UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 1 – Introduction

Dr Neslihan Suzen

Module convener – Dr Neslihan Suzen

About me: I am a Lecturer in Applied Mathematics and a Data Scientist with a PhD in Natural Language Processing (NLP).

My research focuses on using Text Mining, Machine Learning and Semantic Analysis in developing mathematical and computational frameworks to design AI and NLP systems and automated decision support.

Module organisation

Lectures:

- Lectures start on 12th September. Zoom link to lectures and tutorials is available on BB.
- There will be weekly ‘revision lectures’ that will summarise the main topics that we have covered each week.
- Recordings of the lectures will be available at *'Reflect' Recordings* on Blackboard.

Tutorials:

- Every week there will be 1h live tutorial via Zoom on Friday.
- We will discuss solutions to the self-study exercises given at the end of each set of slides and answer any questions you may have.

Module organisation

Questions:

- You can ask questions during the live tutorial sessions.If you have questions outside of lectures, please use the dedicated forum in the *Discussions* menu on Blackboard. If you have private matters to discuss, please send me an email at ns553@Leicester.ac.uk

Learning materials:

- Powerpoint slides will be uploaded regularly on the Blackboard page.

Module organisation

Assessment:

- 40% coursework (2 problem sheets) and 60% final exam in January.

Coursework:

- Each problem sheet counts 20% towards the final mark. The submission deadlines are:
  1. Problem sheet 1: TBC at 12pm (midday) on Blackboard
  2. Problem sheet 2: TBC at 12pm (midday) on Blackboard

Module content and plan at a glance

Linear programming

Queueing theory

Problem Sheet 1

Nonlinear optimization

Problem Sheet 2

Revision/self-study

Mock exam

Prerequisites

- The **main prerequisites** needed to follow this module are:
  - Linear algebra (mostly operations with vectors and matrices)
  - Calculus (mostly continuity and multivariate differentiation)
  - Elementary probability theory
  - Basic coding skills (mainly to solve large problems)

- **Coding:** some exercises require numerical computations. In this module we use Matlab. If you need to refresh your Matlab, have a look at this MathWorks tutorial (you may need to create a mathworks account to access it).

Recommended reading

The recommended textbooks are:

- Hillier and Lieberman, *Introduction to Operations Research,* McGraw-Hill
- Naadimuthu, R. Bronson, *Operations Research*, Schaum's Outlines
- Nocedal and Wright, Numerical optimization, Springer
- Bosch, *Opt Art: from Mathematical Optimization to Visual Design*, Princeton University Press

These are available on Blackboard (go to *Readings and Sources*).

Other useful websites are:

- https://docs.mosek.com/modeling-cookbook/index.html#
- https://www.theorsociety.com/
- https://www.informs.org/
- http://www.pitt.edu/~jrclass/or/or-intro.html
- http://people.brunel.ac.uk/~mastjjb/jeb/or/intro.html

What is operational research?

Definition/description*:

"operational research is a discipline that deals with the development and application of advanced analytical methods to improve decision-making.“

"Employing techniques from other mathematical sciences, such as modeling, statistics, and optimization, operations research arrives at optimal or near-optimal solutions to complex decision-making problems. “

"Operations research is often concerned with determining the extreme values of some real-world objective: the maximum (of profit, performance, or yield) or minimum (of loss, risk, or cost).“

"Originating in military efforts before World War II, its techniques have grown to concern problems in a variety of industries."

* From wikipedia, accessed on 14 September 2021.

Operational research in practice

The key steps of operational research are:

- Understand the problem to be solved
- Formulate a mathematical model
- Collect relevant data
- Compute the solution to the mathematical model
- Validate the solution and possibly perform sensitivity analysis.

Next Lecture outline

Today's goal is to:

- introduce the concepts of optimization and linear programming,
- learn how to solve linear programming problems using a graphical method,
- learn how to write a linear programming problem in standard form,
- and learn how some nonlinear functions can be modeled in a linear fashion.

Today's lecture is loosely based on chapters 2.1 and 2.2 of the Mosek Cookbook.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 2 – Linear programming

Dr Neslihan Suzen

Lecture Outline

Today's goal is to:

- introduce the concepts of optimisation and linear programming,
- learn how to solve linear programming problems using a graphical method,
- learn how to write a linear programming problem in standard form,
- and learn how some nonlinear functions can be modeled in a linear fashion.

Today's lecture is loosely based on chapters 2.1 and 2.2 of the Mosek Cookbook.

What is optimisation?

Optimisation is a technique to determine the best possible solution to a given problem.

This is done by first formulating a problem in mathematical terms, and then by applying suitable optimisation algorithms.

Formulating the problem in mathematical terms can be challenging. Coding it up can also require some skills.

Optimisation – key terms

To formulate the mathematical model, one must first identify:

- the objective: a function that describes how "optimal" a configuration is, and that you aim to maximise or minimise (e.g.: profit, cost, production,...)

- the control/decision variables: what you can modify to maximise/minimise the objective function (e.g.: resource allocation, ...)

- the constraints: the conditions that that the control variable must satisfy (eg: time and resource constraints, ...)

A trivial example – problem statement

You have £15 and you want to purchase as much flour as possible to bake delicious bread.

Your local supermarket sales loose flour at £1 per kg.

How many kg of flour will you buy?

A trivial example – mathematical model

Of course, the answer is 15 kg. To model this task as an optimisation problem, let x be how many kilograms of flour you want to purchase. Then, x is the solution to the optimisation problem

$$\max x \;\; subject\ to\ x\ [kg] \cdot 1 \left[\frac{£}{kg}\right] \leq [£]15 \;\; and \;\;\; x \geq 0.$$

Here,

- the *control variable* is $x \in \mathbb{R}$
- the *objective function* is $f(x) \; = \; x$,
- the *constraints* are $x\ [kg] \cdot 1[£/kg] \leq [£]15$ and $x \geq 0$.

This is a *linear programming problem* because both the objective and the constraints can be expressed with linear functions.

An example – problem statement

Description: A chemical plant produces two compounds (compound A and B).

- Producing 1 kg of compound A requires 1 kg of raw material and 8 processing hours, whereas 1 kg of compound B requires 2 kg of raw material and 7 processing hours.
- Compound A can be sold at £80 per kg, whereas compound B at £120 per kg.
- The market can absorb at most 3 kg of compound A.
- The chemical plant currently owns 6 kg of raw material and has 28 processing hours available.

Question: How many kilograms of each compound should the plant produce to maximise the revenue?

An example – mathematical formulation 1/2

To formulate this problem as a mathematical model, we introduce the control variables x and y to denote the amount of compound A and B to be produced (in kg), respectively.

Then, the *objective* function, which we want to maximise, can be written as:

$$f(x, y) = 80x + 120y$$

and the *constraints* are:

$x + 2y \leq 6$	(constraint on raw material available)
$8x + 7y \leq 28$	(constraint on processing time available)
$x \leq 3$	(constraint imposed by market)
$x, y \geq 0$	(hidden constraints)

An example – mathematical formulation 2/2

Mathematically, we usually write the problem more compactly as shown in the right column.

Maximise:

$$f(x, y) = 80x + 120y$$

subject to:

$$\begin{array}{ccccc} x & + & 2y & \leq & 6 \\ 8x & + & 7y & \leq & 28 \\ x & & & \leq & 3 \\ x & & & \geq & 0 \\ & & y & \geq & 0 \end{array}$$

Maximise:

$$f(x, y) = (80\ 120) \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

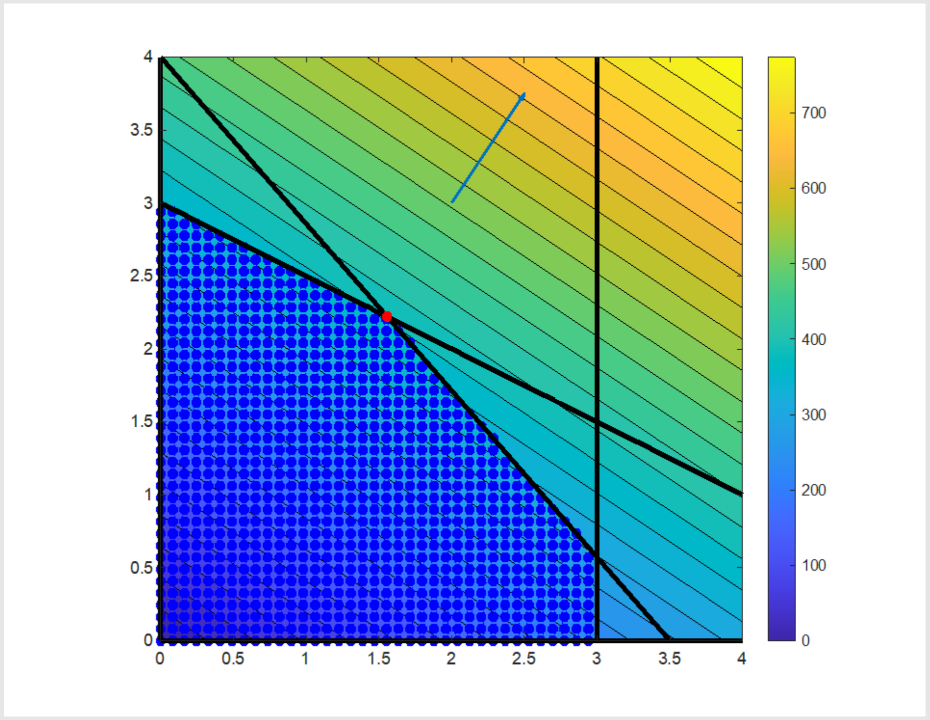
subject to:

$$\begin{pmatrix} 1 & 2 \\ 8 & 7 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 6 \\ 28 \\ 3 \end{pmatrix}$$
$$x, y \geq 0$$

This is a *linear programming problem* because both the objective and the constraints can be expressed with linear functions.

An example – graphical solution (with Matlab)

(with Matlab see OR01_feasible_region.m)



Note:

- the *feasible region* is a convex polygon,
- the optimum is at a vertex,
- the maximum is attained at the intersection of the lines

$$\begin{array}{rcrcr} x & + & 2y & = & 6 \\ 8x & + & 7y & = & 28 \end{array}$$

  In this case, we say that the two inequality constraints

$$\begin{array}{rcrcr} x & + & 2y & \leq & 6 \\ 8x & + & 7y & \leq & 28 \end{array}$$

  are *active* (or *tight*) at the optimum.

Linear programming in standard form

Linear programming problems can be written in many ways, e.g.:

$$1)\quad \begin{array}{ll} min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad 2)\quad \begin{array}{ll} min & c^T x \\ s.t. & Ax \leq b \\ & x \in \mathbb{R}^n \end{array} \qquad 3)\quad \begin{array}{ll} max & c^T x \\ s.t. & Ax \geq b \\ & x \in \mathbb{R}^n \end{array}$$

for some $A \in \mathbb{R}^{m,n}, c \in \mathbb{R}^n, b \in \mathbb{R}^m$. Formulation 1) is called the *standard form*.

Lemma: Representations 1), 2), and 3) (and similar combinations), are all equivalent (for possibly different matrices A, b, and dimensions m, n).

Linear programming in standard form – Proof 1/3

Lemma: The following representations are equivalent.

$$1)\quad \begin{aligned} \min\quad & c^T x \\ s.t.\quad & Ax = b \\ & x \geq 0 \end{aligned} \qquad 2)\quad \begin{aligned} \min\quad & c^T x \\ s.t.\quad & Ax \leq b \\ & x \in \mathbb{R}^n \end{aligned} \qquad 3)\quad \begin{aligned} \max\quad & c^T x \\ s.t.\quad & Ax \geq b \\ & x \in \mathbb{R}^n \end{aligned}$$

Proof: 1) can be written as 2):

$$\begin{aligned} \min\quad & c^T x \\ s.t.\quad & Ax = b \\ & x \geq 0 \end{aligned} \Leftrightarrow \begin{aligned} \min\quad & c^T x \\ s.t.\quad & Ax \leq b \\ & Ax \geq b \\ & Ix \geq 0 \end{aligned} \Leftrightarrow \begin{aligned} \min\quad & c^T x \\ s.t.\quad & Ax \leq b \\ & -Ax \leq -b \\ & -Ix \leq 0 \end{aligned} \Leftrightarrow \begin{aligned} \min\quad & c^T x \\ s.t.\quad & \begin{bmatrix} A \\ -A \\ -I \end{bmatrix} x \leq \begin{pmatrix} b \\ -b \\ 0 \end{pmatrix} \end{aligned}$$

Note that there are no constraints on x, hence $x \in \mathbb{R}^n$.

Linear programming in standard form – Proof 2/3

Lemma: The following representations are equivalent.

$$1)\quad \begin{array}{ll} \min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad 2)\quad \begin{array}{ll} \min & c^T x \\ s.t. & Ax \leq b \\ & x \in \mathbb{R}^n \end{array} \qquad 3)\quad \begin{array}{ll} \max & c^T x \\ s.t. & Ax \geq b \\ & x \in \mathbb{R}^n \end{array}$$

Proof: 2) can be written as 1):

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax \leq b \\ & x \in \mathbb{R}^n \end{array} \Leftrightarrow \begin{array}{ll} \min & c^T (y - z) \\ s.t. & A(y - z) \leq b \\ & y, z \geq 0 \end{array} \Leftrightarrow \begin{array}{ll} \min & \binom{c}{-c} \cdot \binom{y}{z} \\ s.t. & [A, -A]\binom{y}{z} \leq b \\ & y, z \geq 0 \end{array} \Leftrightarrow \begin{array}{ll} \min & \binom{c}{-c} \cdot \binom{y}{z} \\ s.t. & [A, -A, I]\begin{pmatrix} y \\ z \\ s \end{pmatrix} = b \\ & y, z, s \geq 0 \end{array}$$

Remark: the new variable s is known as *slack variable*.

Linear programming in standard form – Proof 3/3

Lemma: The following representations are equivalent.

$$1)\quad \begin{array}{ll} \min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad 2)\quad \begin{array}{ll} \min & c^T x \\ s.t. & Ax \leq b \\ & x \in \mathbb{R}^n \end{array} \qquad 3)\quad \begin{array}{ll} \max & c^T x \\ s.t. & Ax \geq b \\ & x \in \mathbb{R}^n \end{array}$$

Proof: 2) is equivalent to 3):

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax \leq b \\ & x \in \mathbb{R}^n \end{array} \Leftrightarrow \begin{array}{ll} -\max & -(c^T x) \\ s.t. & Ax \leq b \\ & x \in \mathbb{R}^n \end{array} \Leftrightarrow \begin{array}{ll} -\max & c^T(-x) \\ s.t. & -A(-x) \leq b \\ & x \in \mathbb{R}^n \end{array} \Leftrightarrow \begin{array}{ll} -\max & c^T y \\ s.t. & Ay \geq (-b) \\ & y \in \mathbb{R}^n \end{array}$$

Note that if x solves 2) and y solves 3), then $c^T x = -(c^T y)$.

$\square$

UNIVERSITY OF LEICESTER

www.le.ac.uk

Linear modelling - maximum

The inequality $t \geq max\{x_1, x_2, x_3, \dots, x_n\}$ is equivalent to the n inequalities

$$t \ \geq x_1, t \ \geq x_2. \ \dots \ t \ \geq x_n \ .$$

The inequality $t \leq min\{x_1, x_2, x_3, \dots, x_n\}$ can be treated analogously. Similarly,

$$\min \max_{\{i=1,\dots m\}} \{a_i^T x + b_i\} \ s.t. Ax \leq b, x \in \mathbb{R}^n$$

can be rewritten as the linear programming problem

$$\begin{array}{cc} min & t \\ s.t & t \geq a_1^T x + b_1 \\ & \vdots \\ & t \geq a_n^T x + b_n \\ & Ax \leq b \\ & x \in \mathbb{R}^n, t \in \mathbb{R} \end{array}$$

Application: piecewise-affine functions can be used to approximate nonlinear functions accurately (e.g. $|x| = \max(x, -x)$), and hence to approximate a nonlinear (convex) problem with a linear one.

Linear modelling – absolute value and ℓ^∞-norm

When the absolute value of a scalar enters the model as a constraint, e.g.,

$$|x| \leq b,$$

this inequality can be modelled with the 2 inequalities

$$-b \leq x \ \leq b.$$

The ℓ^∞-norm of a vector $x \in \mathbb{R}^n$ is defined as

$$\|x\|_\infty := \max_{\{i=1,\ldots,n\}} |x_i|$$

The inequality $\|x\|_\infty \leq t$ can be modelled with the $2n$ inequalities

$$-t \leq x_1 \leq t, -t \leq x_2 \leq t, \ldots, -t \leq x_n \leq t.$$

Linear modelling - ℓ^1-norm

The ℓ^1-norm of a vector $x \in \mathbb{R}^n$ is defined as

$$\|x\|_1 := |x_1| + |x_2| + \ \dots + |x_n|$$

The inequality $\|x\|_1 \leq t$ can be modelled as follows,

$$|x_1| \leq z_1, |x_2| \leq z_2, \dots, |x_n| \leq z_n, \qquad and \qquad z_1 + z_2 + \ \dots + z_n = t$$

where $z \in \mathbb{R}^n$ is an auxiliary variable

Application: Given an underdetermined linear system $Ax = b$, the optimization problem $\min \|x\|_1 \ s.t. Ax = b, x \in \mathbb{R}^n$ can be used to find a *sparse solution* (a solution with as many zeros as possible). This is the gist of compressed sensing.

Summary

Today we have learnt

- how to solve simple (2 dimensional) linear programming problems using a graphical method.
- that linear programming problems can be written in many equivalent formulations, and that one of them is know as standard form.
- that some nonlinear functions such as *max*, *min*, *abs*, $\|x\|_1$, and $\|x\|_\infty$ can be described using linear functions.

Self-study

1. A farmer owns 80 acres and produces wheat and barley. To produce these crops, the farmer incurs some costs (seeds, fertilizers, etc.). An acre of wheat costs £100, whereas an acre of barley costs £120. To sell the crops at the most favorable market conditions (£1.50 per bushel of wheat and £3.00 per bushel of barley), the farmer stores them in a barn that has capacity for 5'000 bushels. An acre cultivated with wheat produces 115 bushels, whereas an acre allocated to barley produces 35 bushels. Knowing that the farmer currently has £16'000 available for expenses, how should they plant their field?

2. Create your own (2 dimensional) linear programming problem and solve it using the graphical method. Then, write your example in standard form.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 3&4 - Solving linear programming problems in Matlab and Compressed Sensing

Dr Neslihan Suzen

Lecture outline

This lecture's goal is to learn:

- How to solve linear programming problems in Matlab using `linprog`.
- How to solve a compressed sensing problem using linear programming

A previous example - formulation

Mathematically, we usually write the problem as follows:

Maximise:

$$f(x, y) = 80x + 120y$$

subject to:

$$\begin{array}{ccccc} x & + & 2y & \leq & 6 \\ 8x & + & 7y & \leq & 28 \\ x & & & \leq & 3 \\ x & & & \geq & 0 \\ & & y & \geq & 0 \end{array}$$

Maximise:

$$f(x, y) = (80\ 120) \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

subject to:

$$\begin{pmatrix} 1 & 2 \\ 8 & 7 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 6 \\ 28 \\ 3 \end{pmatrix}$$
$$x, y \geq 0$$

This is a *linear programming problem* because both the objective and the constraints can be expressed with linear functions.

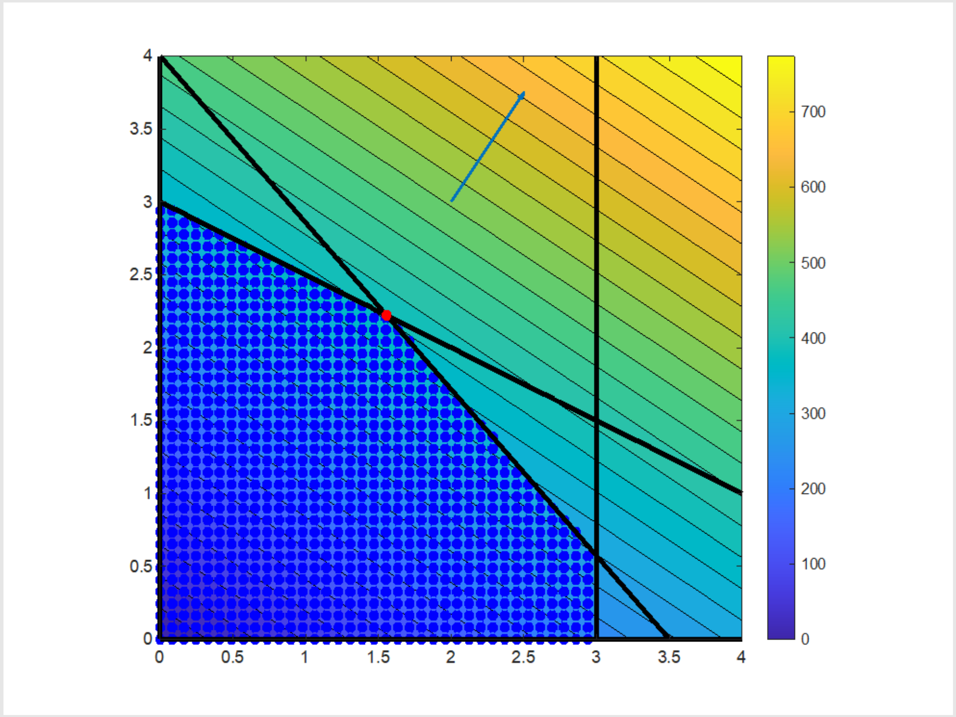
A previous examplem– graphical solution

Maximize:

$$f(x,y) = (80\ 120)\cdot\begin{pmatrix}x\\y\end{pmatrix}$$

subject to:

$$\begin{pmatrix}1 & 2\\8 & 7\\1 & 0\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}\leq\begin{pmatrix}6\\28\\3\end{pmatrix}$$
$$x,y \geq 0$$



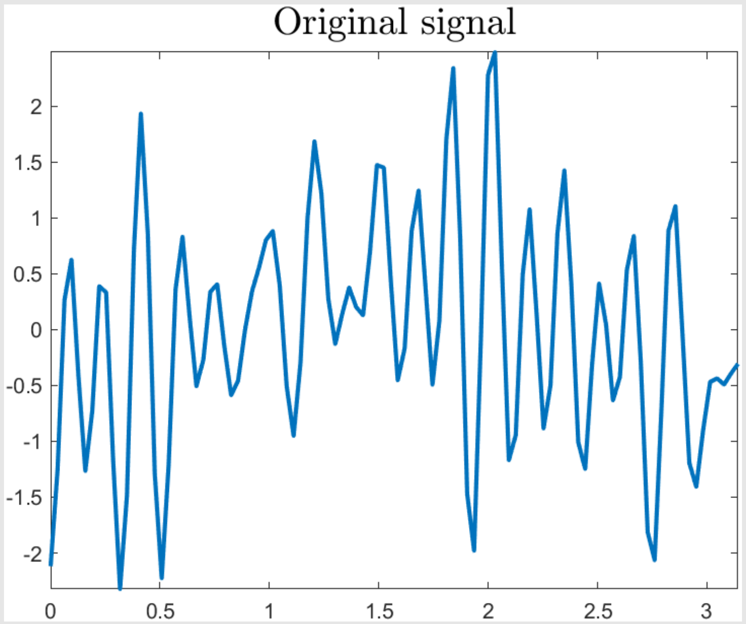
An example – solve in Matlab with linprog
(see matlab's documentation - OR03_linprog.m)

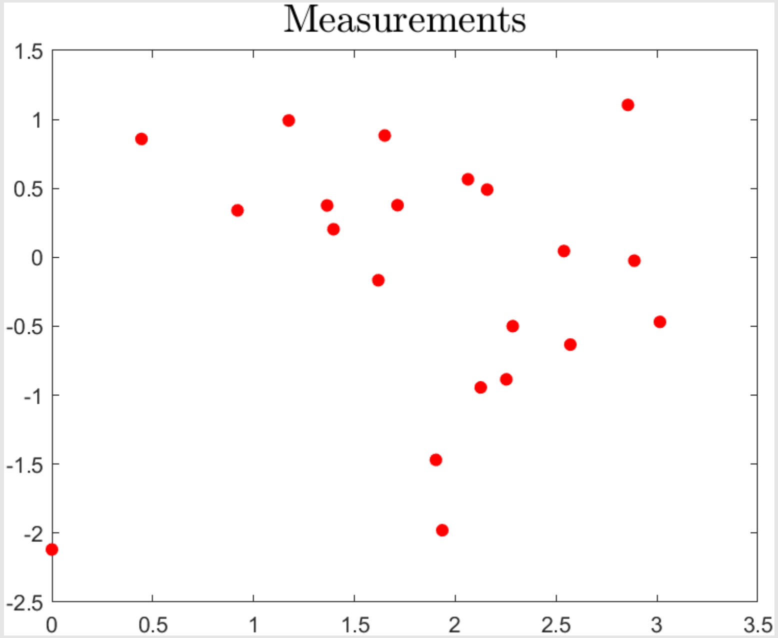
```
% set the problem specs
f = -1*[80;120];           % objective function to minimize
A = [1 2; 8 7; 1 0];       % inequality constraints matrix
b = [6 28 3];              % inequality constraints rhs
Aeq = [];                  % equality constraints matrix
beq = [];                  % inequality constraints rhs
lb = zeros(size(f));       % lower bounds
ub = [];                   % upper bounds

% solve the problem
[x,fval,exitflag,output] = linprog(f,A,b,Aeq,beq,lb,ub);
```

Compressed sensing – an example

Scenario: Your friend is sending you a signal through the aether, but your aether decoder is currently malfunctioning and can only take a discrete number of measurements/samples.





Question: Can you retrieve the signal from these measurements?

Compressed sensing - questions

Question: Can you retrieve the signal from these measurements?

Answer: In general no, but if you know how the signal is generated and you have enough measurements, then you stand a chance.

Assumptions: Your friend tells you that the signal is a linear combination of cosine functions. More precisely, your friend says it is given by

$$f(t) = \sum_{k=1}^{50} w_k \cos(kt),$$

and that very few of the weights w_k are nonzero.

Good news: We can retrieve f by solving a compressed sensing problem.

Compressed sensing – mathematical model

More precisely, we "just" need to solve

$$minimise\ \|w\|_{\ell_1}\ s.t.\ Aw = y, w \in \mathbb{R}^{50},$$

where

- $(t_i, y_i)_{i=1}^{m}$ are the measurements, that is, $y_i = f(t_i), i = 1, \dots, m,$
- and $A \in \mathbb{R}^{m,50}$, $A_{ik} = \cos(t_i k)$.

As seen in OR Lecture 2.pptx, this can be modelled as

$$\begin{array}{rc} min & z_1 + z_2 + \cdots + z_{50} \\ s.t. & -z \leq w \leq z \\ & Aw = y \\ & w \in \mathbb{R}^{50}, z \geq 0 \end{array}$$

See `OR04_compressed_sensing.m`

Summary and self-study

Summary: today we have learnt

- how to solve linear programming problems in matlab using `linprog`.
- how to tackle a compressed sensing problem via linear programming.

Self-study:

- Solve the self-study problems on the last slide of the slides OR Lecture 2_linear_programming.pptx in Matlab using `linprog`.
- In `OR04_compressed_sensing.m`, experiment with the parameters `number of modes, number of nonzero weights, number of measurements`. What do you observe?

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 5&6 – Feasibility and duality in linear programming

Dr Neslihan Suzen

Recap and lecture outline

Summary: so far we have learnt:

- how to solve simple (2 dimensional) linear programming problems using a graphical method
- that linear programming problems can be written in many equivalent formulations, and that one of them is know as standard form
- that some nonlinear functions such as max, min, abs, $||x||_1$, and $||x||_\infty$ can be described using linear functions.
- how to solve linear programming problems in matlab.

Today: Duality theory in linear optimisation, following closely chapters 2.3 and 2.4 of the Mosek Cookbook.

Feasible set

Definition: The *feasible set* of the linear programming problem

$$\begin{aligned} min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

is defined as

$$F_p := \{ x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}$$

and is a convex polytope.

A linear programming problem is *feasible* if $F_p \neq \emptyset$. Otherwise, the problem is *infeasible*.

Example of an infeasible problem

The following linear programming problem

$$\begin{array}{ll} min & (2,3,-1)^T \cdot x \\ s.t. & \begin{pmatrix} 1 & 1 & 2 \\ -4 & -2 & 2 \\ -10 & 0 & 50 \end{pmatrix} x = \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} \\ & x \geq 0 \end{array}$$

is infeasible because multiplying the equality constraint from the left with the vector $y = (1,1,-0.1)^T$ leads to

$$(1,1,-0.1)\begin{pmatrix} 1 & 1 & 2 \\ -4 & -2 & 2 \\ -10 & 0 & 50 \end{pmatrix} x = (-2,-1,-1)^T \cdot x = 0.1 = (1,1,-0.1)\begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix},$$

which is impossible if $x \geq 0$.

Farkas' lemma

Lemma: Let $A \in \mathbb{R}^{m,n}$ and $b \in \mathbb{R}^m$. Then, exactly one of the following is true:

1. There exists $x \geq 0$ such that $Ax = b$.
2. There exists y such that $y^T A \leq 0$ and $y^T b > 0$.

Proof: Let $C = \{Ax \mid x \geq 0\}$ be the closed convex cone spanned by the columns of A. If $b \in C$, then the first alternative is true. Otherwise, standard (but not trivial) hyperplane separation theorems imply the existence of a hyperplane passing through the origin that separates C and $\{b\}$. Let y be a vector normal to this hyperplane. Then, without loss of generality, $y^T b > 0$ and $y^T z \leq 0$ for all $z \in C$, which in turn implies $y^T A \leq 0$. Finally, statements 1. and 2. cannot be true at the same time, otherwise

$$0 < y^T b = y^T(Ax) = (y^T A)x \leq 0,$$

which is a contradiction. $\square$

Duality – primal problem

Primal problem: We consider the following liner programming problem in standard form.

$$\begin{array}{rc} min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array}$$

By convention, the optimal objective value p^* is either:

- $p^* = +\infty$ if the problem is infeasible,
- finite if the problem has an optimal solution,
- $p^* = -\infty$ if the problem is unbounded.

Example: the problem $minimize\ -x\ s.t.\ x \geq 0$ is unbounded.

Duality – Lagrange function

Definition: To the primal problem

$$\begin{array}{ll} min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array}$$

we associate the *Lagrangian* function $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n_+ \rightarrow \mathbb{R}$ defined by

$$L(x, y, s) = c^T x + y^T (b - Ax) - s^T x.$$

The variables $y \in \mathbb{R}^m$ and $s \in \mathbb{R}^n_+$ are called *Lagrange multipliers* or *dual variables*.

Note that, for any feasible $x \in F_p$ and any $(y, s) \in \mathbb{R}^m \times \mathbb{R}^n_+$, we have

$$\begin{aligned} L(x, y, s) &= c^T x + y^T (b - Ax) - s^T x \\ &= c^T x + y^T 0 - s^T x \leq c^T x \end{aligned}$$

Duality – dual function

Definition: The dual function $g: \mathbb{R}^m \times \mathbb{R}^n_+ \to \mathbb{R}$ of the primal problem

$$\begin{array}{ll} min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array}$$

is defined as

$$\begin{aligned} g(y, s) &:= \min_x L(x, y, s) \\ &= \min_x c^T x + y^T (b - Ax) - s^T x \\ &= \min_x x^T (c - A^T y - s) + b^T y \\ &= \begin{cases} b^T y, & if \quad c - A^T y - s = 0, \\ -\infty, & if \quad c - A^T y - s \neq 0. \end{cases} \end{aligned}$$

Duality – dual problem

Since $g(y, s) \leq p^*$ for any $(y, s) \in \mathbb{R}^m \times \mathbb{R}^n_+$, the optimal dual objective value d^* of the *dual problem*

$$\begin{array}{ll} max & b^T y \\ s.t. & c - A^T y = s \\ & s \geq 0, y \in \mathbb{R}^m \end{array}$$

is the best lower bound of the optimal objective value p^* of the primal problem.

Remark: the optimal dual objective value d^* is either:

- $d^* = -\infty$ if the problem is infeasible,
- finite if the dual problem has an optimal solution,
- $d^* = +\infty$ if the problem is unbounded.

Duality – example 1/2

Consider the primal problem

$$\begin{aligned} max \quad & (80\ 120) \cdot \begin{pmatrix} x \\ y \end{pmatrix} \\ s.t. \quad & \begin{pmatrix} 1 & 2 \\ 8 & 7 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} 6 \\ 28 \\ 3 \end{pmatrix} \\ & x, y \geq 0 \end{aligned}$$

which in standard form is

$$\begin{aligned} -min \quad & (-80 \quad -120 \quad 0 \quad 0 \quad 0)\, z \\ & \begin{pmatrix} 1 & 2 & 1 & 0 & 0 \\ 8 & 7 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} z = \begin{pmatrix} 6 \\ 28 \\ 3 \end{pmatrix} \\ & z \geq 0 \end{aligned}$$

Let $-p^*$ denote the optimal objective value.

Duality – example 2/2

The Lagrangian associated to

$$\begin{gathered} \min \quad \begin{pmatrix}-80 & -120 & 0 & 0 & 0\end{pmatrix} z \\ \begin{pmatrix}1 & 2 & 1 & 0 & 0\\ 8 & 7 & 0 & 1 & 0\\ 1 & 0 & 0 & 0 & 1\end{pmatrix} z = \begin{pmatrix}6\\28\\3\end{pmatrix} \\ z \geq 0 \end{gathered}$$

is

$$L(z,y,s) = \begin{pmatrix}-80 & -120 & 0 & 0 & 0\end{pmatrix} z + y^T \left(\begin{pmatrix}6\\28\\3\end{pmatrix} - \begin{pmatrix}1 & 2 & 1 & 0 & 0\\ 8 & 7 & 0 & 1 & 0\\ 1 & 0 & 0 & 0 & 1\end{pmatrix} z \right) - s^T z,$$

and the dual problem is

$$\begin{gathered} \max \quad \begin{pmatrix}6 & 28 & 3\end{pmatrix} y \\ \begin{pmatrix}-80\\-120\\0\\0\\0\end{pmatrix} - \begin{pmatrix}1 & 8 & 1\\ 2 & 7 & 0\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{pmatrix} y = s \\ s \geq 0, y \in \mathbb{R}^3 \end{gathered}$$

- To be continued...

Duality – the dual of the dual 1/3

The dual of

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array}$$

is

$$\begin{array}{ll} \max & b^T y \\ s.t. & c - A^T y = s \ , \\ & s \geq 0, y \in \mathbb{R}^m \end{array}$$

which, we can also write as

$$\begin{array}{ll} \max & b^T y \\ s.t. & c - A^T y \geq 0 \\ & y \in \mathbb{R}^m \end{array}$$

Duality – the dual of the dual 2/3

To derive the dual of the dual problem

$$\begin{array}{ll} max & b^T y \\ s.t. & c - A^T y \geq 0 \\ & y \in \mathbb{R}^m \end{array}$$

we introduce the Lagrangian

$$L: \mathbb{R}^m \times \mathbb{R}^n_+ \to \mathbb{R}, \qquad L(y,x) = b^T y + x^T (c - A^T y)$$

which, for any feasible $y \in F_d$, satisfies $L(y,x) \geq b^T y$.

Duality – the dual of the dual 3/3

Given the Lagrangian

$$L:\ \mathbb{R}^m \times \mathbb{R}_+^n \to \mathbb{R}, \qquad L(y,x) = b^T y + x^T(c - A^T y),$$

the dual function is

$$\begin{aligned} f(x) &:= \max_y L(y,x) = \max_y b^T y + x^T(c - A^T y) \\ &= \max_y y^T(b - Ax) + c^T x = \begin{cases} c^T x, & if \quad Ax = b, \\ \infty, & if \quad Ax \neq b. \end{cases} \end{aligned}$$

and the best upper bound of d^* is the optimal objective value of

$$\begin{array}{rc} min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array}$$

which is the original primal problem.

UNIVERSITY OF LEICESTER

www.le.ac.uk

Weak duality

Lemma: Let p^* and d^* denote the optimal objective value of

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \quad \text{and} \quad \begin{array}{ll} \max & b^T y \\ s.t. & c - A^T y = s \\ & s \geq 0, y \in \mathbb{R}^m \end{array}$$

respectively. Then, $d^* \leq p^*$.

Proof: Let $x \in F_p$ and $(y, s) \in F_d$. Then,

$$b^T y = (Ax)^T y = x^T (A^T y) = x^T (c - s) = c^T x - s^T x \leq c^T x.$$

Since this is true for any $x \in F_p$ and $(y, s) \in F_d$, it follows that $d^* = \max_{y \in F_d} b^T y \leq \min_{x \in F_p} c^T x = p^*$. $\square$

Corollary: if $x \in F_p$ and $(y, s) \in F_d$ are such that $b^T y = c^T x$, then x and (y, s) are optimal solutions to the primal and dual problems, respectively.

Strong duality 1/3

Lemma: Let p^* and d^* denote the optimal objective value of

$$\begin{array}{ll} min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad \text{and} \qquad \begin{array}{ll} max & b^T y \\ s.t. & c - A^T y = s \\ & s \geq 0, y \in \mathbb{R}^m \end{array}$$

respectively. If at least one p^* and d^* is finite, then $d^* = p^*$.

Proof: Assume that p^* is finite and let $\varepsilon > 0$. Since p^* is optimal, the following linear system of equations has no solutions

$$\begin{pmatrix} -c^T \\ A \end{pmatrix} x = \begin{pmatrix} -p^* + \varepsilon \\ b \end{pmatrix}.$$

By Farkas' lemma, there is a vector $z \in \mathbb{R}^{n+1}$ such that

$$(-c, A^T)z \leq 0 \qquad \text{and} \qquad (-p^* + \varepsilon, b^T)z > 0.$$

Strong duality 2/3

Lemma: Let p^* and d^* denote the optimal objective value of

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad \text{and} \qquad \begin{array}{ll} \max & b^T y \\ s.t. & c - A^T y = s \\ & s \geq 0, y \in \mathbb{R}^m \end{array}$$

respectively. If at least one p^* and d^* is finite, then $d^* = p^*$.

Proof: By Farkas' lemma, there is a vector $z \in \mathbb{R}^{n+1}$ such that

$$(-c, A^T)z \leq 0 \qquad \text{and} \qquad (-p^* + \varepsilon, \mathrm{b}^T)z > 0$$

Denote $z = (y_0, y)$, with $y_0 \in \mathbb{R}$ and $y \in \mathbb{R}^n$. Then, since the primal problem is feasible, we conclude that $y_0 \neq 0$. Without loss of generality, we set $y_0 = 1$. Then,

$$A^T y - c \leq 0 \qquad \text{and} \qquad b^T y - p^* + \varepsilon > 0,$$

that is, $c - A^T y \geq 0$ and $b^T y \geq p^* - \varepsilon$.

Strong duality 3/3

Lemma: Let p^* and d^* denote the optimal objective value of

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad \text{and} \qquad \begin{array}{ll} \max & b^T y \\ s.t. & c - A^T y = s \\ & s \geq 0, y \in \mathbb{R}^m \end{array}$$

respectively. If at least one p^* and d^* is finite, then $d^* = p^*$.

Proof: We conclude that

$$c - A^T y \geq 0 \text{ and } b^T y \geq p^* - \varepsilon.$$

The first inequality implies that $y \in F_d$, and letting $\varepsilon \to 0$ in the second one implies

$$b^T y \geq p^*.$$

Hence, $d^* \geq p^*$. Since weak duality implies $d^* \leq p^*$, we conclude that $d^* = p^*$.

The proof starting with d^* being feasible is analogous. $\square$

Primal and dual Farkas' lemma

Lemma: For the primal-dual pair of linear programming problems

$$\begin{array}{ll} \min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad \text{and} \qquad \begin{array}{ll} \max & b^T y \\ s.t. & c - A^T y = s \\ & s \geq 0, y \in \mathbb{R}^m \end{array}$$

the following equivalences hold:

1. The primal problem is infeasible iff there is y such that $A^T y \leq 0$ and $b^T y > 0$.
2. The dual problem is infeasible iff there is $x \geq 0$ such that $Ax = 0$ and $c^T x < 0$.

Summary and self-study

Summary: we have learnt
- that not all linear programming problems are feasible,
- how to derive the dual problem of a primal problem,
- *weak duality*: the optimal dual objective is a lower bound on the primal one,
- *strong duality*: if the primal or the dual are feasible, the bound is sharp.

Self-study: Write the following linear programming problem in standard form.

$$\begin{array}{ll} min & 30x + 21y + 18z \\ s.t. & 3x - 7z \leq 176 \\ & 8z - 2y + x - 6 \geq 12 \\ & 4x + 3y = 19 \\ & x, y, z \in \mathbb{R} \end{array}$$

Then, derive its dual problem identifying clearly the Lagrangian and the dual function.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 7 – Shadow prices and sensitivity analysis

Dr Neslihan Suzen

Recap and lecture outline

Summary: in previous lectures we learnt:

- that not all linear programming problems are feasible,
- how to derive the dual problem of a primal problem,
- weak duality: the optimal dual objective is a lower bound on the primal one,
- strong duality: if the primal or the dual is feasible and p^* or d^* is finite, then the bound is sharp.

Today: Shadow prices and sensitivity analysis, following closely chapter 2.4 of the Mosek Cookbook.

Shadow prices 1/2

Consider the primal-dual pair of linear programming problems

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned} \qquad \text{and} \qquad \begin{aligned} \max \quad & b^T y \\ s.t. \quad & c - A^T y = s \\ & s \geq 0, y \in \mathbb{R}^m \end{aligned}$$

Question: if b is subject to a small perturbation δb, that is, $b \rightarrow b + \delta b$, how does this perturbation affect p^*?

Answer: Assuming the primal-dual pair is feasible, denote by (x^*, y^*, s^*) the primal-dual optimal solution. A perturbation in b affects F_p, but not F_d. Assuming that (y^*, s^*) was a unique vertex of F_d, if δb is small enough, then (y^*, s^*) remains also an optimal solution to the perturbed problem

$$\begin{aligned} \max \quad & (b + \delta b)^T y \\ s.t. \quad & c - A^T y = s \\ & s \geq 0, y \in \mathbb{R}^m \end{aligned}$$

Shadow prices 2/2

Question: if b is subject to a small perturbation δb, that is, $b \to b + \delta b$, how does this perturbation affect p^*?

Answer: The dual solution (y^*, s^*) remains also an optimal solution to the perturbed problem

$$\begin{array}{ll} max & (b+\delta b)^T y \\ s.t. & c - A^T y = s \\ & s \geq 0, y \in \mathbb{R}^m \end{array}$$

and its optimal dual objective d_N^* value is

$$d_N^* = (b+\delta b)^T y^* = b^T y^* + \delta b^T y^* = d^* + \delta b^T y^* = p^* + \delta b^T y^* = p_N^*$$

where p_N^* denotes the optimal objective value of the perturbed primal problem.

Take home message: The dual variable y^* quantifies how sensitive p^* is with respect to perturbations on b.

Shadow prices – example

(see `OR07_shadow_prices.m`**)**

Consider the linear programming problem

$$\begin{aligned} max \qquad & (80\ 120)\cdot\begin{pmatrix}x\\y\end{pmatrix} \\ s.t. \qquad & \begin{pmatrix}1 & 2\\ 8 & 7\\ 1 & 0\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix} \leq \begin{pmatrix}6\\28\\3\end{pmatrix} \\ & x, y \ \geq \ 0 \end{aligned}$$

We can solve this problem in Matlab and read the value of the dual variables with

```
[x, fval, exitflag, output, lambda] =
    linprog(f,A,b,Aeq,beq,lb,ub);
```

which returns $x \cong (1.56, 2.22)$, $p^* \cong 391$, $y \cong (44, 4.4, 0)$. This implies that, for instance, if $\delta b = (1,0,0)^T$, then $p_N^* \cong 391 + 44 = 435$.

Sensitivity analysis – pertubations of f

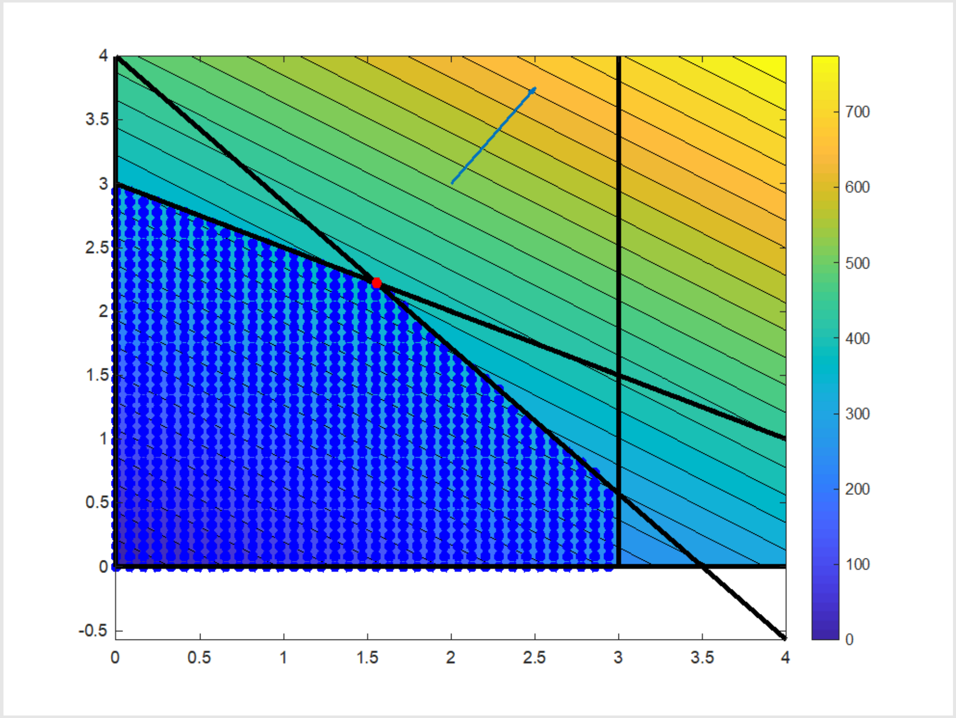
Maximise:

$$f(x,y) = (80\ 120) \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

subject to:

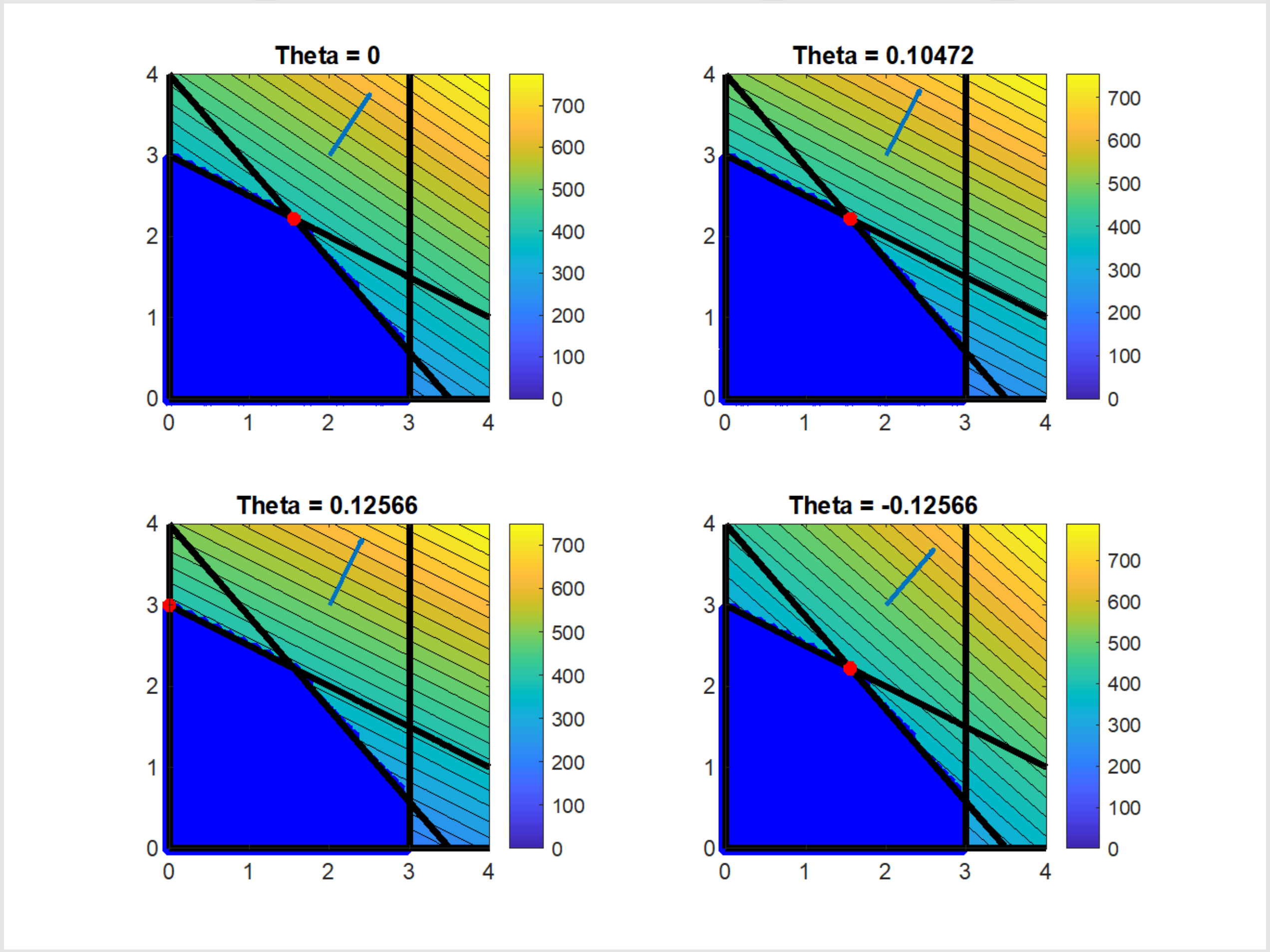
$$\begin{pmatrix} 1 & 2 \\ 8 & 7 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \le \begin{pmatrix} 6 \\ 28 \\ 3 \end{pmatrix}$$
$$x, y \ge 0$$

Question: how does the solution change if we perturb f?



Perturbing the objective function

(see `OR07_sensitivity_analysis.mlx` an `OR07_feasible_region.m`)



Performing sensitivity analysis

Unfortunately, Matlab's `linprog` does not perform sensitivity analysis. There are alternative toolboxes e.g., Mosek, but setting them up can be technical and it's beyond the scope of this lecture.

Instead, sensitivity analysis in R or MS Excel is straightforward. On Blackboard you can find an R-script to perform sensitivity analysis (based on `lpsolve`). If you prefer MS Excel instead, here is a good tutorial. For this example, we get:

Objective's coefficients	Sensitivity analysis – lower bound	Sensitivity analysis – upper bound
80	60	137
120	70	160

Finally, for an interesting critique of sensitivity analysis, read S. W. Wallace.

Summary and self-study

Summary: today we have learnt

- the interpretation of the dual variables as shadow prices,

- how to retrieve the dual variables from `linprog`,

- how perturbations in the objective function can affect the location of the optimum.

Self-study: Analyze the shadow prices of your own linear programming problem. For example, you could use the problem for self-study section in `OR Lecture 2_linear_programming.pptx`.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 8 – Mixed-integer linear problems

Dr Neslihan Suzen

Recap and plan of the day

Summary: so far we learnt:

- how to model linear programming problems and solve them in Matlab,
- how to analyze them using Farkas' lemma,
- the role (and interpretation) of weak and strong duality and sensitivity analysis.

Today: A first glimpse into mixed-integer programming following loosely Ch. 3 of OptArt and Ch. 9 of the book by Hillier and Lieberman.

A&E unit – problem statement

Scenario: Evening shift doctors in the A&E department of a hospital work five consecutive days and have two consecutive days off.

Their five days of work can start on any day of the week and the schedule rotates indefinitely.

The hospital requires the following minimum number of doctors working each day:

Mon	Tue	Wed	Thu	Fri	Sat	Sun
45	50	61	40	60	50	26

Finally, no more than 35 doctors can start their five working days on the same day.

Question: Find a rota to minimize the number of A&E doctors employed.

A&E unit – mathematical model

Let $x_1, \dots, x_7$ denote the how many doctors start on Monday, ..., Sunday. Then, we can model the problems as follows:

$$\begin{array}{rc} min & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 \\ s.t. & x_1 + x_4 + x_5 + x_6 + x_7 \geq 45 \\ & x_1 + x_2 + x_5 + x_6 + x_7 \geq 50 \\ & x_1 + x_2 + x_3 + x_6 + x_7 \geq 61 \\ & x_1 + x_2 + x_3 + x_4 + x_7 \geq 49 \\ & x_1 + x_2 + x_3 + x_4 + x_5 \geq 60 \\ & x_2 + x_3 + x_4 + x_5 + x_6 \geq 50 \\ & x_3 + x_4 + x_5 + x_6 + x_7 \geq 26 \\ & 0 \leq x \leq 35, x \in \mathbb{Z}^7 \end{array}$$

This is a (mixed-)integer linear programming problem.

Solving this example in Matlab with `intlinprog`

(see `OR08_branch_and_bound.m`)

```
f = ones(1,7);
intcon = 1:7;
v = [1 0 0 1 1 1 1];
A = -toeplitz(v([1, 7:-1:2]), v);  % create a circulant
matrix
b = -[45,50,61,49,60,50,26];
lb = zeros(size(f));
ub = 35*ones(1,7);
[x,fval] = intlinprog(f,intcon,A,b,[],[],lb,ub);
```

Matlab's solution is $x = (22,5,22,0,11,12,0)$. Another solution is $x = (22,24,3,11,0,12,0)$.

Mixed-integer optimisation problems are difficult

More precisely, they are *NP hard*, meaning that we can verify that a solution is correct in polynomial time, and we do not know whether we can solve them with a deterministic algorithm in polynomial time (however, if you find out how to do it, you will become very rich and famous).

The beautiful duality theory we developed for linear programming problems does not apply.

In many practical cases, an exact solution cannot be found in reasonable time.

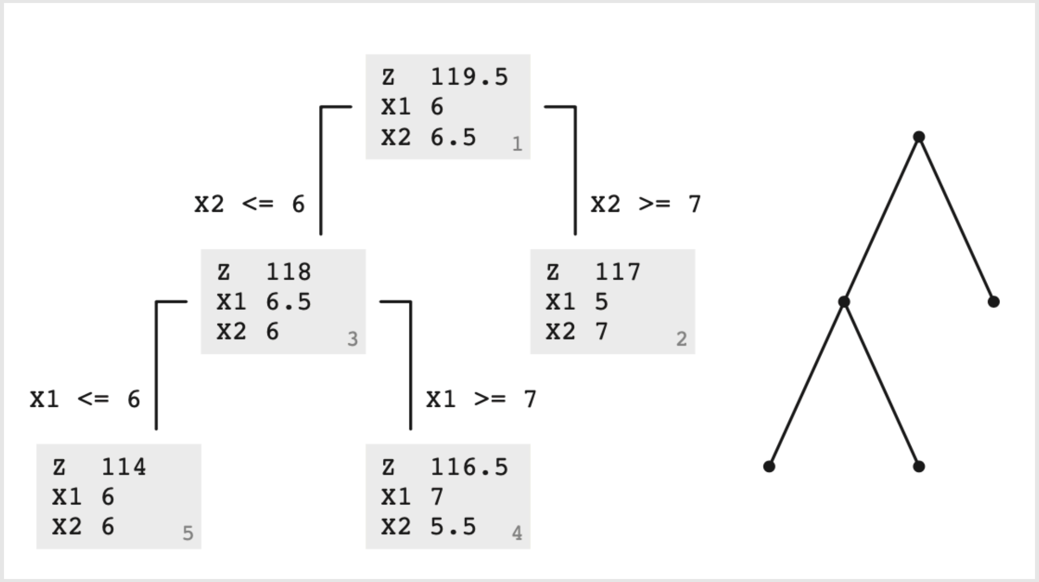
To find out what Matlab's `intlinprog` does, see this documentation page.

The branch and bound method

(see `OR08_branch_and_bound.m`)

Consider the mixed-integer linear programming problem

$$\max 8x + 11y \;\; s.t. \begin{cases} 2x + 2y \leq 25 \\ x + 2y \leq 19 \\ x, y \geq 0, x, y \in \mathbb{N} \end{cases}$$



Branch and bounding the A&E unit problem

Question: What happens if we solve the relaxed version of the A&E problem?

Answer: We get the same solution!!! But why? This was a bit of a coincidence. Indeed, the relaxed problems with `b = -[46,50,61,49,60,50,26];` gives a non-integer (relaxed) solution.

However, sometimes one always gets an integer solution to the relaxed problem. This is known as integer solution property.

Definition: A matrix $A \in \mathbb{R}^{m,n}$ is called totally unimodular if every square non-singular submatrix of A has determinant ± 1.

Theorem: Consider the integer programming problem

$$\max c^T x \;\; s.t \; Ax = b, x \geq 0, x \in \mathbb{Z}^n.$$

If the matrix A is totally unimodular and b is integer valued, then every extreme point of this integer programming problem is integer valued.

Remark: verifying that a matrix is totally unimodular is not always easy, but there are some tricks.

Example with integer solution property

Transportation problem: you employ two artisans who produce chairs and you have three customers who need chairs. Artisan 1 produced 10 chairs and artisan 2 produced 20 chairs. Customer 1 needs 6 chairs, customer 2 needs 8 chairs, and customer 3 needs 16 chairs. The cost to ship a chair from each artisan to any of the three customers is summarized in the following table:

	Customer 1	Customer 2	Customer 3
Artisan 1	£14	£11	£12
Artisan 2	£10	£12	£18

Question: To whom should artisans 1 and 2 ship their chairs to minimize the shipping costs for your company?

Transportation problem - model

Let x_{ij} denote how many chairs artisan i sends to customer j. Then, what we want to solve is

$$\min 14x_{11} + 11x_{12} + 12x_{13} + 10x_{21} + 12x_{22} + 18x_{23}$$

subject to

$$\begin{aligned} x_{11} + x_{21} &= 6, \\ x_{12} + x_{22} &= 8, \\ x_{13} + x_{23} &= 16, \\ x_{11} + x_{12} + x_{13} &= 10, \\ x_{21} + x_{22} + x_{23} &= 20 \end{aligned}$$

and $x_{ij} \geq 0, x_{ij} \in \mathbb{Z}$.

Transportation problem – totally unimodular

The constraints

$$\begin{aligned} x_{11} + x_{21} &= 6, \\ x_{12} + x_{22} &= 8, \\ x_{13} + x_{23} &= 16, \\ x_{11} + x_{12} + x_{13} &= 10, \\ x_{21} + x_{22} + x_{23} &= 20 \end{aligned}$$

can be written as

$$Ax = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \end{pmatrix} = \begin{pmatrix} 6 \\ 8 \\ 16 \\ 10 \\ 20 \end{pmatrix} = b$$

The matrix A is totally unimodular!

Summary and self-study

Summary: today we have learnt

- how to model some mixed-integer linear programming problems,
- how to solve them in Matlab using `intlinprog`,
- and what the branch and bound technique is.

Self-study: Formulate the following problem as mixed-integer linear programming problem.

Scenario: A colleague of mine convenes the module MA3513 Industrial Mathematics Project. In this module, students work in groups for one year on a project set by an industrial partner.
This summer my colleague collected 7 potential projects and now he must allocate students to projects. This year there are 32 students and he would like to let them choose what project they'd like to work on. To this end, he asked them to rank their top-three preferred projects.

Question: How should he allocate students to projects to maximise their happiness while at the same time ensuring that no project has fewer than 4 or more than 5 students?

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 9 – Integer modelling

Dr Neslihan Suzen

Recap and lecture outline

Summary: we have learnt:

- how to model some mixed-integer linear programming problems,
- how to solve them in Matlab using `intlinprog`,
- what the branch and bound technique is,
- and that some mixed-integer problems have a so-called integer solution property.

Today: More modelling using integer variables, following closely chapter 9.1 of the Mosek Cookbook.

Mixed-integer linear modelling

A general *mixed-integer linear modelling problem* takes the form

$$\begin{array}{ll} min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \\ & x_i \in \mathbb{Z} \text{ for } i \in I \end{array}$$

where the set $I \subseteq \{1, \ldots n\}$ specifies which components of x must be integers.

There are wo major modelling techniques:

1. *binary variables* (aka *indicator variables*) take values in $\{0,1\}$ and indicates the absence or presence of a particular event or choice. This can be model by
$$0 \leq x \leq 1 \text{ and } x \in \mathbb{Z}.$$
2. *big-M* conditions: some relations can be modelled linearly only by assuming a fixed bound M on the quantities involved.

Implication of positivity

If $0 \leq x \leq M$ for some $M \in \mathbb{R}$, we can model the conditional statement

$$z = 1 \text{ if } x > 0$$

by setting

$$x \leq Mz \text{ and } z \in \{0,1\}.$$

Note: this only works if we know for sure that $0 \leq x \leq M$. Otherwise, the problem can become infeasible.

Implication of positivity - example

The cost c_i to produce x_i exemplars of a certain item i is often affine, that is,

$$c_i = a_i x_i + b_i$$

where a_i is the material/energy cost to produce a unit of the item i and b_i is an initial investment (such as the purchase of specific equipment).

Example: Lemons are cheaper than oranges, but the lemon press is more expensive. Minimising the cost to produce L liters of juice taken from M_i lemons/oranges available takes the form

$$\begin{array}{lc} min & a^T x + b^T z \\ s.t. & r^T x = L \\ & x \le Mz \\ & x \ge 0, z \in \{0,1\}^2 \end{array}$$

where r_i is the juice extraction amount per lemon/orange.

Semi-continuous variable, indicator constraints

Semi-continuous variables: Let $a, b \in \mathbb{R}, 0 < a \leq b$. The condition $x \in 0 \;\cup [a, b]$ can be modeled by

$$az \leq x \leq bz, \text{and } z \in \{0,1\}.$$

Indicator constraints: Let $a \in \mathbb{R}^n, z \in \{0,1\},\; b, M \in \mathbb{R}, M > b$. The conditions

$$a^T x \leq b \text{ if } z = 1 \text{ and } a^T x \leq M \text{ otherwise}$$

can be modelled as

$$a^T x \leq b + (M - b)(1 - z)$$

Note: if x is bounded (say $|x| \leq m$), then picking $M \;= m^T|a|$ means we do not impose any extra constraint on x if $z = 0$.

Disjunctive constraints

Let $a_1, \dots a_k \in \mathbb{R}^n,\ b \in \mathbb{R}^k$. If we want that at least one of the following constraints is satisfied,

$$a_1^T x \le b_1, \text{or } a_2^T x \le b_2, \dots, \text{or } a_k^T x \le b_k,$$

we may choose $M \in \mathbb{R}$ large enough and use the linear model

$$z_1 + \ \dots + z_k \ge 1$$
$$z \in \{0,1\}^k$$
$$a_i^T x \le b_i + (M - b_i)(1 - z_i), i = 1, \dots, k$$

Constraint satisfaction

If $m < b < M$ we can distinguish between the two options

$$\text{either } m \leq a^T x \leq b \text{ or } b \leq a^T x \leq M$$

with the linear model

$$b + (m - b)z \leq a^T x \leq b + (M - b)(1 - z), z \in \{0,1\}$$

Exact absolute value

Let $x \in \mathbb{R}$. In a previous lecture, we saw how to model $|x| \leq t$.

If $|x| \leq M$, we can model the exact equality $t = |x| \in \mathbb{R}$ as follows

$$\begin{gathered} x = a - b \\ t = a + b \\ a, b \geq 0 \in \mathbb{R} \\ a \leq Mz \\ b \leq M(1 - z) \\ z \in \{0,1\} \end{gathered}$$

Exact ℓ_1-norm

Let $x \in \mathbb{R}^n$. In a previous lecture, we saw how to model

$$\|x\|_{\ell_1} = \sum_{i=1}^{n} |x| \leq t.$$

We can model the exact equality $t = \|x\|_{\ell_1} \in \mathbb{R}$ as follows

$$\begin{gathered} x = a - b \\ \sum_{i=1}^{n} a_i + b_i = t \\ a, b \geq 0 \in \mathbb{R}^n \\ a \leq cz \\ b \leq c(1-z) \\ z \in \{0,1\}^n \end{gathered}$$

Boolean operators

Let $x, y, z \in \{0,1\}$. Then, we can model Boolean operators as follows:

$z = x\ OR\ y$	$x \leq z, y \leq z, z \leq x + y$
$z = x\ AND\ y$	$x \geq z, y \geq z, z + 1 \geq x + y$
$z = NOT\ x$	$z = 1 - x$
$x \Rightarrow y$	$x \leq y$
$At\ most\ one\ of\ x, y, z\ holds$	$x + y + z \leq 1$
$Exactly\ one\ of\ x, y, z\ holds$	$x + y + z = 1$
$At\ least\ one\ of\ x, y, z\ holds$	$x + y + z \geq 1$
$A\ most\ k\ of\ x, y, z\ hold$	$x + y + z \leq k$

Bilinear equality

Let $x, y \in \mathbb{R}$. The bilinear constraint $xy = 0$, which models the alternative

$$x = 0 \quad \text{or} \quad y = 0,$$

can be modelled as

$$\begin{gathered} |x| \leq Mz \\ |y| \leq M(1-z) \\ z \in \{0,1\} \end{gathered}$$

for a suitable constant M.

Summary and self-study

Summary: today we have learnt

- how to model some nonlinear functions using mixed-integer linear programming.

Self-study: Consider the self-study exercise from `OR Lecture 8_mixed_integer.pptx`, but this time assume that I have collected 10 projects instead of 7. How should I modify the corresponding mixed-integer linear programming problem? Note that I cannot run all 10 projects because only I have only 32 students and each project should have at least 4 students.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 10&11– Networks

Dr Neslihan Suzen

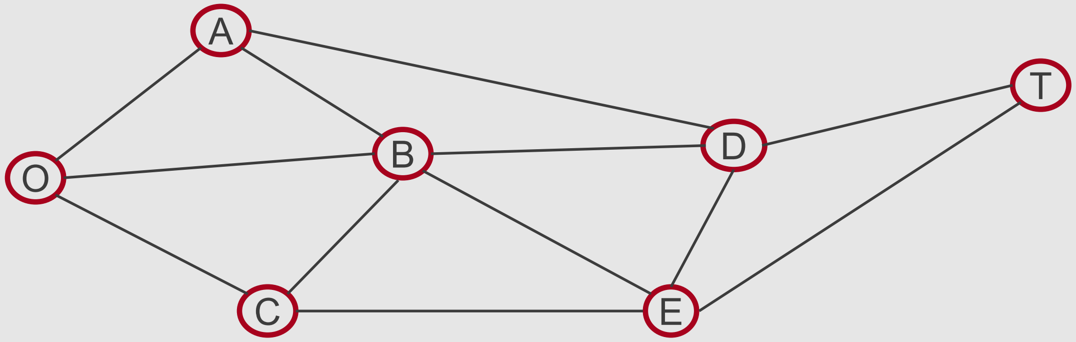
Recap and lecture outline

Summary: We have learnt:

- about sensitivity analysis
- how to model mixed-integer linear programming problems,
- and how to solve them in Matlab using `intlinprog`.

Today: Networks, following loosely Ch. 10 of the book by Hillier and Lieberman.

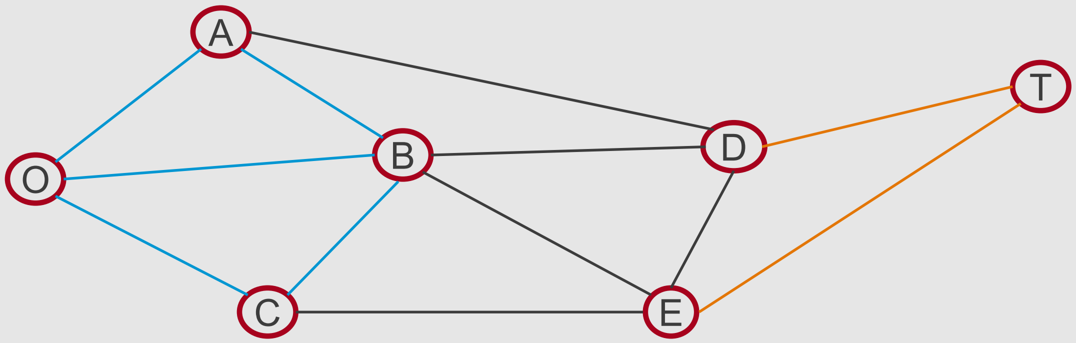
Graphs, nodes, and edges



Definition: A *graph* (V, E) is a set of *nodes* V (depicted as circles) and a set of *edges* E (depicted as lines and also known as links, arcs or branches) that connect certain pairs of nodes.

We assume that at most one edge connects a given pair of nodes and that edges are non-oriented. Then every edge is uniquely determined by the pair of nodes it connects.

Subgraphs and full subgraphs

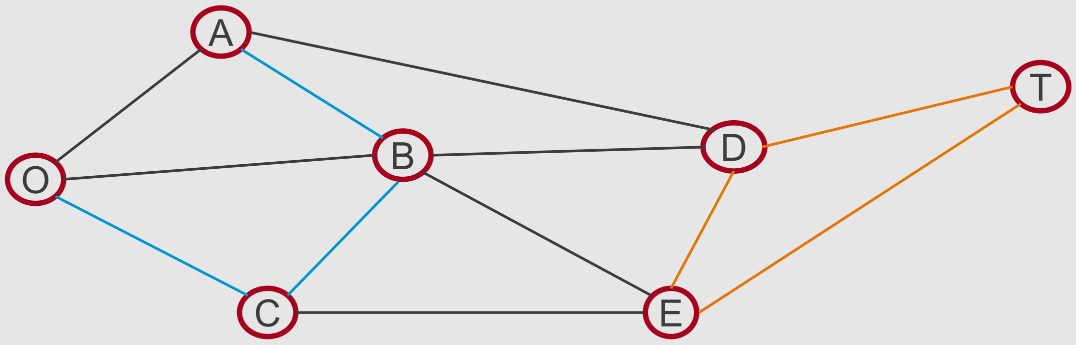


Definition: A *subgraph* $(V', E') \subset (V, E)$ is a graph with $V' \subset V$ and $E' \subset E$.

A subgraph is *full* if an edge connecting two nodes in V that are in V' is also in E' (in other words, if pairs of nodes enter the subgraph along with their edges).

Example: The blue subgraph is full, whereas the orange one is not.

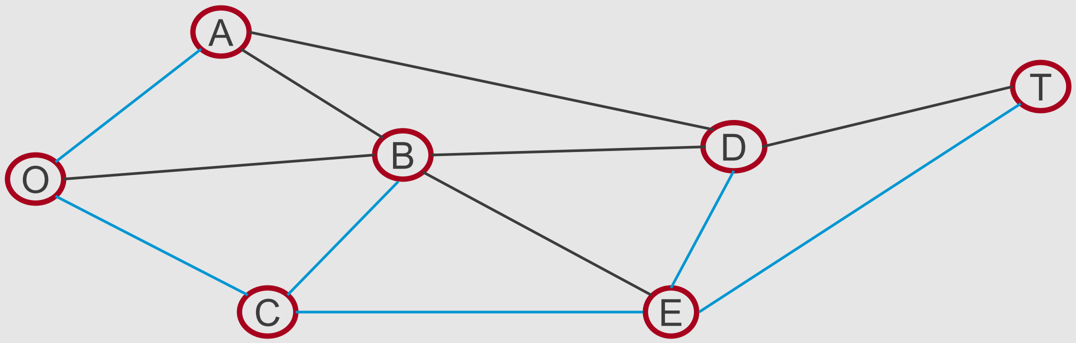
Paths and cycles



Definitions:

- Two edges are *connected* if adjacent to a common node
- A *path* is a (unique) sequence of connected edges with each node being adjacent to at most two edges in the path (e.g. blue line)
- A *cycle* is a path that begins and ends at the same node (e.g. orange line).

Connected graphs and trees



Definitions:

- A graph is connected if each pair of nodes is connected by a path.
- A connected graph is a tree if the connecting paths are unique.
- A spanning tree of a connected graph (V, E) is a subgraph (V', E') with $V' = V$ that is also a tree.

Euler formula for trees

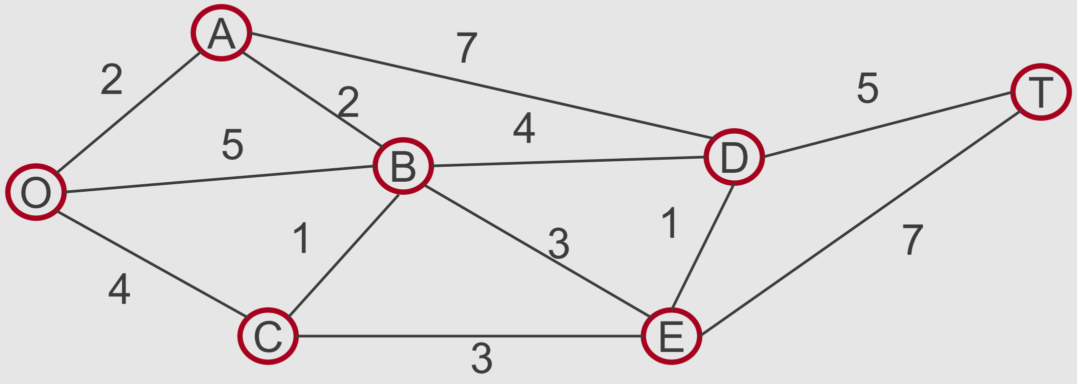
Proposition: Let (V, E) be a connected graph with $n \in \mathbb{N}$ nodes and $e \in \mathbb{N}$ edges. Then, (V, E) is a tree if and only if $n = e + 1$.

Proof:

(⇒) Let (V, E) be a tree with $n \in \mathbb{N}$ nodes. If $n = 2$, the statement is obviously true. If $n > 2$, remove from the tree a node with a single connection (so-called *leaf*) as well as its connection. The result is a tree with $n - 1$ nodes and $e - 1$ edges. Repeating this procedure $n - 1$ additional times (for a total of $n - 2$ removals) lead to a tree with 2 nodes, and necessarily at most one edge. This implies that the original tree had $n - 2 + 1 = n - 1$ edges.

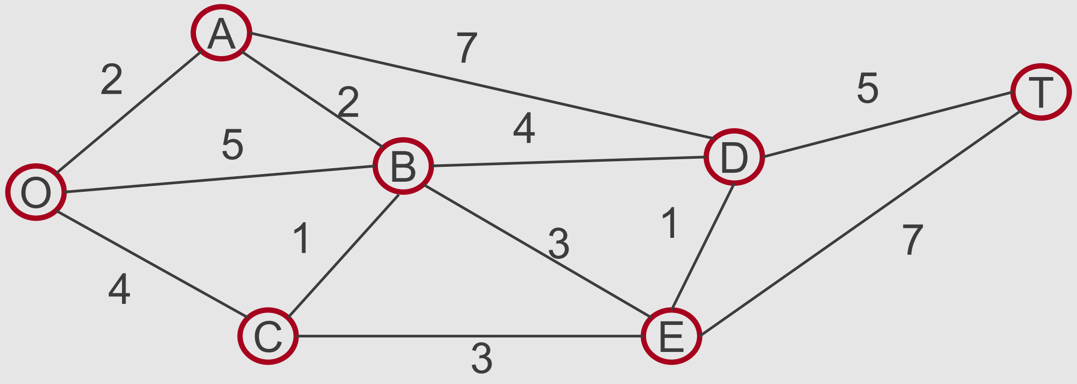
(⇐) Let (V, E) be a connected graph with $n \in \mathbb{N}$ nodes and $n - 1$ edges. Assume that two nodes are connected by two different paths. Then, the union of these paths contains a cycle, with $m \in \mathbb{N}$ nodes and m edges. Connecting the remaining $n - m$ nodes to this cycle requires at least $n - m$ additional edges. This is a contradiction. □

Networks



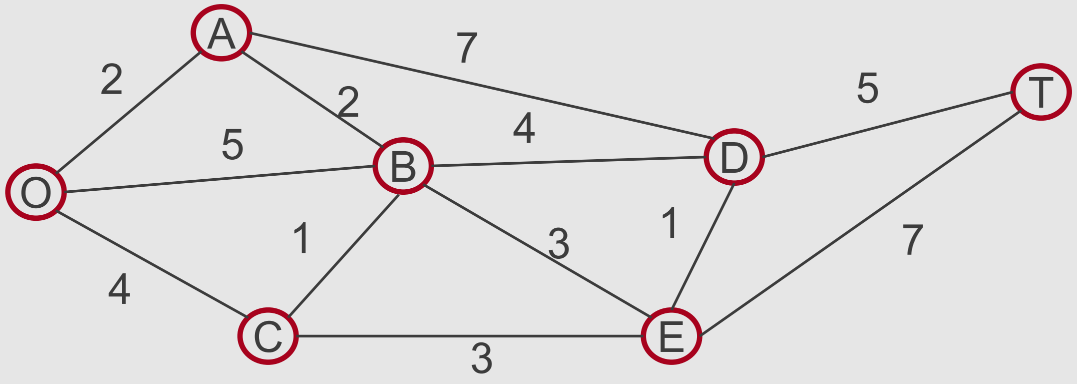
Definition: a *network* (aka *weighted graph*) is a graph(V, E) with a weight $w(e) \in \mathbb{R}$ associated to each edge $e \in E$. In most applications, the weights are positive. In this module, we assume that the weights are indeed positive.

Example of a network - scenario



Scenario: A park has a narrow, winding road system for trams (shown in the figure). Location O is the entrance into the park; other letters designate the locations of ranger stations (and other limited facilities). The numbers are lengths of roads in miles. The park contains a scenic wonder at station T. A small number of trams are used to transport sightseers from the park entrance to station T and back.

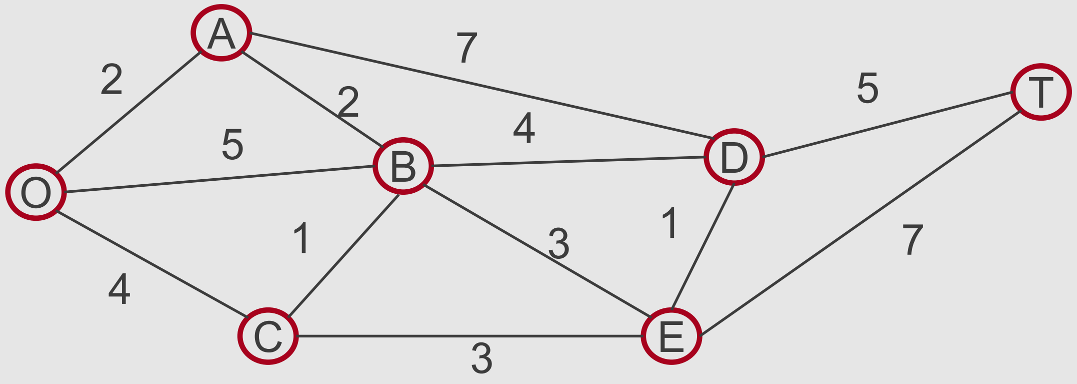
Example of a network – typical problems



Problem 1: The park manager wants to install telephone lines under the roads to establish telephone communication among all the stations. To reduce costs, lines will be installed under just enough roads to ensure every station is connected to the web. Where shall they lay the lines to minimize the total length of lines installed?

Problem 2: What is the shortest route from the entrance to the scenic station T?

Minimal spanning trees and shortest paths



Minimal spanning tree problem: Identify a connected subnetwork that contains all the nodes and such that the sum of the weights of the edges included (the total weight) is minimal.

Shortest path problem: Determine a path to join two nodes such that the sum of the weights of the edges included is minimal.

Minimal spanning tree via linear programming

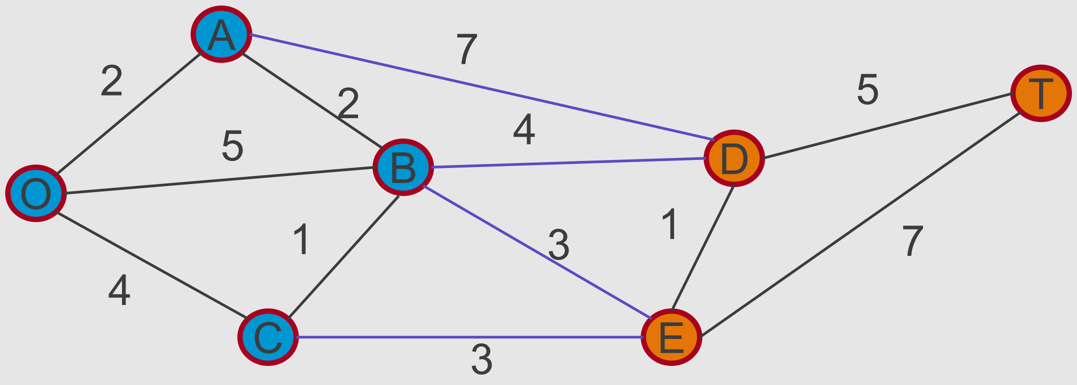
Let $|S|$ denote the cardinality of a generic set S. Let $x \in \{0,1\}^{|E|}$. Then,

$$\min \sum_{e \in E} w(e) x_e \;\; s.t. \begin{cases} \sum_{e \in E} x_e = |V| - 1 \\ \sum_{e \in E'} x_e \leq |V'| - 1 \;\; \forall \text{ full subgraph } (V', E') \\ x_e \in \{0,1\} \; \forall e \in E \end{cases}$$

The second constraint is known as *subtour elimination constraint*: any subset of k vertices must have at most $k - 1$ edges contained in that subset. This ensures that there are no cycles.

Remark: this formulation enjoys the integer solution property!

Network cuts



Definition:

- A *cut* of a network $N = (V, E)$ is a partition $K, \overline{K} \subset V$ such that $K \cap \overline{K} = \emptyset$ and $K \cup \overline{K} = V$
- The *cut-set* of a cut is $C(N, K) = \{(a, b) \in E : a \in K, b \in \overline{K}\}$

Example: K={O,A,B,C}, $\overline{K}$={E,D,T} is a cut, and the set of edges *{(C,E), (B,E), (B,D), (A,D)}* is the cut-set.

Shortest path via linear programming

You can determine the shortest path between two nodes O and T by solving

$$\min \sum_{e\in E} w(e)x_e \quad s.t. \begin{cases} \sum_{e\in C(N,K)} x_e \geq 1 \quad \forall K \in \{V' \subseteq V : O \in V', T \notin V'\} \\ x_e \in \{0,1\}., \forall e \in E \end{cases}$$

The first constraint ensures that the resulting paths connects the source O to the target T.

A key property of cuts

Proposition: Let K be a cut of a network $N = (V, E)$and let $e \in C(N, K)$ satisfy

$$w(e) = \min\ \{w(e) : \ e \ \in C(N, K))$$

Then, e belongs to a minimal spanning tree.

Proof: Assume that (V', E') is a minimal spanning tree of (V, E), and that $e \notin E'$. Then, the network $(V', E' \cup \{e\})$ contains a cycle that connects K and $\overline{K}$ via e and another edge $e_0 \in C(N, K)$ (otherwise(V', E') would be disconnected). Since $w(e) \leq w(e_0)$, the network $(V', E' \cup \{e\} \setminus \{e_0\})$ remains connected, it is a tree by Euler's formula, and its total weight is not bigger than $w(T)$. $\square$

Prim's algorithm (1930, Jarník)

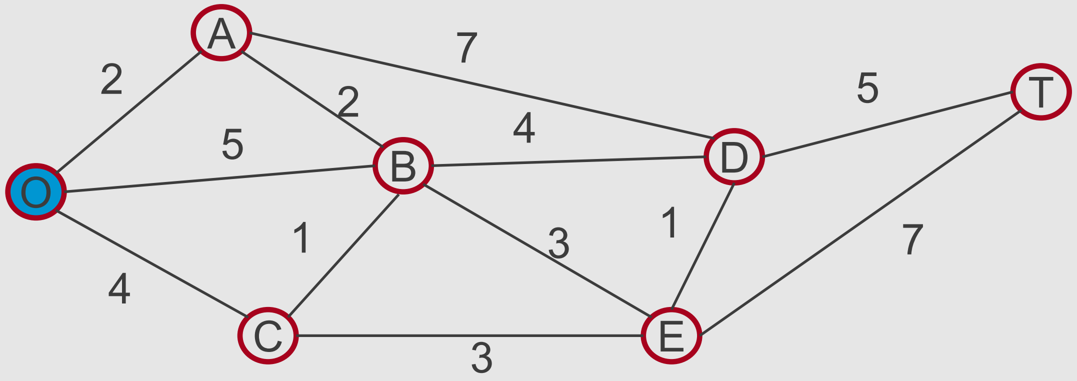
Let $N = (V, E)$ be a connected network with $M = |V| > 1$ nodes.

`Initialization`: pick $n_0 \in V$, build the cut $K_0 = \{n_0\}$, and set $E_0' = \emptyset$

```
Then, for $k = 0: M - 2$
    a. determine the cut-set $C(N, K_k)$
    b. find $(a, b) \in argmin\, \{w(e) : e \in C(N, K_k)\}$
    c. let $n_{k+1} \in \{a, b\} \setminus K_k$
    d. define $K_{k+1} = K_k \cup \{n_{k+1}\}$
    e. define $E_{k+1}' = E_k{}' \cup (a, b)$
```

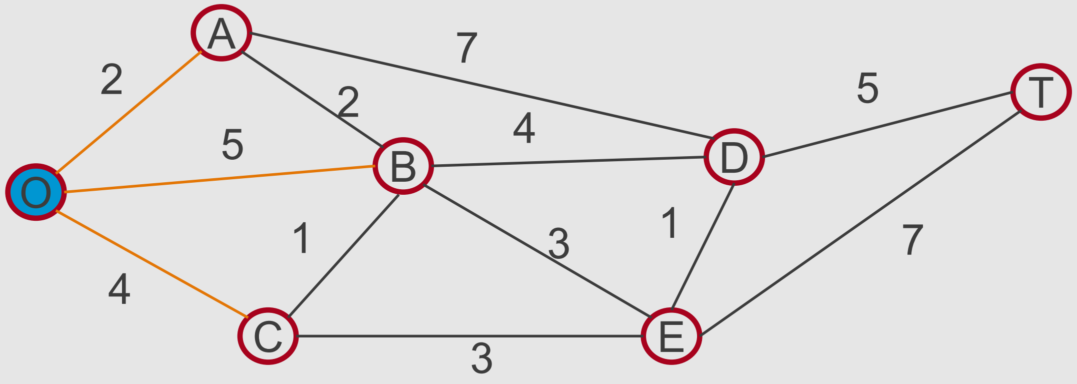
`Output`: (K_{k+1}, E_{k+1}') is a minimal spanning tree.

Prim's algorithm – example (step 1)



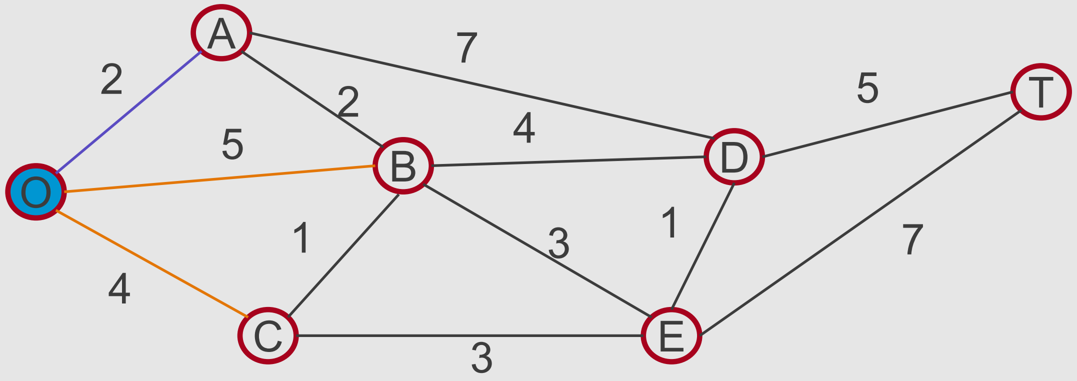
$K_0 = \{O\}, E'_0 = \emptyset$

Prim's algorithm – example (step 2)



$K_0 = \{O\}, E'_0 = \emptyset$
$C(N, K_0)$

Prim's algorithm - example (step 3)

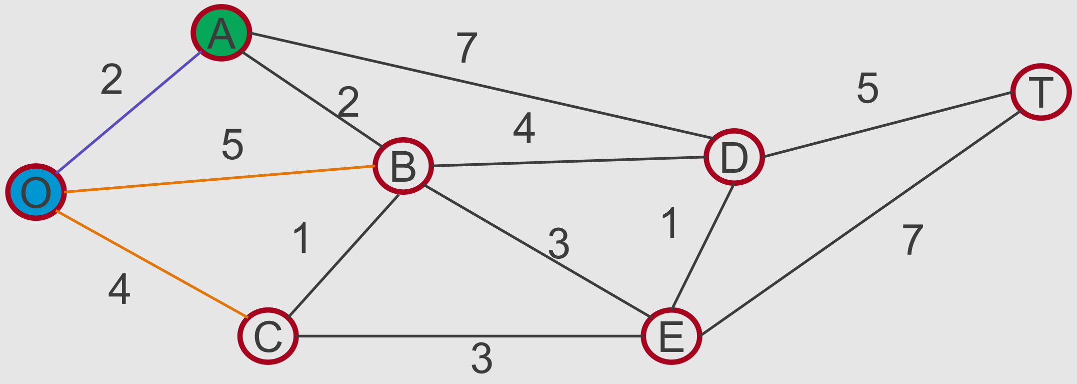


$K_0 = \{O\}, E'_0 = \emptyset$

$C(N, K_0)$

$(\mathrm{a}, b) \in argmin\,\{w(e) : e \in\ C(N, K_k)\}$

Prim's algorithm - example (step 4)



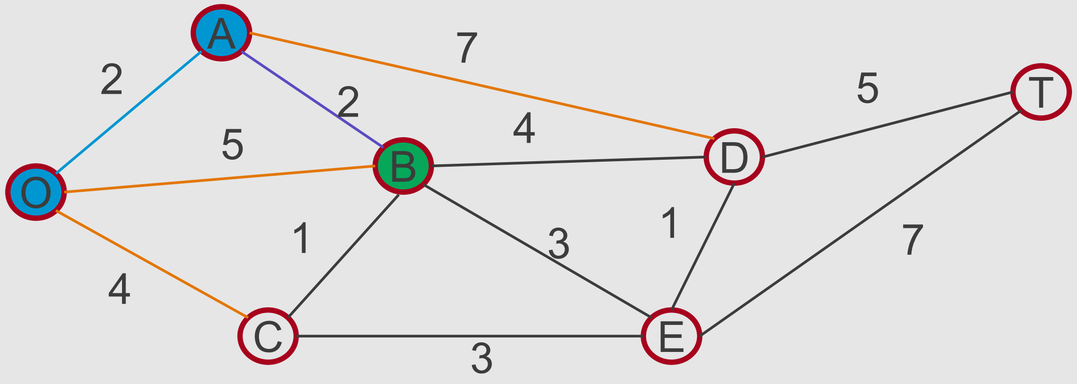
$K_0 = \{O\}, E'_0 = \emptyset$

$C(N, K_0)$

$(O, A) \in argmin\,\{w(e) : e \in\ C(N, K_0)\}$

$n_1\ \in \{O, A\} \setminus K_0 = \{A\}$

Prim's algorithm - example (step 5)



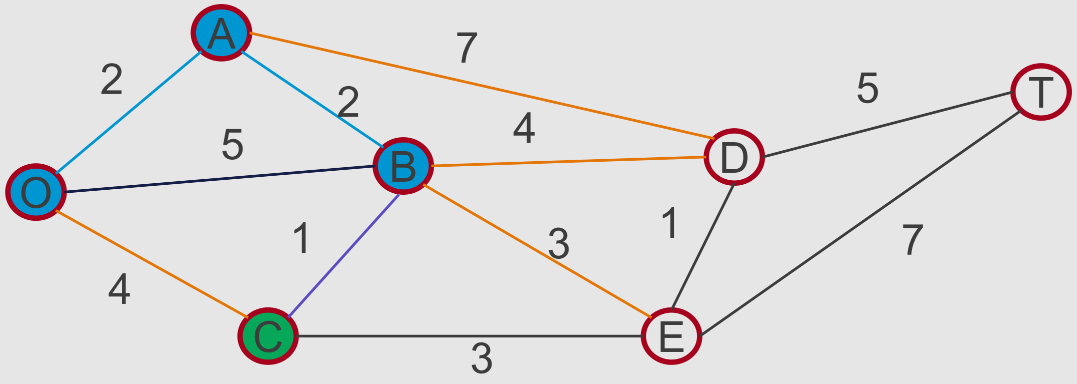
$K_1 = \{O, A\}, E'_1 = \{(O, A)\}$

$C(N, K_1)$

$(A, B) \in argmin\,\{w(e) : e \in\ C(N, K_1)\}$

$n_2\ \in \{A, B\} \setminus K_1 = \{B\}$

Prim's algorithm - example (step 6)



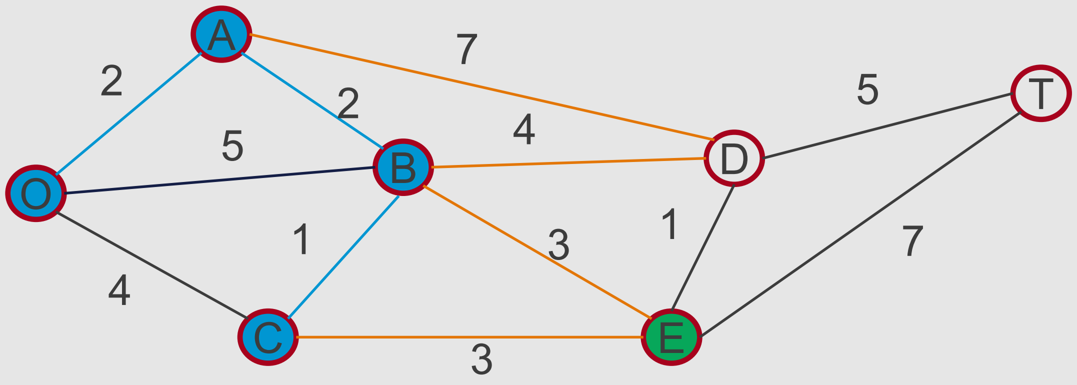
$K_2 = \{O, A, B\}, E_2' = \{(O, A), (A, B)\}$

$C(N, K_2)$

$(B, C) \in argmin\ \{w(e) : e \in\ C(N, K_2)\}$

$n_3\ \in \{B, C\} \setminus K_2 = \{C\}$

Prim's algorithm - example (step 7)



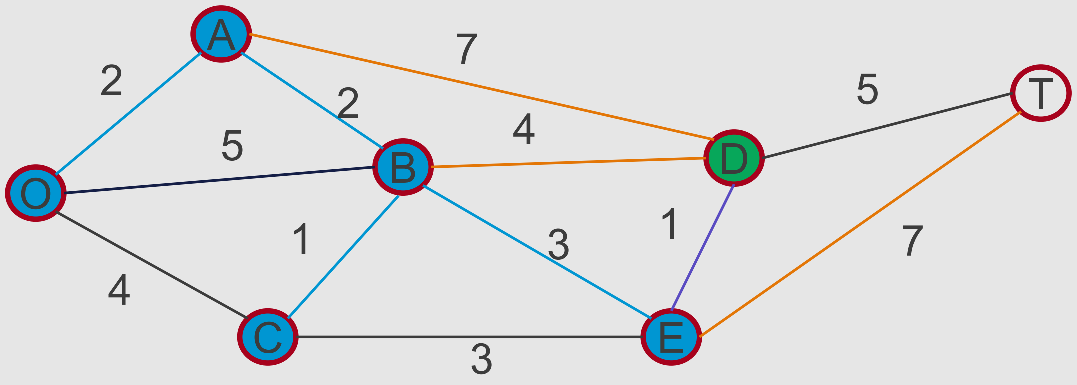
$K_3 = \{O, A, B, C\}, E_3' = \{(O, A), (A, B), (B, C)\}$

$C(N, K_3)$

$(B, E) \in argmin\,\{w(e) : e \in\ C(N, K_3)\}$

$n_4\ \in \{B, E\} \setminus K_3 = \{E\}$

Prim's algorithm - example (step 8)



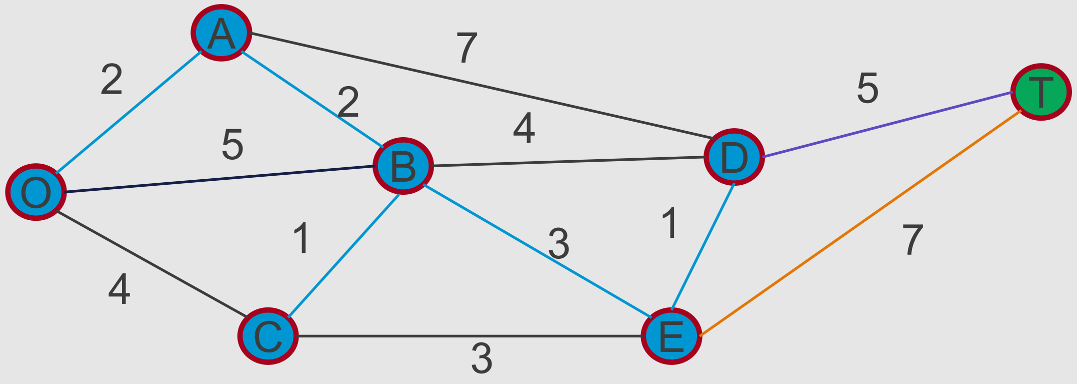
$K_4 = \{O, A, B, C, E\}, E'_4 = \{(O, A), (A, B), (B, C), (B, E)\}$

$C(N, K_4)$

$(D, E) \in argmin\,\{w(e) : e \in\ C(N, K_4)\}$

$n_5\ \in \{D, E\} \setminus K_4 = \{D\}$

Prim's algorithm - example (step 9)



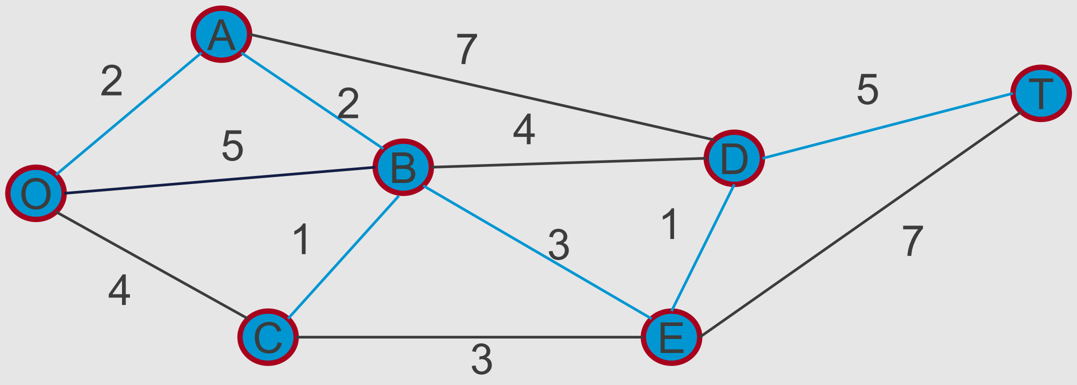
$K_5 = \{O, A, B, C, D, E\}, E'_5 = \{(O, A), (A, B), (B, C), (B, E), (D, E)\}$

$C(N, K_5)$

$(D, T) \in argmin\,\{w(e) : e \in\ C(N, K_5)\}$

$n_5\ \in \{D, T\} \setminus K_5 = \{T\}$

Prim's algorithm – example (step 10)



$K_6 = \{O, A, B, C, D, E, T\}, E'_6 = \{(O, A), (A, B), (B, C), (B, E), (D, E), (D, T)\}$

Dijkstra's algorithm (1956)

Let $N = (V, E)$ be a connected network with M nodes numbered from 1 to M. Dijkstra's algorithms finds all shortest paths from the first node to every other node in N.

Let

- $d \in \mathbb{R}^M_{+,\infty}$, with $d_1 = 0$ and $d_n = \infty$ for $n = 2, \dots, M$,
- $p \in \{0,1, \dots, M\}^{M-1}$, $p_n = 0$ for $n = 1, \dots, M-1$
- $v \in \{0,1\}^M$, $v_n = 0$ for $n = 1, \dots, M$

```
While $\sum_{k=1}^{M} v_k < M$
    pick one $j \in argmin\{d_k : k \in \{1, \dots, M\}, v_k = 0\}$
    set $v_k = 1$
    for all i $\in \{k : k \in \{1, \dots, M\}, v_k = 0, (k, j) \in E\}$
        if $d_j + w\big((j,i)\big) < d_i$
            $d_i = d_j + w\big((j,i)\big)$
            $p_i = j$
```

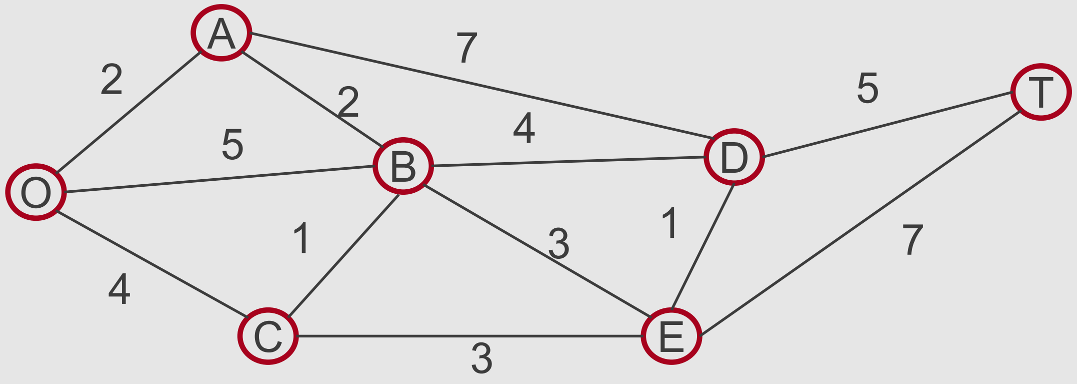
Dijkstra's algorithm (1956) in lay terms

Let $d = (0, \infty, \dots, \infty)$,$p = (\emptyset, \dots, \emptyset)$, and $v = (0, \dots, 0)$, denote the distance, previous-node, and visited-node vectors, respectively.

At each iteration:

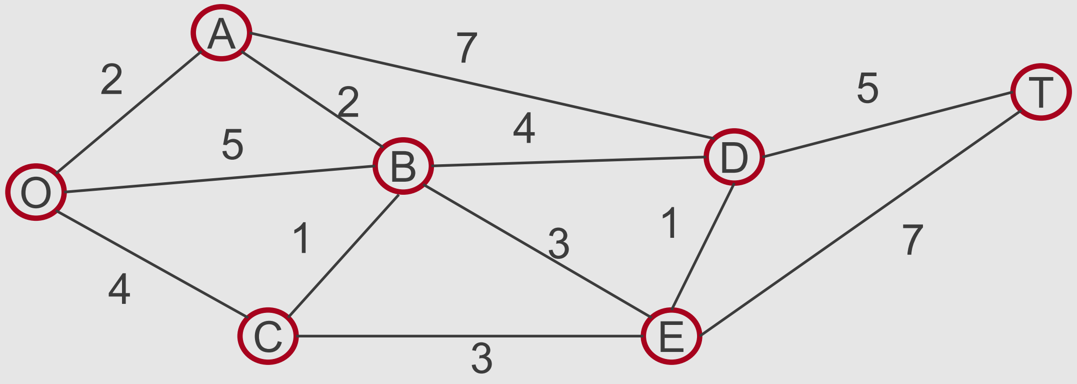
1. pick the most recently visited node (say node j)
2. find its neighbouring nodes that have not been visited yet
3. compute the distances to these neighbours assuming the preceding node is j
4. update the distance and preceding node of these neighbours if passing through node j is a shorter path
5. choose a node whose distance is a minimum among the un-visited nodes and mark it as visited. This is the new most recently visited node. Return to step 1 until all nodes are marked as visited.

Dijkstra's algorithm – example (step 1)



		d	p	v
O	1	0		0
A	2	∞	0	0
B	3	∞	0	0
C	4	∞	0	0
D	5	∞	0	0
E	6	∞	0	0
T	7	∞	0	0

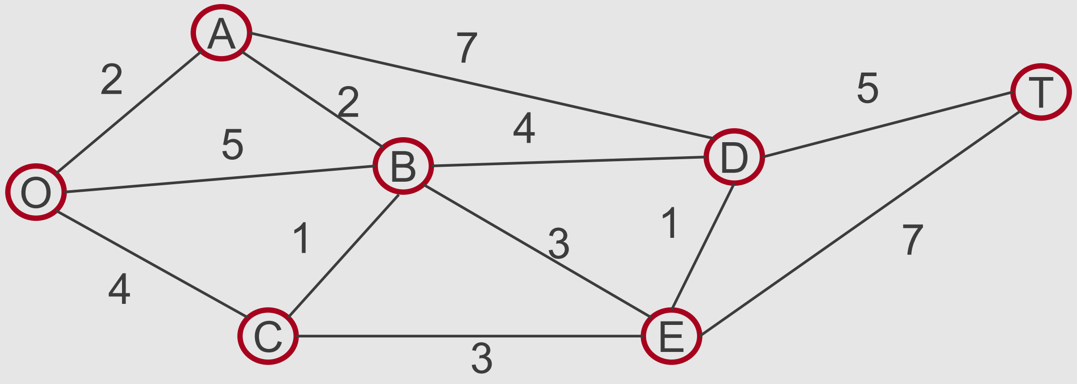
Dijkstra's algorithm – example (step 2)



		d	p	v
O	1	0		0
A	2	∞	0	0
B	3	∞	0	0
C	4	∞	0	0
D	5	∞	0	0
E	6	∞	0	0
T	7	∞	0	0

$$j \in argmin\{d_k : k \in \{1, \dots, M\}, v_k = 0\} = \{1\}$$

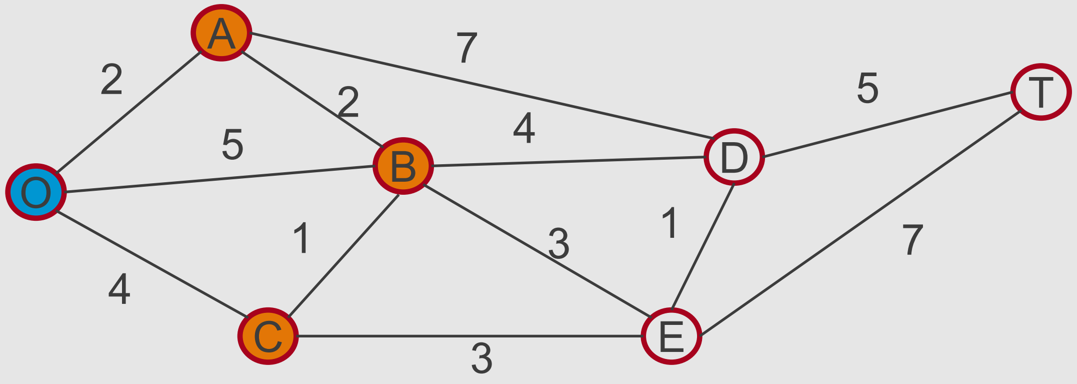
Dijkstra's algorithm – example (step 3)



		d	p	v
O	1	0		1
A	2	∞	0	0
B	3	∞	0	0
C	4	∞	0	0
D	5	∞	0	0
E	6	∞	0	0
T	7	∞	0	0

$$j \in argmin\{d_k : \ k \ \in \{1, \ldots, M\}, v_k = 0\} = \{1\}$$
$$v_j = 1$$

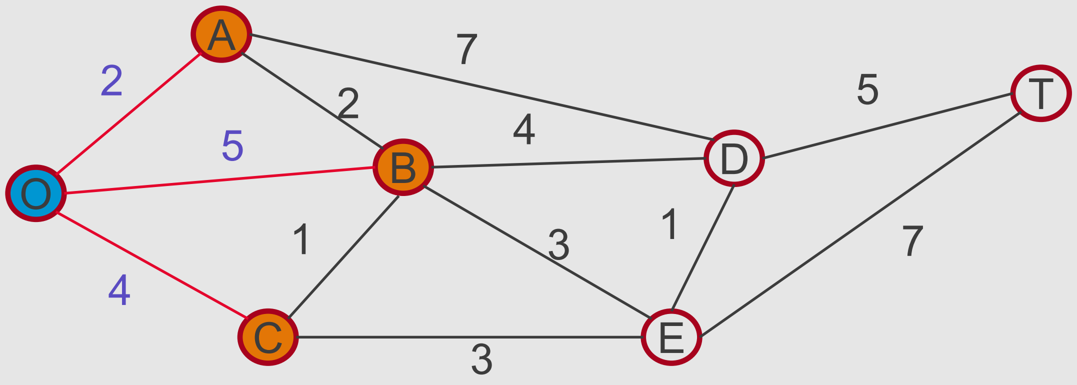
Dijkstra's algorithm – example (step 4)



		d	p	v
O	1	0		1
A	2	∞	0	0
B	3	∞	0	0
C	4	∞	0	0
D	5	∞	0	0
E	6	∞	0	0
T	7	∞	0	0

$$\text{i} \in \{k : k \in \{1, \ldots, M\}, v_k = 0, (k, j) \in E\} = \{2,3,4\}$$

Dijkstra's algorithm – example (step 5)



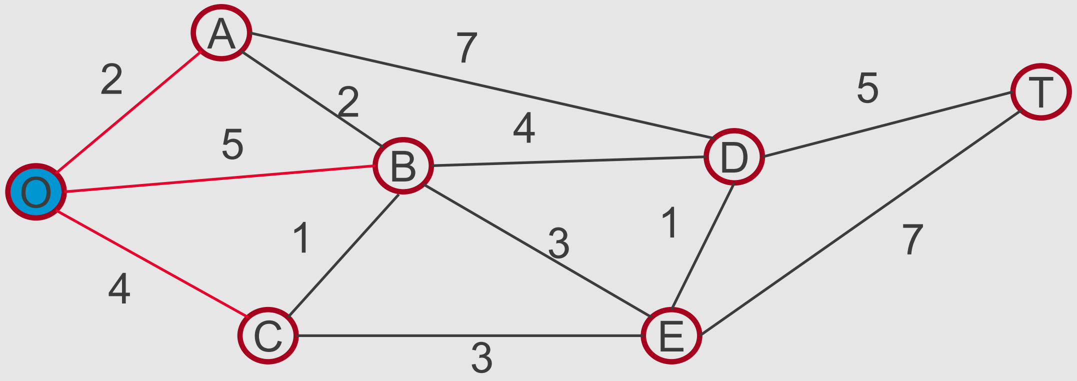
		d	p	v
O	1	0		1
A	2	2	1	0
B	3	5	1	0
C	4	4	1	0
D	5	∞	0	0
E	6	∞	0	0
T	7	∞	0	0

if $d_j + w\big((j,i)\big) < d_i$

$$d_i = d_j + w\big((j,i)\big)$$

$$p_i = j$$

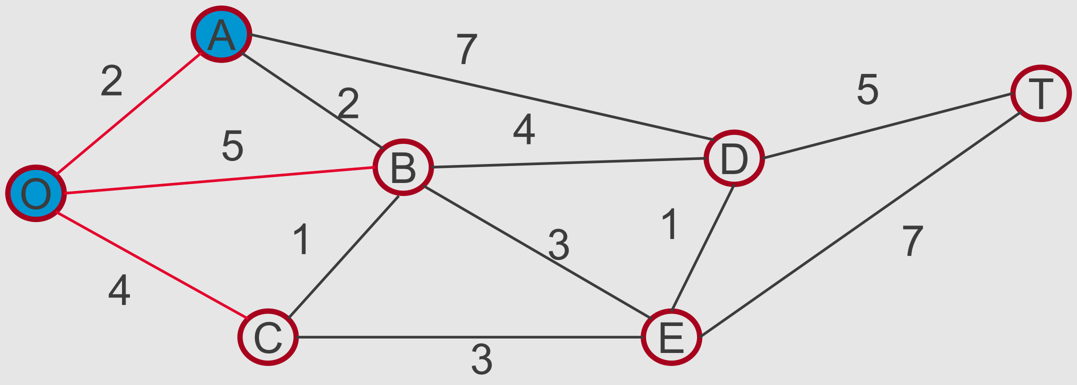
Dijkstra's algorithm – example (step 6)



		d	p	v
O	1	0		1
A	2	2	1	0
B	3	5	1	0
C	4	4	1	0
D	5	∞	0	0
E	6	∞	0	0
T	7	∞	0	0

$$j \in argmin\{d_k \;:\; k \;\in \{1, \ldots, M\}, v_k = 0\} = \{2\}$$

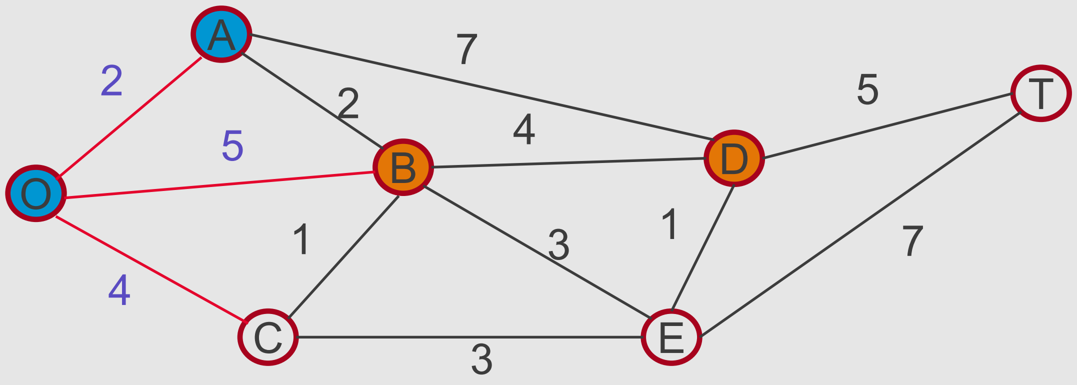
Dijkstra's algorithm – example (step 7)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	5	1	0
C	4	4	1	0
D	5	∞	0	0
E	6	∞	0	0
T	7	∞	0	0

$$j \in argmin\{d_k : \ k \ \in \{1, \dots, M\}, v_k = 0\} = \{2\}$$
$$v_j = 1$$

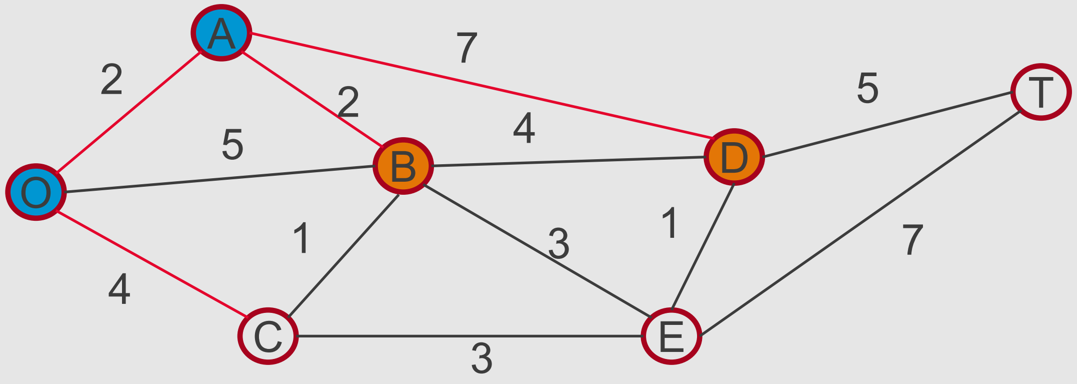
Dijkstra's algorithm – example (step 8)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	5	1	0
C	4	4	1	0
D	5	∞	0	0
E	6	∞	0	0
T	7	∞	0	0

$\mathrm{i} \in \{k : k \in \{1, \dots, M\}, v_k = 0, (k, j) \in E\} = \{3,5\}$

Dijkstra's algorithm – example (step 9)



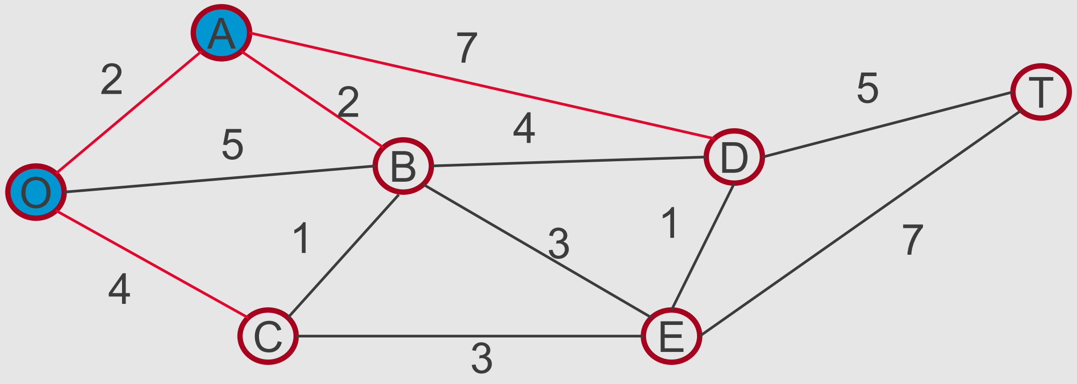
		d	p	v
O	1	0		1
A	2	2	1	1
B	3	$2+2$	2	0
C	4	4	1	0
D	5	$2+7$	2	0
E	6	∞	0	0
T	7	∞	0	0

if $d_j + w\big((j,i)\big) < d_i$

$$d_i = d_j + w\big((j,i)\big)$$

$$p_i = j$$

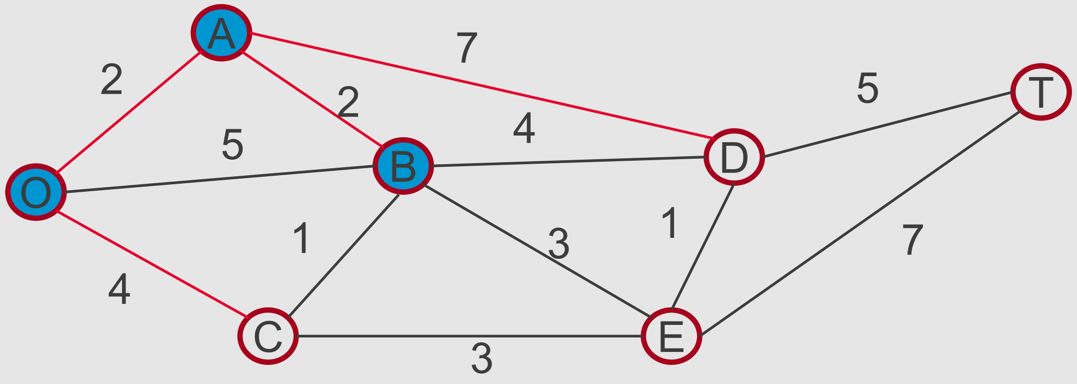
Dijkstra's algorithm – example (step 10)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	0
C	4	4	1	0
D	5	9	2	0
E	6	∞	0	0
T	7	∞	0	0

$$j \in argmin\{d_k : k \in \{1, \dots, M\}, v_k = 0\} = \{3,4\}$$

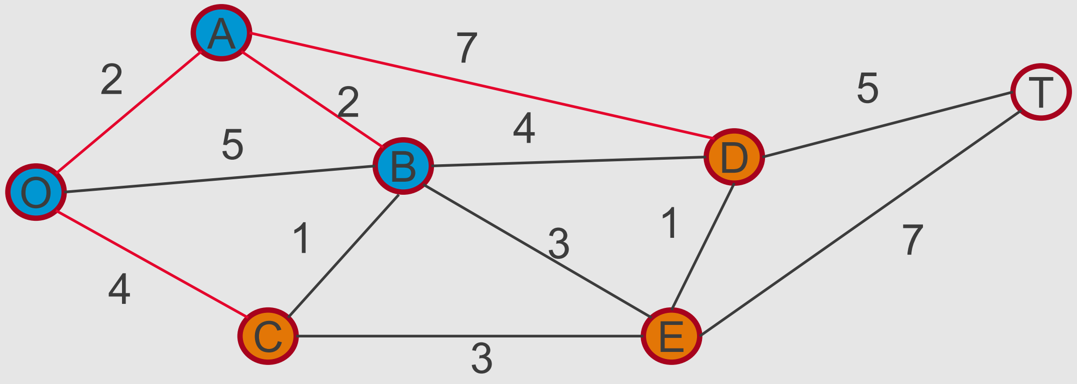
Dijkstra's algorithm – example (step 11)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	0
D	5	9	2	0
E	6	∞	0	0
T	7	∞	0	0

$$j \in argmin\{d_k : \ k \ \in \{1, \dots, M\}, v_k = 0\} = \{1\}$$
$$v_j = 1$$

Dijkstra's algorithm – example (step 12)



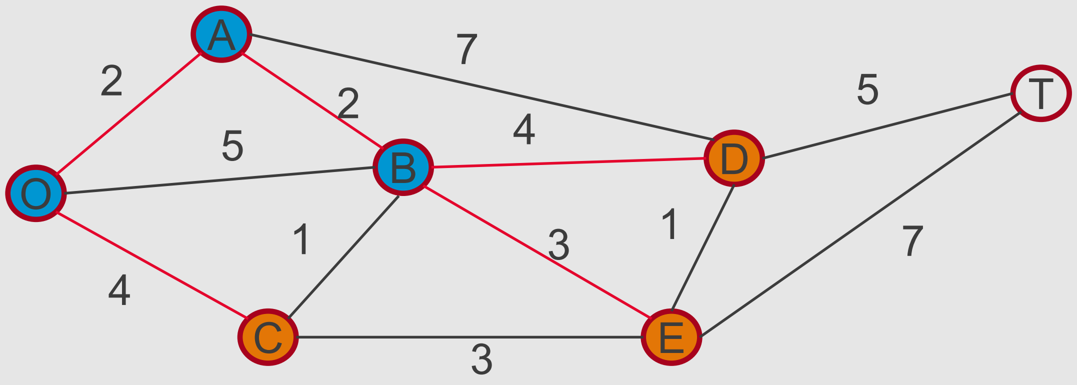
		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	0
D	5	9	2	0
E	6	∞	0	0
T	7	∞	0	0

$$\mathrm{i} \in \{k : k \in \{1, \ldots, M\}, v_k = 0, (k, j) \in E\} = \{4,5,6\}$$

UNIVERSITY OF LEICESTER

www.le.ac.uk

Dijkstra's algorithm – example (step 13)



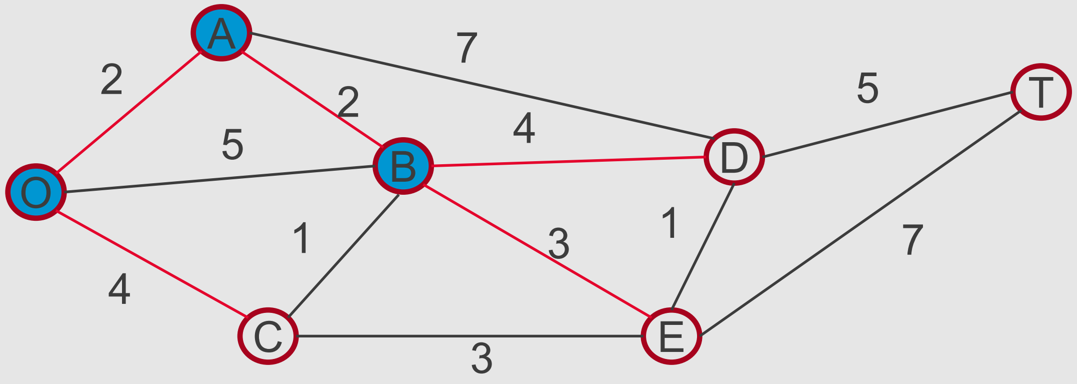
		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	0
D	5	$4+4$	3	0
E	6	$4+3$	3	0
T	7	∞	0	0

if $d_j + w\big((j,i)\big) < d_i$

$$d_i = d_j + w\big((j,i)\big)$$

$$p_i = j$$

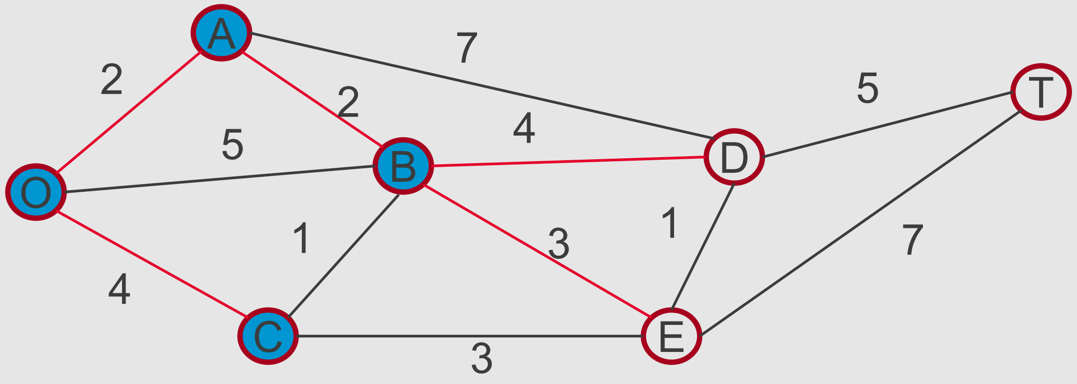
Dijkstra's algorithm – example (step 14)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	0
D	5	8	3	0
E	6	7	3	0
T	7	∞	0	0

$$j \in argmin\{d_k : k \in \{1, \ldots, M\}, v_k = 0\} = \{4\}$$

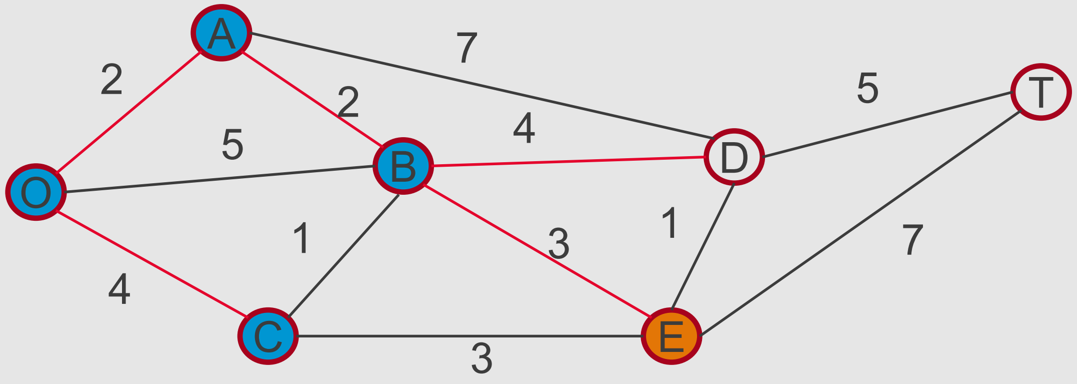
Dijkstra's algorithm – example (step 15)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	0
E	6	7	3	0
T	7	∞	0	0

$$j \in argmin\{d_k : k \in \{1, \dots, M\}, v_k = 0\} = \{4\}$$
$$v_j = 1$$

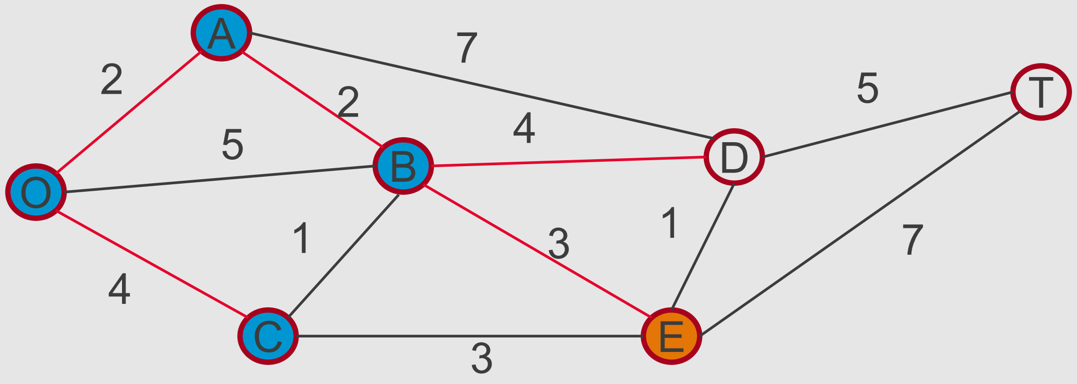
Dijkstra's algorithm – example (step 16)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	0
E	6	7	3	0
T	7	∞	0	0

$\mathrm{i} \in \{k : k \in \{1, \ldots, M\}, v_k = 0, (k, j) \in E\} = \{6\}$

Dijkstra's algorithm – example (step 17)

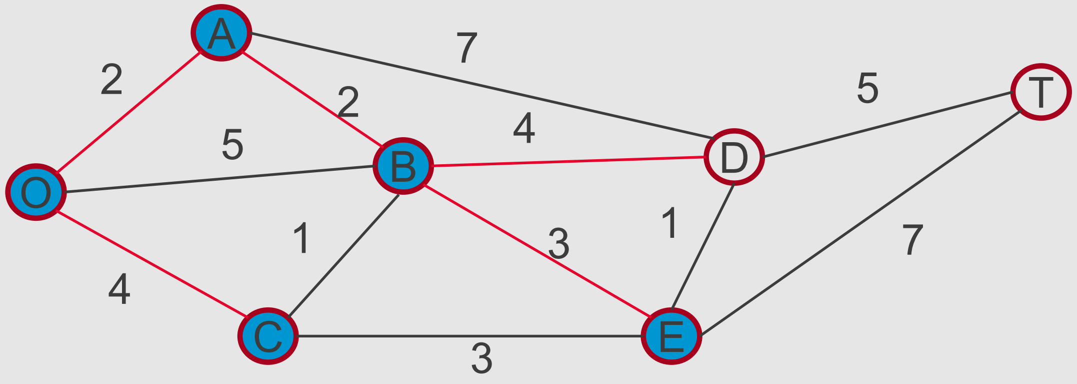


		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	0
E	6	7	3	0
T	7	∞	0	0

if $d_j + w\big((j,i)\big) < d_i$

$$d_i = d_j + w\big((j,i)\big)$$
$$p_i = j$$

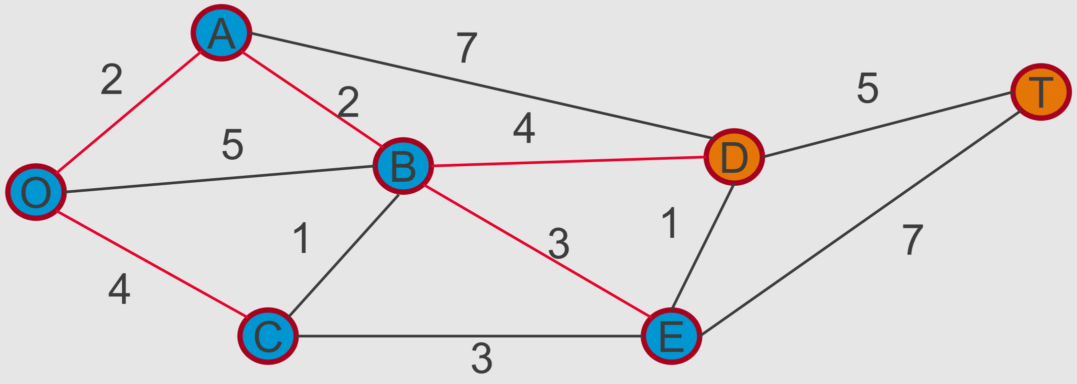
Dijkstra's algorithm – example (step 18)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	0
E	6	7	3	1
T	7	∞	0	0

$$j \in argmin\{d_k : k \in \{1, \dots, M\}, v_k = 0\} = \{6\}$$
$$v_j = 1$$

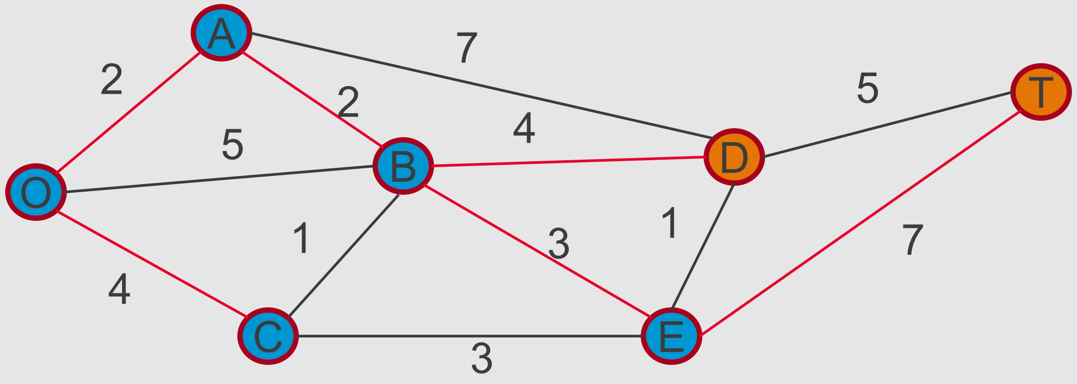
Dijkstra's algorithm – example (step 19)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	0
E	6	7	3	1
T	7	∞	0	0

$\mathrm{i} \in \{k : k \in \{1, \dots, M\}, v_k = 0, (k, j) \in E\} = \{5,7\}$

Dijkstra's algorithm – example (step 20)



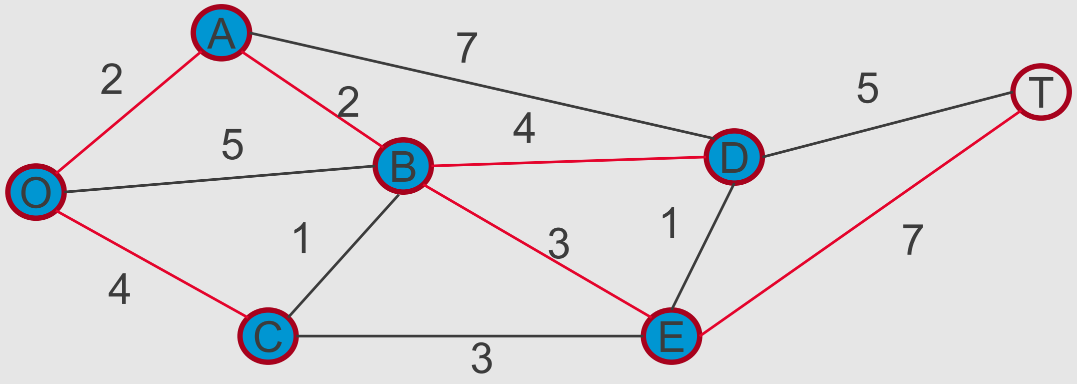
		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	0
E	6	7	3	1
T	7	$7+7$	6	0

if $d_j + w\big((j,i)\big) < d_i$

$$d_i = d_j + w\big((j,i)\big)$$

$$p_i = j$$

Dijkstra's algorithm – example (step 21)



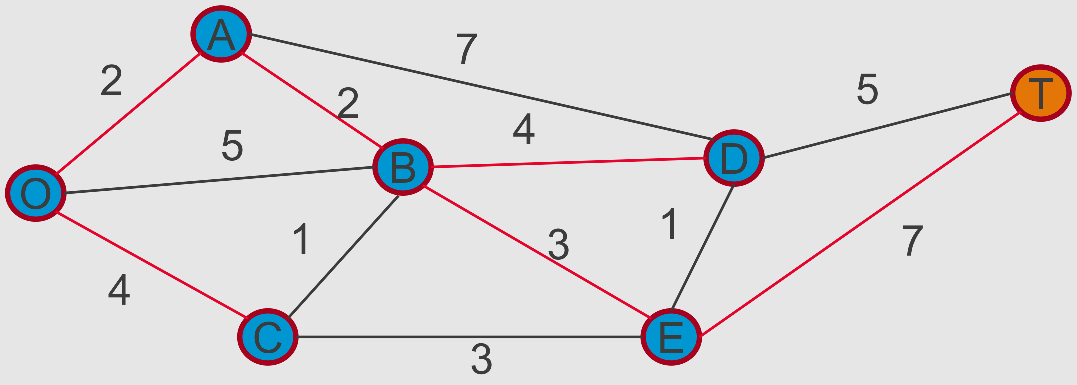
		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	1
E	6	7	3	1
T	7	14	6	0

$$j \in argmin\{d_k : k \in \{1, \ldots, M\}, v_k = 0\} = \{5\}$$
$$v_j = 1$$

UNIVERSITY OF LEICESTER

www.le.ac.uk

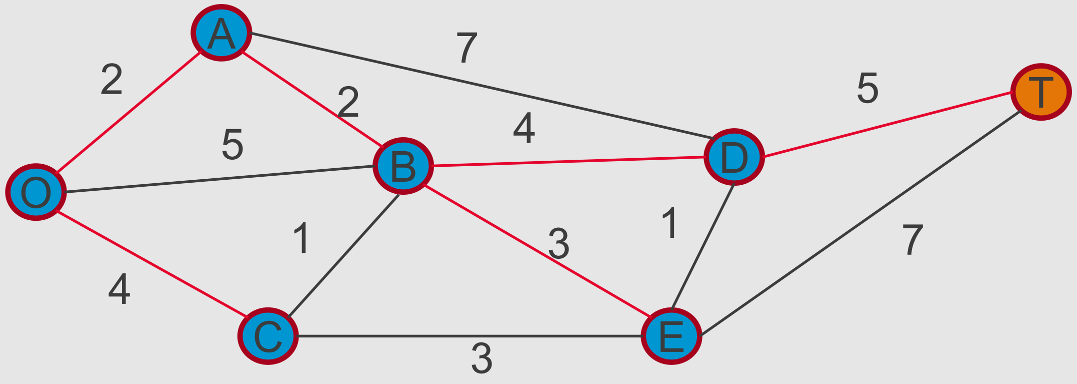
Dijkstra's algorithm – example (step 22)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	1
E	6	7	3	1
T	7	14	6	0

$\mathrm{i} \in \{k : k \in \{1, \ldots, M\}, v_k = 0, (k, j) \in E\} = \{7\}$

Dijkstra's algorithm – example (step 23)



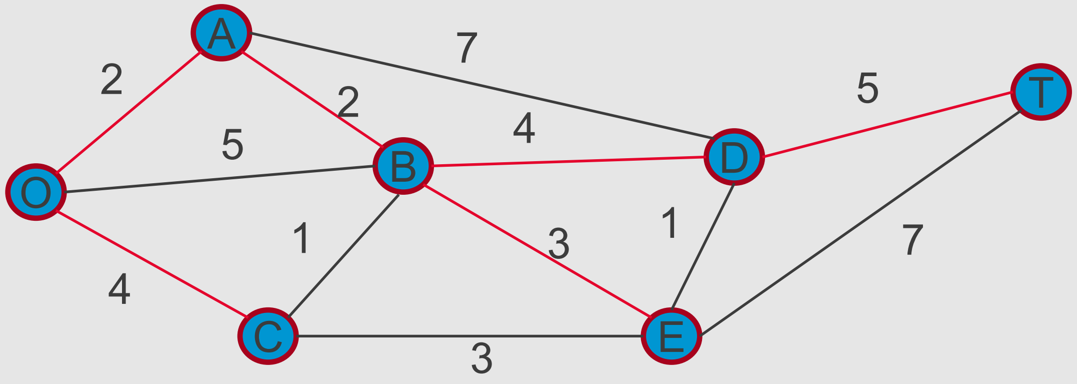
		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	1
E	6	7	3	1
T	7	$8+5$	5	0

if $d_j + w\big((j,i)\big) < d_i$

$$d_i = d_j + w\big((j,i)\big)$$

$$p_i = j$$

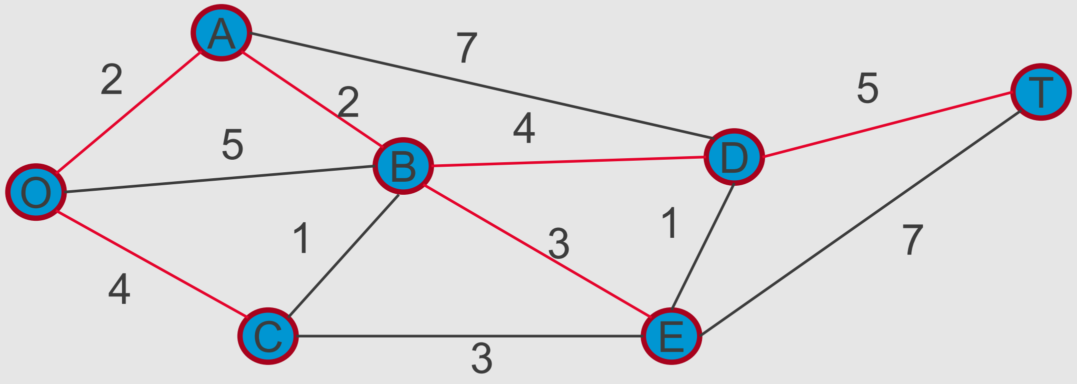
Dijkstra's algorithm – example (step 24)



		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	1
E	6	7	3	1
T	7	13	5	1

$$j \in argmin\{d_k : k \in \{1, \dots, M\}, v_k = 0\} = \{7\}$$
$$v_j = 1$$

Dijkstra's algorithm – example (step 25)

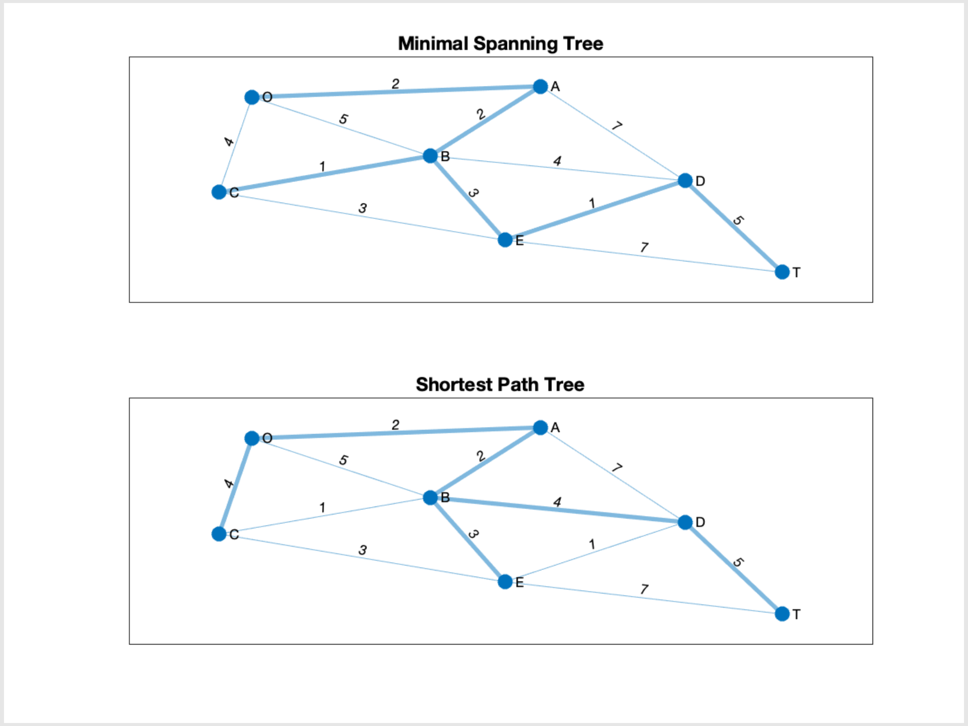


		d	p	v
O	1	0		1
A	2	2	1	1
B	3	4	2	1
C	4	4	1	1
D	5	8	3	1
E	6	7	3	1
T	7	13	5	1

$$\mathrm{i} \in \{k :\ k \in \{1, \dots, M\}, v_k = 0, (k,j) \in E\} = \emptyset$$

Minimal spanning and shortest path trees in Matlab

(see `OR11_networks.m`)

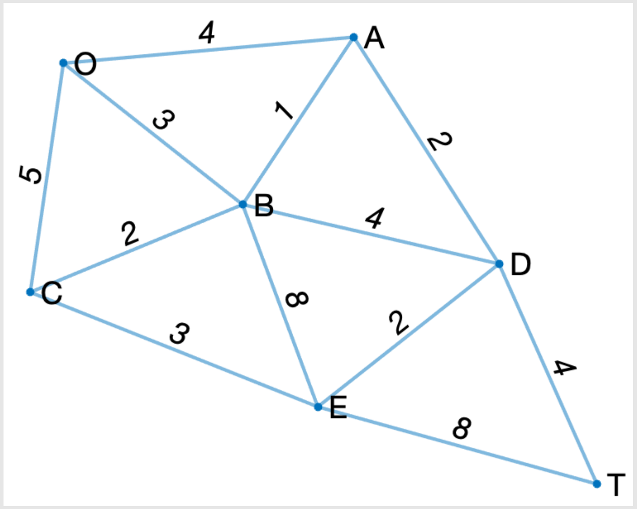


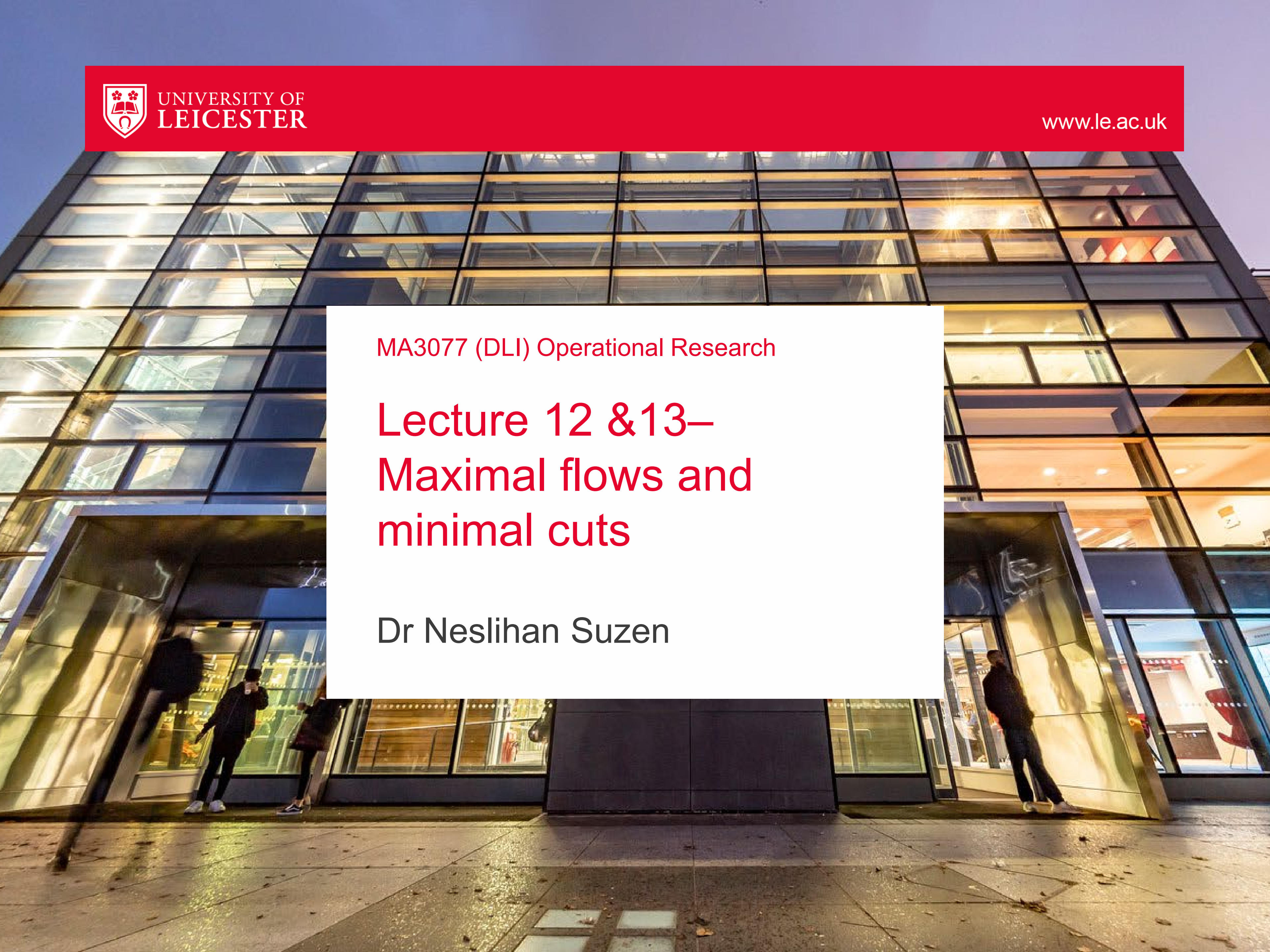
Summary and self-study

Summary: today we have learnt:

- about graphs and networks,
- how to determine the minimal spanning tree of a network,
- how to determine the shortest path tree of a network.

Self-study: Find the minimal spanning tree and the shortest path tree starting at node O of the network on the right.





UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 12 &13– Maximal flows and minimal cuts

Dr Neslihan Suzen

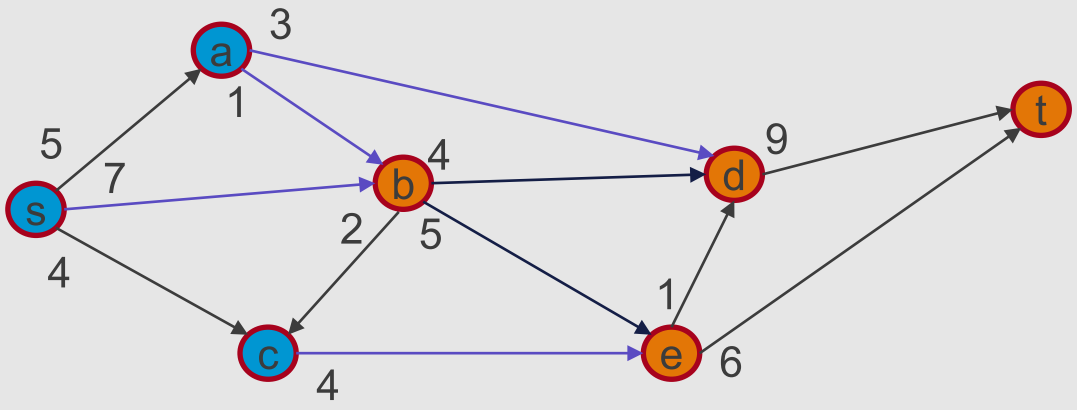
Recap and plan of the day

Summary: in the previous lectures we learnt:

- about graphs and networks
- how to determine a minimal spanning tree,
- how to determine a shortest path tree.

Today: Oriented networks, maximal flows, and minimal cuts, following loosely Ch. 10 of the book by Hillier and Lieberman.

Directed networks - cuts



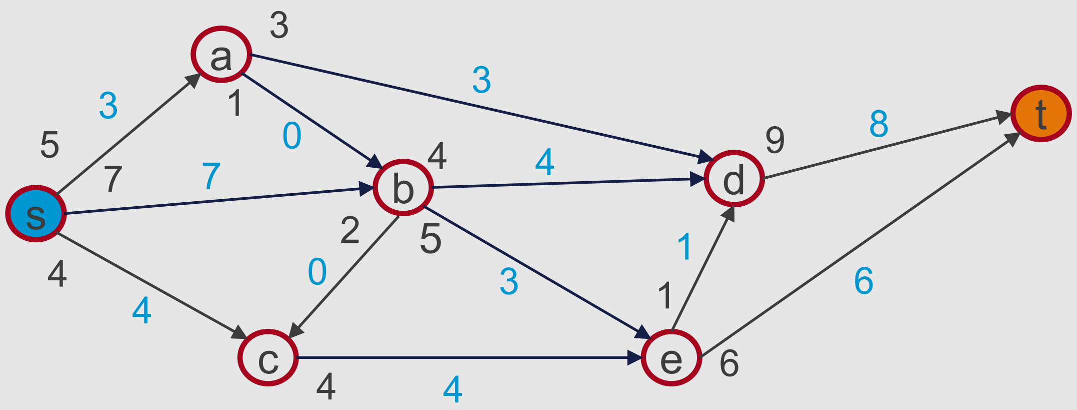
Definition:

- A *network* is a graph (V, E) together with a function $w: E \to \mathbb{R}_+$. In this lecture, the number $w(e)$ denotes the *capacity* of the edge e.
- A network is called *directed* if its edges are directed, that is, if the edges are defined by ordered pairs of vertices.
- A *source* is a node with no incoming edges, whereas a *sink* is a node with no outgoing edges.
- An S-T *cut* of a directed network (V, E) with a source s and a sink t is a partition of $K, \bar{K} \subset V$ such that $s \in K, t \in \bar{K}, K \cap \bar{K} = \emptyset$ and $K \cup \bar{K} = V$
- The *cut-set* of a cut of a directed network $N = (V, E)$ is $C(N, K) = \{(a, b) \in E : a \in K \text{ and } b \in \bar{K}\}$
- The *capacity* of an s-t cut is the sum of the capacities of the edges in its cut-set: $c(N, K) = \sum_{e \in C(N,K)} w(e)$

Example: K={S,A,C}, $\bar{K}$={B,E,D,T} is an S-T cut, $C(N, K) = \{(S, B), (A, B), (A, D), (C, E)\}$ is the cut-set, and $c(N, K) = 3 + 1 + 7 + 4 = 15$.

Challenge: can you find an S-T cut with minimal capacity?

Directed networks - flows



Definition:

Let $N = (V, E)$ be a directed network with source s and sink t. A *flow* is a function $f: E \to \mathbb{R}_+$ that satisfies

- the *capacity constraint*: $f(e) \leq w(e)$ for every $e \in E$
- the *conservation of flows*: for every $v \in V \setminus \{s, t\}$, $\sum_{\{u \in V : (u,v) \in E\}} f(u, v) = \sum_{\{w \in V : (v,w) \in E\}} f(v, w)$

The value of a flow f is defined by

$$|f| = \sum_{\{u \in V : (s,u) \in E\}} f(s, u) = \sum_{\{u \in V : (u,t) \in E\}} f(u, t)$$

Example: the function f is a flow and $|f| = 3 + 7 + 4 = 14 = 8 + 6$

Challenge: can you find a flow with maximal value?

UNIVERSITY OF LEICESTER

www.le.ac.uk

Maximal flows and minimal cuts 1/2

Proposition: Let $N = (V, E)$ be a directed network with source s and sink t. For any flow f and any s-t cut K, it holds $|f| \leq c(N, K)$.

Proof: The conservation of flows implies that, for any $u \in V \setminus \{s, t\}$,

$$\sum_{\{v \in V : (u,v) \in E\}} f(u, v) - \sum_{\{w \in V : (w,u) \in E\}} f(w, u) = 0$$

and the definition of sink implies that $\{w \in V : (w, s) \in E\} = \emptyset$. Therefore,

$$|f| = \sum_{\{v \in V : (s,v) \in E\}} f(s, v) = \sum_{\{v \in V : (s,v) \in E\}} f(s, v) - \sum_{\{w \in N : (w,s) \in E\}} f(w, s)$$

$$= \sum_{u \in K} \left(\sum_{\{v \in V : (u,v) \in E\}} f(u, v) - \sum_{\{w \in V : (w,u) \in E\}} f(w, u) \right)$$

Maximal flows and minimal cuts 2/2

Proposition: Let N be a directed network with source S and sink T. For any flow f and any S-T cut K, it holds $|f| \leq c(N,K)$.

Proof: Therefore,

$$\begin{aligned}
|f| &= \sum_{u\in K}\left(\sum_{\{v\in V:(u,v)\in E\}} f(u,v) - \sum_{\{w\in V:(w,u)\in E\}} f(w,u)\right) \\
&= \sum_{u\in K}\left(\sum_{\{v\in \overline{K}:(u,v)\in E\}} f(u,v) + \sum_{\{v\in K:(u,v)\in E\}} f(u,v) - \sum_{\{w\in \overline{K}:(w,u)\in E\}} f(w,u) - \sum_{\{w\in K:(w,u)\in E\}} f(w,u)\right) \\
&= \sum_{u\in K}\left(\sum_{\{v\in \overline{K}:(u,v)\in E\}} f(u,v) - \sum_{\{w\in \overline{K}:(w,u)\in E\}} f(w,u) + \sum_{\{v\in K:(u,v)\in E\}} f(u,v) - \sum_{\{w\in K:(w,u)\in E\}} f(w,u)\right) \\
&= \sum_{u\in K}\left(\sum_{\{v\in \overline{K}:(u,v)\in E\}} f(u,v) - \sum_{\{w\in \overline{K}:(w,u)\in E\}} f(w,u)\right) + \sum_{u\in K}\left(\sum_{\{v\in K:(u,v)\in E\}} f(u,v) - \sum_{\{w\in K:(w,u)\in E\}} f(w,u)\right) \\
&= \sum_{u\in K}\left(\sum_{\{v\in \overline{K}:(u,v)\in E\}} f(u,v) - \sum_{\{w\in \overline{K}:(w,u)\in E\}} f(w,u)\right) + \sum_{(u,v)\in E, u,v\in K} \left(f(u,v) - f(u,v)\right) \\
&= \sum_{u\in K}\left(\sum_{\{v\in \overline{K}:(u,v)\in E\}} f(u,v) - \sum_{\{w\in \overline{K}:(w,u)\in E\}} f(w,u)\right) \leq \sum_{u\in K}\sum_{\{v\in \overline{K}:(u,v)\in E\}} f(u,v) = \sum_{e\in C(N,K)} f(e) \leq \sum_{e\in C(N,K)} w(e)
\end{aligned}$$

□

Maximal flows and minimal cuts - corollary

Proposition: Let N be a directed network with source S and sink T. For any flow f and any s-t cut K, it holds $|f| \leq c(N, K)$.

Corollary: If $|f| = c(N, K)$, then f is a maximal flow and K a minimal cut.

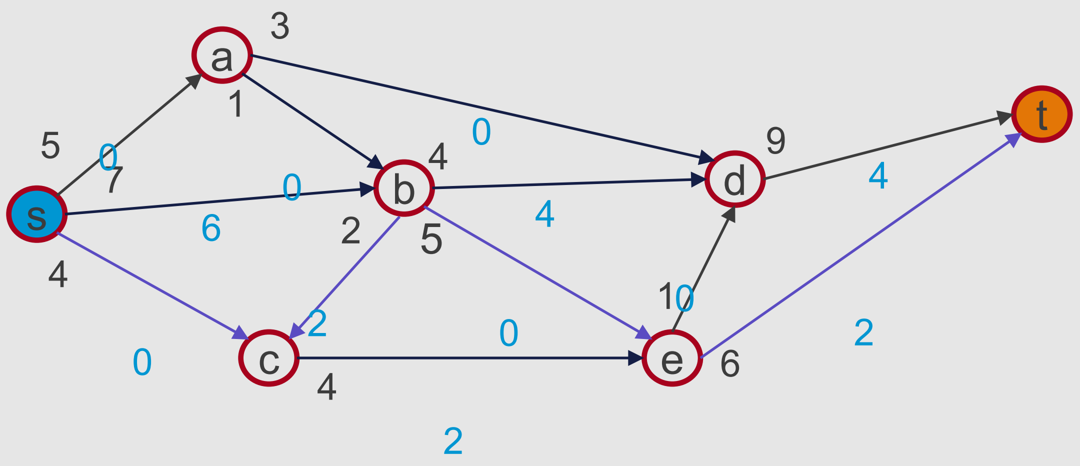
Proof: Let f^* be a maximal flow and K^* a minimal cut. Then

$$|f| \leq |f^*| \leq c(N, K^*) \leq c(N, K) \qquad \text{for any flow } f \text{ and any cut } K.$$

$\square$

- Next lecture

Unsaturated paths



Definition:

Let N be a directed network with source s and sink t, and let f be a flow.

A *path* p in N is a finite sequence $p = \{p_i\}_{i=1}^{\#p}$ of unique nodes such that either $(p_i, p_{i+1}) \in E$ or $(p_{i+1}, p_i) \in E$ for every $i = 1, \ldots, \#p - 1$.

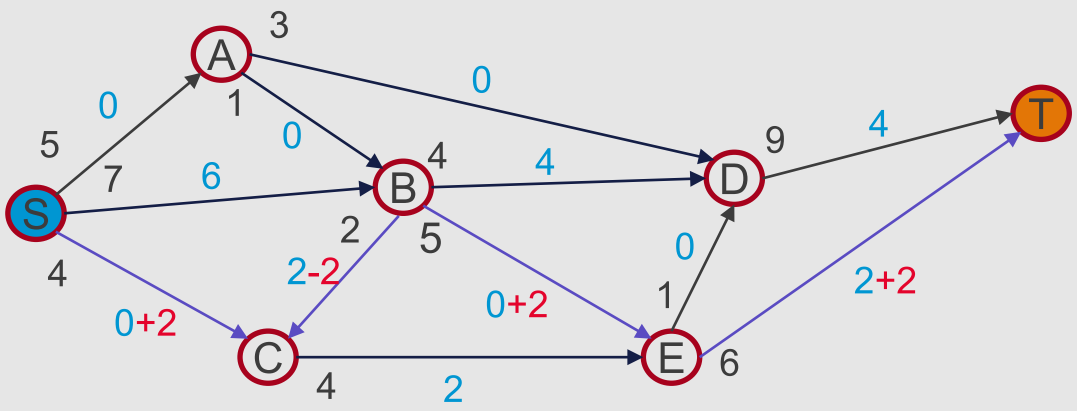
The capacity $\varepsilon(p)$ of a path p in N is defined by

$$\varepsilon(p) := \min_{1 \le i \le \#p-1} \begin{cases} w(p_i, p_{i+1}) - f(p_i, p_{i+1}) & if\,(p_i, p_{i+1}) \in E \\ f(p_{i+1}, p_i) & otherwise \end{cases}$$

A path p with $\varepsilon(p) > 0$ is called f*-unsaturated* . An f-unsaturated path that connects s to t is called f-*augmenting*.

Example: The capacity of the path p={s,c,b,e,t} is $\varepsilon(p) = \min\{4 - 0, 2, 5\ - 0, 6 - 2\} = 2$. Therefore, p is f-augmenting.

Augmented flows



Idea: Let N be a directed network with source s and sink t, let f be a flow, and p an f-augmenting path. Let $\tilde{f}$ be defined as follows

$$\tilde{f}(e) := \begin{cases} f(e) + \varepsilon(p), & \text{if } e = (p_i, p_{i+1}) \text{ for a } p_i \in p, \\ f(e) - \varepsilon(p), & \text{if } e = (p_{i+1}, p_i) \text{ for } a\ p_i \in p, \\ f(e), & otherwise. \end{cases}$$

Then, $\tilde{f}$ is a flow and $|\tilde{f}| = |f| + \varepsilon(p)$.

Example: The value of the flow $\tilde{f}$ obtained by augmenting f with p={s,c,b,e,t} is $|\tilde{f}|$ = 6+0+2 = $|f| + \varepsilon(p)$ = 8.

Maximal flows and minimal cuts revisited

Theorem: Let $N = (V, E)$ be a directed network with source s and sink t.

1. A flow is maximal iff there are no augmenting paths.
2. A flow is maximal iff its value is equal to the capacity of a minimal cut.

Proof: If f is a flow and p is an augmenting path, then the flow $\tilde{f}$ defined in the previous slide satisfies $|\tilde{f}| > |f|$, and f is not maximal. Otherwise, assume there are no augmenting paths, and let

$$K = \{s\} \cup \{v \in V : \exists \text{ path } p \text{ with } p_1 = s, p_{\#p} = v, \text{and } \varepsilon(p) > 0\}.$$

Then, the K defines an s-t cut. Its definition implies that $f(e) = w(e)$ for any $e \in C(N, K)$, and that $f(a, b) = 0$ for any edge (a, b) with $a \in \overline{K}$ and $b \in K$. Therefore,

$$|f| = \sum_{u \in K} \left(\sum_{\{v \in \overline{K} : (u,v) \in E\}} f(u, v) - \sum_{\{w \in \overline{K} : (w,u) \in E\}} f(w, u) \right)$$
$$= \sum_{e \in C(N,K)} f(e) = \sum_{e \in C(N,K)} w(e) = c(N, K).$$

Hence, the previous corollary implies that f is maximal, and the cut defined by K is minimal. □

Maximal flow via linear programming

Let $N = (V, E)$ be a directed network with source s and sink t with $|E|$ many edges. Let $f \in \mathbb{R}^{|E|}$. We can compute the maximal flow by solving

$$\max \sum_{v \in V : (s,v) \in E} f_{sv} \ \ s.t. \begin{cases} f_e \leq w(e) \ \forall e \in E \\ \sum_u f_{uv} - \sum_w f_{vw} = 0 \ \forall v \in V \setminus \{s, t\} \\ f \geq 0 \end{cases}$$

Minimal cut via linear programming

We obtained as the dual to the maximal flow problem. Let $d \in \mathbb{R}^{|E|}$ and $z \in \mathbb{R}^{|V|-2}$

$$\min \sum_{e \in E} w(e) d_e \; s.t. \begin{cases} d_{uv} - z_u + z_v \geq 0 \; \forall (u,v) \in E, u \neq s, v \neq t \\ d_{sv} + z_v \geq 1 \forall (s,v) \in E, v \neq t \\ d_{ut} - z_u \geq 0 \forall (u,z) \in E, u \neq s \\ d_{st} \geq 1 \; if \; (s,t) \in E \end{cases}$$

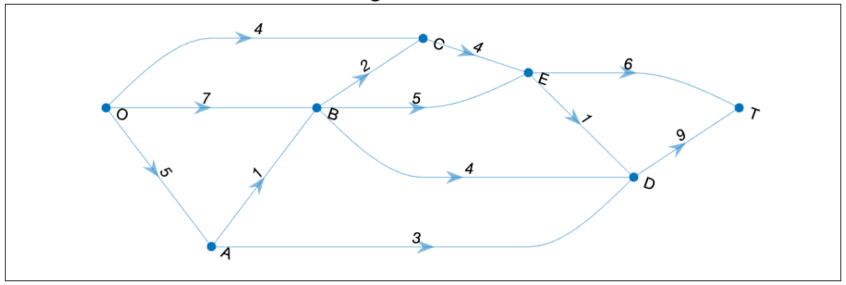
Interpretation: let $K \subset V$ identify a minimal cut

- $d_e = 1 \; if \; e \in C(N,K), d_e = 0$ otherwise
- $z_v = 1 \; if \; v \in K, \; z_v = 0$ otherwise
- $d_{uv} \geq z_v - z_u$ implies that $(u,v) \in C(N,K)$ if $u \in K$ and $v \in \overline{K}$
- $d_{sv} \geq 1 - z_v$ implies that $(s,v) \in C(N,K)$ if $v \in \overline{K}$
- $d_{ut} \geq z_u$ implies that $(u,t) \in C(N,K)$ if $u \in K$

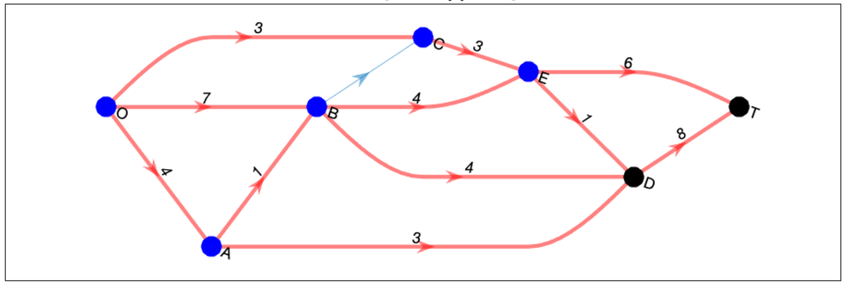
Maximal flows and minimal cuts in Matlab

(see OR13_cutsandflows.m)

Original network



Max Flow from O to T (value |f| = 14) and minimal cut

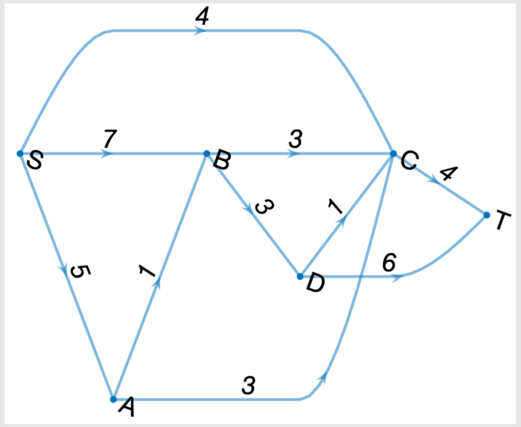


Summary and self-study

Summary: In this lecture and previous lecture we have learnt:

- about directed networks,
- that to a maximal flow corresponds a minimal cut,
- that these are linear programming problems,
- and that you can increase a flow with f-augmenting paths.

Self-study: Consider the directed network with source S and sink T on the right. Let $f: E \to \mathbb{R}_+$ be defined by $f(SB) = 6$, $f(BC) = f(CT) = f(BD) = f(DT) = 3$, and $f(e) = 0$ otherwise. Verify that f is a flow. Then, identify an f-augmenting path and compute its capacity.



MA3077 (DLI) Operational Research

Lecture 14 – Game theory

Dr Neslihan Suzen

Recap and lecture outline

Summary: In previous lectures we learnt:

- about graphs and networks
- how to determine a minimal spanning tree,
- how to determine the shortest path tree
- that maximal flows and minimal cuts are inherently related to each other.

Today: A first glimpes into game theory (following closely Ch. 15 of the book by Hillier and Lieberman).

Two-person, zero-sum games

Definitions:

- Consider a game involving two players.
- The game is called *zero-sum* if one player wins what the other player loses, so that the sum of their net winnings is zero.
- A *strategy* is a predetermined rule that specifies how a player responds to each possible circumstance at each stage of the game.

Two-person, zero-sum games - example

Example: Playing a simplified version of *odds and evens.*

- Each player shows simultaneously one finger or two fingers.
- If the sum of the fingers is even, player A wins the bet and player B pays the bet. Otherwise, player B wins and player A pays.

We can summarize this game in the so-called *payoff table,* which shows the gains of player 1. Negative numbers denote the losses of player 1 (= gains of player 2).

Odds and evens		**Player 2**	
	Strategy	**1**	**2**
Player 1	**1**	1	-1
	2	-1	1

Game theory objectives and assumptions

Goal: The primary objective of game theory is the development of rational criteria for selecting a strategy (possibly the best one).

Key assumptions:

1. Both players are rational.
2. Both players choose their strategies solely to promote their own welfare (no compassion for the opponent , i.e., the game is adversarial, there is no co-operation).

Example 1

Consider the following scenario. Again, the game is assumed to be zero-sum and the table shows the gains of player 1. Negative numbers denote the losses of player 1 (= gains of player 2).

Abstract game		**Player 2**		
	Strategy	**1**	**2**	**3**
Player 1	**1**	1	2	4
	2	1	0	5
	3	0	1	-1

Question: Which strategy should each player select?

Example 1 – dominated strategies 1/6

Consider the following scenario.

Abstract game		**Player 2**		
	Strategy	**1**	**2**	**3**
Player 1	**1**	1	2	4
	2	1	0	5
	3	0	1	-1

Question: Which strategy should each player select?

Definition: A strategy is *dominated* by a second strategy if the latter is at least as good regardless of what the opponent chooses.

Example 1 – dominated strategies 2/6

Consider the following scenario.

Abstract game		**Player 2**		
	Strategy	**1**	**2**	**3**
Player 1	**1**	1	2	4
	2	1	0	5
	3	0	1	-1

Question: Which strategy should each player select?

Player 2 has not dominated strategies, but player 1 does, because strategy 1 dominates strategy 3. Since the players are rational, we can remove strategy 3 of player 1.

Example 1 – dominated strategies 3/6

Consider the following scenario.

Abstract game		Player 2		
	Strategy	**1**	**2**	**3**
Player 1	**1**	1	2	4
	2	1	0	5
	3 ▨	0 ▨	1 ▨	-1 ▨

Question: Which strategy should each player select?

Now player 2 has a dominated strategy, because both strategies 1 and 2 lead to lower losses than strategy 3. Hence, we should remove the latter.

Example 1 – dominated strategies 4/6

Consider the following scenario.

Abstract game		Player 2		
	Strategy	1	2	3 ▨
Player 1	1	1	2	4 ▨
	2	1	0	5 ▨
	3 ▨	0 ▨	1 ▨	-1 ▨

Question: Which strategy should each player select?

Now, player 1's strategy 1 dominates their strategy 2.

Example 1 – dominated strategies 5/6

Consider the following scenario.

Abstract game		**Player 2**		
	Strategy	**1**	**2**	▨ **3**
Player 1	**1**	1	2	▨ 4
	▨ **2**	▨ 1	▨ 0	▨ 5
	▨ **3**	▨ 0	▨ 1	▨ -1

Question: Which strategy should each player select?

Now, player 2's strategy 1 dominates their strategy 2.

Example 1 – dominated strategies 6/6

Consider the following scenario.

Abstract game		Player 2		
	Strategy	1	2	▨ 3
Player 1	1	1	▨ 2	▨ 4
	▨ 2	▨ 1	▨ 0	▨ 5
	▨ 3	▨ 0	▨ 1	▨ -1

Question: Which strategy should each player select?

Each player should pick their own strategy 1. Unfortunately to player 2, this will always lead to a win for player 1.

Game values and fair games

Definition: The payoff to player 1 when both players play optimally is called the *value of the game.*

Definition: If the value of the game is zero, the game is called *fair*.

Example 2

Consider the following scenario.

Abstract game		Player 2		
	Strategy	**1**	**2**	**3**
Player 1	**1**	-3	-2	6
	2	2	0	2
	3	5	-2	-4

Question: Which strategy should each player select?

This game has no dominated strategies.

Example 2 – minimax and maximin 1/2

Consider the following scenario.

Abstract game		**Player 2**			**Min**
	Strategy	**1**	**2**	**3**	
Player 1	**1**	-3	-2	6	-3
	2	2	0	2	**0**
	3	5	-2	-4	-4
Max		5	**0**	6	

Question: Which strategy should each player select?

Idea: Each player minimises their maximum losses. Player 1 determines their *maximin* payoff, whereas player 2 determines their *minimax* loss.

Example 2 - minimax and maximin 2/2

Consider the following scenario.

Abstract game		**Player 2**			**Min**
	Strategy	**1**	**2**	**3**	
Player 1	**1**	-3	-2	6	-3
	2	2	0	2	**0**
	3	5	-2	-4	-4
Max		5	**0**	6	

Question: Which strategy should each player select?

In this case, if both player pick strategy two, neither improves upon their best guarantee and both force the opponent into the same position.

Example 2 – stable solution

Consider the following scenario.

Abstract game		**Player 2**			**Min**
	Strategy	**1**	**2**	**3**	
Player 1	**1**	-3	-2	6	-3
	2	2	0	2	**0**
	3	5	-2	-4	-4
Max		5	**0**	6	

Question: Which strategy should each player select?

In this case, if both player pick strategy two, neither improves upon their best guarantee (not even a-posteriori) and both force the opponent into the same position. This is a *stable solution* (aka *equilibrium solution*).

Example 2 – saddle point

Consider the following scenario.

Abstract game		Player 2			Min
	Strategy	**1**	**2**	**3**	
Player 1	**1**	-3	-2	6	-3
	2	2	0	2	**0**
	3	5	-2	-4	-4
Max		5	**0**	6	

Question: Which strategy should each player select?

Note: The maximin and the minimax values are the same, because this entry is both the minimum in its rows and the maximum in its columns. This is called a *saddle point*.

Summary and self-study

Summary: today we have considered two-person zero-sum games and learnt

- the basic principles of game theory,
- pay-off matrix reduction by dominated strategies,
- what minimax and maximin strategies are
- and that saddle-points characterise stable solution.

Self-study: apply reduction by dominance to the following two-person zero-sum game and then, if necessary, apply the minimax/maximin criterion to determine optimal strategies. Is the solution stable?

Abstract game		**Player 2**			
	Strategy	**1**	**2**	**3**	**4**
Player 1	**1**	2	-2	-5	1
	2	4	-2	-3	-3
	3	0	-1	2	3
	4	3	-3	-3	-4

MA3077 (DLI) Operational Research

Lecture 15 – Stable games and saddle points

Dr Neslihan Suzen

Recap and lecture outline

Recap: in the previous lecture we introduced 2-person zero-sum games and learnt

- pay-off matrix reduction by dominated strategies,
- what minimax and maximin strategies are
- and that saddle-points characterize stable solution.

Today: We prove the tight relationship between stable games and saddle points (following closely Ch. 15 of the book by Hillier and Lieberman).

Lower and upper values of a game

Note: We can fully represent a payoff table of a two-person zero-sum game using a matrix A. We henceforth call such a matrix a *payoff matrix*.

Definition: Let $A \in \mathbb{R}^{m,n}$ be a payoff matrix with entries $a_{ij}, i = 1, \dots, m, j = 1, \dots, n$.

- The *lower value* of player 1 is $A^- := \max_i \min_j a_{ij}$.
- The *upper value* of player 1 is $A^+ := \min_j \max_i a_{ij}$.

Proposition: Let $A \in \mathbb{R}^{m,n}$ be a payoff matrix. Then $A^- \leq A^+$.

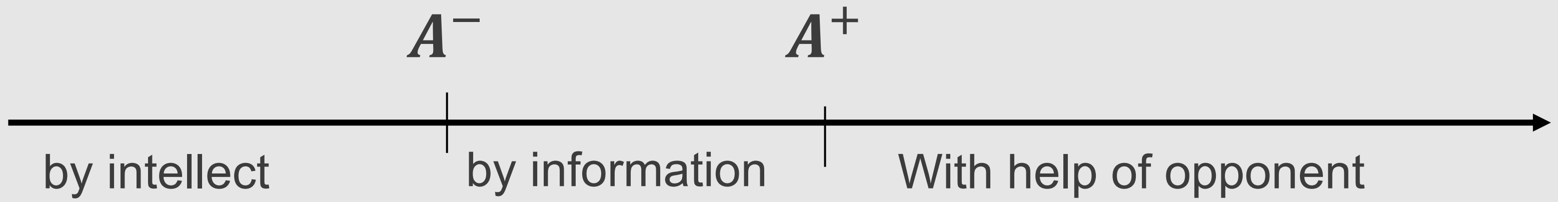
Proof: For any $k \in \{1, \dots, n\}$, $A^- = \max_i \min_j a_{ij} \leq \max_i a_{ik}$. Therefore, taking the minimum over k, we conclude that

$$A^- = \max_i \min_j a_{ij} \leq \min_k \max_i a_{ik} = \mathrm{A}^+$$

$\square$

Interpretation of upper and lower values

We can interpret player's 1 payoff using the following diagram.



Unless player 1 is reckless, they are guaranteed to secure a payoff of A^-. If they have extra information, they can improve upon it, but player 2 can put the upper bound A^+ on this.

Stable games and saddle points

Definition: Let $A \in \mathbb{R}^{m,n}$ be a payoff matrix. The game is called *stable* if $A^- = A^+$, and $A^- = A^+$ is called the value of the game.

Definition: Let $A \in \mathbb{R}^{m,n}$ be a payoff matrix with entries $a_{ij}, i = 1, \dots, m, j = 1, \dots, n$. If there exist a pair $(p, q) \in \{1, \dots m\} \times \{1, \dots, n\}$ such that

$$a_{pq} = \min_j a_{pj} = \max_i a_{iq} \, ,$$

then a_{pq} (or, alternatively, (p, q)) is called a *saddle point* of the game.

Theorem: A two-person zero-sum game with payoff matrix $A \in \mathbb{R}^{m,n}$ is stable if and only if it has a saddle point.

Stable games and saddle points (⇐)

Theorem: A two-person zero-sum game with payoff matrix $A \in \mathbb{R}^{m,n}$ is stable if and only if it has a saddle point.

Proof: (⇐) Let a_{pq} be a saddle point. Then, by definition,

$$\max_i a_{iq} \leq a_{pq} \leq \min_j a_{pj} .$$

Minimizing the left-hand side over q and maximizing the right-hand side over p, we obtain

$$\min_j (\max_i a_{ij}) \leq \max_i a_{iq} \leq a_{pq} \leq \min_j a_{pj} \leq \max_i (\min_j a_{ij})$$

Since $A^+ := \min_j \max_i a_{ij}$ and $A^- := \max_i \min_j a_{ij}$, we conclude that $A^+ \leq A^-$. Finally, since $A^- \leq A^+$ by the previous preposition, we conclude that $A^- = A^+$, and that the game is indeed stable.

Stable games and saddle points (⇒)

Theorem: A two-person zero-sum game with payoff matrix $A \in \mathbb{R}^{m,n}$ is stable if and only if it has a saddle point.

Proof: (⇒) Let $A^- = A^+$ and let a_{pq} be an entry of A such that $a_{pq} = A^- = A^+$. Since $a_{pq} = A^- = \max_i \min_j a_{ij}$ implies that $p \in argmax_i\,(\min_j a_{ij})$, we conclude that

$$a_{pq} = \min_j a_{pj} \, .$$

Similarly, since $a_{pq} = A^+ = \min_j \max_i a_{ij}$ implies that $q \in argmin_j\,(\max_i a_{ij})$, we conclude that

$$a_{pq} = \max_i a_{iq} \, .$$

Therefore, a_{pq} is a saddle point.

□

Reduction by dominance and saddle points

Proposition: Reducing a two-person zero-sum game payoff matrix $A \in \mathbb{R}^{m,n}$ by applying dominance techniques does not affect saddle points.

Proof: Let $a_{pq} = \min_j a_{pj} = \max_i a_{iq}$ be a saddle point. Row k is dominated by row h if $a_{kj} \leq a_{hj}$ for every $j \in \{1, \dots, n\}$. Since,

$$a_{pq} = \max_i a_{iq} \geq a_{hq} \geq a_{kq},$$

removing row k does not affect the value of a_{pq}. Similarly, columns k is dominated by column h if $a_{ih} \leq a_{ik}$ for every $i \in \{1, \dots, m\}$. Since

$$a_{pq} = \min_j a_{pj} \leq a_{ph} \leq a_{pk},$$

removing column k does not affect the value of a_{pq}.

$\square$

Not every game is stable

Consider the following scenario.

Abstract game		Player 2			Min
	Strategy	**1**	**2**	**3**	
Player 1	**1**	0	-2	2	**-2**
	2	5	4	-3	-3
	3	2	3	-4	-4
Max		5	4	**2**	

We observe that $A^{-} = -2 < 2 = A^{+}$, therefore this game is not stable. Indeed, if player 1 chooses strategy 1, then player 2 chooses strategy 2, but then player 1 switches to strategy 2, in which case player 2 selects strategy 3, in which case player 1 apply strategy 1, and so forth...

Summary and self-study

Summary: we have learnt:

- about two-person zero-sum games,
- how to apply reduction by dominance,
- how to determine if a game is stable,
- the reduction by dominance does not affect stability,
- and that not all games are stable.

Self-study: Write a Matlab function `y = is_stable(A)` whose reads in a payoff matrix `A` and returns `y = true` if the associated two-person zero-sum game is stable and `y = false` otherwise.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 16 – Randomized strategies

Dr Neslihan Suzen

Recap and lecture outline

Summary: so far, we learnt:

- about two-person zero sum games,
- how to reduce a payoff table by apply dominance strategies,
- how to determine if a game is stable,
- and that not all games are stable

Today: Randomized strategies to deal with unstable games (following closely Ch. 15 of the book by Hillier and Lieberman).

Not every game is stable

Consider the following scenario.

Abstract game		**Player 2**			**Min**
	Strategy	**1**	**2**	**3**	
Player 1	**1**	0	-2	2	**-2**
	2	5	4	-3	-3
	3	2	3	-4	-4
Max		5	4	**2**	

We observe that $A^{-} = -2 < 2 = A^{+}$, therefore this game is not stable. Indeed, if player 1 chooses strategy 1, then player 2 chooses strategy 2, but then player 1 switches to strategy 2, in which case player 2 selects strategy 3, in which case player 1 apply strategy 1, and so forth...

UNIVERSITY OF LEICESTER

www.le.ac.uk

Mixed strategies

If a game is unstable, deterministic strategies will lead nowhere.

Idea: instead of assuming that a player plays a certain strategy for certain, let the players choose among their strategies randomly. This way, their opponents won't be able to predict the outcomes and adjust their strategy accordingly. More precisely, let $m, n \in \mathbb{N}$ denote the number of strategies available to players 1 and 2, respectively.

- Let $x = (x_i)_{i=1}^m$ be a discrete probability distribution over the set of player 1's strategies, meaning that player 1 selects strategy i with probability x_i.
- Let $y = (y_i)_{i=1}^n$ be a discrete probability distribution over the set of player 2's strategies, meaning that player 2 selects strategy i with probability y_i.

The vectors x and y are called *mixed strategies*, and when players play an actual game, they select a strategy by drawing from this probability distribution. Note that we tacitly assume that $x_i, y_i \geq 0$ and that $\sum_{i=1}^m x_i = 1 = \sum_{i=1}^n y_i$.

Mixed-strategies - example

Consider the following scenario.

Abstract game		**Player 2**			**Min**
	Strategy	**1**	**2**	**3**	
Player 1	**1**	0	-2	2	**-2**
	2	5	4	-3	-3
	3	2	3	-4	-4
Max		5	4	**2**	

If players 1 and 2 use the mixed strategies $x = (0.5,0.5,0)$ and $y = (0,0.5,0.5)$, player 1 chooses randomly between strategies 1 and 2 (e.g., by flipping a coin), whereas player 2 chooses randomly between strategies 2 and 3 (e.g., by flipping another coin).

Expected payoff

To measure the performance of mixed strategies, we use the expected payoff. For player 1, the *expected payoff* can be computed as $x^T A\, y$.

For example, the expected payoff of player 1 in the previous example is

$$(0.5,0.5,0)\begin{pmatrix}0 & -2 & 2\\ 5 & 4 & -3\\ 2 & 3 & -4\end{pmatrix}\begin{pmatrix}0\\ 0.5\\ 0.5\end{pmatrix} = (0.5,0.5,0)\begin{pmatrix}0\\ 0.5\\ -0.5\end{pmatrix} = 0.25.$$

This number represents the average payoff once the game is played many times.

Similarly, the expected loss of player 2 can be computed as $-x^T A\, y$.

Maximin/minimax criterion for mixed strategies

Idea: players minimize their maximum expect loss, that is, that maximum expected loss that can result from any possible random strategy picked by their opponent. Equivalently, players maximize their minimum expected payoff.

Note that, since the payoff matrix contains the payoff to player 1, we say that player 1 adopts a *maximin* strategy, whereas player 2 adopts a *minimax* strategy. Mixed-strategies that satisfy these criteria are called *optimal*.

Characterization of a maximin strategy 1/4

Let $X \coloneqq \{x \in \mathbb{R}^m : x \geq 0, \sum_{i=1}^{m} x_i = 1\}$ and $Y \coloneqq \{y \in \mathbb{R}^n : y \geq 0, \sum_{i=1}^{n} y_i = 1\}$ denote the sets of all feasible mixed strategies. By definition, an optimal strategy x^* for player 1 satisfies

$$x^* \in \operatorname*{argmax}_{z \in X} \min_{y \in Y} z^T A\, y$$

that is, it solves the problem

$$\max_{x \in \mathbb{R}^m} \min_{y \in Y} x^T A\, y \;\; s.t. \begin{cases} (1, \dots, 1)x = 1 \\ x \geq 0 \end{cases}$$

Characterization of a maximin strategy 2/4

An optimal strategy x^* for player 1 solves the problem

$$\max_{x\in\mathbb{R}^m} \min_{y\in Y} x^T A\, y \;\; s.t. \begin{cases} (1,\dots,1)x = 1 \\ x \geq 0 \end{cases}$$

For any $x \in \mathbb{R}^m$ fixed, let $v := \min_{y\in Y} x^T A\, y$, that is, $x^T A\, y \geq v$ is for any $y \in Y$.
This holds also for $y = (1,0,\dots,0)^T$, for $y = (0,1,0,\dots,0)^T$, and so forth. Hence,

$$\sum_{i=1}^{m} A_{ij} x_i \geq v \text{ for any } j = 1,\dots,n.$$

In fact, these two are equivalent because any $y \in Y$ satisfies $\sum_{i=1}^{n} y_i = 1$. To shorten the notation, we write

$$A^T x \geq (v,\dots,v)^T.$$

Characterization of a maximin strategy 3/4

An optimal strategy x^* for player 1 solves the problem

$$\max_{x \in \mathbb{R}^m} \min_{y \in Y} x^T A\, y \;\; s.t. \begin{cases} (1, \dots, 1)x = 1 \\ x \geq 0 \end{cases}$$

Since, for any $x \in \mathbb{R}^m$ fixed, the minimum $v = \min_{y \in Y} x^T A\, y$ is unique, it must be the biggest lower bound for the m inequalities $A^T x \geq (v, \dots, v)^T$. And since this is true for any $x \in \mathbb{R}^m$, an optimal strategy x^* for player 1 solves the problem

$$\max v \;\; s.t. \begin{cases} A^T x \geq (v, \dots, v)^T \\ (1, \dots, 1)x = 1 \\ x \geq 0, v \in \mathbb{R} \end{cases}$$

Characterization of a minimax strategy 4/4

An optimal strategy x^* for player 1 solves the linear programming problem

$$\max v \;\; s.t. \begin{cases} A^T x \geq (v, \dots, v)^T \\ (1, \dots, 1)x = 1 \\ x \geq 0, v \in \mathbb{R} \end{cases}$$

Similarly, an optimal strategy y^* for player 2 solves

$$\min w \; s.t. \begin{cases} Ay \leq (w, \dots, w)^T \\ (1, \dots, 1)y = 1 \\ y \geq 0, w \in \mathbb{R} \end{cases}$$

Equivalence of maximin and minimax outputs 1/3

Theorem: Let player 1 and 2 adopt optimal mixed strategies, and let p^* and d^* denote the optimal expected payoff and the optimal expected loss, respectively. Then, $p^* = d^*$, that is, the game is stable.

Proof: It suffices to show that the linear programming problem that characterizes the optimal strategy of player 2 is dual to the linear programming that characterizes the optimal strategy of player 1. Then, since both problems are feasible, the statement follows by strong duality.

Equivalence of maximin and minimax outputs 2/3

Proof: The primal problem is

$$\max v \;\; s.t. \begin{cases} A^T x \geq \; (v, \dots, v)^T \\ (1, \dots, 1)x = 1 \\ x \geq 0, v \in \mathbb{R} \end{cases}$$

Introduce the Lagrangian $L: \mathbb{R} \times \mathbb{R}^m \times \mathbb{R}^n_+ \times \mathbb{R} \times \mathbb{R}^m_+ \to \mathbb{R}$ defined by

$$L(v, x, y, w, z) \coloneqq v + y^T(A^T x \; - (v, \dots, v)^T) + w(1 - (1, \dots, 1)x) + \mathrm{z}^T x$$

Then, for any feasible (v, x), $L(v, x, w, y) \geq v$. Let $g: \mathbb{R}^n_+ \times \mathbb{R} \times \mathbb{R}^m_+ \to \; \mathbb{R}$ be defined by

$$\begin{aligned} g(y, w, z) &\coloneqq \max_{v,x} L(v, x, y, w, z) \\ &= \max_{v,x} v(1 \; - (1, \dots, 1)y) + x^T(Ay - (w, \dots, w)^T + z) + w \\ &= \begin{cases} w, & 1 \; - (1, \dots, 1)y = 0, Ay - (w, \dots, w)^T + z = 0 \\ \infty, & otherwise. \end{cases} \end{aligned}$$

UNIVERSITY OF LEICESTER

www.le.ac.uk

Equivalence of maximin and minimax outputs 3/3

Proof: Since

$$g(y,w,z) = \begin{cases} w, & 1 - (1,\dots,1)y = 0, Ay - (w,\dots,w)^T + z = 0 \\ \infty, & otherwise \end{cases}$$

the dual problem is

$$\min w \;\; s.t. \begin{cases} Ay = (w,\dots,w)^T - z \\ (1,\dots,1)y = 1 \\ y, z \geq 0, w \in \mathbb{R} \end{cases}$$

which is equivalent to

$$\min w \;\; s.t. \begin{cases} Ay \leq (w,\dots,w)^T \\ (1,\dots,1)y = 1 \\ y \geq 0, w \in \mathbb{R} \end{cases}$$

□

Summary and self-study

Summary: today we have learnt:

- about mixed-strategies for two-person zero-sum games,
- that the maximin/minimax criterion leads to stable solutions.

Self-study: Create your own unstable two-person zero-sum and solve it using mixed-strategies and the maximin/minimax criterion.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 17 – Queueing Theory

Dr Pak-Hin Lee

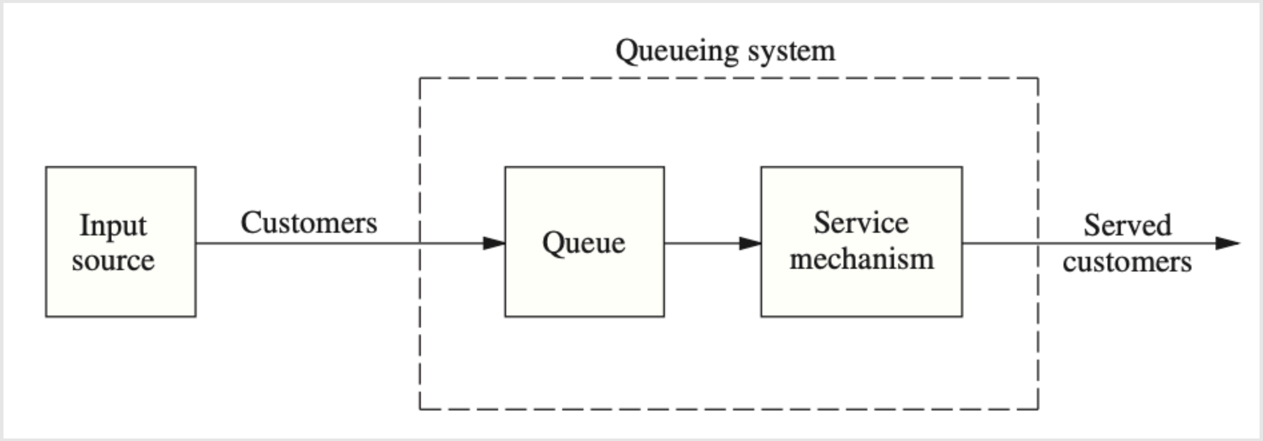
Recap and lecture outline

Summary: in the previous lectures we learnt the theory and some applications of linear programming.

In this lecture: Queueing Theory (following closely Ch. 17 of the book by Hillier and Lieberman).

Basic structure of queueing models

Model: *An input source* generates *customers*, who enter the *queueing system* to access a service and join a *queue* if service is not immediately available. A *queue discipline* prescribes when a queue member can access the *service mechanism*, which provides the desired service. After that, the customer leaves the queueing system.



The input source (calling population)

Definitions:

- Customers arrive from the so-called *calling population*.
- Its *size* is the total number of customers who will eventually access the service. This can be either infinite (easier model, default option) or finite (more technical, because the number of customers in the queue affects the number of potential new customers).
- Customers are generated according to a rule (often a stochastic process).

Assumption: Customers are generated one at a time. The time between consecutive arrivals is called *interarrival time*. Often it is further assumed that customer generation is not affected by the length of a queue (no *balking*).

The queue and the queue discipline

- The queue is where customers wait before being served. Often, it is assumed that the only way out is by being served (no *reneging*).
- A key property of a queue is how many customers it can contain simultaneously.
- Modelling with an infinite queue is easier, because the queue status (called *length*) does not affect the input source.
- The *queue discipline* indicates the order in which members of the queue are selected for service. The most common model is first-come-first-served (known as FIFO: first-in-first-out). Other modelling options are also possible, e.g., at random, or according to some priority procedure.

Example: the waiting room of an A&E unit cannot contain infinitely many patients (they are very rarely turned away, but often discouraged to visit), and it operates a queue discipline based on order of arrival and gravity of the patients' conditions.

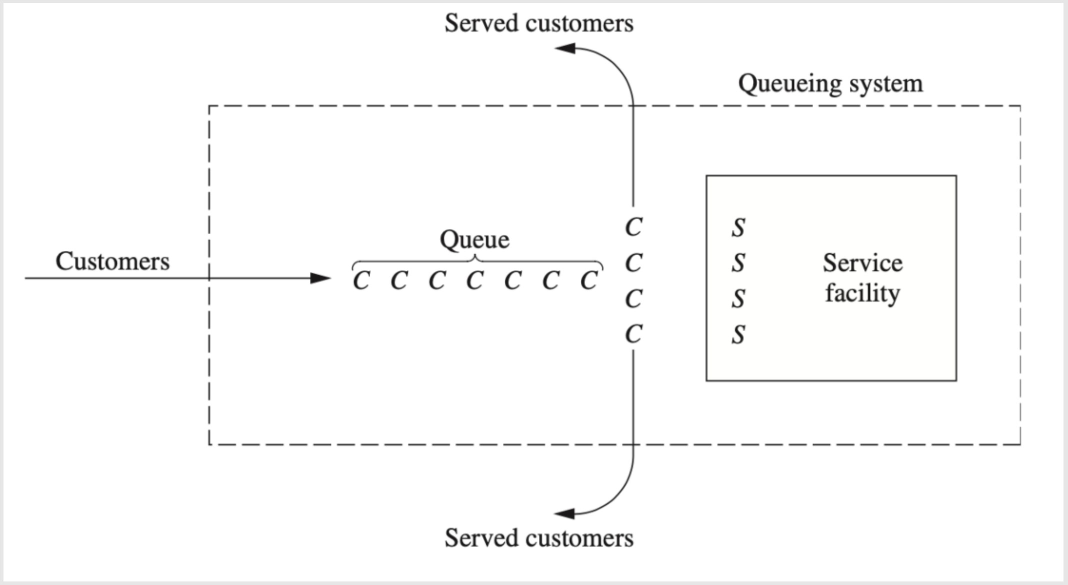
The service mechanism

- The service mechanism consists of one or more *service facilities* (in series).
- Each service facility contains one or more parallel service channels, the *servers*.
- Simple models have only one service facility with possibly more than one server.
- The *service time* is how long a server needs to deliver the service. This is commonly drawn from a probability distribution (which may depend both on the service specifics and the customer type).

Source: https://tinyurl.com/tollboothpic

Elementary queue model

Scenario: A single waiting line (possibly empty at times) in the front of a single service facility with one or more servers. Each customer generated by an input source is serviced by one of the servers after waiting in the queue (waiting line).



Elementary queue model – shortened notation

Common assumption: both interarrival and service times are independent and identically distributed random variables.

Shortened notation:

(Interarrival time distribution) / (Service time distribution) / (Number of servers)

Interarrival time and service distributions can be described by:

- M = exponential distribution (Markovian),
- D = degenerated distribution (constant times),
- E_k = Erlang distribution with shape parameter k,
- G = general distribution.

Terminology and Notation

- **Queue length:** #customers waiting for service to begin.
- $N(t)$ = #customers in queueing system at time t ($t \geq 0$).
- $p(n,t)$ = probability of exactly n customers in queueing system at time t.
- s = number of servers
- λ_n = mean arrival rate (expected number of arrivals per unit time) of new customers when n customers are in system.
- μ_n = mean service rate for overall system (expected number of customers completing service per unit time) when n customers are in system.

Remark: If λ_n and μ_n are independent on n, denote them λ and μ (the latter *per server*). In this case, λ^{-1} and μ^{-1} are the *expected interarrival and service times*, respectively, and $\rho := \frac{(s\mu)^{-1}}{\lambda^{-1}} = \frac{\lambda}{(s\mu)}$ is the *service facility utilization factor*.

Transient and steady-state and conditions

When a queueing system is switched on, the number $N(t)$ of customers in the system varies in time and depends heavily on the initial condition $N(0)$. This is called a *transient condition*. In most instances, however, the systems stabilizes after a certain time and reaches the so-called *steady-state condition*.

Assuming the system is in a steady-state condition, let:

- p_n : the probability of having exactly n customers in the queueing system
- $\bar{\lambda} \ := \ \sum_{n=0}^{\infty} \lambda_n p_n$: average arrival rate over the long run
- $L = \sum_{n=0}^{\infty} n \, p_n$: the expected number of customers in the queueing system
- $L_q = \sum_{n=s}^{\infty} (n - s) \, p_n$: the expected queue length
- W : the expected stay in the system for each individual customer
- W_q : the expected stay in the queue for each individual customer
- W_s : the mean service time

Little's formulae

Little, J. D. C. 1961. A proof for the queuing formula: $L = \lambda W$. *Oper. Res.* 9(3) 383–387.

Theorem: In a steady-state, it holds

$$L = \bar{\lambda} W, \qquad L_q = \bar{\lambda} W_q, \qquad W = W_q + W_s$$

Note that this is independent on the interarrival time distribution, the service time distribution, and the queue discipline!

Example: If,

- I receive an email every $\lambda^{-1} = 5$ minutes (on average),
- I read-reply-delete each email after $W = 20$ minutes (on average),
- and the total number of emails in my inbox is constant (steady-state condition),

then, at any given time there will be $\lambda W = 4$ emails in my inbox. And if it takes me $\mu^{-1} = W_s = 3$ minutes to answer an email, then emails tend to stay unread for $W_q = 17$ minutes.

Summary and self-study

Summary: today we have learnt:

- the basic principles of queueing theory,
- how to use Little's formula for steady states.

Self-study: Create your own queue problem and use Little's formula to determine the average number of customers in the queueing system.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 18 – The exponential distribution

Dr Pak-Hin Lee

Recap and lecture outline

Summary: so far, we have learnt about the rudiments of queueing theory and Little’s formulae.

In this lecture: The exponential distribution (following closely Ch. 17.4 of the book by Hillier and Lieberman).

The exponential distribution

The key characteristics of a queue are the probability distribution of interarrival times and of service times.

In general, these could be whatever, but the problem may become impossible to analyse. A probability distribution with a good balance between applicability and complexity of the resulting queueing model is the exponential distribution.

Definition: A random variable T has an exponential distribution with parameter $\alpha > 0$ if its probability density function is

$$f_T(t) = \begin{cases} \alpha \exp(-\alpha t), & t \geq 0, \\ 0, & t < 0, \end{cases}$$

that is, for any $a, b \in \mathbb{R}$, $\mathrm{P}[a \leq T \leq b] = \int_a^b f_T(x)\,dx$.

The exponential distribution – property 0

Proposition: Let $t > 0$ and let T have an exponential distribution with parameter $\alpha > 0$. Then, $\mathrm{P}[T \le t] = 1 - \exp(-\alpha t)$, $\mathrm{E}[\mathrm{T}] = \alpha^{-1}$, $\mathrm{Var}[\mathrm{T}] = \alpha^{-2}$.

Proof:

$$\mathrm{P}[T \le t] = \int_{-\infty}^{t} f_T(x)\, dx = \int_0^t \alpha e^{-\alpha x}\, dx = [-e^{-\alpha x}]_0^t = 1 - e^{-\alpha t},$$

$$\mathrm{E}[T] = \int_{-\infty}^{\infty} x f_T(x)\, dx = \int_0^{\infty} \alpha x e^{-\alpha x}\, dx = [-x e^{-\alpha x}]_0^{\infty} + \int_0^{\infty} e^{-\alpha x}\, dx = \alpha^{-1},$$

$$\mathrm{Var}[\mathrm{T}] = \mathrm{E}[T^2] - \mathrm{E}[T]^2 = \int_{-\infty}^{\infty} x^2 f_T(x)\, dx - \alpha^{-2} = \int_0^{\infty} x^2 \alpha e^{-\alpha x}\, dx - \alpha^{-2}$$

$$= [-x^2 e^{-\alpha x}]_0^{\infty} + 2\int_0^{\infty} x e^{-\alpha x}\, dx - \alpha^{-2} = \alpha^{-2}.$$

The exponential distribution – property 1

Property 1: For $t > 0$, $f_T(t)$ is strictly decreasing. Therefore, for any $dt > 0$,

$$\mathrm{P}[0 \leq T \leq dt] > \mathrm{P}[t \leq T \leq t + dt].$$

This implies that T is more likely to take values near zero.

If this is used for the interarrival time, it implies that it is unlikely that we need to wait for a long time before a new customer arrives.

This works also well to model service times when the service time depends on the customer (e.g., answering emails). It's less appropriate if the service takes the same time for everyone (e.g., a ride on a roller coaster, in which case the duration of a journey is essentially constant).

The exponential distribution – property 2

Property 2 [Lack of memory]: For $t, dt > 0$,

$$P[T > t + dt \mid T > t] = P[T > dt].$$

Proof:

$$P[T > t + dt \mid T > t] = \frac{P[\{T > t + dt\} \cap \{T > t\}]}{P[T > t]} = \frac{P[T > t + dt]}{P[T > t]}$$

$$= \frac{\exp\big(-\alpha(t + dt)\big)}{\exp(-\alpha t)} = \exp(-\alpha\, dt) = P[T > dt]$$

□

This means that the probability of something happening does not depend on how much time has passed.

The exponential distribution – property 3

Property 3: For any $t \geq 0$ and and small $dt > 0$, $\mathrm{P}[T \leq t + dt \mid T > t] \cong \alpha dt$.

Proof: Using property 2

$$\begin{aligned} \mathrm{P}[T \leq t + dt \mid T > t] &= 1 - \mathrm{P}[T > t + dt \mid T > t] \\ &= 1 - \mathrm{P}[T > dt] = \mathrm{P}[T \leq dt] = 1 - \exp(-\alpha\, dt). \end{aligned}$$

Since, for $dt > 0$ small, $\exp(-\alpha\, dt) \cong 1 - \alpha\, dt$ (by Taylor expansion). We conclude that $\mathrm{P}[T \leq t + dt \mid T > t] \cong \alpha dt$.

$\square$

The exponential distribution – property 4

Property 4: The minimum of finitely many independent exponential random variables has an exponential distribution.

Proof: Let $T_1, T_2, \dots, T_n$ be independent exponential random variables with parameters $\alpha_1, \alpha_2, \dots, \alpha_n > 0$. Let $U \coloneqq \min(T_1, T_2, \dots, T_n)$. Then, for any $t > 0$,

$$P[U > t] = P[T_1 > t, T_2 > t, \dots, T_n > t] = P[T_1 > t]P[T_2 > t] \cdots P[T_n > t]$$
$$= \exp(-\alpha_1 t)\exp(-\alpha_2 t) \cdots \exp(-\alpha_n t) = \exp(-(\alpha_1 + \alpha_2 + \cdots + \alpha_n)t),$$

that is, U has an exponential distribution with parameter $\alpha = \alpha_1 + \alpha_2 + \cdots + \alpha_n$. □

If there are different types of customers, each with their own interarrival time modelled via an exponential distribution, then the interarrival time for a generic customer (a customer of whatever type) is also modelled via an exponential distribution. Similarly, if there are different servers with different execution speeds, the time until the next service completion is also modelled via an exponential distribution, and the queueing system performs like a single very quick server.

The exponential distribution – property 5

Property 5: Let $X(t)$ denote how many times an event T with exponential distribution (with parameter α) occurs by time $t \geq 0$. Then, $X(t)$ has a *Poisson distribution* with parameter αt, that is, for any $n \in \mathbb{N}$, $P[X(t) = n] = \frac{(\alpha t)^n \exp(-\alpha t)}{n!}$.

Proof: (sketch) Let $T_1, T_2, \dots T_{n+1}$ be $n+1$ independent copies of T. Then, the sum $T_1 + T_2 + \cdots + T_{n+1}$ is itself a random variable, and its distribution is an *Erlang distribution* with parameters $(n+1, \alpha)$, that is,

$$P[T_1 + T_2 + \ \dots + T_{n+1} \leq x] = 1 \ - \sum_{k=0}^{n} \frac{(\alpha x)^k \exp(-\alpha x)}{k!}$$

Since T occurs at most n times before a set time t iff $T_1 + T_2 + \cdots + T_{n+1} > t$,

$$P[X(t) \leq n] = P[T_1 + T_2 + \ \dots + T_{n+1} > t] = \sum_{k=0}^{n} \frac{(\alpha t)^k \exp(-\alpha t)}{k!}.$$

The statement follows from $P[X(t) = n] = P[X(t) \leq n] - P[X(t) \leq n - 1]$. □

Expected value of a Poisson distribution

Proposition: Let $X(t)$ have *Poisson distribution* with parameter αt. Then,

$$\mathrm{E}[X(\mathrm{t})] = \alpha t.$$

Proof:

$$\mathrm{E}[X(\mathrm{t})] = \sum_{n=0}^{\infty} n\mathrm{P}[X(t) = n] = \sum_{n=0}^{\infty} n \frac{(\alpha t)^n \exp(-\alpha t)}{n!}$$

$$= \sum_{n=1}^{\infty} \frac{(\alpha t)^n \exp(-\alpha t)}{(n-1)!} = \alpha t \exp(-\alpha t) \sum_{n=1}^{\infty} \frac{(\alpha t)^{n-1}}{(n-1)!} =$$

$$= \alpha t \exp(-\alpha t) \sum_{n=0}^{\infty} \frac{(\alpha t)^n}{(n)!} = \alpha t \exp(-\alpha t) \exp(\alpha t) = \alpha t.$$

□

Sum of independent Poisson processes

Proposition: Let $X_1(t)$ and $X_2(t)$ be independent Poisson random variables with parameters $\alpha_1 t$ and $\alpha_2 t$, respectively. Then, $X(t) := X_1(t) + X_2(t)$ is a Poisson random variable with parameter $(\alpha_1 + \alpha_2)t$.

Proof:

$$\mathrm{P}[X(t) = n] = \mathrm{P}[X_1(t) + X_2(t) = n] = \sum_{k=0}^{n} \mathrm{P}[X_1(t) = k]\mathrm{P}[X_2(t) = n - k]$$

$$= \sum_{k=0}^{n} \frac{(\alpha_1 t)^k \exp(-\alpha_1 t)}{k!} \frac{(\alpha_2 t)^{n-k} \exp(-\alpha_2 t)}{(n-k)!}$$

$$= \frac{\mathrm{t}^n \exp(-(\alpha_1 + \alpha_2)t)}{n!} \sum_{k=0}^{n} \frac{n!\, \alpha_1^k \alpha_2^{n-k}}{k!\,(n-k)!}$$

$$= \frac{(\alpha_1 + \alpha_2)^n \mathrm{t}^n \exp(-(\alpha_1 + \alpha_2)t)}{n!}$$

$\square$

Splitting of a Poisson process 1/2

Proposition: Let $X(t)$ be a Poisson random variable with parameter αt. For a fixed value p, assign each arrival to $X_1(t)$ with probability p, and to $X_2(t)$ otherwise. Then,

- $X_1(t)$ is a Poisson random variable parameter $p\alpha t$,
- $X_2(t)$ is a Poisson random variable parameter $(1-p)\alpha t$,
- $X_1(t)$ and $X_2(t)$ are independent.

Proof: ($X_1(t)$ is a Poisson random variable with parameter $p\alpha t$)

$$\mathrm{P}[X_1(t)=k] = \sum_{m=0}^{\infty} \mathrm{P}[X_1(t)=k \mid X(t)=m]\mathrm{P}[X(t)=m]$$

$$= \sum_{m=k}^{\infty} \frac{m!\,p^k(1-p)^{m-k}}{k!\,(m-k)!}\frac{(\alpha t)^m \exp(-\alpha t)}{m!} = \frac{(p\alpha t)^k \exp(-\alpha t)}{k!}\sum_{m=k}^{\infty}\frac{(\alpha t(1-p))^{m-k}}{(m-k)!}$$

$$= \frac{(p\alpha t)^k \exp(-\alpha t)}{k!}\exp\big(\alpha t(1-p)\big) = \frac{(p\alpha t)^k \exp(-p\alpha t)}{k!}$$

Splitting of a Poisson process 2/2

Proof: (independence)

$$\mathrm{P}[X_1(t) = i, X_2(t) = j] = \sum_{m=0}^{\infty} \mathrm{P}[X_1(t) = i, X_2(t) = j \mid X(t) = m]\mathrm{P}[X(t) = m]$$

$$= P[X_1(t) = i, X_2(t) = j \mid X(t) = i + j]P[X(t) = i + j]$$

$$= P[X_1(t) = i \mid X(t) = i + j]P[X(t) = i + j]$$

$$= \frac{(i+j)!\, p^i (1-p)^j}{i!\, j!} \frac{(\alpha t)^{i+j} \exp(-\alpha t)}{(i+j)!}$$

$$= \frac{(p\alpha t)^i \exp(-p\alpha t)}{i!} \frac{(\alpha t(1-p))^j \exp(-\alpha t(1-p))}{j!} = \mathrm{P}[X_1(t) = i]\mathrm{P}[X_2(t) = j].$$

□

UNIVERSITY OF LEICESTER

www.le.ac.uk

Summary and self-study

Summary: today we have learnt:

- about the exponential distribution and its properties,
- about Poisson processes and their properties.

Self-study: Let the random variable T have exponential distribution with parameter $\alpha = 1$.

- Compute the expected value and the variance of T.
- Let V and W be independent copies of T. Compute the quantities $\mathrm{P}[1 \leq T \leq 3 | V \leq 3]$ and $\mathrm{P}[T + V + W \leq 1]$.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 19 – The M/M/1 queueing system

Dr Pak-Hin Lee

Recapitulation and lecture outline

Summary: so far we have learnt:

- about abstract models for queueing systems,
- about Little's formulae,
- about the exponential distribution and its main properties.

In this lecture: The M/M/1 queueing system (following not-so-closely Ch. 17.6 of the book by Hillier and Lieberman).

The M/M/1 model

In the M/M/1 model:

- the interarrival time of customers is a random variable I with exponential distribution with parameter λ; let $N_I(t)$ denote the number of arrivals by time t.
- the service time is a random variable S with exponential distribution with parameter μ; let $N_S(t)$ denote the number of serviced customers by time t.
- there is only one server.

Goal: study the evolution of $N(t)$:= #customers in the queueing system.

The M/M/1 model - $p(0, t + dt)$

Let $p(n,t)$ denote the probability that $N(t) = n$, let $dt > 0$. Then, if $n = 0$,

$$
\begin{aligned}
&\{N(t+dt) = 0\} = \\
&\{N(\mathrm{t}) = 0\} \cap \{N_I(dt) = 0 \cup N_I(dt) = 1 \cap N_S(dt) = 1 \cup \ \dots\} \\
&\cup \{N(t) = 1\} \cap \{N_I(dt) = 0 \cap N_S(dt) = 1 \cup N_I(dt) = 1 \cap N_S(dt) = 2 \cup \cdots\} \\
&\cup \{N(t) = 2\} \cap \{N_I(dt) = 0 \cap N_S(dt) = 2 \cup N_I(dt) = 1 \cap N_S(dt) = 3 \cup \cdots\} \cup \cdots \\
&= \{N(\mathrm{t}) = 0\} \cap \{N_I(dt) = 0\} \cup \{N(t) = 1\} \cap \{N_I(dt) = 0 \cap N_S(dt) = 1\} \cup \cdots
\end{aligned}
$$

$$
\begin{aligned}
&p(0, t+dt) = P[N(t+dt) = 0] = \\
&= \mathrm{P}[N(\mathrm{t}) = 0, N_I(dt) = 0] + \mathrm{P}[N(t) = 1, N_I(dt) = 0, N_S(dt) = 1] + \mathrm{P}[\dots] \\
&= \mathrm{P}[N(t) = 0]\mathrm{P}[N_I(dt) = 0] + \mathrm{P}[N(\mathrm{t}) = 1]\mathrm{P}[N_I(dt) = 0]\mathrm{P}[N_S(dt) = 1] + \mathrm{P}[\dots]
\end{aligned}
$$

The M/M/1 model - $p(0, t+dt)$

Let $p(n,t)$ denote the probability that $N(t) \; = \; n$ and let $dt > 0$. Then, if $n = 0$,

$$p(0, t+dt)$$
$$= \mathrm{P}[N(t) = 0]\mathrm{P}[N_I(dt) = 0] + \mathrm{P}[N(\mathrm{t}) = 1]\mathrm{P}[N_I(dt) = 0]\mathrm{P}[N_S(dt) = 1] + \mathrm{P}[\ldots]$$

Recall that, for $dt > 0$ small,

$$P[N_I(dt) = 0] = \frac{(\lambda dt)^0 \exp(-\lambda dt)}{0!} \cong 1 - \lambda dt,$$
$$P[N_S(dt) = 1] = \frac{(\mu dt)^1 \exp(-\mu dt)}{1!} \cong \; \mu dt,$$

Therefore, $p(0, t+dt) = p(0,t)(1 - \lambda dt) + p(1,t)\mu dt + dt^2(\ldots)$

The M/M/1 model - $p(n, t + dt)$

Similarly, for $n > 0$,

$$
\begin{aligned}
&\{N(t + dt) = n\} \\
&= \bigcup_{m=0}^{\infty} \{N(t) = m\} \cap \bigcup_{k=n-m}^{\infty} \{N_I(dt) = k\} \cap \{N_S(dt) = k - (n - m)\} \\
&= \{N(t) = n - 1\} \cap \{N_I(dt) = 1\} \cap \{N_S(dt) = 0\} \\
&\quad \cup \{N(t) = n\} \cap \{N_I(dt) = 0\} \cap \{N_S(dt) = 0\} \\
&\quad \cup \{N(t) = n + 1\} \cap \{N_I(dt) = 0\} \cap \{N_S(dt) = 1\} \cup \cdots
\end{aligned}
$$

Therefore,

$$p(n, t + dt) = p(n - 1, t)\lambda dt + p(n, t)(1 - \lambda dt - \mu dt) + p(n + 1, t)\mu dt + dt^2(\ldots)$$

The M/M/1 model – Kolmogorov equations

We just showed that, after dropping dt^2 terms,

$$p(0,t+dt) \cong p(0,t)(1-\lambda dt) + p(1,t)\mu dt$$
$$p(n,t+dt) \cong p(n-1,t)\lambda dt + p(n,t)(1-\lambda dt-\mu dt) + p(n+1,t)\mu dt,$$

that is,

$$\frac{p(0,t+dt)-p(0,t)}{dt} \cong p(1,t)\mu - p(0,t)\lambda$$
$$\frac{p(n,t+dt)-p(n,t)}{dt} \cong p(n-1,t)\lambda - p(n,t)(\lambda+\mu) + p(n+1,t)\mu$$

Passing to the limit $dt \to 0$, we obtain the so-called *Kolmogorov equations*

$$\partial_t p(0,t) = p(1,t)\mu - p(0,t)\lambda$$
$$\partial_t p(n,t) = p(n-1,t)\lambda - p(n,t)(\lambda+\mu) + p(n+1,t)\mu$$

The M/M/1 model – transient behaviour

To study the transient behavior of the M/M/1 model, we need to set some initial conditions and then solve

$$\partial_t p(0,t) = p(1,t)\mu - p(0,t)\lambda$$
$$\partial_t p(n,t) = p(n-1,t)\lambda - p(n,t)(\lambda+\mu) + p(n+1,t)\mu$$

under the additional constrains: $p(i,t) \geq 0, p(0,t) + p(1,t) + \ ... = 1$

The Kolmogorov's equation for the M/M/1 model are solvable analytically, but the formulas are nasty and involve Bessel functions with imaginary arguments. And this is just the simplest model! Numerically, we can simulate a queue of (sufficiently long but) finite length (will look at this next week).

M/M/1 model – steady states

Setting $\partial_t p(0,t) = 0$ and $\partial_t p(n,t) = 0$, we obtain

$$p(1,t)\mu = p(0,t)\lambda$$
$$p(n,t)(\lambda+\mu) = p(n-1,t)\lambda + p(n+1,t)\mu$$

Proposition: Assuming the queue can be infinitely long, the solution to the linear recurrence relation above is

$$p(n,t) = \left(\frac{\lambda}{\mu}\right)^n p(0,t)$$

Proof: To shorten the notation, let $p_n := p(n,t)$. Since $p_1 = \frac{\lambda}{\mu} p_0$, proceeding by induction we obtain

$$p_{n+1} = p_n\left(\frac{\lambda}{\mu}+1\right) - p_{n-1}\frac{\lambda}{\mu} = \left(\frac{\lambda}{\mu}\right)^n p_0\left(\frac{\lambda}{\mu}+1\right) - \left(\frac{\lambda}{\mu}\right)^{n-1} p_0\frac{\lambda}{\mu} = \left(\frac{\lambda}{\mu}\right)^{n+1} p_0$$

$\square$

M/M/1 model – existence of steady states

Recall that $\rho \coloneqq \frac{\lambda}{\mu}$ is the *service facility utilisation factor*. The steady state $\{p_n\}_{n\geq 0}$ exists if $\sum_{n=0}^{\infty} p_n = 1$. However, since, $p_n = \rho^n p_0$, if $\rho < 1$

$$\sum_{n=0}^{\infty} p_n = \sum_{n=0}^{\infty} \rho^n p_0 = \frac{p_0}{1-\rho},$$

we conclude that the steady state exists only if $p_0 = 1 - \rho$. Note that this implies that $\lambda < \mu$ must hold.

Example: If I receive an email every $\lambda^{-1} = 5$ minutes and it takes me takes me $\mu^{-1} = 3$ minutes to answer it, then $\rho = \frac{3}{5} = 0.6$, $p_0 = 0.4$, $p_1 = 0.24$, ...

M/M/1 model – server idle time

The server does not work if there are no customers in the system. This happens with probability $p_0 = 1 - \rho$, which implies that the server is busy with probability ρ.

This is compatible with the interpretation of $\rho \coloneqq \frac{\lambda}{\mu}$ being the *service facility utilisation factor.*

If $\mu \gg \lambda$, meaning that service time is much faster then the arrival of customers, then $\rho \ll 1$, meaning that the server is often idle.

Note that this does not depend on the queue discipline!

M/M/1 model – example

If I receive an email every $\lambda^{-1} = 5$ minutes and it takes me takes me $\mu^{-1} = 3$ minutes to answer it, then $\rho = \frac{3}{5} = 0.6$ and $p_0 = 0.4$. The expected number of emails is in my inbox is

$$L = \sum_{n=0}^{\infty} n\, p_n = 0.4 \sum_{n=0}^{\infty} n\, 0.6^n = 0.4 \frac{0.6}{(1-0.6)^2} = 1.5.$$

In fact, a smidge more generally, in any M/M/1 we have

$$L = \sum_{n=0}^{\infty} n\, p_n = (1-\rho) \sum_{n=0}^{\infty} n\, \rho^n = (1-\rho) \frac{\rho}{(1-\rho)^2} = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu-\lambda}$$

Then, using Little's formula, we conclude that the expected stay of an email in my inbox is $W = \lambda^{-1} L = 7.5$ minutes. In the next slide we show that $W = (\mu - \lambda)^{-1}$.

M/M/1 model – Waiting Time Distribution 1/2

Proposition: Let the queue discipline be implemented via a first-in-first-out policy. Then, the customer waiting time w is a random variable with exponential distribution with parameter $\mu - \lambda > 0$, and $W = \mathrm{E}[w] = (\mu - \lambda)^{-1}$.

Proof: A customer waits longer than $t \geq 0$ if there are n customers ahead of them in the queue and providing $n + 1$ services takes longer than t, that is,

$$\mathrm{P}[w > t] = \sum_{n=0}^{\infty} p_n \mathrm{P}[S_1 + S_2 + \cdots + S_{n+1} > t]$$

$$= \sum_{n=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^n p_0 \sum_{k=0}^{n} \frac{(\mu t)^k \exp(-\mu t)}{k!} =$$

$$= \exp(-\mu t)\, p_0 \sum_{n=0}^{\infty} \left(\frac{\lambda}{\mu}\right)^n \sum_{k=0}^{n} \frac{(\mu t)^k}{k!} = \exp(-(\mu - \lambda)t)\,,$$

where the last step is shown in the next slide.

M/M/1 model – Waiting Time Distribution 2/2

$$p_0 \sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^n \sum_{k=0}^{n} \frac{(\mu t)^k}{k!} = \left(1-\frac{\lambda}{\mu}\right)\sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^n \sum_{k=0}^{n} \frac{(\mu t)^k}{k!}$$

$$= 1 + \sum_{n=1}^{\infty}\left(\frac{\lambda}{\mu}\right)^n \sum_{k=0}^{n} \frac{(\mu t)^k}{k!} - \sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^{n+1} \sum_{k=0}^{n} \frac{(\mu t)^k}{k!}$$

$$= 1 + \sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^{n+1} \left(\frac{(\mu t)^{n+1}}{(n+1)!} + \sum_{k=0}^{n} \frac{(\mu t)^k}{k!}\right) - \sum_{n=0}^{\infty}\left(\frac{\lambda}{\mu}\right)^{n+1} \sum_{k=0}^{n} \frac{(\mu t)^k}{k!}$$

$$= 1 + \sum_{n=0}^{\infty} \frac{(\lambda t)^{n+1}}{(n+1)!} = \sum_{n=0}^{\infty} \frac{(\lambda t)^n}{n!} = \exp(\lambda t)$$

$\square$

Note: the derivation is easier if one uses the equivalent formula

$$P[S_1 + \cdots + S_{n+1} > t] = \int_t^{\infty} \frac{\mu(\mu x)^n \exp(-\mu x)}{n!}\, dx\,.$$

M/M/1 model – average waiting time

Since the customer waiting time w is a random variable with exponential distribution with parameter $\mu - \lambda > 0$, and $W = \mathrm{E}[w] = (\mu - \lambda)^{-1}$, we can compute

$$\mathrm{P}[w > W] = \exp(-(\mu - \lambda)W) = \exp(-(\mu - \lambda)(\mu - \lambda)^{-1}) = \exp(-1) \cong 0.368,$$

that is, most customers wait for less time than the average, but some unlucky ones wait for a long time.

Summary and self-study

Summary: today we have learnt:

- how to derive Kolmogorov's equations for M/M/1 model,
- how to determine the steady states of M/M/1 model,
- that its waiting times have exponential distribution.

Self-study: Consider an M/M/1 model with $\lambda = 2$ and $\mu = 3$ and FIFO queue discipline. Compute p_0, L, and W assuming that the queue can be infinitely long.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 20 – Birth and death processes

Dr Pak-Hin Lee

Recap and lecture outline

Recap: so far, we have learnt:

- how to describe abstract queueing systems models,
- the relationships between the exponential distribution and Poisson processes,
- how to analyse the M/M/1 system.

In this lecture: The birth-and-death process to generalise the M/M/1 model (following more-or-less-closely Ch. 17.5 and Ch. 17.6 of the book by Hillier and Lieberman).

The M/M/1 model - summary

Assumptions:

- the interarrival time of customers has exponential distribution with parameter λ.
- the service time has exponential distribution with parameter μ,
- there is only one server.

Goal: Study the evolution of $N(t)$ (#customers in the queueing system).

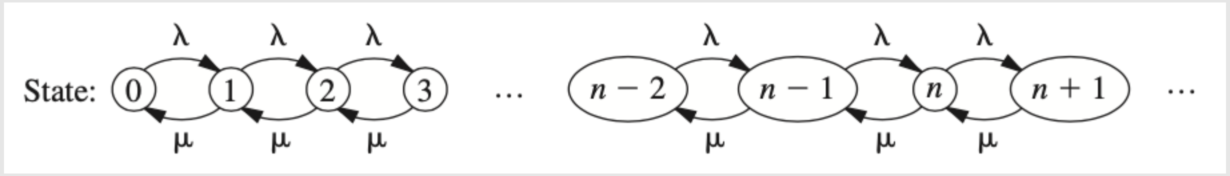
Answer: Let $p(n,t)$ denote the probability that $N(t) = n$. Then,

$$\partial_t p(0,t) = p(1,t)\mu - p(0,t)\lambda$$
$$\partial_t p(n,t) = p(n-1,t)\lambda - p(n,t)(\lambda+\mu) + p(n+1,t)\mu$$

The steady-states are: $p_0 = 1-\rho$ and $p_n = \left(\frac{\lambda}{\mu}\right)^n p_0 = \rho^n\, p_0$ for $n > 0$.

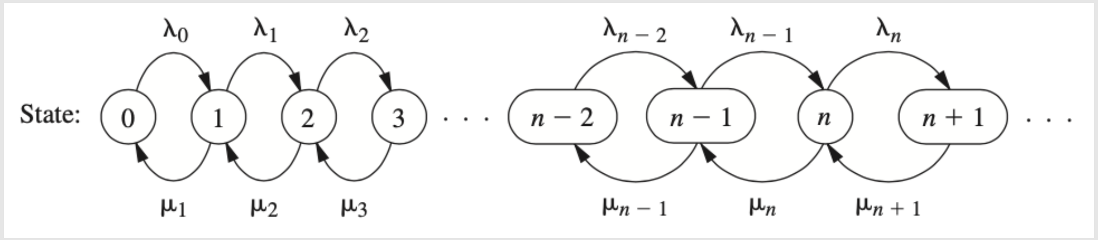
The M/M/1 model as a birth-and-death process

We can sketch the behaviour of $N(t)$ with the following diagram:



The circles denote the possible states of $N(t)$, whereas the arrows denote the parameters of the *average birth and death rates*, respectively

We now generalise this by letting these parameters depend on the state of $N(t)$:



Birth-and-death processes 1/2

Let $p(n,t)$ denote the probability that $N(t) = n$. By repeating the same derivation as for the M/M/1 model, we can show that

$$\begin{aligned} \partial_t p(0,t) &= p(1,t)\mu_1 - p(0,t)\lambda_0 \\ \partial_t p(n,t) &= p(n-1,t)\lambda_{n-1} - p(n,t)(\lambda_n + \mu_n) + p(n+1,t)\mu_{n+1} \quad \text{for } n \geq 1. \end{aligned}$$

The associated steady-states equations are :

$$\begin{aligned} p_1\mu_1 &= p_0\lambda_0 \\ p_{n-1}\lambda_{n-1} + p_{n+1}\mu_{n+1} &= p_n(\lambda_n + \mu_n) \quad \text{for } n \geq 1, \end{aligned}$$

where $p_i := p(i,t)$

Birth-and-death processes 2/2

By induction we can prove that the solution to the steady-states equations

$$p_1\mu_1 = p_0\lambda_0$$
$$p_{n-1}\lambda_{n-1} + p_{n+1}\mu_{n+1} = p_n(\lambda_n + \mu_n) \quad \text{for } n \geq 1,$$

satisfies $p_{n+1}= \frac{\lambda_n}{\mu_{n+1}} p_n$, that is, $p_n = c_n p_0$ for $n > 0$, where

$$c_n = \frac{\lambda_{n-1}\lambda_{n-2} \dots \lambda_0}{\mu_n\mu_{n-1} \dots \mu_1}.$$

The constraint $\sum_{n=0}^{\infty} p_n = p_0 + \sum_{n=1}^{\infty} c_n p_0 = 1$ implies that $p_0 = (1 + \sum_{n=1}^{\infty} c_n)^{-1}$.

Finally, we can adapt Little's law to cover birth-and-death processes. Let $\bar{\lambda} \coloneqq \sum_{n=0}^{\infty} \lambda_n p_n$ denote the mean arrival rate. Then $L = \bar{\lambda}\, W$ and $L_q = \bar{\lambda}\, W_q$

Example

The owner of (and unique worker in) the small local coffee shop at the corner has decided to hire a trainee. When there's only one customer, the trainee takes the order and prepares the required beverages. However, if there's more than one customer, the owner steps in to provide service to an additional customer. On average, the trainee needs 6 minutes to serve a customer, whereas the owner only needs 3 minutes.

Assuming these have exponential distribution and that customers arrive at the coffee shop following a Poisson distribution with mean rate of 20 customers per hour, compute: the steady-states of the number of customers in the coffee shop, the probability that the queue is empty, the expected number of customers and of customers in the queue, the associated waiting times, and the mean service time.

Example – solution 1/3

Since customers arrive at the coffee shop following a Poisson distribution with mean rate of 20 customers per hour, and since this does not depend on how many customers are already in the coffee shop, we conclude that

$$\lambda_0 = \lambda_1 = \lambda_2 = \cdots = \overline{\lambda} = 20.$$

Since only the trainee works when there's one customer, but both work if there's more than one customer, the service rates are $\mu_1 = \frac{60}{6} = 10$ (customers per hour) and $\mu_2 = \mu_3 = \cdots = \frac{60}{6} + \frac{60}{3} = 30$.

Therefore, $c_1 = \frac{\lambda_0}{\mu_1} = \frac{20}{10} = 2$ and $c_n = \frac{20}{10}\left(\frac{20}{30}\right)^{n-1} = 2\left(\frac{2}{3}\right)^{n-1}$ for $n > 1$.

Example – solution 2/3

Since $c_1 = 2$ and $c_n = 2\left(\frac{2}{3}\right)^{n-1}$ for $n > 1$, we conclude that

$$p_0 = \left(1 + \sum_{n=1}^{\infty} c_n\right)^{-1} = \left(1 + 2 + 2\sum_{n=2}^{\infty}\left(\frac{2}{3}\right)^{n-1}\right)^{-1} = \left(1 + 2\sum_{n=0}^{\infty}\left(\frac{2}{3}\right)^{n}\right)^{-1} = \frac{1}{7}$$

and that $p_n = c_n p_0 = \frac{2}{7}\left(\frac{2}{3}\right)^{n-1}$ for $n > 0$.

The queue is empty if there are at most two customers. The probability of this event (in a steady-state scenario) is $p_0 + p_1 + p_2 = \frac{1}{7} + \frac{2}{7} + \frac{4}{21} = \frac{13}{21}$.

Example – solution 3/3

The mean number of customers and the mean length of the queue are

$$L = \sum_{n=0}^{\infty} n\, p_n = \sum_{n=1}^{\infty} n \frac{2}{7}\left(\frac{2}{3}\right)^{n-1} = \frac{2}{7}\sum_{n=1}^{\infty} n\left(\frac{2}{3}\right)^{n-1} = \frac{2}{7}\frac{3}{2}\sum_{n=1}^{\infty} n\left(\frac{2}{3}\right)^{n}$$

$$= \frac{3}{7}\sum_{n=1}^{\infty} n\left(\frac{2}{3}\right)^{n} = \frac{\frac{3}{7}\frac{2}{3}}{\left(1-\frac{2}{3}\right)^{2}} = \frac{18}{7}$$

$$L_q = \sum_{n=2}^{\infty}(n-2)\, p_n = \sum_{n=0}^{\infty} n \frac{2}{7}\left(\frac{2}{3}\right)^{n+1} = \frac{4}{21}6 = \frac{8}{7}$$

Finally, by Little's formula,

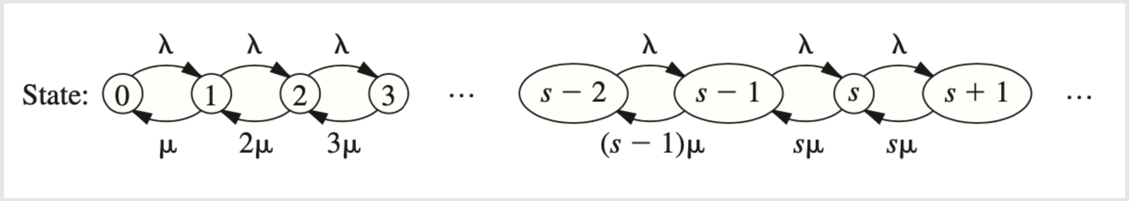
$$W = \frac{\mathrm{L}}{\bar{\lambda}} = \frac{9}{70} \cong 7'42'', \qquad W_q = \frac{L_q}{\bar{\lambda}} = \frac{4}{70} \cong 3'25'', \qquad W_s = W - W_q = 4'17''$$

The M/M/s model 1/2

Assume that, instead of only one, we now have $s > 1$ independent servers with exponential distribution with parameter μ. In the associated birth-and-death process, the system service rate μ_n represents the mean rate of service completions for the overall queueing system when there are n customers in the system. In this case we distinguish two cases:

- if $n \leq s$, then $\mu_n = n\mu$, (some server are inactive),
- otherwise, if $n > s$, then $\mu_n = s\mu$ (all servers are active).

If the interarrival rate of customers is exponentially distributed with parameter $\lambda_0 = \lambda_1 = \cdots = \lambda$, we can summarise the M/M/s model with the following sketch.



The M/M/s model 2/2

In this case, since $\mu_n = \min(n, s)\mu$,

$$c_n = \frac{\lambda_{n-1}\lambda_{n-2}\dots\lambda_0}{\mu_n\mu_{n-1}\dots\mu_1} = \begin{cases} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n, & if\ n \le s \\ \frac{1}{s!\,s^{n-s}}\left(\frac{\lambda}{\mu}\right)^n, & if\ n > s \end{cases}$$

and

$$p_0 = \left(1 + \sum_{n=1}^{\infty} c_n\right)^{-1} = \left(1 + \sum_{n=1}^{s-1}\frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n + \sum_{n=s}^{\infty}\frac{1}{s!\,s^{n-s}}\left(\frac{\lambda}{\mu}\right)^n\right)^{-1}$$

$$= \left(\sum_{n=0}^{s-1}\frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n + \frac{1}{s!}\left(\frac{\lambda}{\mu}\right)^s\sum_{n=s}^{\infty}\left(\frac{\lambda}{s\mu}\right)^{n-s}\right)^{-1} = \left(\sum_{n=0}^{s-1}\frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n + \frac{\left(\frac{\lambda}{\mu}\right)^s}{s!\,(1-\frac{\lambda}{s\mu})}\right)^{-1}$$

The M/M/s model – averaged quantities

To determine L, L_q, W, W_q it is easier to start with L_q. Since we have n servers,

$$L_q = \sum_{n=s}^{\infty}(n-s)p_n = \sum_{n=0}^{\infty} np_{n+s} = \sum_{n=0}^{\infty} nc_{n+s}\, p_0 = p_0 \sum_{n=0}^{\infty} \frac{n}{s!\, s^n}\left(\frac{\lambda}{\mu}\right)^{n+s}$$

$$= \frac{p_0}{s!}\left(\frac{\lambda}{\mu}\right)^{s} \sum_{n=0}^{\infty} n\left(\frac{\lambda}{s\mu}\right)^{n} = \frac{p_0}{s!}\left(\frac{\lambda}{\mu}\right)^{s} \frac{\frac{\lambda}{s\mu}}{\left(1-\frac{\lambda}{s\mu}\right)^2}$$

From here, we can compute $W_q = \frac{L_q}{\lambda}$, and then $W = W_q + \frac{1}{\mu}$,

and then $L = \lambda W = \lambda\left(W_q + \frac{1}{\mu}\right) = L_q + \frac{\lambda}{\mu}$.

Summary and self-study

Summary: today we have learnt:

- how to interpret the M/M/1 model as a birth-and-death process,
- how to determine the steady states of birth-and-death process,
- how to analyse the M/M/s model (using a birth-and-death process).

Self-study: Consider an M/M/2 model with $\lambda = 2$ and $\mu = 3$ and FIFO queue discipline. Compute $p_0,\ L,\ L_q, W_q$ and W assuming that the queue can be infinitely long.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 21 – Finite queues

Dr Pak-Hin Lee

Recap and lecture outline

Recap: so far we have the basics of queueing theory, Little's law, and how to analyse the M/M/s model with infinitely long queues as a birth-and-death process.

In this lecture: The M/M/s with finite queue, aka the M/M/s/K model (following Ch. 17.6 of the book by Hillier and Lieberman). The key difference of this model is that if the queueing system is full, then no further customer is allowed to enter the queueing system.

The M/M/1 model – transient behavior

To study the transient behavior of the M/M/1 model, we need to set some initial conditions and then solve

$$\partial_t p(0,t) = p(1,t)\mu - p(0,t)\lambda$$
$$\partial_t p(n,t) = p(n-1,t)\lambda - p(n,t)(\lambda+\mu) + p(n+1,t)\mu$$

under the additional constrains: $p(i,t) \geq 0,\ p(0,t) + p(1,t) + \ ... = 1$

The steady-states are: $p_0 = 1 - \rho$ and $p_n = \rho^n p_0$ for $n > 0$, where $\rho = \frac{\lambda}{\mu}$.

The M/M/1/K model

If the queueing system hosts at most K customers, then the Kolmogorov equations are

$$\begin{aligned}
&\partial_t p(0,t) = p(1,t)\mu - p(0,t)\lambda \\
&\partial_t p(n,t) = p(n-1,t)\lambda - p(n,t)(\lambda+\mu) + p(n+1,t)\mu \quad n = 1,\dots,K-1 \\
&\partial_t p(K,t) = p(K-1,t)\lambda - p(K,t)\mu
\end{aligned}$$

because no one can arrive if $N(t) = K$. Then, the steady state conditions read

$$\begin{aligned}
&p_1\mu = p_0\lambda \\
&p_n(\lambda+\mu) = p_{n-1}\lambda + p_{n+1}\mu \quad n = 1,\dots,K-1 \\
&p_K\mu = p_{K-1}\lambda
\end{aligned}$$

and the solution is still $p_n = \left(\frac{\lambda}{\mu}\right)^n p_0 = \rho^n p_0$ for $n = 1,\dots,K$.

The M/M/1/K model – steady states

The solution is still $p_n = \rho^n p_0$ for $n = 1, \dots, K$. Recall that if $\rho \neq 1, \text{then}$

$$\sum_{n=0}^{K} p_n = \sum_{n=0}^{K} \rho^n p_0 = \frac{p_0(1-\rho^{K+1})}{1-\rho}$$

Since $\sum_{n=0}^{K} p_n = 1,$ we conclude that $p_0 = \frac{1-\rho}{1-\rho^{K+1}}$, and $p_n = \frac{\rho^n(1-\rho)}{1-\rho^{K+1}}$.

On the other hand, if $\rho = 1$, then, $p_0 = p_1 = \cdots = p_K$ and $\sum_{n=0}^{K} p_n = 1$ implies

$$p_0 = p_1 = \cdots = p_K = \frac{1}{K+1}.$$

The M/M/1/K model – expected # of customers

If $\rho \neq 1$, then

$$L = \sum_{n=0}^{K} n p_n = \sum_{n=0}^{K} n \frac{\rho^n (1-\rho)}{1-\rho^{K+1}} = \frac{1-\rho}{1-\rho^{K+1}} \sum_{n=0}^{K} n\rho^n = \frac{(1-\rho)}{1-\rho^{K+1}} \sum_{n=0}^{K} \rho \frac{d}{d\rho} \rho^n$$

$$= \frac{(1-\rho)\rho}{1-\rho^{K+1}} \frac{d}{d\rho} \sum_{n=0}^{K} \rho^n = \frac{(1-\rho)\rho}{1-\rho^{K+1}} \frac{d}{d\rho} \left(\frac{1-\rho^{K+1}}{1-\rho} \right)$$

$$= \frac{(1-\rho)\rho}{1-\rho^{K+1}} \frac{-(K+1)\rho^K (1-\rho) + (1-\rho^{K+1})}{(1-\rho)^2} = \frac{\rho}{1-\rho} - \frac{(K+1)\rho^{K+1}}{1-\rho^{K+1}}$$

If $\rho = 1$, then

$$L = \sum_{n=0}^{K} n p_n = \sum_{n=0}^{K} \frac{n}{K+1} = \frac{1}{K+1} \frac{K(K+1)}{2} = \frac{K}{2}.$$

The M/M/1/K model – average stays

Since

$$\bar{\lambda} = \sum_{n=0}^{\infty} \lambda p_n = \lambda \sum_{n=0}^{K-1} p_n = \lambda(1 - p_K),$$

if $\rho \neq 1$ we have

$$W = \frac{L}{\bar{\lambda}} = \frac{1}{\lambda(1 - p_K)} \left(\frac{\rho}{1 - \rho} - \frac{(K+1)\rho^{K+1}}{1 - \rho^{K+1}} \right),$$

whereas if $\rho = 1$ we have

$$W = \frac{L}{\bar{\lambda}} = \frac{K}{2\lambda(1 - p_K)}.$$

M/M/1 model with finite length - example

If I receive an email every $\lambda^{-1} = 5$ minutes, it takes me takes me $\mu^{-1} = 3$ minutes to answer it, and my inbox can contain at most 5 emails, then $\rho = \frac{3}{5} = 0.6$,

$$p_0 = \frac{1 - 0.6}{1 - 0.6^{5+1}} \cong 0.4196,$$

$p_1 \cong 0.2517,$ $p_2 \cong 0.1510,$ $p_3 \cong 0.0906,$ $p_4 \cong 0.0544,$ $p_5 \cong 0.0326$,

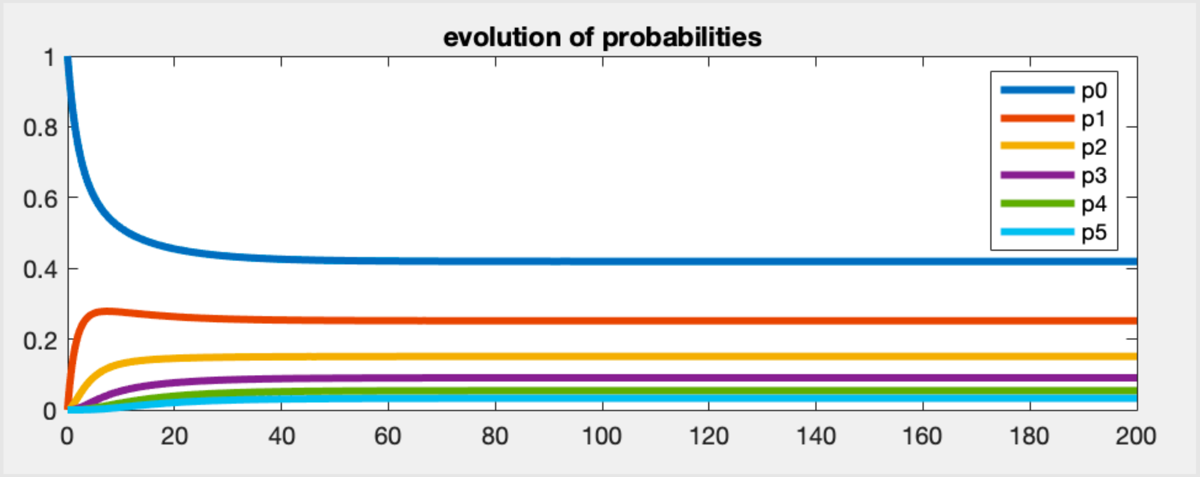
and $L = \sum_{n=0}^{5} n\, p_n \cong 1.2064.$

M/M/1/K model – transient behaviour

`OR_Lect_21_finite_queue.m`

For finite length queues, we can also simulate the transient behaviour by solving Kolmogorov's equations using Matlab's ODE suite.

For example, for $K = 5$, using $p(\cdot, t = 0) = (1, 0, 0, 0, 0, 0)^T$, $\lambda = 5^{-1}$, $\mu = 3^{-1}$,



and observe that $p(\cdot, t = 0) \cong$ (0.4196,0.2517,0.1511,0.0906,0.0544,0.0326), which coincides with the steady states computed in the previous slide.

UNIVERSITY OF LEICESTER

www.le.ac.uk

The M/M/s/K model

If the queueing system can host at most K and contains s identical servers, then the capacity of the queue is $K - s$. Note that the case $K = s$ is allowed.

This means that $\lambda_0 = \lambda_1 = \cdots = \lambda_{K-1} =: \lambda$ and $\lambda_n = 0$ for $n \geq K$.

As for the service rate, we have $\mu_n = \min(n, s)\mu$ for $n \geq 1$.

As this is a birth-and-death process, the steady states are $p_n = c_n p_0$, where

$$c_n = \frac{\lambda_{n-1}\lambda_{n-2} \dots \lambda_0}{\mu_n \mu_{n-1} \dots \mu_1} = \begin{cases} \frac{1}{n!}\left(\frac{\lambda}{\mu}\right)^n & for\ 0 \leq n \leq s, \\ \frac{1}{s!}\left(\frac{\lambda}{\mu}\right)^s \left(\frac{\lambda}{s\mu}\right)^{n-s} & for\ s \leq n \leq K, \\ 0 & for\ n > K, \end{cases}$$

and $p_0 = (1 + \sum_{n=1}^{\infty} c_n)^{-1} = (1 + \sum_{n=1}^{K} c_n)^{-1}$.

Summary and self-study

Summary: today we have studied the M/M/s/K, that is, the M/M/s model with the further restriction that the queueing system can host at most K customers.

Self-study: Prove that for the M/M/s/K model, the following formulas hold:

$$L_q = \frac{p_0 s^s \rho^{s+1}}{s!\,(1-\rho)^2}\Big(1 - \rho^{K-s}\big(1-(K-s)(\rho-1)\big)\Big), \qquad \text{where } \rho = \frac{\lambda}{s\mu},$$

$$L = \sum_{n=0}^{s-1} n p_n + L_q + s\left(1 - \sum_{n=0}^{s-1} p_n\right)$$

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 22: Part I – Finite calling population

Dr Pak-Hin Lee

Recap and lecture outline

Recap: in the last lecture we studied the M/M/s/K model,

In this lecture: The M/M/s model with finite calling population, following Ch. 17.6 of the book by Hillier and Lieberman.

The M/M/s model with finite calling population

Let assume the calling population contains only N customers. This implies that if the system currently contains $n \leq N$ customers, then there are only $N - n$ potential customers remaining in the calling population.

We also assume that customers return to the calling population once the exit the queueing system.

Example: A computer room in the library contains N computers (the calling population). The library employs a technician (that is, one server) that repairs these computers when they stop working. The queueing system comprises all computers are not working at a given time. Once a computer is repaired, it returns to the calling population of computers that are available to students.

Interarrival time

Note that each member is either inside or outside the queueing system. That is, if n customers in the system, then $N - n$ are outside the system.

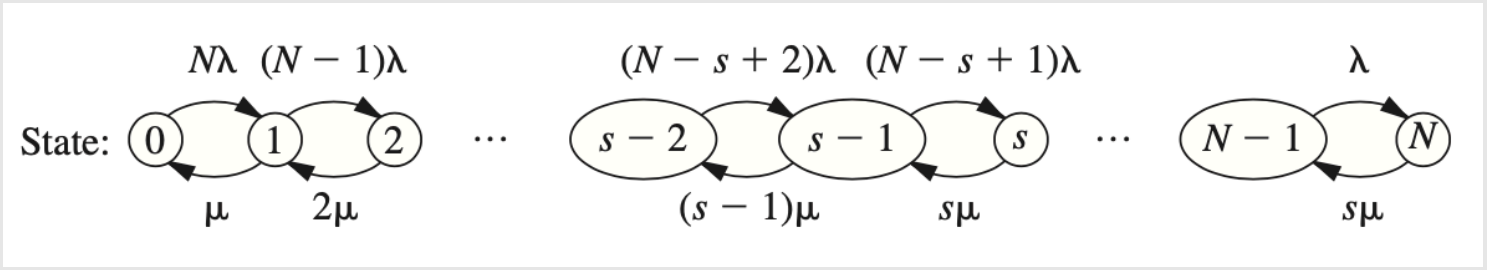
Assume the outside distribution of each member is exponential with parameter λ.

Idea: For a fixed time $t \geq 0$, if none of them has entered the system it means that the minimum among their interarrival times is bigger than t.

This means that the interarrival time of the calling population is the minimum among the interarrival times of each member outside the system. Property 4 on slide 8 of lecture 19 implies that this distribution is exponential with $\lambda_n = (N - n)\lambda$.

Using birth-and-death processes

The parameters are $\lambda_n = \max((N-n)\lambda, 0)$ and $\mu_n = \min(n, s)\mu$. The diagram is



As usual, $p_n = c_n p_0$ for $n > 0$, where

$$c_n = \frac{\lambda_{n-1}\lambda_{n-2}\dots\lambda_0}{\mu_n\mu_{n-1}\dots\mu_1} = \begin{cases} \binom{N}{n}\left(\frac{\lambda}{\mu}\right)^n & for\ 0 \le n \le s, \\ \binom{N}{n}\left(\frac{\lambda}{\mu}\right)^n \frac{1}{s^{n-s}} & for\ s \le n \le N, \\ 0 & for\ n > N, \end{cases}$$

and $p_0 = (1 + \sum_{n=1}^{\infty} c_n)^{-1} = (1 + \sum_{n=1}^{N} c_n)^{-1}$.

UNIVERSITY OF LEICESTER

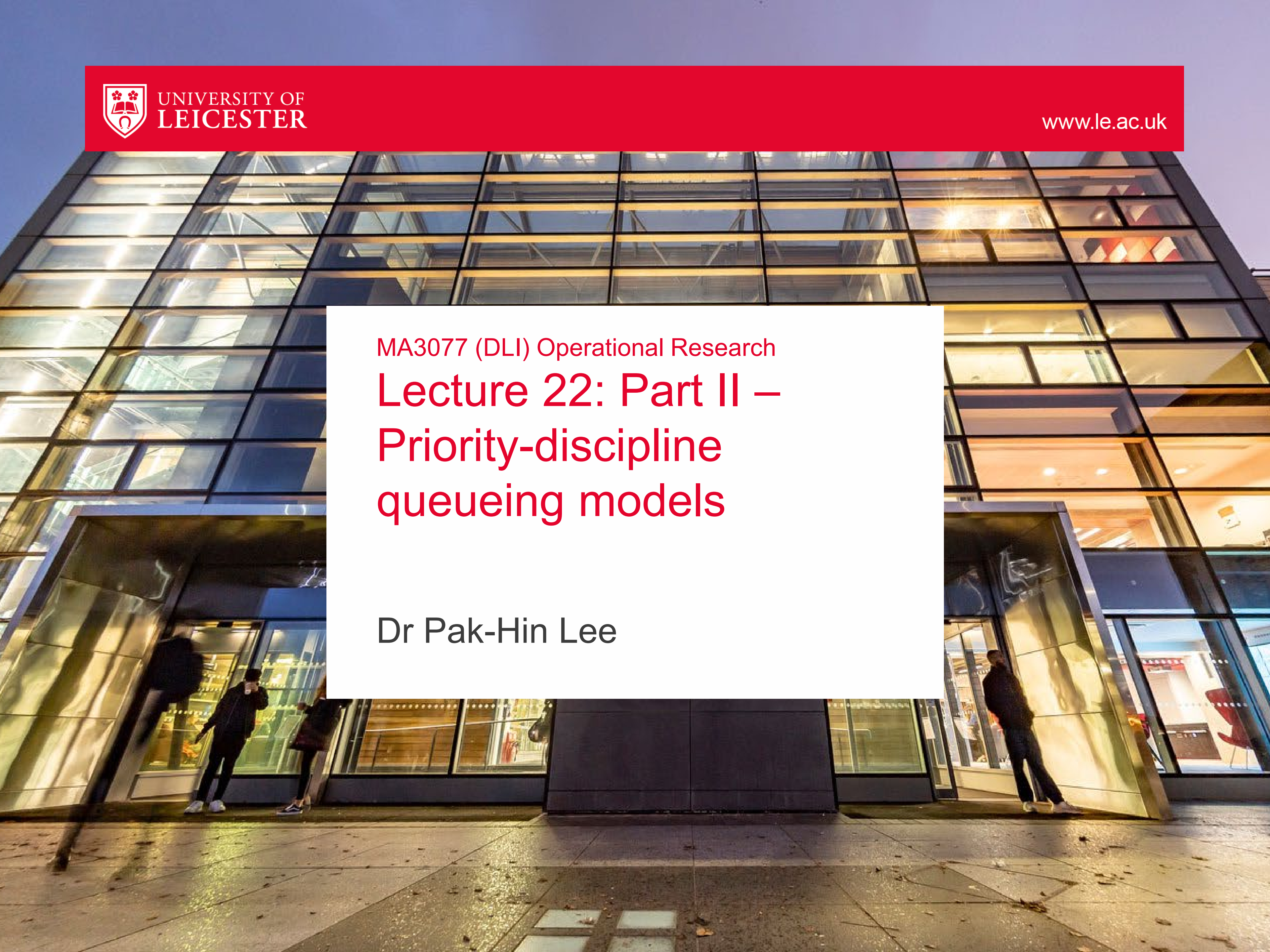
www.le.ac.uk

Summary and self-study

Summary: today we have studied the M/M/s with finite calling population size N.

Self-study: Show that for an M/M/s model with finite calling population size N the following formula holds:

$$L = \sum_{n=0}^{s-1} np_n + L_q + s\left(1 - \sum_{n=0}^{s-1} p_n\right)$$



UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 22: Part II – Priority-discipline queueing models

Dr Pak-Hin Lee

Recap and lecture outline

Recap: so far we have learnt a few variations on the M/M/s models. More precisely, we looked at the:

- M/M/s/K model
- M/M/s model with finite calling populations.

In this lecture: The M/M/s model with priority queue disciplines, following closely Ch. 17.8 of the book by Hillier and Lieberman.

Priority-discipline queueing model

Idea: the queue discipline is based on a *priority system:* the order in which members of the queue are selected for service is based on their assigned priorities.

Assumptions:

- there are N priority classes. Class 1 has the highest priority and class N the lowest. Within each class, customers receive service on a first-in-first-out basis,
- The interarrival time distribution of each class is exponential (with parameters $\lambda_1, \dots, \lambda_n$),
- The service time is exponential and does not depend on the priority class.
- The system is *preemptive:* a customer being served is immediately bumped back into the queue if a higher-priority customer enters the system.

Generic customers (i.e., if we ignore class)

The interarrival time of a generic customer (that is, if we ignore their class) is also a Poisson process (cf. slide 11 on Lecture 17). Since the service time is class independent, the model for generic customers is simply M/M/s with a different queue-discipline, and the corresponding formulas for L, L_q, W, W_q still apply.

However, the distribution of customer waiting time w is not anymore exponential. The variance is larger because the waiting times of high-priority customers is much smaller than the waiting times of lower-priority customers. Also, the total number of customers in the system contains mostly customers in lower-priority classes.

Waiting time distribution with $s = 1$ – 1st class

Assuming the system has only one server, for first class customers:

- the interarrival time distribution exponential with parameter λ_1
- service time is exponential with parameter μ
- queue discipline is simply FIFO.

This implies that we can just reuse the results for the M/M/1/FIFO model. Let w_1 be the random variable describing the first class customer waiting time. Then, the distribution of w_1 is exponential with parameter $\mu - \lambda_1$ (cf. slide 13, Lecture 20), and

$$W_1 = \mathrm{E}[w_1] = (\mu - \lambda_1)^{-1}.$$

Waiting time distribution with $s = 1$ – 2nd class

Let w_{12} be the random variable describing the waiting time of a generic customer. Then the distribution of w_{12} is exponential with parameter $\mu - \lambda_1 - \lambda_2$, and

$$W_{12} = \mathrm{E}[w_{12}] = (\mu - \lambda_1 - \lambda_2)^{-1}$$

Let w_2 be the random variable describing the waiting time of a customer of class 2, and let $W_2 = \mathrm{E}[w_2]$. Since the probability that a generic customer is in class 1 is $\lambda_1/(\lambda_1 + \lambda_2)$ and in class 2 is $\lambda_2/(\lambda_1 + \lambda_2)$ (cf. slide 12, Lecture 19), it holds

$$W_{12} = \frac{\lambda_1}{\lambda_1 + \lambda_2} W_1 + \frac{\lambda_2}{\lambda_1 + \lambda_2} W_2$$

Therefore,

$$W_2 = \frac{(\lambda_1 + \lambda_2)W_{12} - \lambda_1 W_1}{\lambda_2} = \frac{1}{\lambda_2}\left(\frac{\lambda_1 + \lambda_2}{\mu - \lambda_1 - \lambda_2} - \frac{\lambda_1}{\mu - \lambda_1}\right)$$
$$= \frac{(\mu - \lambda_1)(\lambda_1 + \lambda_2) - \lambda_1(\mu - \lambda_1 - \lambda_2)}{\lambda_2(\mu - \lambda_1)(\mu - \lambda_1 - \lambda_2)} = \frac{\mu}{(\mu - \lambda_1)(\mu - \lambda_1 - \lambda_2)}.$$

Waiting time formula in preemptive model

We can generalize this to N different classes. We obtain the equality

$$\left(\sum_{i=1}^{k} \lambda_i\right) W_{1k} = \sum_{i=1}^{k} \lambda_i W_i$$

that is,

$$\lambda_k W_k = \left(\sum_{i=1}^{k} \lambda_i\right) W_{1k} - \sum_{i=1}^{k-1} \lambda_i W_i \,.$$

One can show by induction that the solution is

$$W_k = (\mu B_k B_{k-1})^{-1},$$

where $B_0 = 1$ and $B_k = 1 - \mu^{-1}(\lambda_1 + \cdots + \lambda_k)$.

Summary and self-study

Summary: today we have had a look at queueing systems with preemptive priority queue discipline.

Self-study: Consider an M/M/1 model with $\lambda_1 = 0.2$, $\lambda_2 = 0.6$, $\mu = 3$, and with preemptive priority queue discipline. Compute W_2.

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 23 – Fundamentals of unconstrained optimisation

Dr Pak-Hin Lee

Recap and plan of the day

Recap: so far, we have learnt:

- the theory and some practice of linear programming,
- how to analyse queues.

In this lecture: Fundamentals of unconstrained nonlinear optimisation, following closely Ch. 2 of the book *Numerical Optimization* by Jorge Nocedal and Stephen J. Wright.

Global and local minimisers

Given a smooth function $f\colon \mathbb{R}^n \to \mathbb{R}$, solve

$$\min_x f(x),$$

that is, find $x^* \in \mathbb{R}^n$ so that $f(x^*) \leq f(x)$ for all $x \in \mathbb{R}^n$. Such an x^* is called a *global minimiser.*

Most often, finding a global minimiser is just not possible, and we must content ourselves with a *local minimiser*, that is, an $x^* \in \mathbb{R}^n$ so that $f(x^*) \leq f(x)$ for every x in a neighbourhood of x^* (e.g., for all x such that $\|x - x^*\| < \varepsilon$ for a certain $\varepsilon > 0$).

Global/local minimisers are called *strict global/local minimisers* if $\leq$ can be replaced by $<$ in their definition.

Example:

- if $f(x) = 2$ for every $x \in \mathbb{R}^n$, then every $x \in \mathbb{R}^n$ is a global minimiser,
- if $f(x) = x^T x$, then $x^* = 0 \in \mathbb{R}^n$ is a strict global minimiser.

Taylor's theorem

Theorem: Let $f\colon \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable and let $p \in \mathbb{R}^n$. Then,

$$f(x+p) = f(x) + \nabla f(x+tp)^T p \qquad \text{for some} \quad t \in (0,1).$$

If $f\colon \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable, then

$$f(x+p) = f(x) + \nabla f(x)^T p + \frac{1}{2} p^T \nabla^2 f(x+tp) p \qquad \text{for some} \quad t \in (0,1).$$

Definition: A point $x \in \mathbb{R}^n$ such that $\nabla f(x) = 0$ is called a *stationary point*.

First-order necessary conditions

Theorem: Suppose $f: \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable (at least in a neighbourhood of x^*). If $x^* \in \mathbb{R}^n$ is a local minimiser, then $\nabla f(x^*) = 0$.

Proof: (by contrapositive)

If $\nabla f(x^*) \neq 0$, set $p \coloneqq -\nabla f(x^*)$, so that $p^T \nabla f(x^*) = -\|\nabla f(x^*)\|^2 < 0$. Since ∇f is continuous near x^*, so is the function $t \mapsto p^T \nabla f(x^* + tp)$, and there is a $T > 0$ such that $p^T \nabla f(x^* + tp) < 0$ for any $t \in [0, T]$. Let $r \in (0, T]$ and let $q_r := rp$. Then, by Taylor's theorem, there is an $s \in (0,1)$ such that,

$$f(x^* + q_r) = f(x^*) + q_r^T \nabla f(x^* + s q_r) = f(x^*) + r p^T \nabla f(x^* + (sr)p) < f(x^*)$$

(because $sr < r \leq T$). Since r can be arbitrarily small, $x^* + q_r = x^* + rp$ can be arbitrarily close to x^*, which implies that x^* cannot be a local minimiser.

$\square$

Second-order necessary conditions

Definition: A matrix B is positive definite if $p^T Bp > 0$ for all $p \neq 0$, and positive semidefinite if $p^T Bp \geq 0$ for all p.

Theorem: Suppose $f: \mathbb{R}^n \to \mathbb{R}$ and $\nabla^2 f$ exists and is continuous (at least in a neighbourhood of x^*). If $x^* \in \mathbb{R}^n$ is a local minimiser of f, then $\nabla f(x^*) = 0$ and $\nabla^2 f(x^*)$ is positive semidefinite.

Proof: (by contrapositive)

Assume there is a $p \in \mathbb{R}^n$ such that $p^T \nabla^2 f(x^*) p < 0$. Since $\nabla^2 f$ is continuous near x^*, so is the function $t \mapsto p^T \nabla^2 f(x^* + tp)p$. Let $T > 0$ be such that $p^T \nabla^2 f(x^* + tp)p < 0$ for all $t \in [0, T]$. Let $r \in (0, T]$. Then, by Taylor's theorem, there is an $s \in (0,1)$ such that,

$$f(x^* + rp) = f(x^*) + r\nabla f(x^*)^T p + \frac{r^2}{2} p^T \nabla^2 f(x^* + srp)p < f(x^*)$$

(because $sr < r \leq T$). Since r can be arbitrarily small, $x^* + rp$ can be arbitrarily close to x^*, which implies that x^* cannot be a local minimiser.

□

Second-order sufficient conditions

Theorem: Suppose $f: \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable (at least in a neighbourhood of x^*) and $x^* \in \mathbb{R}^n$ is a stationary point. If $\nabla^2 f(x^*)$ is positive definite, then x^* is a strict local minimiser of f.

Proof: (direct proof)

By continuity, $\nabla^2 f$ is positive definite also in a neighbourhood D of x^*. Let $p \in \mathbb{R}^n$ be any sufficiently short vector, so that $x^* + p \in D$. Then, by Taylor's theorem, there is a $t \in (0,1)$ such that

$$f(x^* + p) = f(x^*) + \nabla f(x^*)^T p + \frac{1}{2} p^T \nabla^2 f(x + tp)p = f(x^*) + \frac{1}{2} p^T \nabla^2 f(x + tp)p.$$

Since $t \in (0,1)$, $x^* + tp \in D$, and $p^T \nabla^2 f(x + tp)p > 0$. Hence, $f(x^* + p) > f(x^*)$ and therefore x^* is a strict local minimiser. $\square$

Convex functions 1/2

Definition: A function $f: \mathbb{R}^n \to \mathbb{R}$ is convex if, for any distinct $p, q \in \mathbb{R}^n$ and any $t \in [0,1]$, it holds

$$f(tp + (1-t)q) \leq tf(p) + (1-t)f(q).$$

Theorem: If $f: \mathbb{R}^n \to \mathbb{R}$ is convex, any local minimiser $x^* \in \mathbb{R}^n$ is also a global minimiser. In addition, if f is differentiable, then any stationary point x^* is a global minimiser.

Proof: (by contradiction)

If x^* is only a local minimiser, there is a $z \in \mathbb{R}^n$ such that $f(z) < f(x^*)$. Then, by convexity, any $t \in (0, 1]$ it holds

$$f(tz + (1-t)x^*) \leq tf(z) + (1-t)f(x^*) < tf(x^*) + (1-t)f(x^*) = f(x^*).$$

This implies that x^* cannot be a local minimiser, because the point $tz + (1-t)x^*$ can be arbitrarily close to x^*.

Convex functions 2/2

Definition: A function $f: \mathbb{R}^n \to \mathbb{R}$ is convex if, for any distinct $p, q \in \mathbb{R}^n$ and any $t \in [0,1]$, it holds

$$f(tp + (1-t)q) \leq tf(p) + (1-t)f(q).$$

Theorem: If $f: \mathbb{R}^n \to \mathbb{R}$ is convex, any local minimiser $x^* \in \mathbb{R}^n$ is also a global minimiser. In addition, if f is differentiable, then any stationary point x^* is a global minimiser.

Proof: (by contradiction)

If x^* is not a global minimiser, there is a $z \in \mathbb{R}^n$ such that $f(z) < f(x^*)$. Then, by convexity and the definition of derivative

$$\nabla f(x^*)^T(z - x^*) = \frac{d}{dt} f\big(x^* + t(z - x^*)\big)\Big|_{t=0} = \lim_{t\to 0} \frac{f\big(x^* + t(z - x^*)\big) - f(x^*)}{t}$$
$$\leq \lim_{t\to 0} \frac{tf(z) - tf(x^*)}{t} = f(z) - f(x^*) < 0,$$

which contradicts our assumption that x^* is a stationary point. $\square$

Summary and self-study

Summary: today we have learnt the fundamentals of nonlinear optimisation.

Self-study: Determine all stationary points of the function

$$f\colon \mathbb{R} \to \mathbb{R}, \qquad x \mapsto f(x) := \cos(\exp(x)).$$

Is this function convex?

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 24 – line search algorithms

Dr Pak-Hin Lee

Recap and plan of the day

Recap: in the previous lecture we studied the fundamentals of unconstrained optimisation.

In this lecture: Line search methods to compute approximate minimisers, following Ch. 3 of the book *Numerical Optimization* by Jorge Nocedal and Stephen J. Wright.

Line search methods

Given a smooth function $f\colon \mathbb{R}^n \to \mathbb{R}$, solve $\min_x f(x)$.

Basic idea: given a starting point $x_0 \in \mathbb{R}^n$, create a sequence $\{x_k\}_{k\geq 0}$ so that $\{f(x_k)\}_{k\geq 0}$ is decreasing. To construct the next iterate x_{k+1}, we can use information about f and the previously computed iterates $x_0, x_1, \dots, x_k$.

Line search strategy: at the k-th iteration,

- choose a direction $p_k \in \mathbb{R}^n$,
- determine a *step length* s_k by solving $\min_{s>0} f(x_k + sp_k)$ approximately,
- set $x_{k+1} = x_k + s_k p_k$
- repeat until convergence criteria are met.

Steepest descent direction 1/2

The most obvious candidate is $p_k = -\nabla f(x_k)$. This is the *steepest descent direction*. Indeed, assuming that f is twice continuously differentiable, for any fixed direction p Taylor's theorem implies that

$$f(x_k + sp) = f(x_k) + sp^T \nabla f(x_k) + \frac{s^2}{2} p^T \nabla^2 f(x_k + tsp) p$$

for a $t \in (0,1)$. By differentiating with respect to s, we obtain

$$\frac{\mathrm{d}}{\mathrm{ds}} f(x_k + sp) \Big|_{s=0} = p^T \nabla f(x_k),$$

that is, $p^T \nabla f(x_k)$ is the rate of change of f at x_k in the direction p. In light of this, the best direction is the one that solves

$$\min_{\|p\|=1} p^T \nabla f(x_k)$$

(the constraint $\|p\| = 1$ is included because we are only interested in a direction).

Steepest descent direction 1/2

The best direction is the one that solves $\min_{\|p\|=1} p^T \nabla f(x_k)$.

Since $p^T \nabla f(x_k) = \|p\| \|\nabla f(x_k)\| \cos(\alpha)$, where α is the angle between p and $\nabla f(x_k)$, solving $\min_{\|p\|=1} p^T \nabla f(x_k)$ is equivalent to solving

$$\min_{\alpha} \|\nabla f(x_k)\| \cos(\alpha)$$

whose minimiser is $\alpha^* = \pi$. Therefore, the optimal direction is $p^* = -\frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$, as claimed.

The *steepest descent method* (aka the gradient descent method) uses $p_k = -\nabla f(x_k)$ at each iteration. Note that p_k may not satisfy $\|p_k\| = 1$. This will affect the choice of the step size s_k, but not the resulting new iterate $x_{k+1} = x_k + s_k p_k$.

Descent directions

You can choose $p_k = -\nabla f(x_k)$, but don't have to if you don't want to. Any direction that makes an angle strictly less than $\frac{\pi}{2}$ in absolute value with $-\nabla f(x_k)$ is a descent direction as long as the step length is sufficiently small. Indeed, by Taylor's theorem,

$$f(x_k + sp) = f(x_k) + sp^T\nabla f(x_k) + \frac{s^2}{2}p^T\nabla^2 f(x_k + tsp)p$$

for a $t \in (0,1)$. If $s > 0$ is sufficiently small, $|p^T\nabla f(x_k)| > \frac{s}{2}\max_{0\le r\le s}|p^T\nabla^2 f(x_k + rp)p|$, and if $p^T\nabla f(x_k) < 0$, then

$$\begin{aligned} f(x_k + sp) &= f(x_k) + s\left(p^T\nabla f(x_k) + \frac{s}{2}p^T\nabla^2 f(x_k + tsp)p\right) \\ &= f(x_k) + s\left(\frac{s}{2}p^T\nabla^2 f(x_k + tsp)p - |p^T\nabla f(x_k)|\right) \\ &\le f(x_k) + s\left(\frac{s}{2}\max_{0\le r\le s}|p^T\nabla^2 f(x_k + rp)p| - |p^T\nabla f(x_k)|\right) < f(x_k). \end{aligned}$$

The Newton direction 1/2

For a fixed $p \in \mathbb{R}^n$, consider the approximation arising by truncating the Taylor series expansion of $f(x_k + p)$ after the second order, that is,

$$f(x_k + p) \approx f(x_k) + p^T \nabla f(x_k) + \frac{1}{2} p^T \nabla^2 f(x_k) p =: m_k(p)$$

If $\nabla^2 f(x_k)$ is positive definite, choose p_k as the minimiser of $\min_{p \in \mathbb{R}^n} m_k(p)$. Since $m_k(p)$ is convex, we only need to find a stationary point. The gradient of $m_k(p)$ with respect to p is (recall that if f is twice differentiable, $\nabla^2 f$ is symmetric)

$$\nabla_p m_k(p) = \nabla^2 f(x_k) p + \nabla f(x_k).$$

Therefore, the *Newton direction* is $p_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$

Note that in this case we don't (need to) add the constraint $\|p_k\| = 1$ because $m_k(p) \to \infty$ if $\|p\| \to \infty$.

The Newton direction 2/2

The *Newton direction* is $p_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$

Remarks:

- When defining p_k as the minimiser of $\min_{p \in \mathbb{R}^n} m_k(p)$, we don't (need to) add the constraint $\|p_k\| = 1$ because $m_k(p) \to \infty$ if $\|p\| \to \infty$, and we often use $s_k = 1$.
- The Newton direction is reliable when the approximation $f(x_k + p) \approx m_k(p)$ is accurate. In fact, if f is tree times continuously differentiable, $f(x_k + p) - m_k(p)$ is of order $\|p\|^3$, and hence accurate if $\|p\| < 1$ is sufficiently small.
- If $\nabla^2 f(x_k)$ is positive definite, the newton direction is a descent direction. Indeed,
$$\nabla f(x_k)^T p_k = -\nabla f(x_k)^T \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \leq -c_k \|\nabla f(x_k)\|^2$$
for a $c_k > 0$ (because $\left(\nabla^2 f(x_k)\right)^{-1}$ is also positive definite).

Steepest descent vs Newton's method

The steepest descent method:

- requires computing a good step size s_k
- needs only the first derivative of f.
- it typically converges linearly, that is $\|x^* - x_{k+1}\| = C\|x^* - x_k\|$ for a constant $0 < C < 1$ (but C can be arbitrarily close to 1 in tough problems).

Newton's method:

- comes with a natural step size $s_k = 1$, but requires solving linear system for all k
- requires first and second derivatives of f, and $\nabla^2 f(x_k)$ must be positive definite,
- it typically has local quadratic rate of convergence, that is, when x_k is close to x^*, then $\|x^* - x_{k+1}\| = C\|x^* - x_k\|^2$ for a constant $0 < C < 1$ (in which case only few steps are needed to achieve high accuracy.

Quasi-Newton's methods - BFGS

Idea: use values of ∇f on previous (and current) iterates to build an approximation B_k of $\nabla^2 f(x_k)$. Then, use B_k to compute p_k as in Newton's method, that is,

$$p_k = -(B_k)^{-1}\nabla f(x_k)$$

BFGS method:

$$d_k = x_{k+1} - x_k$$
$$y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$$
$$B_{k+1} = B_k - \frac{B_k d_k d_k^T B_k}{d_k^T B_k d_k} + \frac{y_k y_k^T}{y_k^T d_k}$$

Remarks:

- If B_0 is positive definite, all B_k's are positive definite,
- $B_{k+1}^{-1} = \left(I - \rho_k d_k y_k^T\right) B_k^{-1}(I - \rho_k d_k y_k^T) + \rho_k d_k d_k^T$, where $\rho_k = \left(y_k^T d_k\right)^{-1}$,
- It usually converges superlinearly and only needs ∇f .

Summary and self-study

Summary: today we have learnt:

- the strategy of line search methods
- steepest descent / Newton / Quasi-Newton methods

Self-study:

1. Let $x_0 = \log_e\left(\frac{\pi}{2}\right)$. Consider the function

$$f: \mathbb{R} \to \mathbb{R}, \qquad x \mapsto f(x) := \cos(\exp(x)).$$

a) Perform one step of Newton's method starting from x_0,

b) Perform one step of the steepest descent method starting from x_0 and using the smallest positive optimal step size.

2. Compute the gradient and Hessian of the function $f: \mathbb{R}^n \to \mathbb{R}$ with

$x \mapsto f(x) := x^T x$

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 25 – line search step selection

Dr Pak-Hin Lee

Recap and plan of the day

Recap: so far, we have learnt:

- the fundamentals of nonlinear optimization
- the strategy of line search methods
- steepest descent / Newton / Quasi-Newton methods

In this lecture: Step length selection and convergence rates, following closely Ch. 3 of the book *Numerical Optimization* by Jorge Nocedal and Stephen J. Wright.

Line search methods

Given a smooth function $f: \mathbb{R}^n \to \mathbb{R}$, solve $\min_x f(x)$.

Basic idea: given a starting point $x_0 \in \mathbb{R}^n$, create a sequence $\{x_k\}_{k\geq 0}$ so that $\{f(x_k)\}_{k\geq 0}$ is decreasing. To construct the next iterate x_{k+1}, we can use information about f and the previously computed iterates $x_0, x_1, \dots, x_k$.

Line search strategy: at the k^{th} iteration,

- choose a direction $p_k \in \mathbb{R}^n$,
- determine a *step length* s_k by solving $\min_{s>0} f(x_k + sp_k)$ approximately,
- set $x_{k+1} = x_k + s_k p_k$
- repeat until converge criteria are met.

Descent direction

The descent direction p_k should satisfy $p_k^T \nabla f(x_k) < 0$. It often has the form

$$p_k = -B_k^{-1} \nabla f(x_k)$$

where B_k is a symmetric and nonsingular matrix (often positive definite).

Example:

- $B_k = I$ for a steepest descent direction,
- $B_k = \nabla f^2(x_k)$ for a Newton's direction,
- $B_k \approx \nabla f^2(x_k)$ for a BFGS direction.

Step length

The step length s_k solves $\min_{s>0} f(x_k + sp_k)$ approximately.

Ideally, we would like to achieve a substantial reduction in f without investing too many resources in determining s_k.

Solving $\min_{s>0} f(x_k + sp_k)$ exactly is usually too expensive, because it may require many evaluations of f and ∇f.

To find an approximate solution, an popular strategy is to:

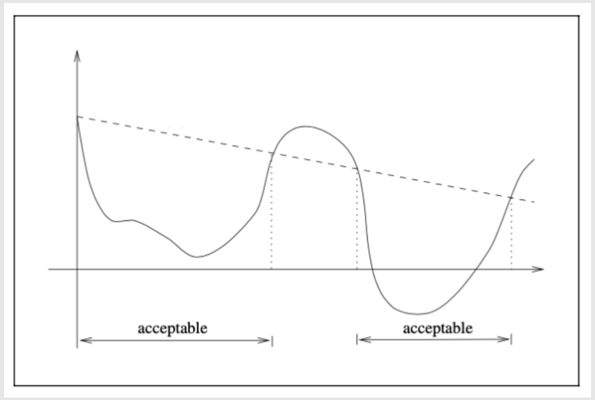
- first find an interval of desirable step lengths,
- and then applies a bisection or interpolation strategy to determine a good step length within this interval.

Armijo rule

We may be tempted to accept a step length as soon as $f(x_k + s_k p_k) < f(x_k)$. Unfortunately, this is not sufficient to guarantee convergence: $f(x_k)$ may stagnate before reaching the minimum. We need to enforce a sufficient decrease condition!

Armijo condition: for a fixed $c_1 \in (0,1)$, do not accept s_k if it does not satisfy the following inequality.

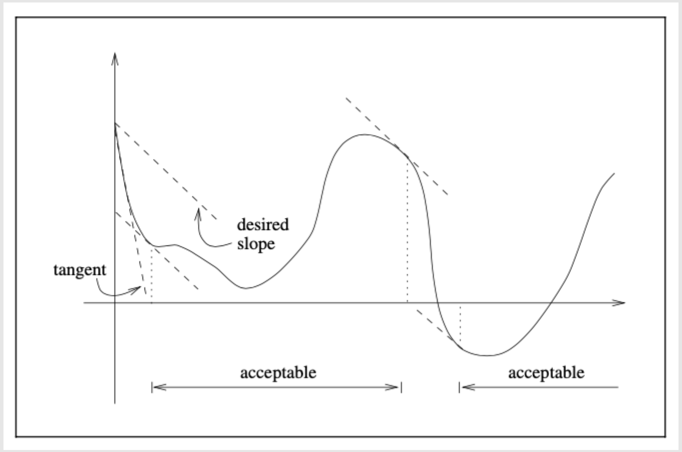
$$f(x_k + s_k p_k) \leq f(x_k) + s_k (c_1 \nabla f(x_k)^T) p_k.$$



The curvature condition

Armijo rule is a good start, but is not enough to guarantee substantial decrease at each iteration, because it is satisfied by any sufficiently small step size (you can prove this using Taylor's theorem). To rule this out, we add an extra requirement, the *curvature condition*: for a fixed $c_2 \in (0,1)$ the step length s_k must satisfy

$$\frac{d}{ds} f(x_k + sp_k)\Big|_{s=s_k} = \nabla f(x_k + s_k p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k = c_2 \frac{d}{ds} f(x_k + sp_k)\Big|_{s=0}$$



The Wolfe conditions

Armijo rule and the curvature condition are known as the *Wolfe conditions*:

$$f(x_k + s_k p_k) \leq f(x_k) + s_k(c_1 \nabla f(x_k)^T) p_k$$
$$\nabla f(x_k + s_k p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k$$

for fixed constants $0 < c_1 < c_2 < 1$.

To avoid too positive derivatives at x_{k+1}, one can consider the strong Wolve conditions instead:

$$f(x_k + s_k p_k) \leq f(x_k) + s_k(c_1 \nabla f(x_k)^T) p_k$$
$$|\nabla f(x_k + s_k p_k)^T p_k| \leq |c_2 \nabla f(x_k)^T p_k|$$

for fixed constants $0 < c_1 < c_2 < 1$.

Existence of step sizes

Lemma: Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable, and let p_k be a descent direction at x_k, and let $0 < c_1 < c_2 < 1$. If $f(x_k + sp_k) > -\infty$ for any $s > 0$, then there exists step lengths satisfying the Wolfe conditions and the strong Wolfe conditions.

Remark: The Wolfe conditions are invariant to linear scaling of the objective function as well as to affine change of variables. They are widely applicable.

Backtracking

In practice, the curvature condition is not required if one implements a backtracking strategy.

Backtracking algorithm:
Choose $v > 0, r \in (0,1), c \in (0,1)$. Then, at each iteration k

- Set $s_k = v$
- While $f(x_k + s_k p_k) > f(x_k) + s_k(c_1 \nabla f(x_k)^T)p_k$
  update $s_k = rs_k$

The backtracking strategy ensures that the chosen step s_k is the biggest step in the sequence $\{r^k v\}_{k\geq 0}$ that satisfies Armijo rule. More sophisticated step selections algorithms are described in Ch. 3.5 of the book by Nocedal and Wright.

Convergence of line search methods 1/2

Theorem [Zoutendijk]: Let $\{x_k\}_{k\geq 0}$ be a sequence generated by a line search method based on descent directions $\{p_k\}_{k\geq 0}$ and step lengths $\{s_k\}_{k\geq 0}$ that satisfy the Wolfe conditions. Assume that $f\colon \mathbb{R}^n \to \mathbb{R}$ is (at least locally) Lipschitz-continuously differentiable. Let $\{\alpha_k\}_{k\geq 0}$denote the sequence of angles between the descent directions p_k and the gradient values $\nabla f(x_k)$. Then,

$$\sum_{k\geq 0} \cos^2(\alpha_k)\|\nabla f(x_k)\| < \infty.$$

In particular, if there is a constant $c > 0$ such that $\cos(\alpha_k) > c$ for every $k \geq 0$, then

$$\lim_{k\to\infty} \|\nabla f(x_k)\| = 0,$$

that is, $\{x_k\}_{k\geq 0}$ converges to a stationary point.

Convergence of line search methods 2/2

Corollary: If $p_k = -B_k^{-1}\nabla f(x_k)$ and the matrices $\{B_k\}_{k\geq 0}$ are positive definite and their condition number is bounded uniformly, that is, if there is a constant $M > 0$ such that

$$\|B_k\|\left\|B_k^{-1}\right\| \leq M \quad \text{for all } k \geq 0,$$

then $\alpha_k > \arccos(M^{-1})$ for every $k \geq 0$, and $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$.

Steepest descent's rate of convergence

Theorem: Let $\{x_k\}_{k\geq 0}$ be a sequence generated by a line search method based on steepest descent directions $\{p_k\}_{k\geq 0}$ with optimal step lengths $\{s_k\}_{k\geq 0}$.

Assume that:

- $f: \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable,
- that $\{x_k\}_{k\geq 0}$ converges to a point x^* ,
- and that $\nabla^2 f(x^*)$ is positive definite.

Let $0 < \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ be the eigenvalues of $\nabla^2 f(x^*)$. Then, for any k sufficiently large,

$$f(x_{k+1}) - f(x^*) \leq r^2\big(f(x_k) - f(x^*)\big) \qquad \text{for any } r \in \left(\frac{\lambda_n - \lambda_1}{\lambda_n + \lambda_1}, 1\right).$$

Newton's method's local rate of convergence

Theorem: Let $\{x_k\}_{k\geq 0}$ be a sequence generated by Newton's method with $s_k = 1$.

Assume that:

- $f: \mathbb{R}^n \to \mathbb{R}$ is (at least locally) twice Lipschitz-continuously differentiable,
- $\nabla^2 f(x^*)$ is positive definite on a stationary point x^* ($\Rightarrow x^*$ a strict local minimizer).

Then,

- if x_0 is sufficiently close to x^*, then $x_k \to x^*$ with quadratic rate of convergence,
- $\|\nabla f(x_k)\| \to 0$ with quadratic rate of convergence.

Stopping criteria

Line search strategy: at the k-th iteration,

- choose a direction $p_k \in \mathbb{R}^n$,
- determine a *step length* s_k by solving $\min_{s>0} f(x_k + sp_k)$ approximately,
- set $x_{k+1} = x_k + s_k p_k$
- repeat until converge criteria are met.

Possible convergence criteria: pick small constants $C_1, C_2, C_3 > 0$ and

- stop if $\|\nabla f(x_k)\| < C_1$, or
- stop if $|f(x_k) - f(x_{k-1})| < C_2$, or
- stop if $\|x_k - x_{k-1}\| = s_{k-1}\|p_{k-1}\| < C_3$, or

any *relative tolerance* counterpart. Alternatively, stop after $N \in \mathbb{N}$ iterations.

Summary and self-study

Summary: this week we have learnt:

- the fundamentals of nonlinear optimization
- the strategy of line search methods
- steepest descent / Newton / Quasi-Newton methods
- step selection criteria and rates of convergence.

Self-study: Play with the `OR25_Rosenbrock.m` file and modify it to minimize the function

$$f: \mathbb{R}^2 \to \mathbb{R}, \qquad f(x) = \frac{1}{2} x^T Q x - b^T x$$

for various matrixes Q with different condition numbers (see page 42 in the book of Nocedal and Wright).

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 26 – Theory of constrained optimisation

Dr Pak-Hin Lee

Recap and plan of the day

Summary: so far we learnt:

- the fundamentals of nonlinear optimisation
- the strategy of line search methods
- steepest descent / Newton / Quasi-Newton methods
- step selection criteria and rates of convergence

In this lecture: Constrained optimisation theory, following closely Ch. 12 (Intro and 12.3) of the book *Numerical Optimization* by Jorge Nocedal and Stephen J. Wright.

Constrained optimisation

We consider problems of the form

$$\min_{x} f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in E \\ c_i(x) \geq 0, & i \in I \end{cases}$$

where E and I are two finite sets of indices and the functions $f: \mathbb{R}^n \to \mathbb{R}$ and $c_i: \mathbb{R}^n \to \mathbb{R}$ are sufficiently smooth functions.

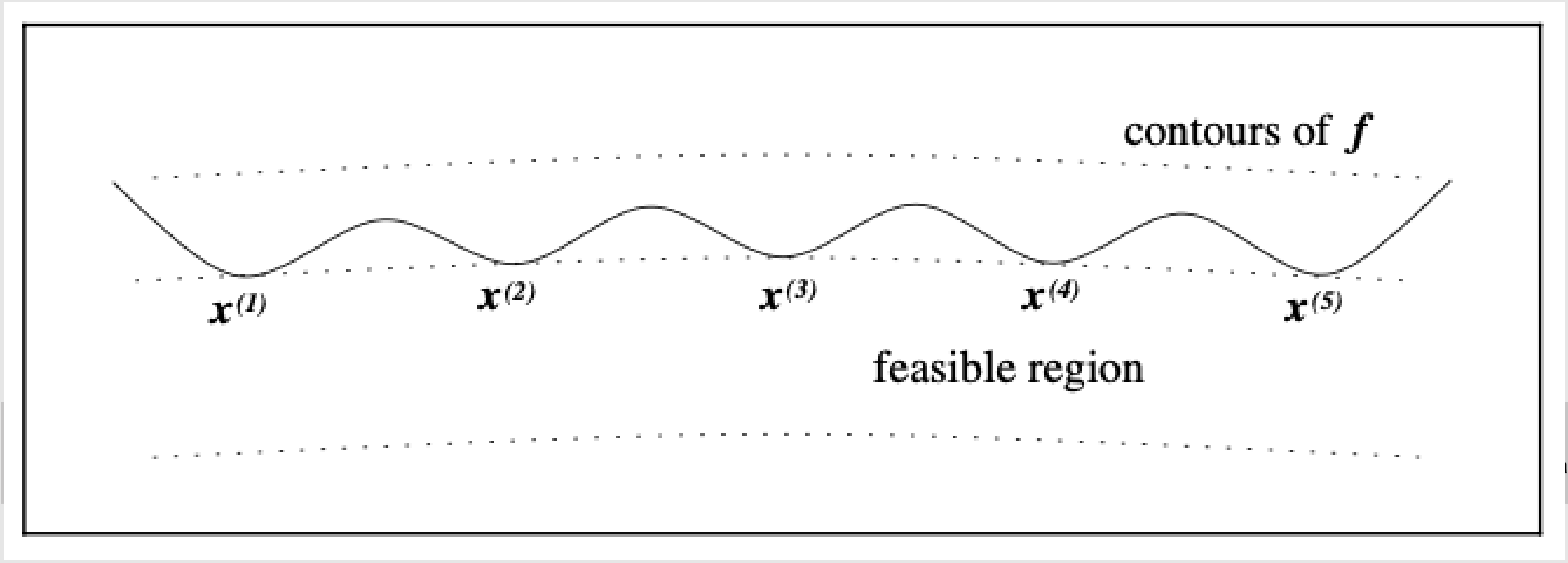
The function f is the objective function, the functions c_i, $i \in E$ are the equality constraints, and the functions c_i, $i \in I$ are the inequality constraints. The feasible set is

$$F := \{x \in \mathbb{R}^n \mid c_i(x) = 0 \text{ for } i \in E \text{ and } c_i(x) \geq 0 \; for \; i \in I\}$$

Local solutions

A point x^* is a *local solution* to $\min_{x \in F} f(x)$ if $x^* \in F$ and if there is a neighbourhood N of x^* so that $f(x^*) \leq f(x)$ for all $x \in N \cap F$.
A point x^* is a strict *local solution* to $\min_{x \in F} f(x)$ if $x^* \in F$ and if there is a neighborhood N of x^* so that $f(x^*) < f(x)$ for all $x \in N \cap F$ with $x \neq x^*$.

Example: the problem $\min_{x \in \mathbb{R}^2} (x_2 + 100)^2 + 0.01x_1^2$ subject to $x_2 - \cos(x_1) \geq 0$ has local solutions near the points $x^* = (k\pi, -1)^T$ for $k = \pm 1, \pm 3, \pm 5, \ldots$



Active set

The *active set* $A(x)$ at any feasible point x contains the indices of all equality constraints as well as the indices of the inequality constraints for which $c_i(x) = 0$, that is

$$A(x) = E \cup \{i \in I \mid c_i(x) = 0\}.$$

If x is a feasible point, an inequality constraint with index $i \in I$ is said to be *active* if $c_i(x) = 0$ and *inactive* if $c_i(x) > 0$.

Linear independence of constraint qualification (LICQ): We say that the LICQ hold at a feasible point x if the vectors in the set of active constraint gradients $\{\nabla c_i(x) \mid i \in A(x)\}$ are linearly independent.

First-order optimality conditions

Theorem: Let the LICQ hold at a local solution x^* to

$$\min_x f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in E \\ c_i(x) \geq 0, & i \in I \end{cases}$$

with smooth f and $c_i, i \in E \cup I$. Define the Lagrangian $L: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ as

$$L(x, y) := f(x) - \sum_{i=1}^{m} y_i c_i(x),$$

where $m = \#E + \#I$. Then, there exists $y^* \in \mathbb{R}^m$ such that

$$\begin{aligned} \nabla_x L(x^*, y^*) &= 0 & & \\ c_i(x^*) &= 0, & & for\ all\ i \in E \\ c_i(x^*) &\geq 0, & & for\ all\ i \in I \\ y_i^* &\geq 0, & & for\ all\ i \in I \\ y_i^* c_i(x^*) &= 0, & & for\ all\ i \in E \cup I \end{aligned}$$

Remarks on first-order optimality conditions

Remarks:

- The first-order optimality conditions are also known as Karush-Kuhn-Tucker (KKT) conditions.
- The conditions $y_i^* c_i(x^*) = 0$ for all $i \in E \cup I$ are called complementarity conditions. They imply that, if the constraint with index i is inactive, then $y_i^* = 0$.
- In light of the complementarity conditions, we can rewrite

$$\nabla_x L(x^*, y^*) = \nabla f(x) - \sum_{i=1}^{m} y_i \nabla c_i(x) = \nabla f(x) - \sum_{i \in A(x)} y_i \nabla c_i(x) = 0$$

  This implies that y^* is unique. Indeed, if there was another one, say z^*, then

$$0 = \nabla_x L(x^*, y^*) - \nabla_x L(x^*, z^*) = \sum_{i \in A(x)} (z_i - y_i) \nabla c_i(x),$$

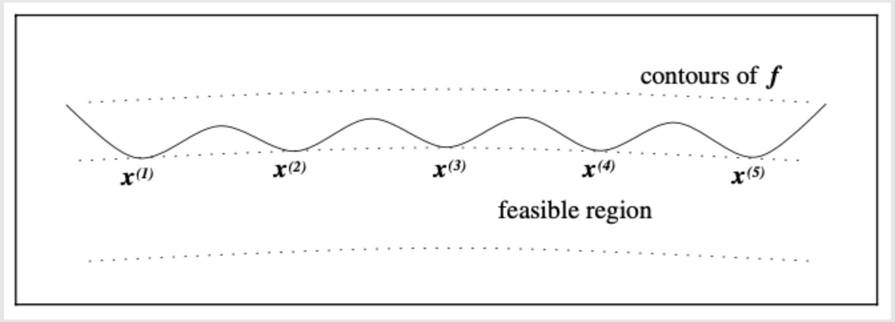
  which is not possible if $\{\nabla c_i(x) \mid i \in A(x)\}$ are linearly independent.

Example - Lagrangian

Consider the constrained optimisation problem

$$\min_{x\in\mathbb{R}^2} (x_2 + 100)^2 + 0.01x_1^2 \text{ subject to } x_2 - \cos(x_1) \geq 0$$

This has local solutions close to the points $x^* = (k\pi, -1)^T$ for $k = \pm 1, \pm 3, \pm 5, \ldots$



The Lagrangian is $L: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}, L(x, y) := (x_2 + 100)^2 + 0.01x_1^2 - y(x_2 - \cos(x_1))$

Example - KKT

The Lagrangian is $L: \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}, L(x, y) := (x_2 + 100)^2 + 0.01x_1^2 \; - y(x_2 - \cos(x_1))$

Then,$\nabla_x L(x, y) = (0.02x_1 - y\sin(x_1)\,, 2(x_2 + 100) - y)^T$, and the KKT conditions read

$$\begin{aligned} 0.02x_1^* - y^* \sin(x_1^*) &= 0 \\ 2(x_2^* + 100) - y^* &= 0 \\ x_2^* - \cos(x_1^*) &\geq 0, \\ y^* &\geq 0, \\ y^*(x_2^* - \cos(x_1^*)) &= 0. \end{aligned}$$

The equation $2(x_2^* + 100) - y^* = 0$ implies that $y^* = 2(x_2^* + 100)$, that is $y^* \neq 0$ unless $x_2^* = -100$. However, since $x_2^* \geq \cos(x_1^*)$, we conclude that $y^* > 0$, meaning that local solutions must satisfy $x_2^* = \cos(x_1^*)$ (that this, the constraint is active at local solutions). Therefore, the optimal solutions x^* must satisfy

$$0.02x_1^* - 2(\cos(x_1^*) + 100)\sin(x_1^*) = 0$$

Example – bound on solutions

Therefore, the optimal solutions x^* satisfy

$$0.02x_1^* - 2(\cos(x_1^*) + 100)\sin(x_1^*) = 0,$$

or, equivalently, $x_1^* = 10'000\sin(x_1^*) + 50\sin(2x_1^*)$. This equation implies that the local minima can happen only in (roughly) the interval $x_1^* \in [-10'050, 10'050]$. Note if we replace $x_2^* = \cos(x_1^*)$ in the objective function, we obtain the 1-dimensional problem

$$\min_{z\in\mathbb{R}} f(z) = (\cos(z) + 100)^2 + 0.01z^2$$

and that

$$f'(z) = 0.02z - 2(\cos(z) + 100)\sin(z).$$

Summary and self-study

Summary: today we learnt the fundamentals of constrained nonlinear optimisation and the first order optimality conditions (KKT conditions).

Self-study: Consider the constrained problem

$$\min_{x \in \mathbb{R}^2} \left(x_1 - \frac{3}{2}\right)^2 + \left(x_2 - \frac{1}{2}\right)^4 \ \textit{subject to} \ \begin{cases} 1 - x_1 - x_2 \geq 0 \\ 1 - x_1 + x_2 \geq 0 \\ 1 + x_1 - x_2 \geq 0 \\ 1 + x_1 + x_2 \geq 0 \end{cases}$$

Derive the KKT conditions and compute the Lagrange multiplier $y^* \in \mathbb{R}^4$

UNIVERSITY OF LEICESTER

www.le.ac.uk

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 27 – Penalty and augmented Lagrangian Methods

Dr Pak-Hin Lee

Recap and plan of the day

Recap: in the previous lecture we learnt:

- the fundamentals of nonlinear constrained optimisation
- the first-order necessary conditions for optimality (KKT conditions)

In this lecture: We discuss practical algorithms to solve (at least approximately) nonlinear constrained optimisation problems. We follow closely Ch. 17 of the book *Numerical Optimization* by Jorge Nocedal and Stephen J. Wright.

Constrained optimisation

We consider problems of the from

$$\min_x f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in E \\ c_i(x) \geq 0, & i \in I \end{cases}$$

where E and I are two finite sets of indices, and $f: \mathbb{R}^n \to \mathbb{R}$ and $c_i: \mathbb{R}^n \to \mathbb{R}$ are assumed to be sufficiently smooth functions.

Basic idea: Instead of solving the constrained problem, solve a sequence of unconstrained problems that converge to the original one. We consider two popular options:

- the quadratic penalty method,
- and the augmented Lagrangian method.

The quadratic penalty method

Idea: Instead of solving

$$\min_x f(x) \quad \text{subject to} \quad \begin{cases} c_i(x) = 0, & i \in E \\ c_i(x) \geq 0, & i \in I \end{cases}$$

define the *quadratic penalty function*

$$Q(x,p) := f(x) + \frac{p}{2}\sum_{i\in E} c_i^2(x) + \frac{p}{2}\sum_{i\in i} (\max(-c_i(x),0))^2$$

where $p > 0$ is a *penalty parameter*. Then, consider a sequence of positive penalty parameters $\{p_k\}_{k\geq 0}$ with $p_k \to \infty$ and solve (approximately)

$$\min_x Q(x,p_k) = f(x) + \frac{p_k}{2}\sum_{i\in E} c_i^2(x) + \frac{p_k}{2}\sum_{i\in i} (\max(-c_i(x),0))^2$$

sequentially, using the minimiser x_k of $Q(x,p_k)$ as initial guess to compute x_{k+1}.

The quadratic penalty method - example

Consider the optimisation problem $\min_{x \in \mathbb{R}^2} x_1 + x_2$ subject to $x_1^2 + x_2^2 - 2 = 0$.

The exact solution is $x^* = (-1, -1)^T$.

To solve this problem with the quadratic penalty method, define

$$Q(x, p) = x_1 + x_2 + \frac{p}{2}(x_1^2 + x_2^2 - 2)^2.$$

We can solve the associated sequence of unconstrained optimisation problems using Matlab, see `OR_Lect_27_quadratic_penalty.m`

The quadratic penalty method – penalty selection

The parameter sequence $\{p_k\}_{k\geq 0}$ can be chosen adaptively according to various criteria and heuristics (e.g., if it's tough to minimise $Q(x, p_k)$, in the next iteration use a p_{k+1} that is close to p_k).

Sometimes things just go wrong and one needs to restart using larger parameter penalisation values.

In the limit $p_k \to \infty$, the Hessian $\nabla^2 Q(x, p_k)$ becomes arbitrarily ill conditioned near the minimiser. This negatively affects the convergence of steepest-descent and quasi-Newton methods. Newton's method is theoretically unaffected by this, but practically one can still face numerical difficulties to compute the Newton direction.

The quadratic penalty method - convergence

Theorem: Assume that $I = \emptyset$. Let $\{p_k\}_{k\geq 0}$ be a sequence of positive penalty parameters with $p_k \to \infty$, and, for each $k \geq 0$, let x_k be the exact global minimiser of $Q(x, p_k)$. Then, every limit point x^* of $\{x_k\}_{k\geq 0}$ is a global minimiser of $\min_x f(x)$ subject to $c_i(x) = 0,\ i \in E$.

Theorem: Assume that $I = \emptyset$. Let $\{p_k\}_{k\geq 0}$ be a sequence of positive penalty parameters with $p_k \to \infty$. Let $\{e_k\}_{k\geq 0}$ be a sequence of positive tolerances with $e_k \to 0$. For each $k \geq 0$, let x_k be an approximate minimiser of $Q(x, p_k)$ that satisfies $||\nabla_x Q(x, p_k)|| \leq e_k$. Let x^* be a limit point of $\{x_k\}_{k\geq 0}$. Then,

- if x^* is infeasible, then it is a stationary point of the function $\sum_{i\in E} c_i^2(x)$,
- otherwise, if the gradients $\{\nabla c_i(x)\}_{i\in E}$ are linearly independent, x^* is a KKT point, and for any subsequence $\{x_k\}_{k\in K}$ with $x_k \to x^*$, $-p_k c_i(x_k) \to y_i^*$ for every $i \in E$.

Augmented Lagrangian method - idea

The previous theorem states that $-p_k c_i(x_k) \to y_i^*$ for every $i \in E$, that is

$$c_i(x_k) \approx \frac{-y_i^*}{p_k}.$$

To ensure that $c_i(x_k) \approx 0$, we need $p_k \gg 1$.

Idea: Add an estimate of the Lagrange multipliers y_i^* to the penalty method. More precisely, define $L_A: \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ by

$$L_A(x, y, p) \coloneqq f(x) - \sum_{i \in E} y_i c_i(x) + \frac{p}{2} \sum_{i \in E} c_i^2(x)$$

Then, for a p_k and an estimate y_i^k of y_i^*, compute x_k by minimising $L_A\left(x, y^k, p_k\right)$.

Augmented Lagrangian method – update of y^k

The point x_k is an approximate minimiser of

$$L_A\left(x, y^k, p_k\right) = f(x) - \sum_{i \in E} y_i^k c_i(x) + \frac{p_k}{2} \sum_{i \in E} c_i^2(x).$$

if $\nabla_x L_A\left(x_k, y^k, p_k\right) \approx 0$, that is, if

$$0 \approx \nabla_x L_A\left(x_k, y^k, p_k\right) = \nabla f(x_k) - \sum_{i \in E} \left(y_i^k - p_k c_i(x_k)\right) \nabla c_i(x_k).$$

Comparing this to

$$0 = \nabla L(x^*, y^*) = \nabla f(x_k) - \sum_{i \in E} y_i^* \nabla c_i(x_k)$$

we infer $y_i^* \approx y_i^k - p_k c_i(x_k)$, which suggests the update $y_i^{k+1} = y_i^k - p_k c_i(x_k)$.

Note that $c_i(x_k) \approx (y_i^k - y_i^*)/p_k \ll 1/p_k$ if $y_i^k \approx y_i^*$. So, we don't need $p_k \to \infty$.

Augmented Lagrangian method - convergence

Theorem: Assume that $I = \emptyset$. Let x^* be a local solution at which the LICQ is satisfied. Let some additional second-order sufficient conditions be satisfied for $y = y^*$(see Thm 12.6 in book by Nocedal and Wright). Then, there is a $p^* > 0$ such that x^* is a strict local minimiser of $L_A(x, y^*, p)$ for all $p \geq p^*$. Additionally, there exist scalars $a, b, c > 0$ such that, for all y^k, p_k such that $\|y^k - y^*\| \leq a p_k$:

- the problem
$$\min_x L_A(x, y^k, p_k) \;\; subject\ to \;\; \|x - x^*\| \leq b$$
has a unique solution x_k, and $\|x_k - x^*\| \leq c \, \|y^k - y^*\| / p_k$,
- $\|y^{k+1} - y^*\| \leq c \, \|y^k - y^*\| / p_k$
- $\nabla_x^2 L_A(x_k, y^*, p_k)$ is positive definite and the LICQ are satisfied at each x_k

Summary and self-study

Summary: today we learnt:

- the fundamentals of constrained nonlinear optimisation
- its first order optimality conditions
- two strategies to solve constrained problems as a sequence of unconstrained ones

Self-study: Consider the optimisation problem

$$\min_{x\in\mathbb{R}^2} x_1 + x_2 \quad \text{subject to } x_1^2 + x_2^2 - 2 = 0$$

Perform one step of the quadratic penalty method using the penalty parameter $p_0 = 2$. To solve the internal iteration, use Newton's method with initial guess $x_0 = (0,0)^T$

UNIVERSITY OF LEICESTER

www.le.ac.uk

MA3077 (DLI) Operational Research

Lecture 28 – optimization with state constraints

Dr Pak-Hin Lee

Recap and plan of the day

Summary: so far, we have learnt some theory and practice of constrained and unconstrained optimisation

In this lecture: Optimisation with a different type of constraints: state constraints. This topic is a bit advanced and is not covered in introductory books. Since we do not have good references for this topic, the presentation will be as pedestrian as possible. If you have any questions, please do not hesitate to contact me. I'll be happy to provide further explanations.

Optimisation with state constraints - example

Example: Consider the following problem: for a given vector $u_t \in \mathbb{R}^2$,

$$\min_{x\in\mathbb{R}} f(x,u) := \|u - u_t\|^2 \quad \mathit{subject\ to} \quad \begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix} u = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

The constraint $\begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix} u = \binom{1}{0}$ is called *state constraint*: to evaluate the objective function $f(x,u)$ one must solve the state constraint first. By defining $A(x) := \begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix}$, and the vector $b = \binom{1}{0}$, we see that this example is of the form

$$\min_{x} f(x,u) \quad \mathit{subject\ to} \quad A(x)u = b.$$

Optimization with state constraints

The previous example was of the form

$$\min_x f(x,u) \quad \textit{subject to} \quad A(x)u = b.$$

We observe that the state constraint is nonlinear in x, but linear in u. In general state constraints can be nonlinear also in u, that is, one can face problems of the form

$$\min_x f(x,u) \quad \textit{subject to} \quad F(x,u) = 0.$$

Remark: The constraints we encountered so far, which were only on the variable x, are known as *control constraints.* Of course, an optimization problem can have both control and state constraints. For example,

$$\min_{x\in\mathbb{R}} \|u - u_t\|^2 \quad \textit{subject to} \quad \begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix} u = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \textit{and}\ 0 \le x < 2\pi.$$

Reduced functionals

If the state constraint $F(x,u) = 0$ admits a unique solution u for each value of x, that is, if there is a solution function S such that $F(x,S(x)) = 0$ for every x, then we can convert the state constraint problem into an unconstrained one by defining the the $reduced\ functional\ g(x) \coloneqq f\big(x,S(x)\big)$.

Example: the state constrained problem

$$\min_{x\in\mathbb{R}} f(x,u) := \|u - u_t\|^2 \quad subject\ to \quad \begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix} u = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

can be rewritten as the unconstrained problem $\min_{x\in\in\mathbb{R}} g(x)$,where

$$g(x) \coloneqq \left\| \begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix}^{-1} \begin{pmatrix} 1 \\ 0 \end{pmatrix} - u_t \right\|^2$$

(in this case, we used $S(x) = \begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix}^{-1} \binom{1}{0}$).

Computing solution functions is expensive

Unfortunately, computing the solution function S can be very expensive (if at all possible). In particular, it can be much more expensive than a single solve of $F(x, u) = 0$ for a single value of x.

Example: Consider the linear system $A(x)u = b$ where $A(x) \in \mathbb{R}^{n,n}$, $b \in \mathbb{R}^{n}$, and $A(x)$ is also (invertible and) tridiagonal, that is

$$A_{i,j}(x) = 0 \; for \; all \;\; j \in \{1,2,\dots,n\} \setminus \{i-1, i, i+1\}.$$

Then, the linear system $A(x)u = b$ can be solved very quickly (in linear time with respect to n) and with little memory consumption using Thomas algorithm. On the other hand, computing the inverse $B(x) \coloneqq A(x)^{-1}$ requires at least n^2-many operations, and so does computing the matrix-vector product $B(x)b$ (because $B(x)$ is not sparse).

Differentiation 1/2

To solve

$$\min_x f(x,u) \quad \textit{subject to} \quad A(x)u = b$$

using a steepest descent (or BFGS) methods, we need to compute the total derivative $\frac{d}{dx} f(x,u)$. Because u depends on x via the state constraint, we employ the chain rule and obtain

$$\frac{d}{dx} f(x,u) = (\nabla_x f(x,u))^T + (\nabla_u f(x,u))^T \frac{d}{dx} u.$$

Question: what is $u' := \frac{d}{dx} u$?

Answer: differentiating the state constraint $A(x)u = b$, we obtain

$$\frac{d}{dx}(A(x)u) = (\nabla_x A(x))^T u + A(x)u' = 0 = \frac{d}{dx} b$$

Differentiation 2/2

The function u' is the solution to $(\nabla_x A(x))^T u + A(x)u' = 0$. Note that if $A(x) \in \mathbb{R}^{n,n}$ and $x \in \mathbb{R}^m$, then $\nabla_x A(x) \in \mathbb{R}^{m,n,n}$ and $u' \in \mathbb{R}^{n,m}$!

Example: If the optimization problem is

$$\min_{x\in\mathbb{R}^3} f(x,u) \quad subject\ to \quad \begin{pmatrix} 3+\cos(x_1) & -\sin(x_2) \\ \sin(x_3) & 3 \end{pmatrix} u = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

then $u' = (u_{x_1}, u_{x_2}, u_{x_3}) \in \mathbb{R}^{2,3}$ is the solution to

$$\begin{pmatrix} -\sin(x_1) & 0 \\ 0 & 0 \end{pmatrix} u + A(x)u_{x_1} = 0$$

$$\begin{pmatrix} 0 & -\cos(x_2) \\ 0 & 0 \end{pmatrix} u + A(x)u_{x_2} = 0$$

$$\begin{pmatrix} 0 & 0 \\ \cos(x_3) & 0 \end{pmatrix} u + A(x)u_{x_3} = 0$$

Differentiation via Lagrangians 1/2

We just saw that to evaluate $\frac{d}{dx}f(x,u)$ when $f(x,u)$ is constrained to $A(x)u = b$, we need to solve $A(x)u = b$ and $(\nabla_x A(x))^T u + A(x)u' = 0$ first.

The linear system $(\nabla_x A(x))^T u + A(x)u' = 0$ is very large! Can we avoid that? Yes!

Consider the Lagrangian $L: \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$,

$$L(x,u,p) = f(x,u) - p^T(A(x)u - b).$$

Note that, if u satisfies $A(x)u = b$, then $L(x,u,p) = f(x,u)$ for any $p \in \mathbb{R}^n$. Thus, compute

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}x}L(x,u,p) &= (\nabla_x f(x,u))^T + (\nabla_u f(x,u))^T u' - p^T((\nabla_x A(x))^T u + A(x)u') \\ &= (\nabla_x f(x,u))^T - p^T(\nabla_x A(x))^T u + \big((\nabla_u f(x,u))^T - p^T A(x)\big)u'\end{aligned}$$

Differentiation via Lagrangians 2/2

Since, for any $p \in \mathbb{R}^n$, $L(x,u,p) = f(x,u)$ when $A(x)u = b$, it must also hold that

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}x} L(x,u,p) &= (\nabla_x f(x,u))^T + (\nabla_u f(x,u))^T u' - p^T((\nabla_x A(x))^T u + A(x)u') \\ &= (\nabla_x f(x,u))^T - p^T(\nabla_x A(x))^T u + \big((\nabla_u f(x,u))^T - p^T A(x)\big)u' \\ &= (\nabla_x L(x,u,p))^T + (\nabla_u L(x,u,p))^T u' \\ &= \frac{d}{dx} f(x,u)\end{aligned}$$

for any $p \in \mathbb{R}^n$. If p is such that $(\nabla_u f(x,u))^T - p^T A(x) = 0$, i.e., $A(x)^T p = \nabla_u f(x,u)$, then, $\nabla_u L(x,u,p) = 0$, and

$$\frac{\mathrm{d}}{\mathrm{d}x} L(x,u,p) = (\nabla_x L(x,u,p))^T \frac{d}{dx} f(x,u)$$

Remark: the equation $A(x)^T p = \nabla_u f(x,u)$ is called the *adjoint equation*, and its solution is the *adjoint variable*.

Differentiation via Lagrangians – example 1/2

Example: The Lagrangian necessary to differentiate

$$\min_{x\in\mathbb{R}^3} \|u-u_t\|^2 \quad subject\ to \quad \begin{pmatrix} 3+\cos(x_1) & -\sin(x_2) \\ \sin(x_3) & 3 \end{pmatrix} u = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

is $L\colon \mathbb{R}^3 \times \mathbb{R}^2 \times \mathbb{R}^2$, $L(x,u,p) = \|u-u_t\|^2 - p^T\left(\begin{pmatrix} 3+\cos(x_1) & -\sin(x_2) \\ \sin(x_3) & 3 \end{pmatrix} u - \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right)$.

Differentiating $L(x,u,p)$ with respect to the variable p gives the state equation

$$\begin{pmatrix} 3+\cos(x_1) & -\sin(x_2) \\ \sin(x_3) & 3 \end{pmatrix} u = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Differentiating $L(x,u,p)$ with respect to the variable u gives the adjoint equation

$$\begin{pmatrix} 3+\cos(x_1) & \sin(x_3) \\ -\sin(x_2) & 3 \end{pmatrix} p = 2(u-u_t)$$

Differentiation via Lagrangians – example 2/2

The Lagrangian is $L\colon \mathbb{R}^3 \times \mathbb{R}^2 \times \mathbb{R}^2$,

$$L(x,u,p) = \|u - u_t\|^2 - p^T\left(\begin{pmatrix} 3+\cos(x_1) & -\sin(x_2) \\ \sin(x_3) & 3 \end{pmatrix} u - \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right).$$

The state and adjoint equations are

$$\begin{pmatrix} 3+\cos(x_1) & -\sin(x_2) \\ \sin(x_3) & 3 \end{pmatrix} u = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \; and \begin{pmatrix} 3+\cos(x_1) & \sin(x_3) \\ -\sin(x_2) & 3 \end{pmatrix} p = 2(u - u_t)$$

Differentiating $L(x,u,p)$ with respect to the variable x gives

$$\begin{aligned} \nabla_x L(x,u,p) &= -\nabla_x (p^T \begin{pmatrix} 3+\cos(x_1) & -\sin(x_2) \\ \sin(x_3) & 3 \end{pmatrix} u) \\ &= -\nabla_x (p_1 u_1 (3+\cos(x_1)) - p_1 u_2 \sin(x_2) + p_2 u_1 \sin(x_3) \\ &= (p_1 u_1 \sin(x_1), p_1 u_2 \cos(x_2), -p_2 u_1 \cos(x_3))^T = \frac{\mathbf{d}}{\mathbf{d}\boldsymbol{x}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})^T \end{aligned}$$

Summary and self-study

Summary: today we have learnt how to tackle state constrained optimization problems by computing the derivatives required by the steepest descent method (or by BFGS) using the Lagrange method.

Self-study: Solve

$$\min_{x\in\mathbb{R}} f(x,u) := \left\| u - \begin{pmatrix} 2 \\ 2 \end{pmatrix} \right\|^2 \quad \textit{subject to} \quad \begin{pmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{pmatrix} u = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

using Matlab's function `fminunc` . To provide both $f(x,u)$ and its derivative, use need to use the commands (see Matlab's documentation):

```
options = optimoptions('fminunc', 'SpecifyObjectiveGradient',
true)
x = fminunc(@..., ..., options)
```

Summary and self-study

Summary: this week we have learnt:

- the fundamentals of constrained optimization
- the first order necessary conditions for constrained optimization
- the penalty method and the augmented Lagrangian method
- how to differentiate state constrained functionals.