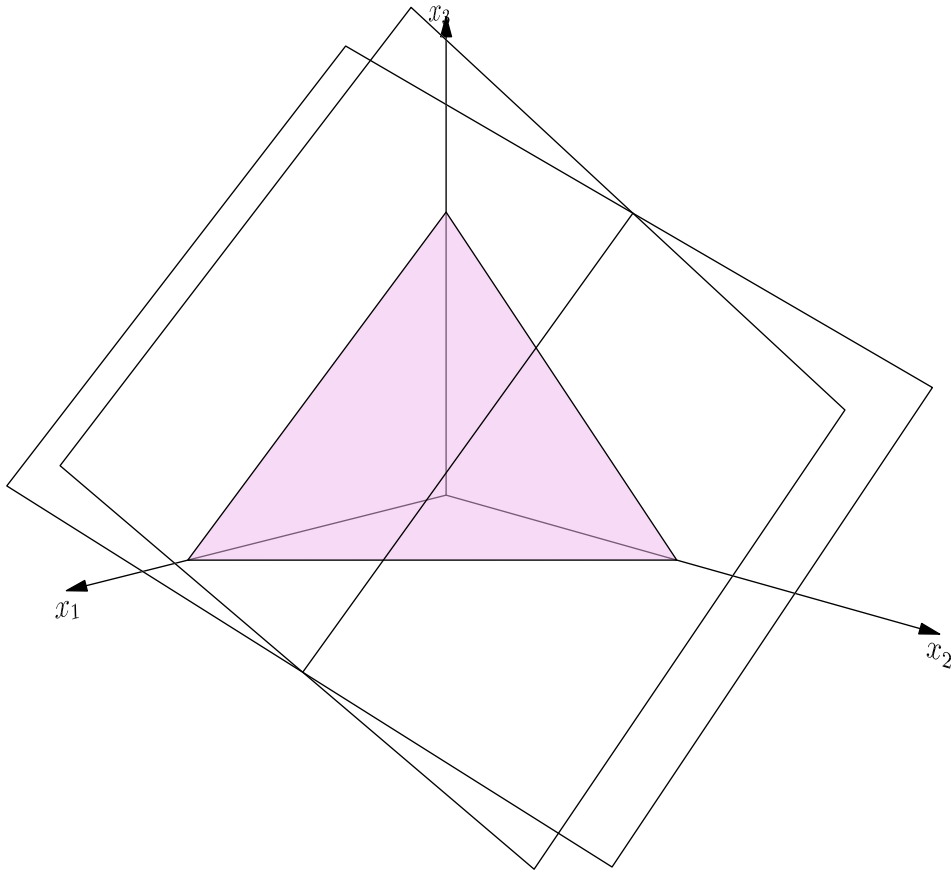
Solving Linear Programming



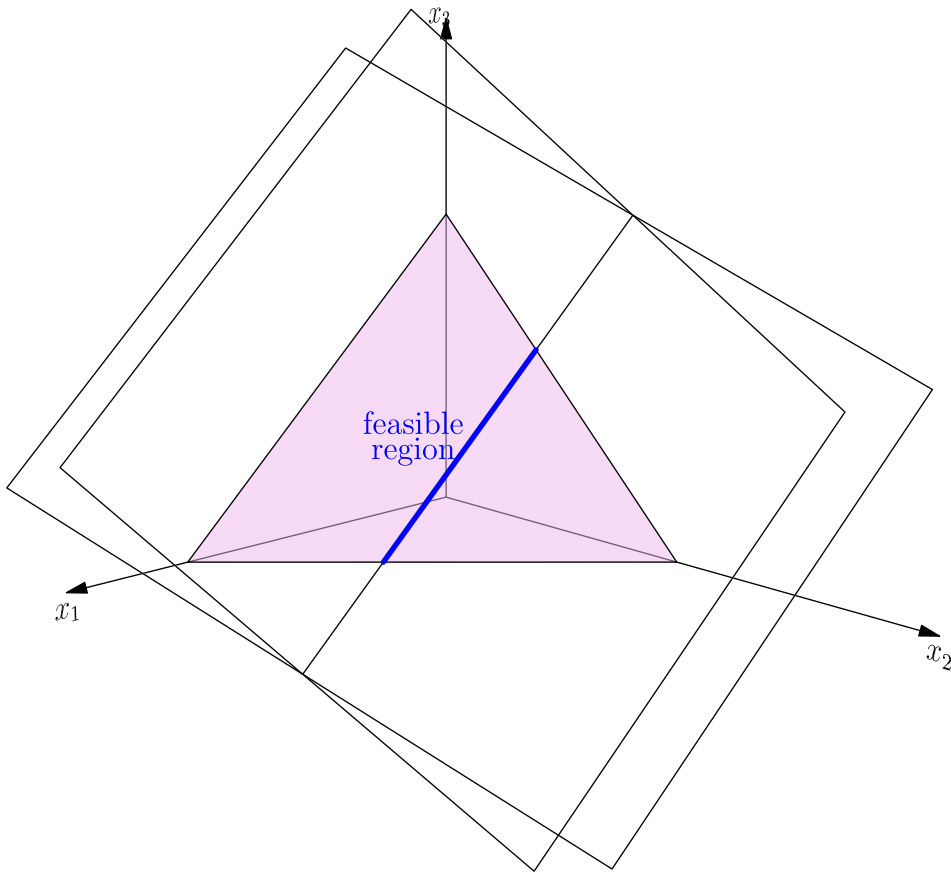
Solving Linear Programming



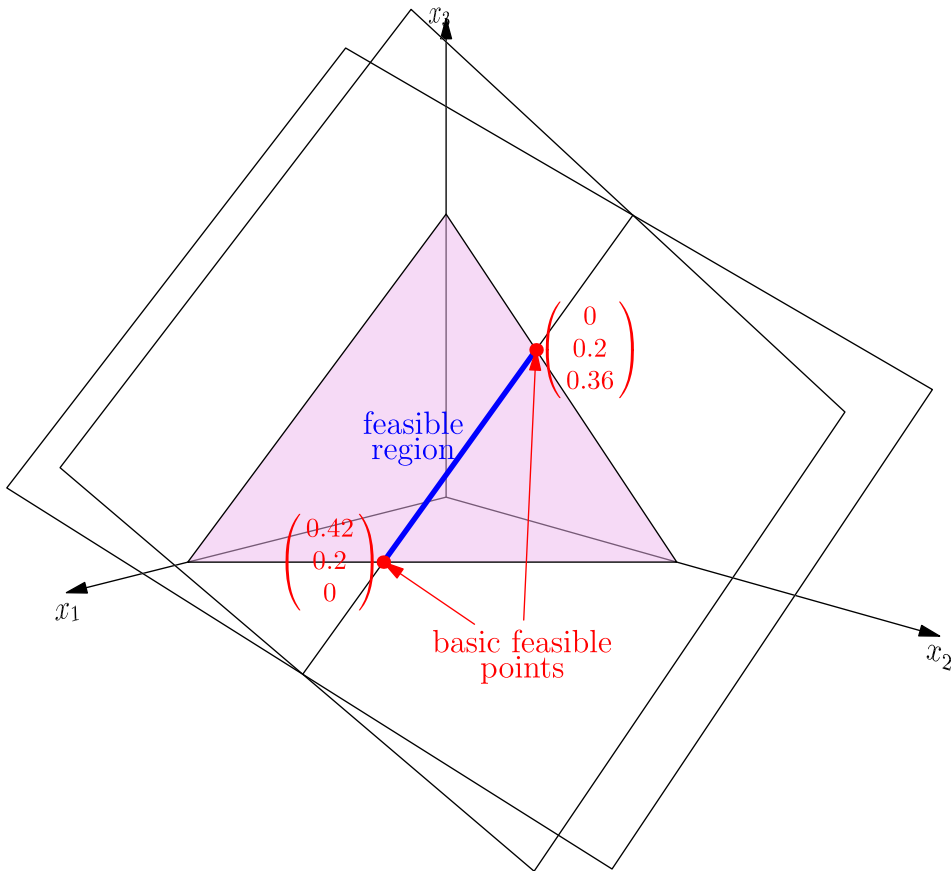
Solving Linear Programming



Solving Linear Programming

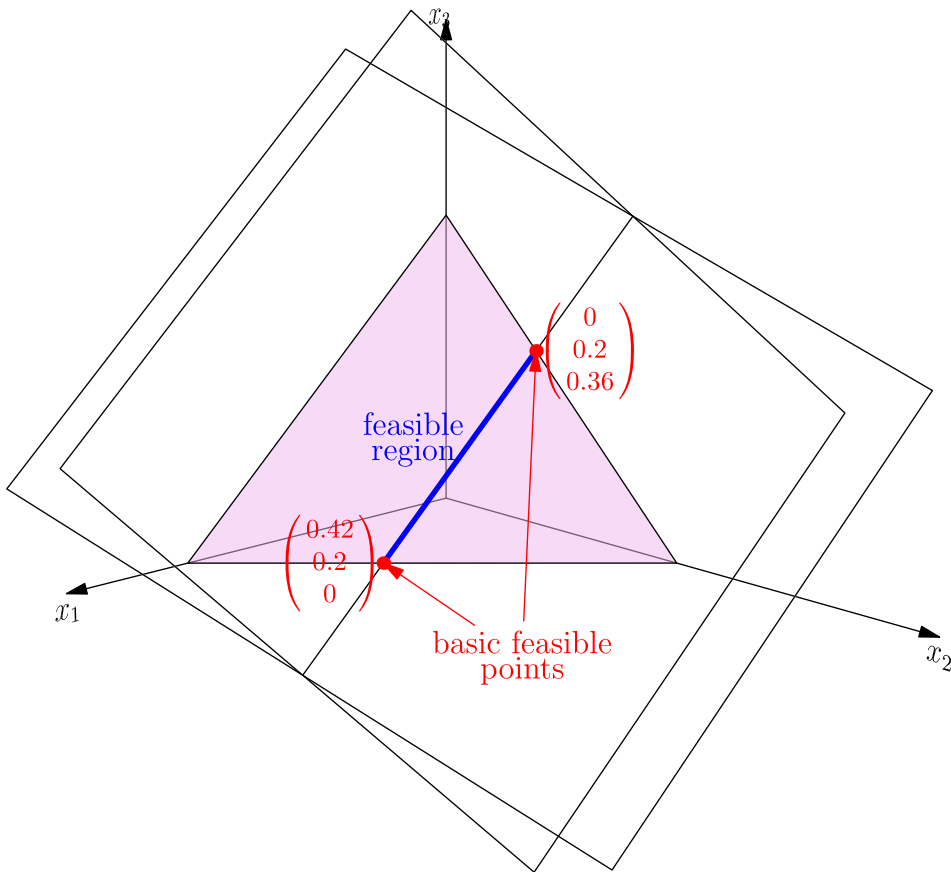


Solving Linear Programming

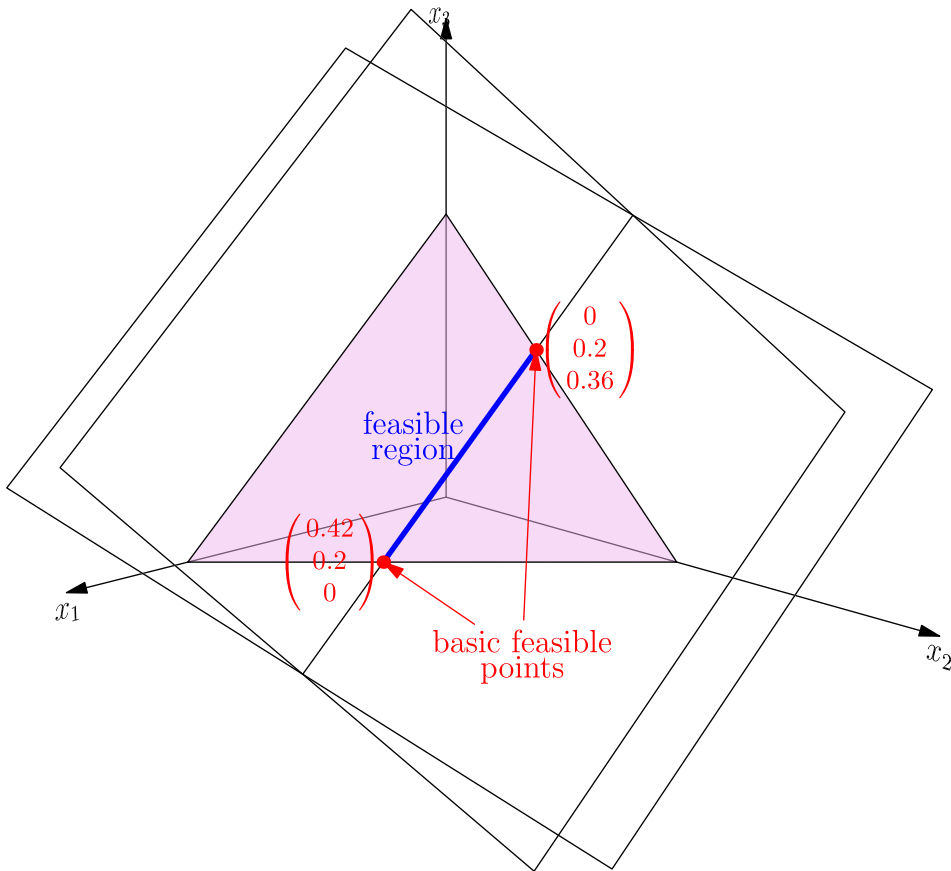


Solving Linear Programming

- The basic feasible points for LP problems with n variables and m constraints have at least $n - m$ zero variables

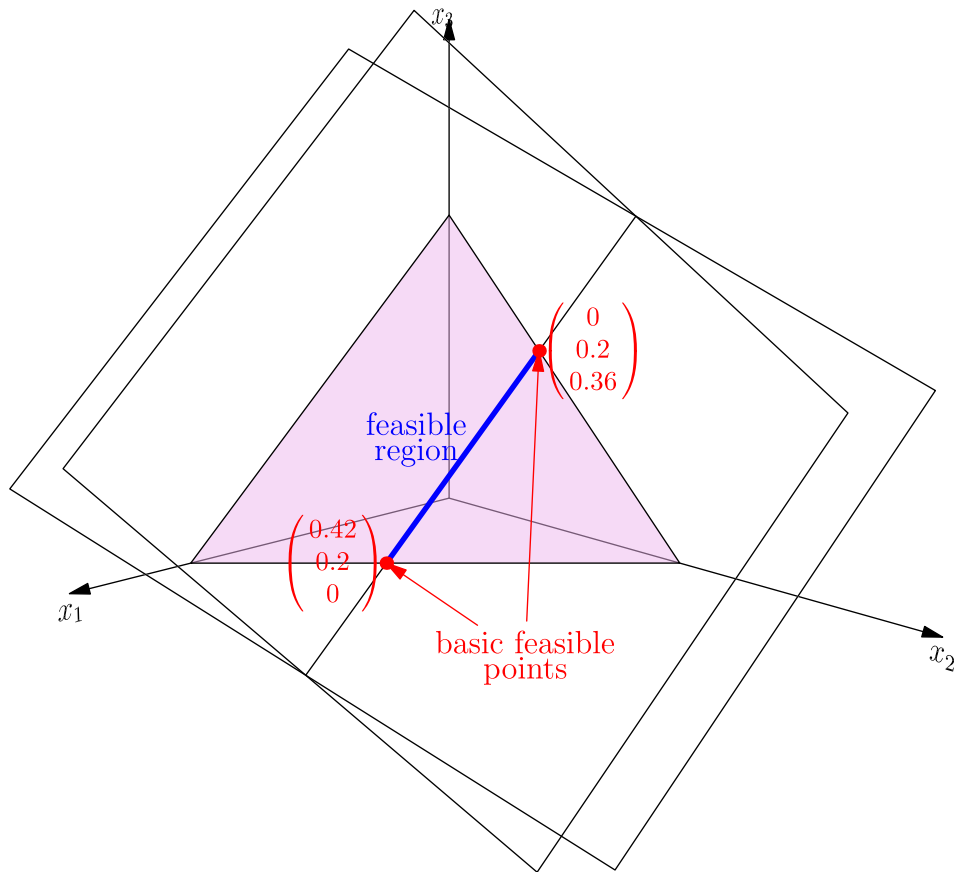


Solving Linear Programming



- The basic feasible points for LP problems with n variables and m constraints have at least $n - m$ zero variables
- Typical number of basic feasible solutions is $\binom{n}{m} \geq \left(\frac{n}{m}\right)^m$

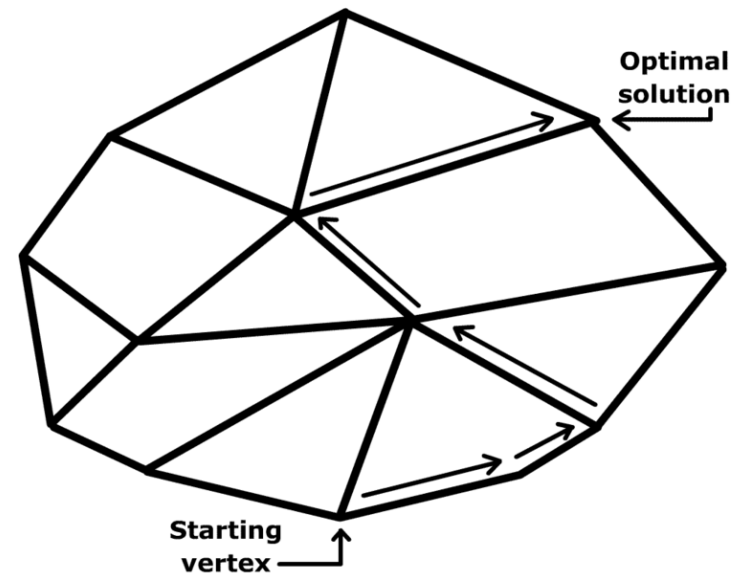
Solving Linear Programming



- The basic feasible points for LP problems with n variables and m constraints have at least $n - m$ zero variables
- Typical number of basic feasible solutions is $\binom{n}{m} \geq \left(\frac{n}{m}\right)^m$
- Simplex algorithm organises iterative search for global solutions

Outline

1. Recap
2. **Basic Feasible Solutions**
3. Simplex Method
4. Classic LP Problems



Basic Feasible Solution

- *A basic feasible solution or basic feasible point is a solution that lies at a vertex of the feasible space*
- To solve a linear program we will start at a basic feasible point and move to the neighbour which best improves the objective function
- When we cannot find a better solution we are at the optimal solution
- This is an example of an iterative improvement algorithm which gives an optimal solution

Basic Feasible Solution

- A *basic feasible solution* or *basic feasible point* is a solution that lies at a vertex of the feasible space
- To solve a linear program we will start at a basic feasible point and move to the neighbour which best improves the objective function
- When we cannot find a better solution we are at the optimal solution
- This is an example of an iterative improvement algorithm which gives an optimal solution

Basic Feasible Solution

- A *basic feasible solution* or *basic feasible point* is a solution that lies at a vertex of the feasible space
- To solve a linear program we will start at a basic feasible point and move to the neighbour which best improves the objective function
- When we cannot find a better solution we are at the optimal solution
- This is an example of an iterative improvement algorithm which gives an optimal solution

Basic Feasible Solution

- A *basic feasible solution* or *basic feasible point* is a solution that lies at a vertex of the feasible space
- To solve a linear program we will start at a basic feasible point and move to the neighbour which best improves the objective function
- When we cannot find a better solution we are at the optimal solution
- This is an example of an iterative improvement algorithm which gives an optimal solution

Constraints

- There are two types of constraints
 1. n non-negativity constraints $x_i \geq 0$
 2. m additional constraints, which we can take to be equalities
$$\mathbf{A}x = b$$
- Note that some of the variables might be slack variables
- We consider the case when there are more variables than additional constraints, i.e. $n > m$
- This is usually be the case, but. . .
- If this isn't true it turns out you can consider an equivalent problem (dual problem) where you have a variable for each constraint and a constraint for each variable

Constraints

- There are two types of constraints
 1. n non-negativity constraints $x_i \geq 0$
 2. m additional constraints, which we can take to be equalities
$$\mathbf{A}x = b$$
- Note that some of the variables might be slack variables
- We consider the case when there are more variables than additional constraints, i.e. $n > m$
- This is usually be the case, but. . .
- If this isn't true it turns out you can consider an equivalent problem (dual problem) where you have a variable for each constraint and a constraint for each variable

Constraints

- There are two types of constraints
 1. n non-negativity constraints $x_i \geq 0$
 2. m additional constraints, which we can take to be equalities
$$\mathbf{A}x = b$$
- Note that some of the variables might be slack variables
- We consider the case when there are more variables than additional constraints, i.e. $n > m$
- This is usually be the case, but. . .
- If this isn't true it turns out you can consider an equivalent problem (dual problem) where you have a variable for each constraint and a constraint for each variable

Constraints

- There are two types of constraints
 1. n non-negativity constraints $x_i \geq 0$
 2. m additional constraints, which we can take to be equalities
$$\mathbf{A}x = b$$
- Note that some of the variables might be slack variables
- We consider the case when there are more variables than additional constraints, i.e. $n > m$
- This is usually be the case, but. . .
- If this isn't true it turns out you can consider an equivalent problem (dual problem) where you have a variable for each constraint and a constraint for each variable

Constraints

- There are two types of constraints
 1. n non-negativity constraints $x_i \geq 0$
 2. m additional constraints, which we can take to be equalities
$$\mathbf{A}x = b$$
- Note that some of the variables might be slack variables
- We consider the case when there are more variables than additional constraints, i.e. $n > m$
- This is usually be the case, but. . .
- If this isn't true it turns out you can consider an equivalent problem (dual problem) where you have a variable for each constraint and a constraint for each variable

Basic Variable

- In total we have n equality and m non-negativity constraints
- n constraints must be satisfied to be at a vertex of feasible region
- So at least $n - m$ of the non-negativity constraints are satisfied (i.e. $x_i = 0$)
- The $n - m$ variables that are zero are said to be **non-basic variables**
- The other m variables are said to be **basic variables**

Basic Variable

- In total we have n equality and m non-negativity constraints
- n constraints must be satisfied to be at a vertex of feasible region
- So at least $n - m$ of the non-negativity constraints are satisfied (i.e. $x_i = 0$)
- The $n - m$ variables that are zero are said to be **non-basic variables**
- The other m variables are said to be **basic variables**

Basic Variable

- In total we have n equality and m non-negativity constraints
- n constraints must be satisfied to be at a vertex of feasible region
- So at least $n - m$ of the non-negativity constraints are satisfied (i.e. $x_i = 0$)
- The $n - m$ variables that are zero are said to be **non-basic variables**
- The other m variables are said to be **basic variables**

Basic Variable

- In total we have n equality and m non-negativity constraints
- n constraints must be satisfied to be at a vertex of feasible region
- So at least $n - m$ of the non-negativity constraints are satisfied (i.e. $x_i = 0$)
- The $n - m$ variables that are zero are said to be **non-basic variables**
- The other m variables are said to be **basic variables**

Basic Variable

- In total we have n equality and m non-negativity constraints
- n constraints must be satisfied to be at a vertex of feasible region
- So at least $n - m$ of the non-negativity constraints are satisfied (i.e. $x_i = 0$)
- The $n - m$ variables that are zero are said to be **non-basic variables**
- The other m variables are said to be **basic variables**

Initial Basic Feasible Solution

- One of the tricky bits of tackling a linear program is to find an initial feasible solution
- We do this in **phase one** of the simplex program
- To do this for each additional constraint we add a new **auxiliary variable** ξ_k , e.g.

$$\forall k \in \{1, 2, \dots, m\} \quad \xi_k + \sum_i A_{ki} x_i = b_k \geq 0$$

- We then can find a basic feasible solution by setting $x_i = 0$ so

$$\xi_k = b_k \quad \forall k \in \{1, 2, \dots, m\}$$

Initial Basic Feasible Solution

- One of the tricky bits of tackling a linear program is to find an initial feasible solution
- We do this in **phase one** of the simplex program
- To do this for each additional constraint we add a new **auxiliary variable** ξ_k , e.g.

$$\forall k \in \{1, 2, \dots, m\} \quad \xi_k + \sum_i A_{ki} x_i = b_k \geq 0$$

- We then can find a basic feasible solution by setting $x_i = 0$ so

$$\xi_k = b_k \quad \forall k \in \{1, 2, \dots, m\}$$

Initial Basic Feasible Solution

- One of the tricky bits of tackling a linear program is to find an initial feasible solution
- We do this in **phase one** of the simplex program
- To do this for each additional constraint we add a new **auxiliary variable** ξ_k , e.g.

$$\forall k \in \{1, 2, \dots, m\} \quad \xi_k + \sum_i A_{ki} x_i = b_k \geq 0$$

- We then can find a basic feasible solution by setting $x_i = 0$ so

$$\xi_k = b_k \quad \forall k \in \{1, 2, \dots, m\}$$

Initial Basic Feasible Solution

- One of the tricky bits of tackling a linear program is to find an initial feasible solution
- We do this in **phase one** of the simplex program
- To do this for each additional constraint we add a new **auxiliary variable** ξ_k , e.g.

$$\forall k \in \{1, 2, \dots, m\} \quad \xi_k + \sum_i A_{ki} x_i = b_k \geq 0$$

- We then can find a basic feasible solution by setting $x_i = 0$ so

$$\xi_k = b_k \quad \forall k \in \{1, 2, \dots, m\}$$

Eliminating Auxiliary Variables

- In phase one we run a simplex algorithm with an auxiliary cost function

$$\min f_{\text{aux}}(\mathbf{x}, \boldsymbol{\xi}) = \sum_{k=1}^m \xi_k$$

- This should find a solution where all the $\xi_k = 0$
- If no solution exists it means there is no feasible solution and we're finished
- If there is a solution then we can eliminate the auxiliary variables and we have a feasible solution

Eliminating Auxiliary Variables

- In phase one we run a simplex algorithm with an auxiliary cost function

$$\min f_{\text{aux}}(\mathbf{x}, \boldsymbol{\xi}) = \sum_{k=1}^m \xi_k$$

- This should find a solution where all the $\xi_k = 0$
- If no solution exists it means there is no feasible solution and we're finished
- If there is a solution then we can eliminate the auxiliary variables and we have a feasible solution

Eliminating Auxiliary Variables

- In phase one we run a simplex algorithm with an auxiliary cost function

$$\min f_{\text{aux}}(\mathbf{x}, \boldsymbol{\xi}) = \sum_{k=1}^m \xi_k$$

- This should find a solution where all the $\xi_k = 0$
- If no solution exists it means there is no feasible solution and we're finished
- If there is a solution then we can eliminate the auxiliary variables and we have a feasible solution

Eliminating Auxiliary Variables

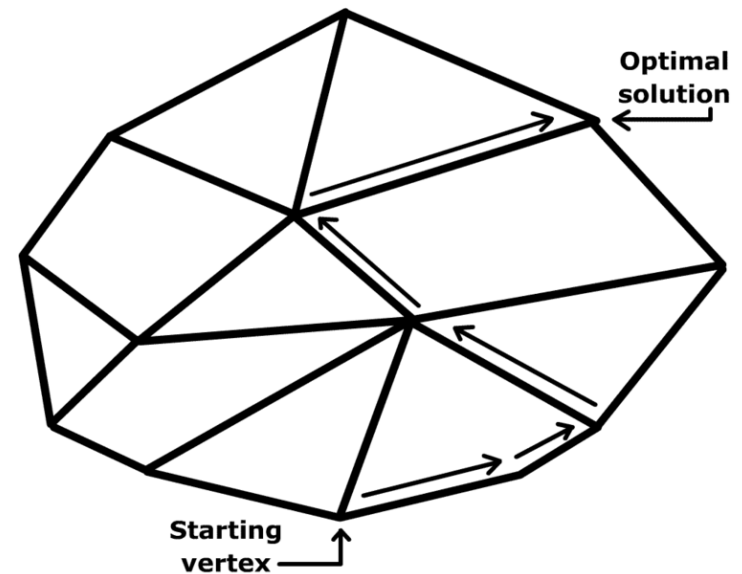
- In phase one we run a simplex algorithm with an auxiliary cost function

$$\min f_{\text{aux}}(\mathbf{x}, \boldsymbol{\xi}) = \sum_{k=1}^m \xi_k$$

- This should find a solution where all the $\xi_k = 0$
- If no solution exists it means there is no feasible solution and we're finished
- If there is a solution then we can eliminate the auxiliary variables and we have a feasible solution

Outline

1. Recap
2. Basic Feasible Solutions
3. **Simplex Method**
4. Classic LP Problems



Phase Two

- In phase two we now have an initial basic feasible solution (with $n - m$ zero variables)
- We then run the simplex algorithm on the original objective function $f(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x}$
- That is we move to a neighbouring vertex which gives the best increase in the objective function
- To help organise this search we write the objective function and constraints in a **restricted normal form** and then build a **tableau** showing the *basic variables* and the *non-basic variables*

Phase Two

- In phase two we now have an initial basic feasible solution (with $n - m$ zero variables)
- We then run the simplex algorithm on the original objective function $f(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x}$
- That is we move to a neighbouring vertex which gives the best increase in the objective function
- To help organise this search we write the objective function and constraints in a **restricted normal form** and then build a **tableau** showing the *basic variables* and the *non-basic variables*

Phase Two

- In phase two we now have an initial basic feasible solution (with $n - m$ zero variables)
- We then run the simplex algorithm on the original objective function $f(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x}$
- That is we move to a neighbouring vertex which gives the best increase in the objective function
- To help organise this search we write the objective function and constraints in a **restricted normal form** and then build a **tableau** showing the *basic variables* and the *non-basic variables*

Phase Two

- In phase two we now have an initial basic feasible solution (with $n - m$ zero variables)
- We then run the simplex algorithm on the original objective function $f(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x}$
- That is we move to a neighbouring vertex which gives the best increase in the objective function
- To help organise this search we write the objective function and constraints in a **restricted normal form** and then build a **tableau** showing the *basic variables* and the *non-basic variables*

Restricted Normal Form

- To perform the moves between vertices it helps to represent the problem in a **restricted normal form**
- Starting from a basic feasible point we have a constraint for each basic (non-zero) variable
- We write the constraints as an equality between basic and non-basic (zero valued) variables
- Similarly we write the objective function in terms of non-basic variables
- This is always possible as we can use the constraints to eliminate the basic variables

Restricted Normal Form

- To perform the moves between vertices it helps to represent the problem in a **restricted normal form**
- Starting from a basic feasible point we have a constraint for each basic (non-zero) variable
- We write the constraints as an equality between basic and non-basic (zero valued) variables
- Similarly we write the objective function in terms of non-basic variables
- This is always possible as we can use the constraints to eliminate the basic variables

Restricted Normal Form

- To perform the moves between vertices it helps to represent the problem in a **restricted normal form**
- Starting from a basic feasible point we have a constraint for each basic (non-zero) variable
- We write the constraints as an equality between basic and non-basic (zero valued) variables
- Similarly we write the objective function in terms of non-basic variables
- This is always possible as we can use the constraints to eliminate the basic variables

Restricted Normal Form

- To perform the moves between vertices it helps to represent the problem in a **restricted normal form**
- Starting from a basic feasible point we have a constraint for each basic (non-zero) variable
- We write the constraints as an equality between basic and non-basic (zero valued) variables
- Similarly we write the objective function in terms of non-basic variables
- This is always possible as we can use the constraints to eliminate the basic variables

Restricted Normal Form

- To perform the moves between vertices it helps to represent the problem in a **restricted normal form**
- Starting from a basic feasible point we have a constraint for each basic (non-zero) variable
- We write the constraints as an equality between basic and non-basic (zero valued) variables
- Similarly we write the objective function in terms of non-basic variables
- This is always possible as we can use the constraints to eliminate the basic variables

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

$$x_1 = 0.2, \quad x_3 = 0, \quad f(\mathbf{x}) = -7$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

$$x_4 = 0.12, \quad x_3 = 0, \quad f(\mathbf{x}) = -7.1$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

$$x_5 = 0.12, \quad x_3 = 0, \quad f(\mathbf{x}) = -7.6$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

$$x_7 = 0.6, \quad x_2 = 0, \quad f(\mathbf{x}) = -3.2$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

where

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9 \geq 0$$

$$x_3 = 1 - 5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9 \geq 0$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_7 = x_8 = x_9 = 0$$

Best pivot: $x_7 = 0.6$, $x_2 = 0$, $f(\mathbf{x}) = -3.2$ swap x_2 with x_7

		x_1	x_4	x_5	x_7	x_8	x_9
$f(\mathbf{x})$	-8	5	7	3	8	-6	-3
x_2	3	6	6	-5	-5	5	-5
x_3	1	-5	-8	-8	4	-8	-5
x_6	8	-8	-7	-2	-5	-3	4

Tableau

Swap x_2 with x_7

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9$$

$$\Rightarrow x_7 = 0.6 + 1.2x_1 + 1.2x_4 - x_5 - 0.2x_2 + x_8 - x_9$$

Tableau

Swap x_2 with x_7

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9$$

$$\Rightarrow x_7 = 0.6 + 1.2x_1 + 1.2x_4 - x_5 - 0.2x_2 + x_8 - x_9$$

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

$$\Rightarrow \max_{\mathbf{x}} f(\mathbf{x}) = -3.2 + 15x_1 + 17x_4 - 5x_5 - 1.6x_2 + 2x_8 - 11x_9$$

Tableau

Swap x_2 with x_7

$$x_2 = 3 + 6x_1 + 6x_4 - 5x_5 - 5x_7 + 5x_8 - 5x_9$$

$$\Rightarrow x_7 = 0.6 + 1.2x_1 + 1.2x_4 - x_5 - 0.2x_2 + x_8 - x_9$$

$$\max_{\mathbf{x}} f(\mathbf{x}) = -8 + 5x_1 + 7x_4 + 3x_5 + 8x_7 - 6x_8 - 3x_9$$

$$\Rightarrow \max_{\mathbf{x}} f(\mathbf{x}) = -3.2 + 15x_1 + 17x_4 - 5x_5 - 1.6x_2 + 2x_8 - 11x_9$$

$$x_3 = -5x_1 - 8x_4 - 8x_5 + 4x_7 - 8x_8 - 5x_9$$

$$\Rightarrow x_3 = 3.4 - 0.2x_1 - 3.2x_4 - 12x_5 - 0.8x_2 - 4x_8 - 9x_9$$

$$x_6 = 8 - 8x_1 - 7x_4 - 2x_5 - 5x_7 - 3x_8 + 4x_9$$

$$\Rightarrow x_6 = 5 - 14x_1 - 13x_4 + 3x_5 + x_2 - 8x_8 + 9x_9$$

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -3.2 + 15x_1 + 17x_4 - 5x_5 - 1.6x_2 + 2x_8 - 11x_9$$

where

$$x_7 = 0.6 + 1.2x_1 + 1.2x_4 - x_5 - 0.2x_2 + x_8 - x_9 \geq 0$$

$$x_3 = 3.4 - 0.2x_1 - 3.2x_4 - 12x_5 - 0.8x_2 - 4x_8 - 9x_9 \geq 0$$

$$x_6 = 5 - 14x_1 - 13x_4 + 3x_5 + x_2 - 8x_8 + 9x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_2 = x_8 = x_9 = 0$$

		x_1	x_4	x_5	x_2	x_8	x_9
$f(\mathbf{x})$	-3.2	15	17	-5	-1.6	2	-11
x_7	0.6	1.2	1.2	-1	-0.2	1	-1
x_3	3.4	-0.2	-3.2	-12	-0.8	-4	-9
x_6	5	-14	-13	3	1	-8	9

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = -3.2 + 15x_1 + 17x_4 - 5x_5 - 1.6x_2 + 2x_8 - 11x_9$$

where

$$x_7 = 0.6 + 1.2x_1 + 1.2x_4 - x_5 - 0.2x_2 + x_8 - x_9 \geq 0$$

$$x_3 = 3.4 - 0.2x_1 - 3.2x_4 - 12x_5 - 0.8x_2 - 4x_8 - 9x_9 \geq 0$$

$$x_6 = 5 - 14x_1 - 13x_4 + 3x_5 + x_2 - 8x_8 + 9x_9 \geq 0$$

$$x_1 = x_4 = x_5 = x_2 = x_8 = x_9 = 0$$

Best pivot: $x_4 = 0.38$, $x_6 = 0$, $f(\mathbf{x}) = 3.2$ swap x_6 with x_4

		x_1	x_4	x_5	x_2	x_8	x_9
$f(\mathbf{x})$	-3.2	15	17	-5	-1.6	2	-11
x_7	0.6	1.2	1.2	-1	-0.2	1	-1
x_3	3.4	-0.2	-3.2	-12	-0.8	-4	-9
x_6	5	-14	-13	3	1	-8	9

Tableau

Swap x_6 with x_4

$$x_6 = 5 - 14x_1 - 13x_4 + 3x_5 + x_2 - 8x_8 + 9x_9$$

$$\Rightarrow x_4 = 0.38 - 1.1x_1 - 0.077x_6 + 0.23x_5 + 0.077x_2 - 0.62x_8 + 0.69x_9$$

Tableau

Swap x_6 with x_4

$$x_6 = 5 - 14x_1 - 13x_4 + 3x_5 + x_2 - 8x_8 + 9x_9$$

$$\Rightarrow x_4 = 0.38 - 1.1x_1 - 0.077x_6 + 0.23x_5 + 0.077x_2 - 0.62x_8 + 0.69x_9$$

$$\max_{\mathbf{x}} f(\mathbf{x}) = -3.2 + 15x_1 + 17x_4 - 5x_5 - 1.6x_2 + 2x_8 - 11x_9$$

$$\Rightarrow \max_{\mathbf{x}} f(\mathbf{x}) = 3.2 - 3.3x_1 - 1.3x_6 - 1.2x_5 - 0.32x_2 - 8.2x_8 + 0.49x_9$$

Tableau

Swap x_6 with x_4

$$x_6 = 5 - 14x_1 - 13x_4 + 3x_5 + x_2 - 8x_8 + 9x_9$$

$$\Rightarrow x_4 = 0.38 - 1.1x_1 - 0.077x_6 + 0.23x_5 + 0.077x_2 - 0.62x_8 + 0.69x_9$$

$$\max_{\mathbf{x}} f(\mathbf{x}) = -3.2 + 15x_1 + 17x_4 - 5x_5 - 1.6x_2 + 2x_8 - 11x_9$$

$$\Rightarrow \max_{\mathbf{x}} f(\mathbf{x}) = 3.2 - 3.3x_1 - 1.3x_6 - 1.2x_5 - 0.32x_2 - 8.2x_8 + 0.49x_9$$

$$x_7 = 0.6 + 1.2x_1 + 1.2x_4 - x_5 - 0.2x_2 + x_8 - x_9$$

$$\Rightarrow x_7 = 1.1 - 0.092x_1 - 0.092x_6 - 0.72x_5 - 0.11x_2 + 0.26x_8 - 0.17x_9$$

$$x_3 = 3.4 - 0.2x_1 - 3.2x_4 - 12x_5 - 0.8x_2 - 4x_8 - 9x_9$$

$$\Rightarrow x_3 = 2.2 + 3.2x_1 + 0.25x_6 - 13x_5 - 1x_2 - 2x_8 - 11x_9$$

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = 3.2 - 3.3x_1 - 1.3x_6 - 1.2x_5 - 0.32x_2 - 8.2x_8 + 0.49x_9$$

where

$$x_7 = 1.1 - 0.092x_1 - 0.092x_6 - 0.72x_5 - 0.11x_2 + 0.26x_8 - 0.17x_9 \geq 0$$

$$x_3 = 2.2 + 3.2x_1 + 0.25x_6 - 13x_5 - 1x_2 - 2x_8 - 11x_9 \geq 0$$

$$x_4 = 0.38 - 1.1x_1 - 0.077x_6 + 0.23x_5 + 0.077x_2 - 0.62x_8 + 0.69x_9 \geq 0$$

$$x_1 = x_6 = x_5 = x_2 = x_8 = x_9 = 0$$

		x_1	x_6	x_5	x_2	x_8	x_9
$f(\mathbf{x})$	3.2	-3.3	-1.3	-1.2	-0.32	-8.2	0.49
x_7	1.1	-0.092	-0.092	-0.72	-0.11	0.26	-0.17
x_3	2.2	3.2	0.25	-13	-1	-2	-11
x_4	0.38	-1.1	-0.077	0.23	0.077	-0.62	0.69

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = 3.2 - 3.3x_1 - 1.3x_6 - 1.2x_5 - 0.32x_2 - 8.2x_8 + 0.49x_9$$

where

$$x_7 = 1.1 - 0.092x_1 - 0.092x_6 - 0.72x_5 - 0.11x_2 + 0.26x_8 - 0.17x_9 \geq 0$$

$$x_3 = 2.2 + 3.2x_1 + 0.25x_6 - 13x_5 - 1x_2 - 2x_8 - 11x_9 \geq 0$$

$$x_4 = 0.38 - 1.1x_1 - 0.077x_6 + 0.23x_5 + 0.077x_2 - 0.62x_8 + 0.69x_9 \geq 0$$

$$x_1 = x_6 = x_5 = x_2 = x_8 = x_9 = 0$$

Best pivot: $x_9 = 0.19$, $x_3 = 0$, $f(\mathbf{x}) = 3.3$ swap x_3 with x_9

		x_1	x_6	x_5	x_2	x_8	x_9
$f(\mathbf{x})$	3.2	-3.3	-1.3	-1.2	-0.32	-8.2	0.49
x_7	1.1	-0.092	-0.092	-0.72	-0.11	0.26	-0.17
x_3	2.2	3.2	0.25	-13	-1	-2	-11
x_4	0.38	-1.1	-0.077	0.23	0.077	-0.62	0.69

Tableau

Swap x_3 with x_9

$$x_3 = 2.2 + 3.2x_1 + 0.25x_6 - 13x_5 - 1x_2 - 2x_8 - 11x_9$$

$$\Rightarrow x_9 = 0.19 + 0.29x_1 + 0.022x_6 - 1.1x_5 - 0.093x_2 - 0.18x_8 - 0.089x_3$$

Tableau

Swap x_3 with x_9

$$x_3 = 2.2 + 3.2x_1 + 0.25x_6 - 13x_5 - 1x_2 - 2x_8 - 11x_9$$

$$\Rightarrow x_9 = 0.19 + 0.29x_1 + 0.022x_6 - 1.1x_5 - 0.093x_2 - 0.18x_8 - 0.089x_3$$

$$\max_{\mathbf{x}} f(\mathbf{x}) = 3.2 - 3.3x_1 - 1.3x_6 - 1.2x_5 - 0.32x_2 - 8.2x_8 + 0.49x_9$$

$$\Rightarrow \max_{\mathbf{x}} f(\mathbf{x}) = 3.3 - 3.1x_1 - 1.3x_6 - 1.7x_5 - 0.37x_2 - 8.3x_8 - 0.044x_3$$

Tableau

Swap x_3 with x_9

$$x_3 = 2.2 + 3.2x_1 + 0.25x_6 - 13x_5 - 1x_2 - 2x_8 - 11x_9$$

$$\Rightarrow x_9 = 0.19 + 0.29x_1 + 0.022x_6 - 1.1x_5 - 0.093x_2 - 0.18x_8 - 0.089x_3$$

$$\max_{\mathbf{x}} f(\mathbf{x}) = 3.2 - 3.3x_1 - 1.3x_6 - 1.2x_5 - 0.32x_2 - 8.2x_8 + 0.49x_9$$

$$\Rightarrow \max_{\mathbf{x}} f(\mathbf{x}) = 3.3 - 3.1x_1 - 1.3x_6 - 1.7x_5 - 0.37x_2 - 8.3x_8 - 0.044x_3$$

$$x_7 = 1.1 - 0.092x_1 - 0.092x_6 - 0.72x_5 - 0.11x_2 + 0.26x_8 - 0.17x_9$$

$$\Rightarrow x_7 = 1 - 0.14x_1 - 0.096x_6 - 0.53x_5 - 0.092x_2 + 0.29x_8 + 0.015x_3$$

$$x_4 = 0.38 - 1.1x_1 - 0.077x_6 + 0.23x_5 + 0.077x_2 - 0.62x_8 + 0.69x_9$$

$$\Rightarrow x_4 = 0.52 - 0.88x_1 - 0.062x_6 - 0.56x_5 + 0.012x_2 - 0.74x_8 - 0.062x_3$$

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = 3.3 - 3.1x_1 - 1.3x_6 - 1.7x_5 - 0.37x_2 - 8.3x_8 - 0.044x_3$$

where

$$x_7 = 1 - 0.14x_1 - 0.096x_6 - 0.53x_5 - 0.092x_2 + 0.29x_8 + 0.015x_3 \geq 0$$

$$x_9 = 0.19 + 0.29x_1 + 0.022x_6 - 1.1x_5 - 0.093x_2 - 0.18x_8 - 0.089x_3 \geq 0$$

$$x_4 = 0.52 - 0.88x_1 - 0.062x_6 - 0.56x_5 + 0.012x_2 - 0.74x_8 - 0.062x_3 \geq 0$$

$$x_1 = x_6 = x_5 = x_2 = x_8 = x_3 = 0$$

		x_1	x_6	x_5	x_2	x_8	x_3
$f(\mathbf{x})$	3.3	-3.1	-1.3	-1.7	-0.37	-8.3	-0.044
x_7	1	-0.14	-0.096	-0.53	-0.092	0.29	0.015
x_9	0.19	0.29	0.022	-1.1	-0.093	-0.18	-0.089
x_4	0.52	-0.88	-0.062	-0.56	0.012	-0.74	-0.062

Tableau

$$\max_{\mathbf{x}} f(\mathbf{x}) = 3.3 - 3.1x_1 - 1.3x_6 - 1.7x_5 - 0.37x_2 - 8.3x_8 - 0.044x_3$$

where

$$x_7 = 1 - 0.14x_1 - 0.096x_6 - 0.53x_5 - 0.092x_2 + 0.29x_8 + 0.015x_3 \geq 0$$

$$x_9 = 0.19 + 0.29x_1 + 0.022x_6 - 1.1x_5 - 0.093x_2 - 0.18x_8 - 0.089x_3 \geq 0$$

$$x_4 = 0.52 - 0.88x_1 - 0.062x_6 - 0.56x_5 + 0.012x_2 - 0.74x_8 - 0.062x_3 \geq 0$$

$$x_1 = x_6 = x_5 = x_2 = x_8 = x_3 = 0$$

Optimum

		x_1	x_6	x_5	x_2	x_8	x_3
$f(\mathbf{x})$	3.3	-3.1	-1.3	-1.7	-0.37	-8.3	-0.044
x_7	1	-0.14	-0.096	-0.53	-0.092	0.29	0.015
x_9	0.19	0.29	0.022	-1.1	-0.093	-0.18	-0.089
x_4	0.52	-0.88	-0.062	-0.56	0.012	-0.74	-0.062

Awkward Problems

- If there are any column with all entries positive then this variable can be increase forever—this is a signal that the linear programming problem is unbounded
- You can also find that a basic variable becomes zero—this is known as a degenerate feasible vector
- It can be removed by exchanging variables on the left of the inequality with variables on the right
- This makes the algorithm a bit more complex to implement

Awkward Problems

- If there are any column with all entries positive then this variable can be increase forever—this is a signal that the linear programming problem is unbounded
- You can also find that a basic variable becomes zero—this is known as a degenerate feasible vector
- It can be removed by exchanging variables on the left of the inequality with variables on the right
- This makes the algorithm a bit more complex to implement

Awkward Problems

- If there are any column with all entries positive then this variable can be increase forever—this is a signal that the linear programming problem is unbounded
- You can also find that a basic variable becomes zero—this is known as a degenerate feasible vector
- It can be removed by exchanging variables on the left of the inequality with variables on the right
- This makes the algorithm a bit more complex to implement

Awkward Problems

- If there are any column with all entries positive then this variable can be increase forever—this is a signal that the linear programming problem is unbounded
- You can also find that a basic variable becomes zero—this is known as a degenerate feasible vector
- It can be removed by exchanging variables on the left of the inequality with variables on the right
- This makes the algorithm a bit more complex to implement

High Performance Solvers

- Although the tableau method is the “classic solver” it doesn’t cut the mustard for large scale problems
- The simplex update can also be viewed as solving a linear set of equations which is facilitated by performing an LU-decomposition
- However, the constraints are often very sparse so good solvers try to take advantage of the sparsity
- Top end simplex algorithms are rather complex
- There is a second approach known as the interior point method which is competitive on large problems

High Performance Solvers

- Although the tableau method is the “classic solver” it doesn’t cut the mustard for large scale problems
- The simplex update can also be viewed as solving a linear set of equations which is facilitated by performing an LU-decomposition
- However, the constraints are often very sparse so good solvers try to take advantage of the sparsity
- Top end simplex algorithms are rather complex
- There is a second approach known as the interior point method which is competitive on large problems

High Performance Solvers

- Although the tableau method is the “classic solver” it doesn’t cut the mustard for large scale problems
- The simplex update can also be viewed as solving a linear set of equations which is facilitated by performing an LU-decomposition
- However, the constraints are often very sparse so good solvers try to take advantage of the sparsity
- Top end simplex algorithms are rather complex
- There is a second approach known as the interior point method which is competitive on large problems

High Performance Solvers

- Although the tableau method is the “classic solver” it doesn’t cut the mustard for large scale problems
- The simplex update can also be viewed as solving a linear set of equations which is facilitated by performing an LU-decomposition
- However, the constraints are often very sparse so good solvers try to take advantage of the sparsity
- Top end simplex algorithms are rather complex
- There is a second approach known as the interior point method which is competitive on large problems

High Performance Solvers

- Although the tableau method is the “classic solver” it doesn’t cut the mustard for large scale problems
- The simplex update can also be viewed as solving a linear set of equations which is facilitated by performing an LU-decomposition
- However, the constraints are often very sparse so good solvers try to take advantage of the sparsity
- Top end simplex algorithms are rather complex
- There is a second approach known as the interior point method which is competitive on large problems

Time Complexity of Simplex

- The time complexity of the updates is $O(n^2)$
- The critical question is how many updates are necessary
- It turns out that typically this is $O(n)$ making the simplex algorithm $O(n^3)$
- However, it is possible to cook up problems where there is a “long path” from the initial solution to the optimum which is exponentially big
- Thus the worst case time is exponential, although this almost never happens in practice

Time Complexity of Simplex

- The time complexity of the updates is $O(n^2)$
- The critical question is how many updates are necessary
- It turns out that typically this is $O(n)$ making the simplex algorithm $O(n^3)$
- However, it is possible to cook up problems where there is a “long path” from the initial solution to the optimum which is exponentially big
- Thus the worst case time is exponential, although this almost never happens in practice

Time Complexity of Simplex

- The time complexity of the updates is $O(n^2)$
- The critical question is how many updates are necessary
- It turns out that typically this is $O(n)$ making the simplex algorithm $O(n^3)$
- However, it is possible to cook up problems where there is a “long path” from the initial solution to the optimum which is exponentially big
- Thus the worst case time is exponential, although this almost never happens in practice

Time Complexity of Simplex

- The time complexity of the updates is $O(n^2)$
- The critical question is how many updates are necessary
- It turns out that typically this is $O(n)$ making the simplex algorithm $O(n^3)$
- However, it is possible to cook up problems where there is a “long path” from the initial solution to the optimum which is exponentially big
- Thus the worst case time is exponential, although this almost never happens in practice

Time Complexity of Simplex

- The time complexity of the updates is $O(n^2)$
- The critical question is how many updates are necessary
- It turns out that typically this is $O(n)$ making the simplex algorithm $O(n^3)$
- However, it is possible to cook up problems where there is a “long path” from the initial solution to the optimum which is exponentially big
- Thus the worst case time is exponential, although this almost never happens in practice

Interior Point Method

- An alternative to the simplex method is the interior point method which always remains in the feasible region, away from the constraints
- These method iterate towards the constraints and are provably polynomial
- For small linear programming problems they are out-performed in practice by the simplex method
- On large and very large problems they seem to perform as well if not better than the simplex method
- The high-end solvers will have a variety of interior point methods tailored to the particular problem

Interior Point Method

- An alternative to the simplex method is the interior point method which always remains in the feasible region, away from the constraints
- These method iterate towards the constraints and are provably polynomial
- For small linear programming problems they are out-performed in practice by the simplex method
- On large and very large problems they seem to perform as well if not better than the simplex method
- The high-end solvers will have a variety of interior point methods tailored to the particular problem

Interior Point Method

- An alternative to the simplex method is the interior point method which always remains in the feasible region, away from the constraints
- These method iterate towards the constraints and are provably polynomial
- For small linear programming problems they are out-performed in practice by the simplex method
- On large and very large problems they seem to perform as well if not better than the simplex method
- The high-end solvers will have a variety of interior point methods tailored to the particular problem

Interior Point Method

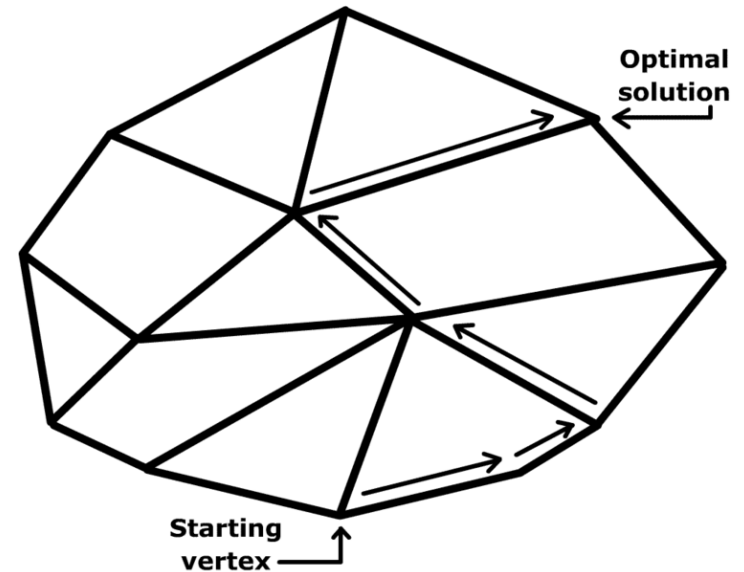
- An alternative to the simplex method is the interior point method which always remains in the feasible region, away from the constraints
- These method iterate towards the constraints and are provably polynomial
- For small linear programming problems they are out-performed in practice by the simplex method
- On large and very large problems they seem to perform as well if not better than the simplex method
- The high-end solvers will have a variety of interior point methods tailored to the particular problem

Interior Point Method

- An alternative to the simplex method is the interior point method which always remains in the feasible region, away from the constraints
- These method iterate towards the constraints and are provably polynomial
- For small linear programming problems they are out-performed in practice by the simplex method
- On large and very large problems they seem to perform as well if not better than the simplex method
- The high-end solvers will have a variety of interior point methods tailored to the particular problem

Outline

1. Recap
2. Basic Feasible Solutions
3. Simplex Method
4. **Classic LP Problems**



LP Problems

- Any problem that can be set up as a linear program can be solved in polynomial time
- One way is just to feed it to a LP-solver
- Sometimes the problems are important enough and have such a distinctive formulation that faster specialised algorithms have been developed
- We consider a couple of classic problems: *maximum flow* and *linear assignment*

LP Problems

- Any problem that can be set up as a linear program can be solved in polynomial time
- One way is just to feed it to a LP-solver
- Sometimes the problems are important enough and have such a distinctive formulation that faster specialised algorithms have been developed
- We consider a couple of classic problems: *maximum flow* and *linear assignment*

LP Problems

- Any problem that can be set up as a linear program can be solved in polynomial time
- One way is just to feed it to a LP-solver
- Sometimes the problems are important enough and have such a distinctive formulation that faster specialised algorithms have been developed
- We consider a couple of classic problems: *maximum flow* and *linear assignment*

LP Problems

- Any problem that can be set up as a linear program can be solved in polynomial time
- One way is just to feed it to a LP-solver
- Sometimes the problems are important enough and have such a distinctive formulation that faster specialised algorithms have been developed
- We consider a couple of classic problems: *maximum flow* and *linear assignment*

Maximum Flow

- In maximum flow we consider a directed graph representing a network of pipes
- We choose one vertex as the source and a second vertex as a sink
- Each edge has a flow capacity that cannot be exceeded
- The problem is to maximise the flow between source of sink
- This can be used to model the flow of a fluid, parts in an assembly line, current in an electrical circuit or packets through a communication network

Maximum Flow

- In maximum flow we consider a directed graph representing a network of pipes
- We choose one vertex as the source and a second vertex as a sink
- Each edge has a flow capacity that cannot be exceeded
- The problem is to maximise the flow between source of sink
- This can be used to model the flow of a fluid, parts in an assembly line, current in an electrical circuit or packets through a communication network

Maximum Flow

- In maximum flow we consider a directed graph representing a network of pipes
- We choose one vertex as the source and a second vertex as a sink
- Each edge has a flow capacity that cannot be exceeded
- The problem is to maximise the flow between source of sink
- This can be used to model the flow of a fluid, parts in an assembly line, current in an electrical circuit or packets through a communication network

Maximum Flow

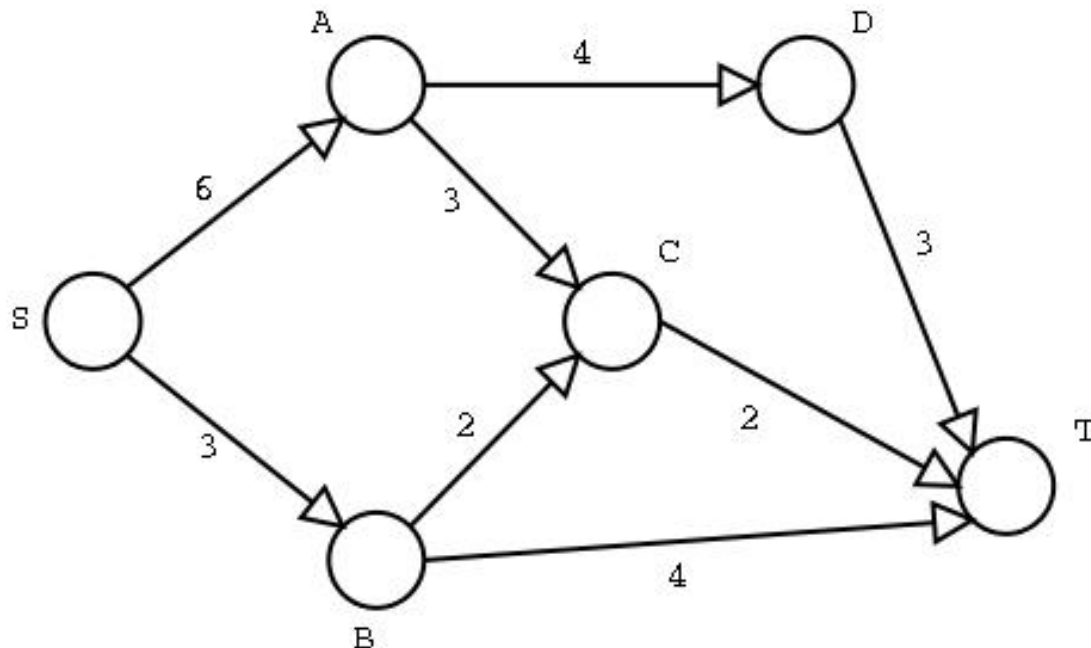
- In maximum flow we consider a directed graph representing a network of pipes
- We choose one vertex as the source and a second vertex as a sink
- Each edge has a flow capacity that cannot be exceeded
- The problem is to maximise the flow between source of sink
- This can be used to model the flow of a fluid, parts in an assembly line, current in an electrical circuit or packets through a communication network

Maximum Flow

- In maximum flow we consider a directed graph representing a network of pipes
- We choose one vertex as the source and a second vertex as a sink
- Each edge has a flow capacity that cannot be exceeded
- The problem is to maximise the flow between source of sink
- This can be used to model the flow of a fluid, parts in an assembly line, current in an electrical circuit or packets through a communication network

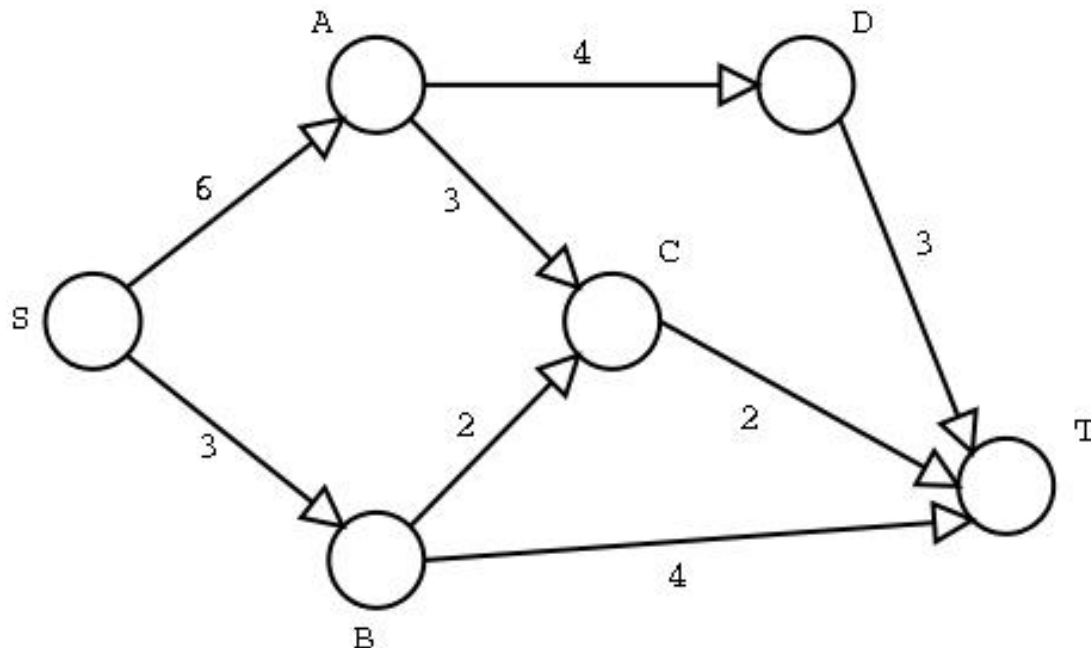
Example

- Consider a firm that has to ship haggis from Edinburgh to Southampton
- The shipping firm transports this in crates which it sends through intermediate cities
- The number of crates is limited by the size of the lorries it uses



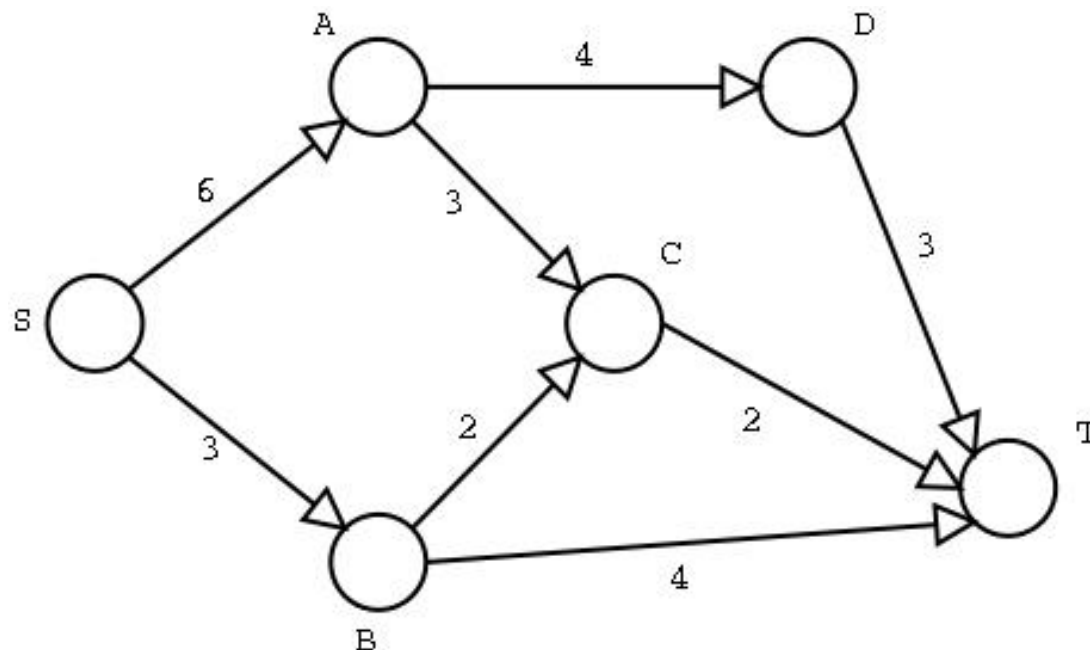
Example

- Consider a firm that has to ship haggis from Edinburgh to Southampton
- The shipping firm transports this in crates which it sends through intermediate cities
- The number of crates is limited by the size of the lorries it uses



Example

- Consider a firm that has to ship haggis from Edinburgh to Southampton
- The shipping firm transports this in crates which it sends through intermediate cities
- The number of crates is limited by the size of the lorries it uses



Flow

- We are given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each edge has a capacity $c(i, j)$
- We define the flow from i to j as $f(i, j)$ with $0 \leq f(i, j) \leq c(i, j)$
- For all vertices except the source (s) and sink (t) we assume

$$\forall i \in \mathcal{V} / \{s, t\} \quad \sum_{j \in \mathcal{V} | (i, j) \in \mathcal{E}} f(i, j) = \sum_{j \in \mathcal{V} | (j, i) \in \mathcal{E}} f(j, i)$$

(i.e. no flow is lost from source to sink)

- We want to maximise the flow from the source

$$\sum_{i \in \mathcal{V} | (s, i) \in \mathcal{E}} f(s, i)$$

Flow

- We are given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each edge has a capacity $c(i, j)$
- We define the flow from i to j as $f(i, j)$ with $0 \leq f(i, j) \leq c(i, j)$
- For all vertices except the source (s) and sink (t) we assume

$$\forall i \in \mathcal{V} / \{s, t\} \quad \sum_{j \in \mathcal{V} | (i, j) \in \mathcal{E}} f(i, j) = \sum_{j \in \mathcal{V} | (j, i) \in \mathcal{E}} f(j, i)$$

(i.e. no flow is lost from source to sink)

- We want to maximise the flow from the source

$$\sum_{i \in \mathcal{V} | (s, i) \in \mathcal{E}} f(s, i)$$

Flow

- We are given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each edge has a capacity $c(i, j)$
- We define the flow from i to j as $f(i, j)$ with $0 \leq f(i, j) \leq c(i, j)$
- For all vertices except the source (s) and sink (t) we assume

$$\forall i \in \mathcal{V} / \{s, t\} \quad \sum_{j \in \mathcal{V} | (i, j) \in \mathcal{E}} f(i, j) = \sum_{j \in \mathcal{V} | (j, i) \in \mathcal{E}} f(j, i)$$

(i.e. no flow is lost from source to sink)

- We want to maximise the flow from the source

$$\sum_{i \in \mathcal{V} | (s, i) \in \mathcal{E}} f(s, i)$$

Flow

- We are given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where each edge has a capacity $c(i, j)$
- We define the flow from i to j as $f(i, j)$ with $0 \leq f(i, j) \leq c(i, j)$
- For all vertices except the source (s) and sink (t) we assume

$$\forall i \in \mathcal{V} / \{s, t\} \quad \sum_{j \in \mathcal{V} | (i, j) \in \mathcal{E}} f(i, j) = \sum_{j \in \mathcal{V} | (j, i) \in \mathcal{E}} f(j, i)$$

(i.e. no flow is lost from source to sink)

- We want to maximise the flow from the source

$$\sum_{i \in \mathcal{V} | (s, i) \in \mathcal{E}} f(s, i)$$

Solving Maximum Flow

- As set up we have a linear objective function with linear constraints
- We can therefore solve this problem with a LP-solver
- (Note the solution will typically involve a fraction flow)
- However, this is such a classic problem with a distinctive structure that we can solve it more quickly with other algorithms
- The classic algorithm is the Ford-Fulkerson method with run time $O(|\mathcal{E}| \times f_{\max})$ where f_{\max} is the maximum flow, although we won't cover this in the course

Solving Maximum Flow

- As set up we have a linear objective function with linear constraints
- We can therefore solve this problem with a LP-solver
- (Note the solution will typically involve a fraction flow)
- However, this is such a classic problem with a distinctive structure that we can solve it more quickly with other algorithms
- The classic algorithm is the Ford-Fulkerson method with run time $O(|\mathcal{E}| \times f_{\max})$ where f_{\max} is the maximum flow, although we won't cover this in the course

Solving Maximum Flow

- As set up we have a linear objective function with linear constraints
- We can therefore solve this problem with a LP-solver
- (Note the solution will typically involve a fraction flow)
- However, this is such a classic problem with a distinctive structure that we can solve it more quickly with other algorithms
- The classic algorithm is the Ford-Fulkerson method with run time $O(|\mathcal{E}| \times f_{\max})$ where f_{\max} is the maximum flow, although we won't cover this in the course

Solving Maximum Flow

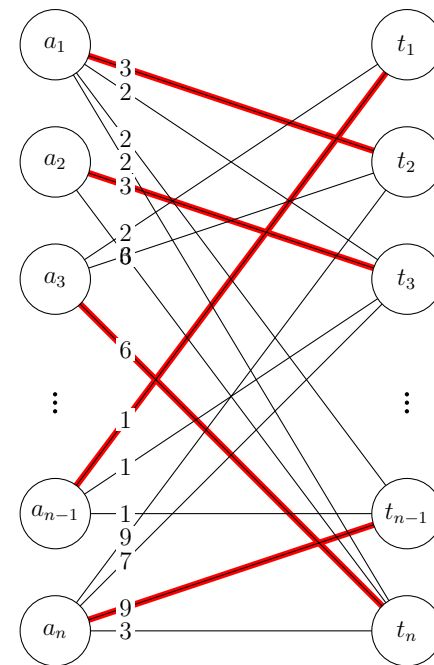
- As set up we have a linear objective function with linear constraints
- We can therefore solve this problem with a LP-solver
- (Note the solution will typically involve a fraction flow)
- However, this is such a classic problem with a distinctive structure that we can solve it more quickly with other algorithms
- The classic algorithm is the Ford-Fulkerson method with run time $O(|\mathcal{E}| \times f_{\max})$ where f_{\max} is the maximum flow, although we won't cover this in the course

Solving Maximum Flow

- As set up we have a linear objective function with linear constraints
- We can therefore solve this problem with a LP-solver
- (Note the solution will typically involve a fraction flow)
- However, this is such a classic problem with a distinctive structure that we can solve it more quickly with other algorithms
- The classic algorithm is the Ford-Fulkerson method with run time $O(|\mathcal{E}| \times f_{\max})$ where f_{\max} is the maximum flow, although we won't cover this in the course

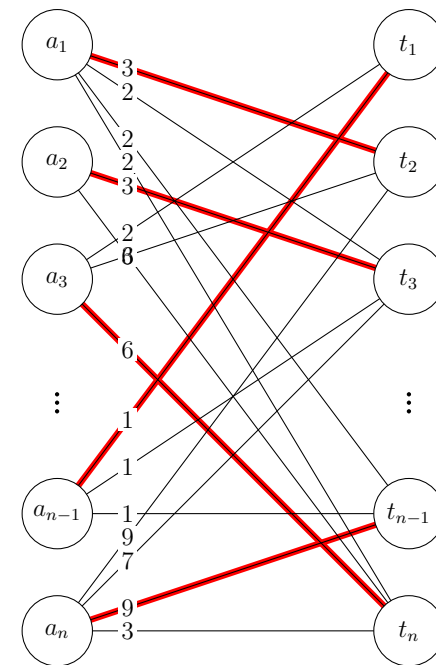
Linear Assignment

- We are given a set of n agents, \mathcal{A} , and n tasks, \mathcal{T}
- Each agent has a cost associated with performing a task $c(a, t)$
- We want to assign an agent to one task so as to minimise the total cost
- Consider a taxi firm with taxi's at 5 different locations and 5 requests to fulfil. The cost is the distance to the clients. Which taxi should go to which client?



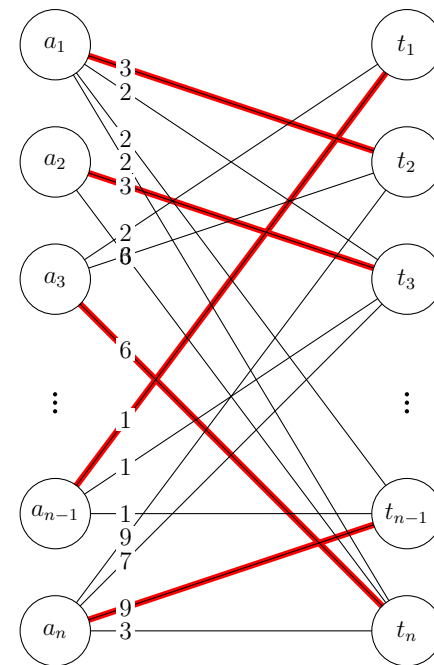
Linear Assignment

- We are given a set of n agents, \mathcal{A} , and n tasks, \mathcal{T}
- Each agent has a cost associated with performing a task $c(a, t)$
- We want to assign an agent to one task so as to minimise the total cost
- Consider a taxi firm with taxi's at 5 different locations and 5 requests to fulfil. The cost is the distance to the clients. Which taxi should go to which client?



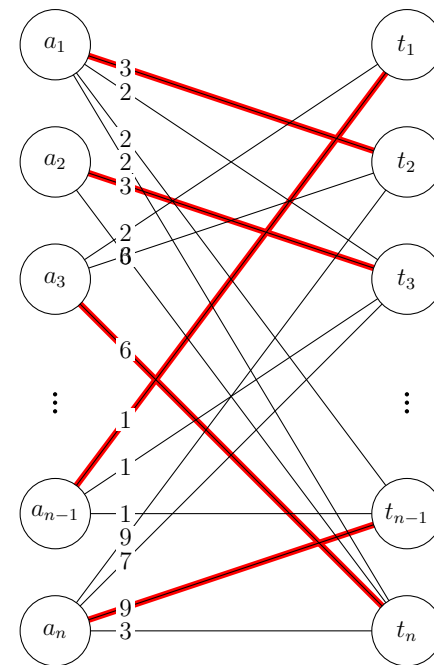
Linear Assignment

- We are given a set of n agents, \mathcal{A} , and n tasks, \mathcal{T}
- Each agent has a cost associated with performing a task $c(a, t)$
- We want to assign an agent to one task so as to minimise the total cost
- Consider a taxi firm with taxi's at 5 different locations and 5 requests to fulfil. The cost is the distance to the clients. Which taxi should go to which client?



Linear Assignment

- We are given a set of n agents, \mathcal{A} , and n tasks, \mathcal{T}
- Each agent has a cost associated with performing a task $c(a, t)$
- We want to assign an agent to one task so as to minimise the total cost
- Consider a taxi firm with taxi's at 5 different locations and 5 requests to fulfil. The cost is the distance to the clients. Which taxi should go to which client?



LA as LP

- The linear assignment problem can be set as a linear programming problem

$$\min_x \sum_{a \in \mathcal{A}, t \in \mathcal{T}} c(a, t) x_{a, t}$$

subject to

$$\forall a \in \mathcal{A} \quad \sum_{t \in \mathcal{T}} x_{a, t} = 1$$

$$\forall t \in \mathcal{T} \quad \sum_{a \in \mathcal{A}} x_{a, t} = 1$$

$$\forall (a, t) \in (\mathcal{A}, \mathcal{T}) \quad x_{a, t} \geq 0$$

Hungarian Algorithm

- Linear assignment is another classic problem that is commonly encountered
- Although it can be solved using a generic LP-solver this is not the most efficient algorithm
- The most efficient algorithm is the Hungarian algorithms
- This is rather complex (having once implemented it I can tell you from bitter experience it ain't easy)
- Its worst case time is $O(n^3)$ although it frequently takes $\Theta(n^2)$

Hungarian Algorithm

- Linear assignment is another classic problem that is commonly encountered
- Although it can be solved using a generic LP-solver this is not the most efficient algorithm
- The most efficient algorithm is the Hungarian algorithms
- This is rather complex (having once implemented it I can tell you from bitter experience it ain't easy)
- Its worst case time is $O(n^3)$ although it frequently takes $\Theta(n^2)$

Hungarian Algorithm

- Linear assignment is another classic problem that is commonly encountered
- Although it can be solved using a generic LP-solver this is not the most efficient algorithm
- The most efficient algorithm is the Hungarian algorithms
- This is rather complex (having once implemented it I can tell you from bitter experience it ain't easy)
- Its worst case time is $O(n^3)$ although it frequently takes $\Theta(n^2)$

Hungarian Algorithm

- Linear assignment is another classic problem that is commonly encountered
- Although it can be solved using a generic LP-solver this is not the most efficient algorithm
- The most efficient algorithm is the Hungarian algorithms
- This is rather complex (having once implemented it I can tell you from bitter experience it ain't easy)
- Its worst case time is $O(n^3)$ although it frequently takes $\Theta(n^2)$

Hungarian Algorithm

- Linear assignment is another classic problem that is commonly encountered
- Although it can be solved using a generic LP-solver this is not the most efficient algorithm
- The most efficient algorithm is the Hungarian algorithms
- This is rather complex (having once implemented it I can tell you from bitter experience it ain't easy)
- Its worst case time is $O(n^3)$ although it frequently takes $\Theta(n^2)$

Quadratic Programming

- If we have linear constraints and a quadratic objective function then we have a quadratic programming problem
- Again this can be solved in polynomial time
- Many of the ideas used are the same as for linear programming
- This also has important applications in science and engineering

Quadratic Programming

- If we have linear constraints and a quadratic objective function then we have a quadratic programming problem
- Again this can be solved in polynomial time
- Many of the ideas used are the same as for linear programming
- This also has important applications in science and engineering

Quadratic Programming

- If we have linear constraints and a quadratic objective function then we have a quadratic programming problem
- Again this can be solved in polynomial time
- Many of the ideas used are the same as for linear programming
- This also has important applications in science and engineering

Quadratic Programming

- If we have linear constraints and a quadratic objective function then we have a quadratic programming problem
- Again this can be solved in polynomial time
- Many of the ideas used are the same as for linear programming
- This also has important applications in science and engineering

Lessons

- Linear programming is a classic problem
- We know a huge number of problems are solvable in polynomial time because they can be formulated as linear programs
- Linear programs occur sufficiently often that they are hugely important
- They aren't easy to solve, although standard simplex is not massively complex
- For particular LP problems with distinctive structure there are sometimes better algorithms than generic LP-solvers

Lessons

- Linear programming is a classic problem
- We know a huge number of problems are solvable in polynomial time because they can be formulated as linear programs
- Linear programs occur sufficiently often that they are hugely important
- They aren't easy to solve, although standard simplex is not massively complex
- For particular LP problems with distinctive structure there are sometimes better algorithms than generic LP-solvers

Lessons

- Linear programming is a classic problem
- We know a huge number of problems are solvable in polynomial time because they can be formulated as linear programs
- Linear programs occur sufficiently often that they are hugely important
- They aren't easy to solve, although standard simplex is not massively complex
- For particular LP problems with distinctive structure there are sometimes better algorithms than generic LP-solvers

Lessons

- Linear programming is a classic problem
- We know a huge number of problems are solvable in polynomial time because they can be formulated as linear programs
- Linear programs occur sufficiently often that they are hugely important
- They aren't easy to solve, although standard simplex is not massively complex
- For particular LP problems with distinctive structure there are sometimes better algorithms than generic LP-solvers

Lessons

- Linear programming is a classic problem
- We know a huge number of problems are solvable in polynomial time because they can be formulated as linear programs
- Linear programs occur sufficiently often that they are hugely important
- They aren't easy to solve, although standard simplex is not massively complex
- For particular LP problems with distinctive structure there are sometimes better algorithms than generic LP-solvers