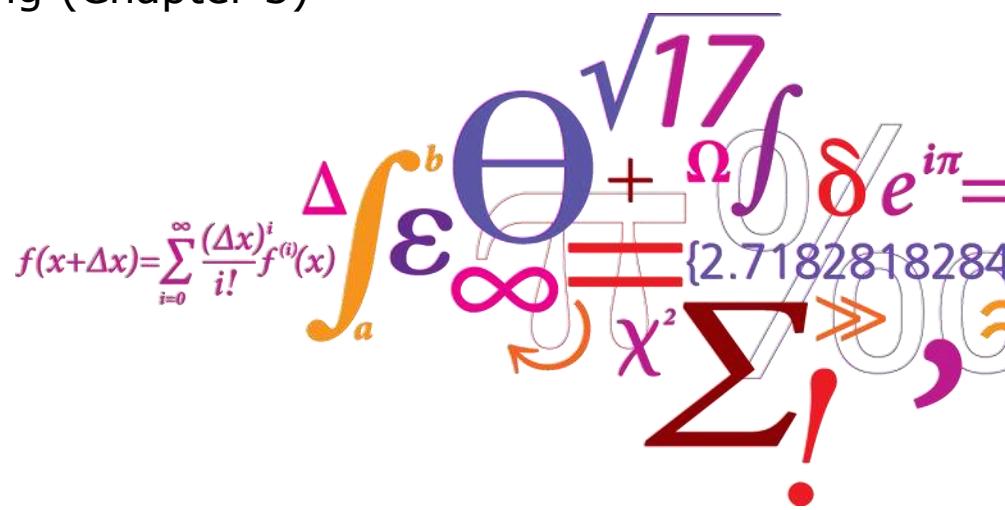


42101 Introduction to Operations Research

- What is Operations Research?
- Overview of the course
- Introduction to Linear Programming (Chapter 3)

Stefan Røpke

Richard Lusby



What is Operations Research (OR)?

- Very short definition:
Application of mathematical techniques to decision making
- Examples of applications
 - Planning of production: which products should be produced at what time?
 - Work planning: Which employee should be on duty at what time?
- ... it becomes clear as we go on!

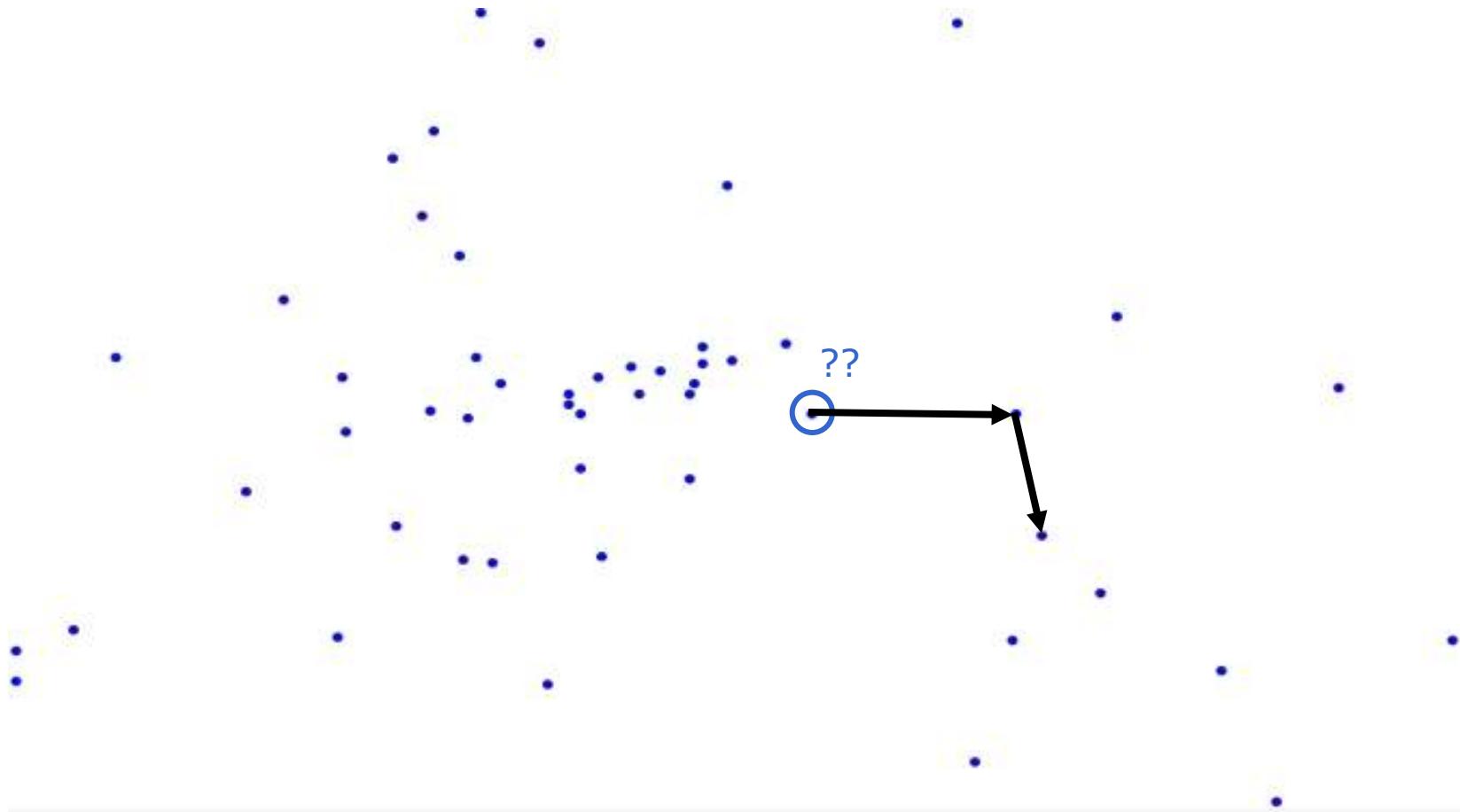
What is Operations Research?

- Useful skills when studying Operations Research
 - Mathematics
 - Computer Science
 - Machine learning/Artificial intelligence
 - Ability to understand new application areas
 - Common sense

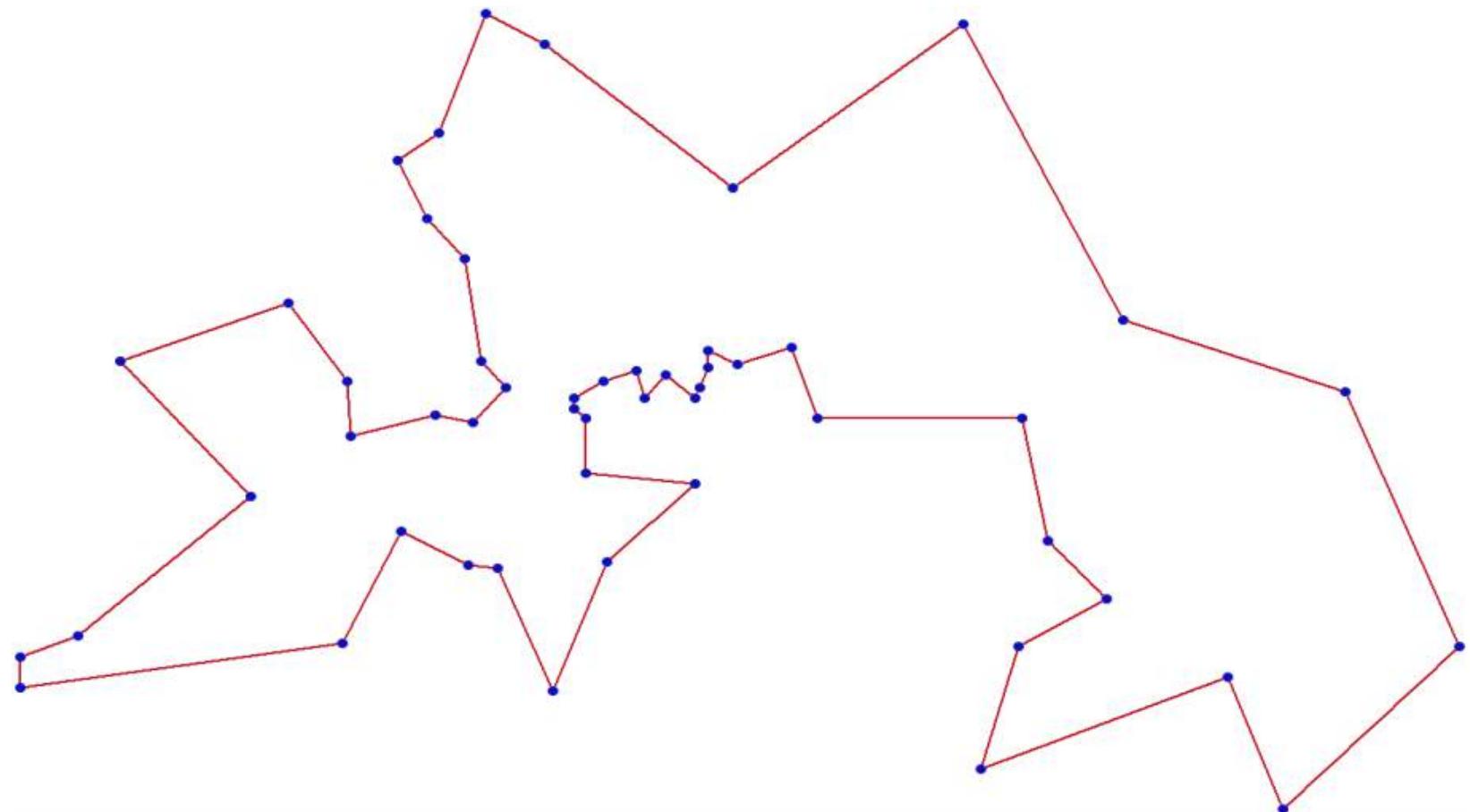
Very short history of operations research

- Debatable when and where operations research started.
- However, a mile stone is the use of operations research in UK during World War 2 in order to improve war efforts.
- After the second world war the techniques were used for peaceful purposes and accelerated due to the arrival of computers
- Now OR techniques are used in many places. Everyday we are in contact with products and serviced that in one way or the other have been influenced by OR.

The traveling salesman problem (TSP)

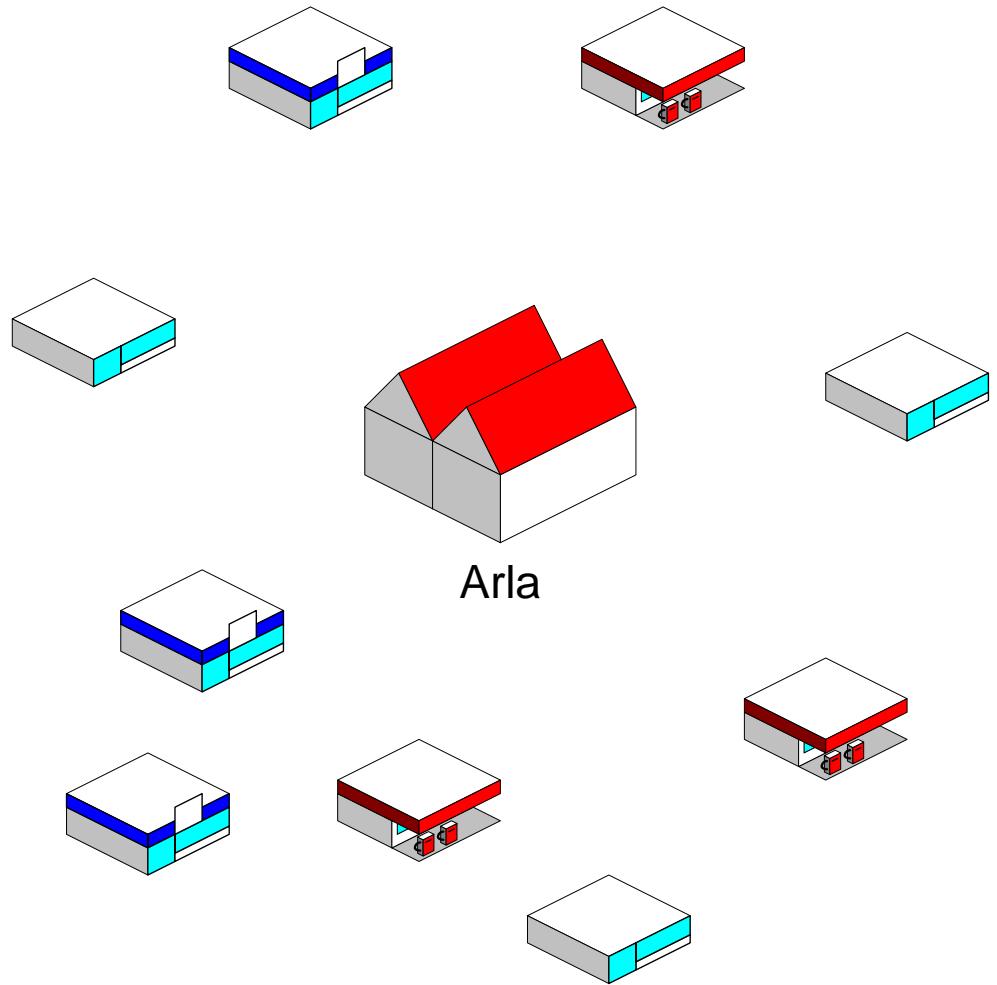


The traveling salesman problem (TSP)



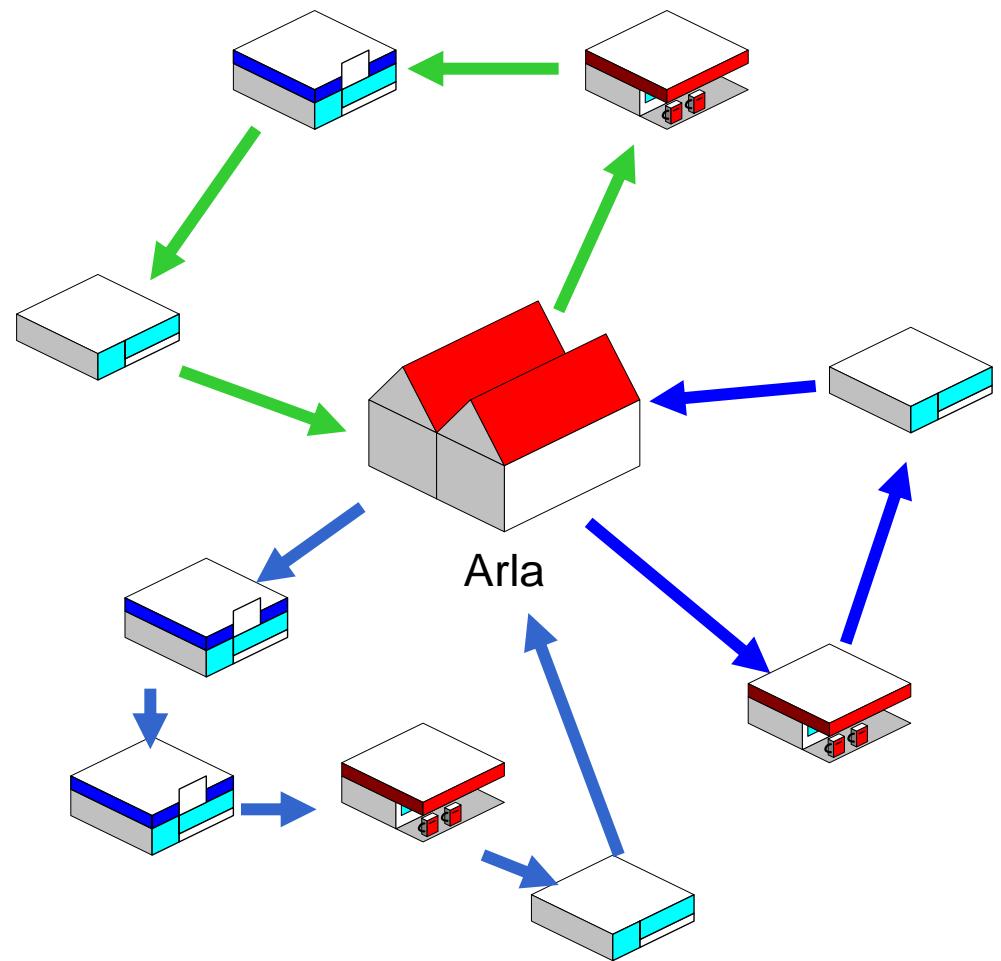
Distribution planning

Our task: supply super markets, gas stations, corner stores etc. with dairy (milk) products.
Each truck can at most serve 4 shops.



Distribution planning

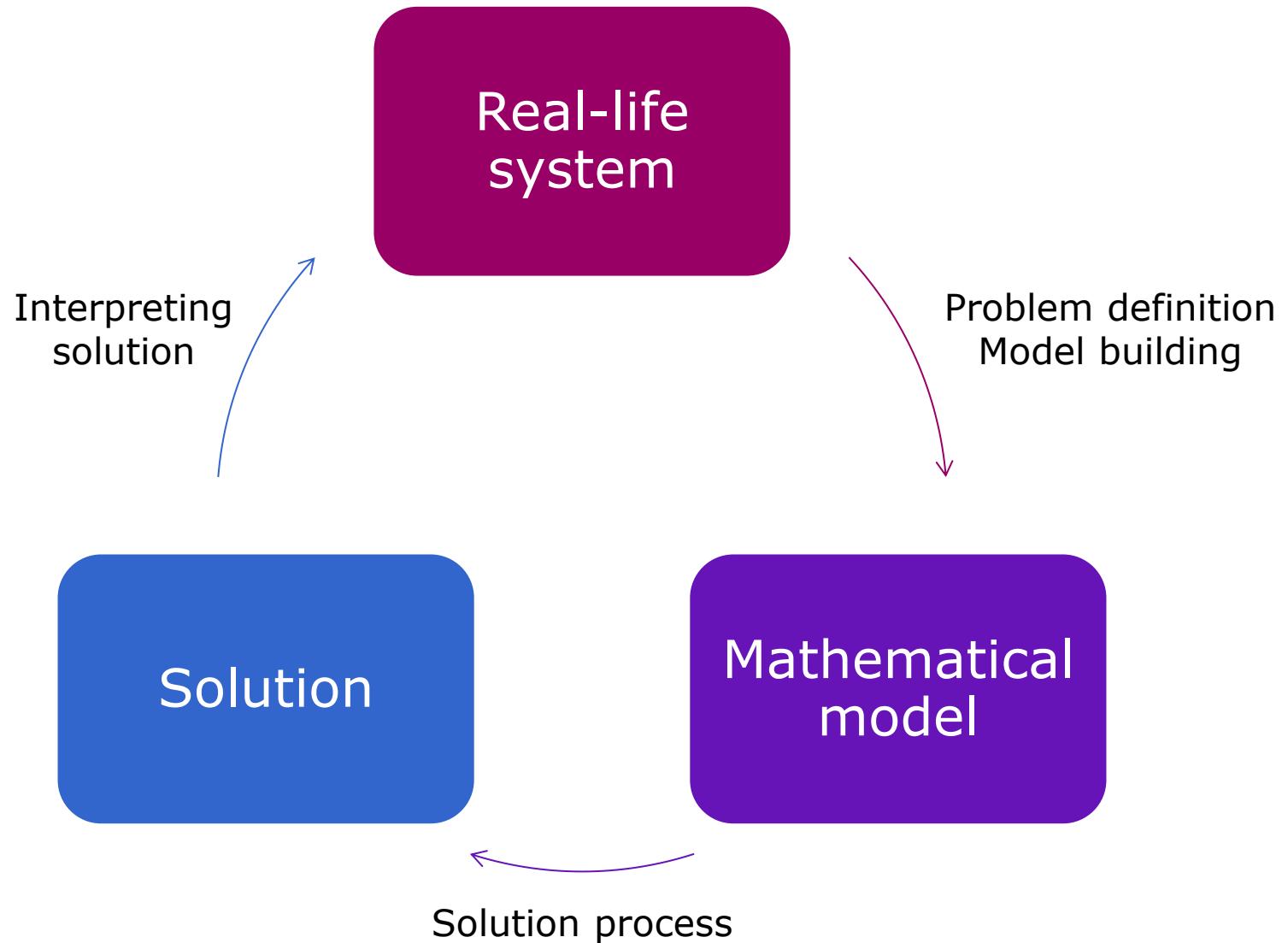
Our task: supply super markets, gas stations, corner stores etc. with dairy (milk) products.
Each truck can at most serve 4 shops.



Operations research is everywhere

- Products are often brought to the end customer using trucks, ships or planes. OR may have been involved in planning the transport.
- The production of products you buy may have been planned using OR
- When you ride on the bus, metro or the S-train OR may have been used in the planning processes
- OR methods may have been used to design the electronics in your smartphone, computer, etc. Used to optimize signal path, for example.

Work cycle in OR



Course overview

#	Date	Topic	Reading material	Teacher	Project
1	02-sep	Introduction to Operations Research and Linear Programming	3.1-3.4. (1.1-1.4) (2.1-2.7)	SR	
2	09-sep	Modeling with linear programming The Simplex algorithm	3.1-3.4. 4.1-4.4. (Appendix 6)	SR	
3	16-sep	Basic graph theory. Handling non-standard LPs	4.5-4.7. 10.1-10.2.	SR	
4	23-sep	Simplex using matrix computations	5.1-5.5. (Appendix 4)	SR	Project 1 start. Hand in: 13th of October
5	30-sep	Duality and sensitivity analysis	6.1-6.6 7.1-7.2	SR	
6	07-okt	Modeling with integer variables	12.1-12.4.	SR	
	14-okt	Autumn holiday			

Course overview

7	21-okt	Modeling with integer variables	12.1-12.4.	RL	
8	28-okt	Modeling with integer variables. Greedy algorithm	12.1-12.4.	RL	Project 2 start (preliminary)
9	04-nov	Modeling with integer variables. Solving a model iteratively	12.1-12.4.	RL	
10	11-nov	Total unimodularity: Assignment problem Min cost flow	9.3,9.4, 10.6 Lecture notes on the assignment problem.	RL	
11	18-nov	Linear relaxation, Branch and bound	12.5-12.7	RL	
12	25-nov	Dual simplex transport problem	8.1, 9.1 Lecture notes on the network simplex algorithm	RL	
13	02-dec	Solve old exam		RL / SR	
???		Question hour. Date is to be decided.		RL / SR	
11-dec		Exam. See www.eksamensplan.dtu.dk			

Practical info

- **Main book**

- Hillier, Lieberman: Introduction to Operations Research, 10th edition.
[earlier version are similar, but page and chapter references in class
may be “off”]
 - Material on DTU inside.

- **Prerequisites** (Simultaneous Linear Equations, Matrix operations)

- Read appendix 4 and 6
 - Appendix 6 you can get from the book web-page or from DTU inside.

- **DTU inside**

- Course overview, slides, projects, additional reading material
 - “Final” version of slides ready around noon Sunday.

- **Language:** English

- You can hand in projects in Danish or English
 - You can answer exam in Danish or English

Practical information

- **Lectures**

- Monday from 13 to 15 building 341, room 21

- **Exercises**

- Monday from 15 to 17
- Building 324, room 040, 050 and 060.

- **Preparation:**

- Read the text and look at slides before lecture.
- If you do not finish exercises, then complete them before next lecture.

- **Teachers**

- Stefan Røpke, ropke@dtu.dk (first part)
- Richard Lusby, rmlu@dtu.dk (second part)

- **Teaching assistants**

Kristine Børsting	Morten Bondorf Gerdes	Elisabeth Marie Heegaard
		

Demanding course?

- You have to make an effort to meet the learning objectives!
 - Be active from the start – we keep building on material from past lectures.
 - Read the book. It's not enough to be present at the lectures.
 - Solve all exercises
 - Exercises from 15 to 17 are more important compared to the lectures.

Two projects

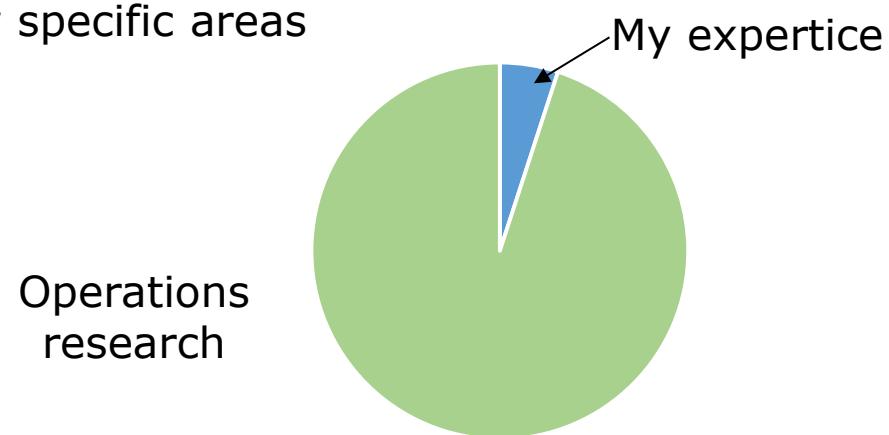
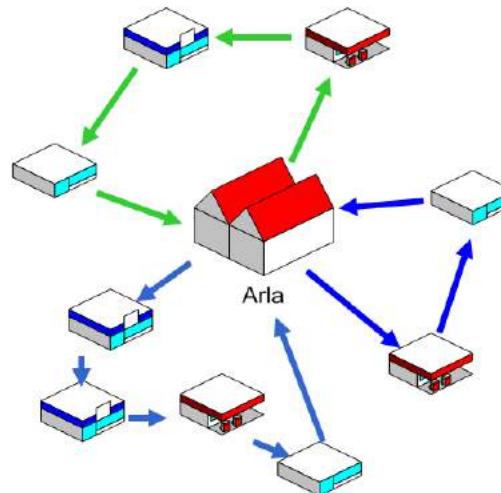
- Show what you can do with OR.
- Go a little further with some topics.
- Preparation for the exam.
- Projects start on the 23th of September and 28th of October (Preliminary plan)
- Content is the material covered until (and including) the project start day
- Do projects in groups of two to three
- You need to pass both projects in order to attend the exam. We make an overall assessment of the two projects.
 - Perfect project: 100 point
 - Passed: 50-100 point

Exam

- 4 hours written exam
- “All aids allowed”.
- Combination of multiple choice and text questions. Mainly multiple-choice (could be multiple-choice only).

About myself

- Contact:
 - Email: ropke@dtu.dk
 - Office 224, building 424.
- Background
 - Been employed at DTU since 2008
 - Professor since 2012.
 - Education in Computer Science (University of Copenhagen)
 - I do research in operations research.
 - Is especially interested in applications within transport and solution methods.
 - World class research in a few specific areas



Introduction to Linear programming (LP)

Programming = make a program = planning

Content of chapter 3

- 3. Introduction to LP
- 3.1 Prototype example: Wyndor
- 3.2 LP-model
- 3.3 Assumptions for LP
- 3.4 More examples

3.1 Wyndor: problem

- 2 products (glass doors, windows)
- Requires time on 3 plants

	1	2	Capacity (hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

- Profit is per unit and should be multiplied by 1000\$
- We want to plan the production so that our profit is maximized

3.1 Wyndor model

Decision variables:

x_1 = production of product 1 (units/week)

x_2 = production af product 2 (units/week)

	1	2	Capacity (Hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

3.1 Wyndor model

Decision variables:

x_1 = production of product 1 (units/week)

x_2 = production af product 2 (units/week)

Symbolic model (LP)

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

$$x_1 \geq 0, \quad x_2 \geq 0$$

	1	2	Capacity (Hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

3.1 Wyndor model

Decision variables:

x_1 = production of product 1 (units/week)

x_2 = production af product 2 (units/week)

Symbolic model (LP)

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

Components of the model:

- Decision variables (what we can decide)
- Objective function (what we wish to optimize)
- Constraints (that we have to respect)

	1	2	Capacity (Hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

3.1 Wyndor model

Decision variables:

x_1 = production of product 1 (units/week)

x_2 = production af product 2 (units/week)

Symbolic model (LP)

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

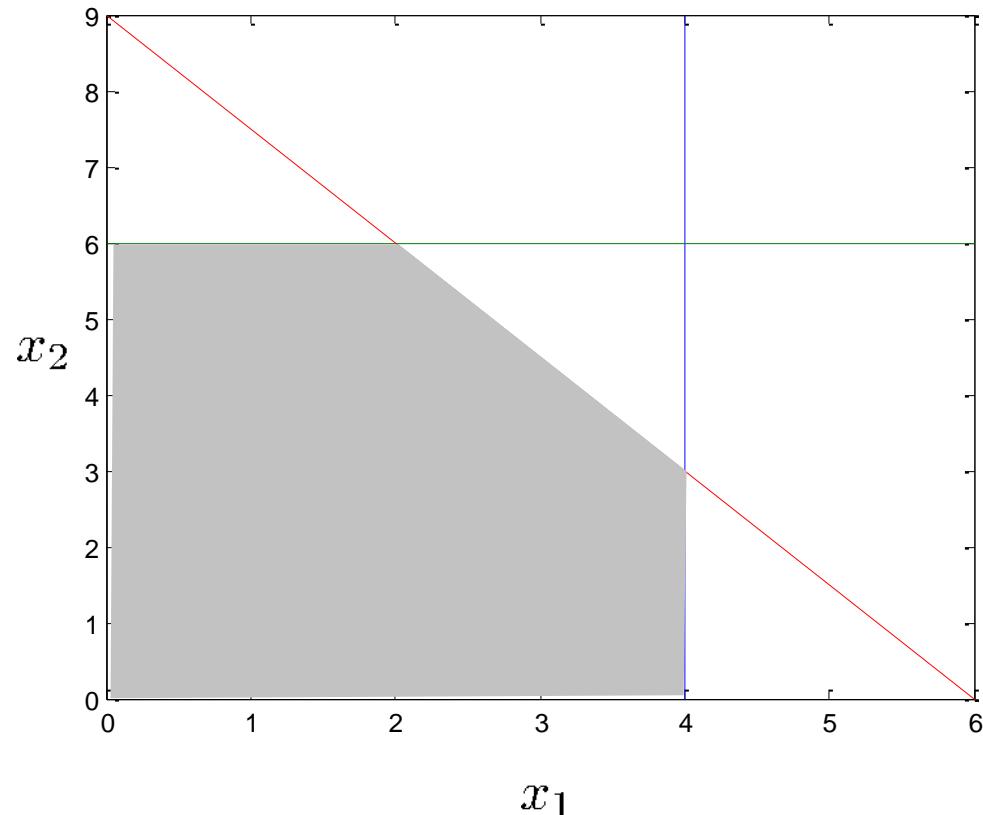
$$x_1 \geq 0, x_2 \geq 0$$

	1	2	Capacity (Hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

Is it ok if decision variable is non-integer?

E.g. $x_1 = 2.76$?

Wyndor solution



$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4 \quad \text{●}$$

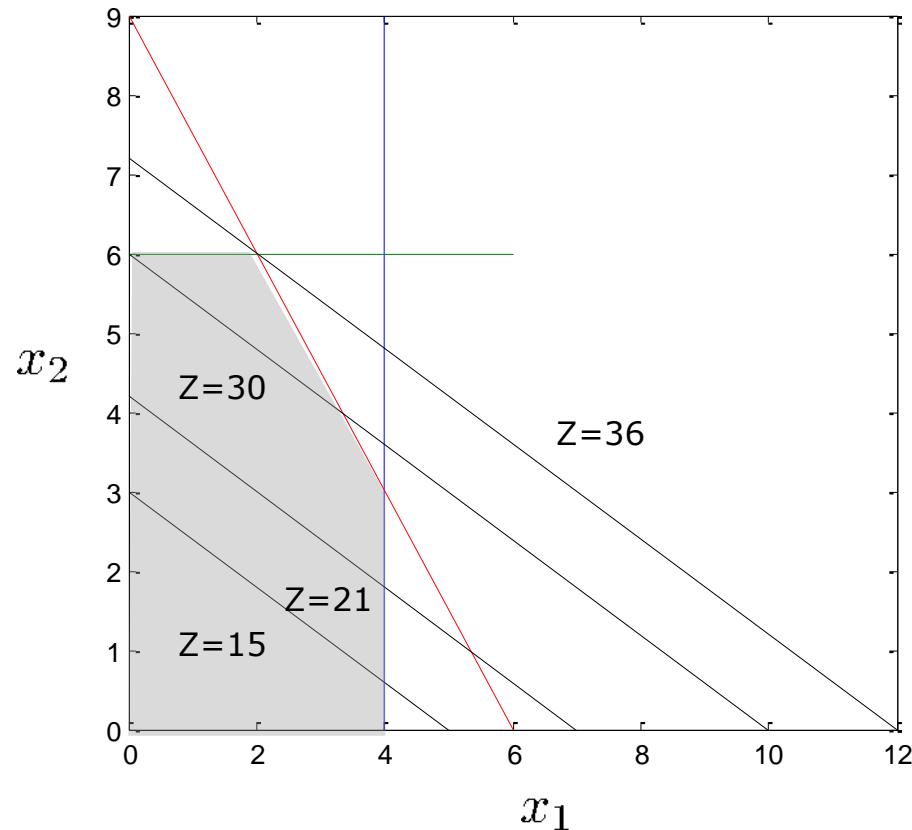
$$2x_2 \leq 12 \quad \text{●}$$

$$3x_1 + 2x_2 \leq 18 \quad \text{●}$$

and

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

Wyndor solution



$$\begin{aligned}
 x_1^* &= 2 \text{ units per week} \\
 x_2^* &= 6 \text{ units per week} \\
 Z^* &= 36k\$ \text{ per week} \\
 &= 36,000\$ \text{ per week}
 \end{aligned}$$

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4 \quad \bullet$$

$$2x_2 \leq 12 \quad \bullet$$

$$3x_1 + 2x_2 \leq 18 \quad \bullet$$

and

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

The graphical method

1. Draw the line for the objective function for a fixed Z value. Make sure that the line intersects the feasible area.
2. While keeping the slope fixed, "move" this line toward increasing values for Z .
3. The point (or the line) that the line touch just before leaving the feasible area is the optimal solution(s).

3.2 LP-model in general

- Allocating m resources to n activities
- Decision variables: x_j : level of activity j
- Parameters (input)
 - c_j : How useful is activity j
 - b_i : How many units of resource i are available
 - a_{ij} : units of resource i consumed by each unit of activity j
- Z : measure of performance

$$\max Z = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

subject to

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \leq b_2$$

⋮

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m$$

$$x_1, x_2, \dots, x_n \geq 0$$

3.2 LP-model in general

- Allocating m resources to n activities
- Decision variables: x_j : level of activity j
- Parameters (input)
 - c_j : How useful is activity j
 - b_i : How many units of resource i are available
 - a_{ij} : units of resource i consumed by each unit of activity j
- Z : measure of performance

Not always useful or possible to interpret an LP model this way

“Our” standard form

$$\max Z = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

subject to

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \leq b_2$$

⋮

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m$$

$$x_1, x_2, \dots, x_n \geq 0$$

LP's as above are said to be on the *standard form*.

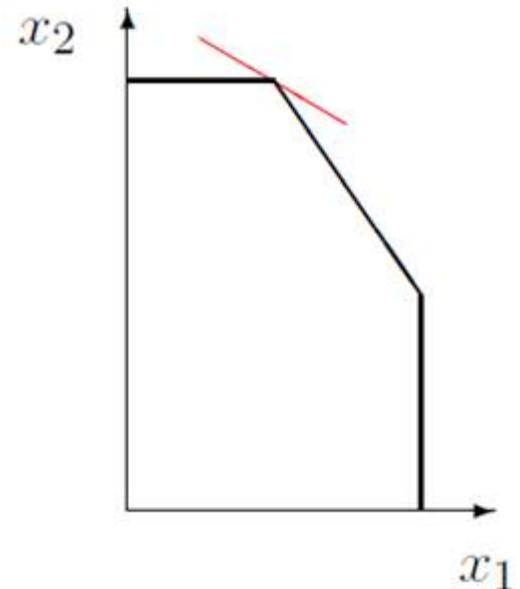
Deviation from the standard form:

- Minimization
- \geq constraints
- $=$ constraints
- Decision variables that can take negative values.

Terminology

Terminology

- Feasible solution
- Infeasible solution
- Feasible region
- Optimal solution
- Objective function
- Objective value
- Unbounded solution
- Corner-point feasible solution – CPF



Assumptions for linear programming

$$\max Z = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

subject to

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \leq b_2$$

⋮

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m$$

$$x_1, x_2, \dots, x_n \geq 0$$

- Linear objective function and constraints.
- c_j and a_{ij} are constants, only x_j are variables. Objective function and constraints are **always** on this form.
- Variables are real numbers. We cannot force them to be integers, for example.

3.4 Examples

- p. 45 Design of Radiation Therapy 
- p. 47 Regional Planning
- p. 51 Controlling Air Pollution
- p. 53 Reclaiming Solid Wastes
- p. 57 Personnel Scheduling 
- p. 60 Distributing Goods Through a Distribution Network

p. 45 Design of Radiation Therapy

- A tumor is treated with radiation therapy.
- Doctors are considering two different entry points for the radiation beams

	Beam 1	Beam 2	Restrictions
Healthy anatomy	0.4	0.5	
Critical tissues	0.3	0.1	≤ 2.7
Tumor region	0.5	0.5	$= 6$
Center of tumor	0.6	0.4	≥ 6

- All numbers in the table are in kilorads
- Numbers for beam 1 and 2 are for a normal dose. We can give higher and lower doses as necessary.
- How to model this? Think of
 - Decision variables
 - Constraints
 - Objective function

p. 45 Radiation Therapy: model

- x_j : intensity of beam j ($j = 1, 2$)
- Z : radiation dose to healthy anatomy (kilorads)

$$\min Z = 0.4x_1 + 0.5x_2$$

subject to

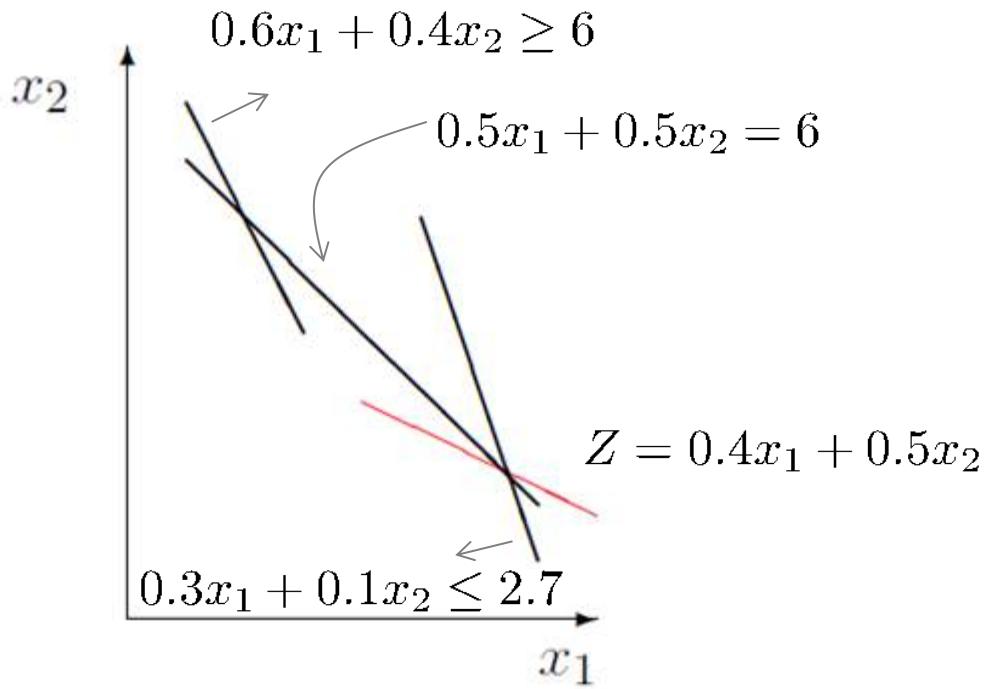
$$0.3x_1 + 0.1x_2 \leq 2.7$$

$$0.5x_1 + 0.5x_2 = 6$$

$$0.6x_1 + 0.4x_2 \geq 6$$

$$x_1, x_2 \geq 0$$

p. 45 Radiation Therapy: solution



$$\min Z = 0.4x_1 + 0.5x_2$$

subject to

$$0.3x_1 + 0.1x_2 \leq 2.7$$

$$0.5x_1 + 0.5x_2 = 6$$

$$0.6x_1 + 0.4x_2 \geq 6$$

$$x_1, x_2 \geq 0$$

$x_1^* = 7.5$ times standard dose

$x_2^* = 4.5$ times standard dose

$Z^* = 5.25$ kilorads

Modelbuilding – how to?

- First of all: Understand what the model has to decide.
- Figure out what decision variables are needed
- How do you interpret the variables? Write down the interpretation of the variables and consider the unit attached to the variable.
- Examples
 - Wyndor: x_1 : Production of doors (number of door batches per week)
 - Radiation therapy: x_1 : Intensity of beam 1 (relative to a standard beam)
- Write up objective function and constraints

Personnel Scheduling

Time	Shift					Agents needed
	1	2	3	4	5	
6-8	<input checked="" type="checkbox"/>					48
8-10	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				79
10-12	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				65
12-14	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			87
14-16		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			64
16-18		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		73
18-20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			82
20-22			<input checked="" type="checkbox"/>			43
22-24			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		52
24-6				<input checked="" type="checkbox"/>		15
Cost (\$ per day per agent)	170	160	175	180	195	

- How to model this? Think of
 - Decision variables
 - Constraints
 - Objective function

Personnel Scheduling: model

x_j : number of agents used on shift j ($j = 1, 2, \dots, 5$)

$$\min 170x_1 + 160x_2 + 175x_3 + 180x_4 + 195x_5$$

subject to

$$\begin{aligned}
 x_1 & \geq 48 \\
 x_1 + x_2 & \geq 79 \\
 x_1 + x_2 & \geq 65 \\
 x_1 + x_2 + x_3 & \geq 87 \\
 x_2 + x_3 & \geq 64 \\
 x_3 + x_4 & \geq 73 \\
 x_3 + x_4 & \geq 82 \\
 x_4 & \geq 43 \\
 x_4 + x_5 & \geq 52 \\
 x_5 & \geq 15 \\
 x_1, x_2, x_3, x_4, x_5 & \geq 0 \text{ and integer}
 \end{aligned}$$

Personnel Scheduling - solution

Time	Shift					Agents needed
	1	2	3	4	5	
6-8	<input checked="" type="checkbox"/>					48
8-10	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				79
10-12	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				65
12-14	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			87
14-16		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			64
16-18		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		73
18-20		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			82
20-22			<input checked="" type="checkbox"/>			43
22-24			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		52
24-6				<input checked="" type="checkbox"/>		15
Cost (\$ per day per agent)	170	160	175	180	195	

$$Z^* = 30610\$ \text{ per day}$$

$$x_1^* = 48, x_2^* = 31, x_3^* = 39, x_4^* = 43, x_5^* = 15$$

Very short history of Linear Programming

- First used in general form: Kantorovich (1939)
- Dantzig proposed the simplex algorithm (1947)
 - We learn about this method next week
- Numerous application since then

Solve LP models using software

- See Julia slides

Exercises

- See tasks on DTU inside
- Let's move to the exercise rooms:
- Building 324, room 040, 050 and 060

Simplex method

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

Mathematical symbols and equations are overlaid on the slide, including:
- A large orange integral symbol with 'a' and 'b' bounds.
- A purple theta symbol with a blue square root of 17 above it.
- A pink infinity symbol.
- A red sigma symbol with a red double arrow above it.
- A red exclamation mark at the bottom right.
- A red equals sign with a red arrow pointing to the right.
- A red brace grouping several numbers: {2.7182818284, 3.1415926535, 1.6180339887, 0.5772156649, 0.3333333333, 0.2366471679, 0.1465571231, 0.0989010938, 0.0666666667, 0.0467330902, 0.0333333333, 0.0243978179, 0.0178579446, 0.0133333333, 0.0098901093, 0.0072727273, 0.0052366471, 0.0038901093, 0.0028571429, 0.0020357944, 0.0014285714, 0.0010000000, 0.0007085714, 0.0005000000, 0.0003571429, 0.0002500000, 0.0001785714, 0.0001250000, 0.0000890109, 0.0000613644, 0.0000428571, 0.0000300000, 0.0000210000, 0.0000148414, 0.0000100000, 0.0000069388, 0.0000047657, 0.0000032857, 0.0000022737, 0.0000015849, 0.0000011111, 0.0000007408, 0.0000005000, 0.0000003434, 0.0000002333, 0.0000001628, 0.0000001111, 0.0000000740, 0.0000000500, 0.0000000343, 0.0000000233, 0.0000000162, 0.0000000111, 0.0000000074, 0.0000000050, 0.0000000034, 0.0000000023, 0.0000000016, 0.0000000011, 0.0000000007, 0.0000000005, 0.0000000003, 0.0000000002, 0.0000000001}

LP models

Two examples from last week:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

$$\min 170x_1 + 160x_2 + 175x_3 + 180x_4 + 195x_5$$

subject to

$$x_1 \geq 48$$

$$x_1 + x_2 \geq 79$$

$$x_1 + x_2 \geq 65$$

$$x_1 + x_2 + x_3 \geq 87$$

$$x_2 + x_3 \geq 64$$

$$x_3 + x_4 \geq 73$$

$$x_3 + x_4 \geq 82$$

$$x_4 \geq 43$$

$$x_4 + x_5 \geq 52$$

$$x_5 \geq 15$$

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

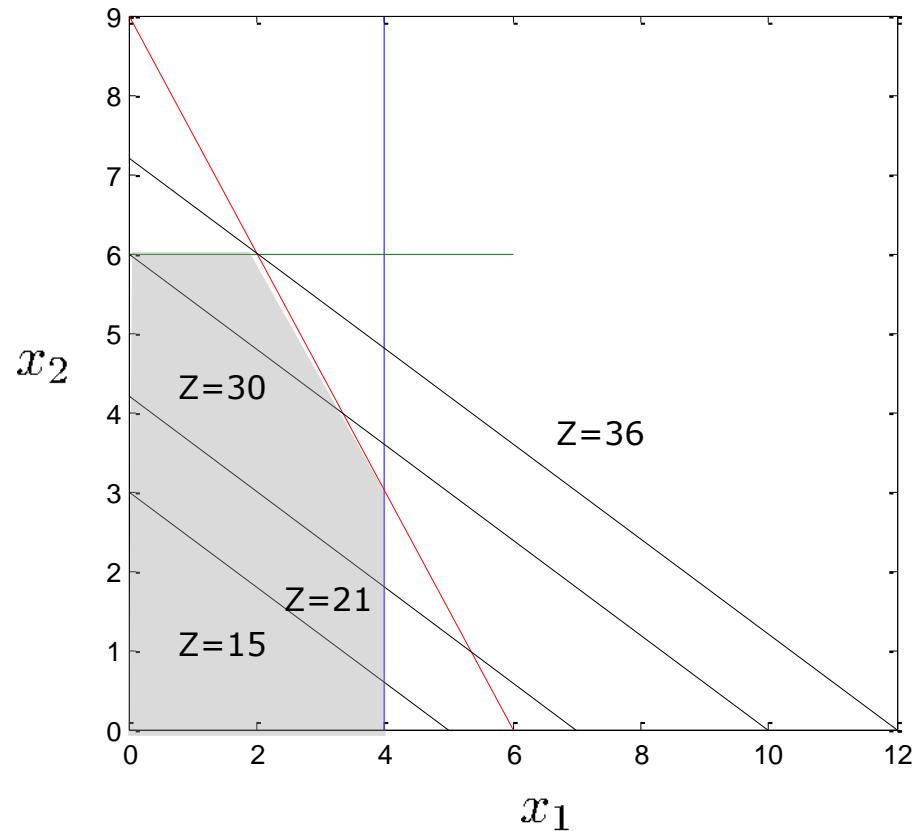
How do we solve the problem on the right?

The simplex algorithm

A method for solving any LP

- **Algorithm:**
A precise description of the operations needed to solve a specific problem.
Similar to a cooking recipe (but more precise).
- Some or all of you probably already knows algorithms for solving simple problems
 - Sorting
 - Searching through sorted data

Wyndor solution



$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

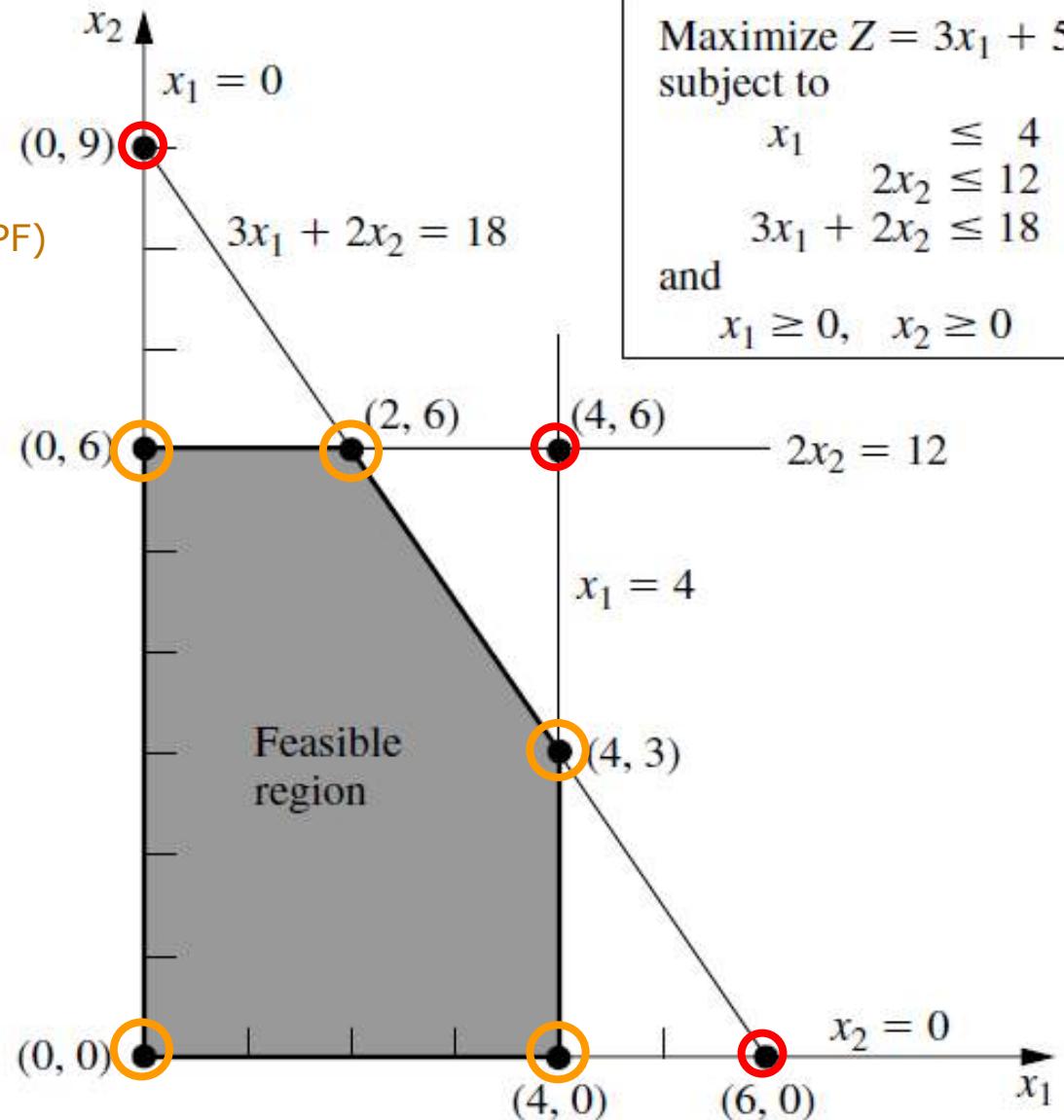
To find the optimal solution it is enough to examine all corner points (intersection between two lines).

Geometry

Corner point feasible solutions (CPF)

Corner point infeasible solutions

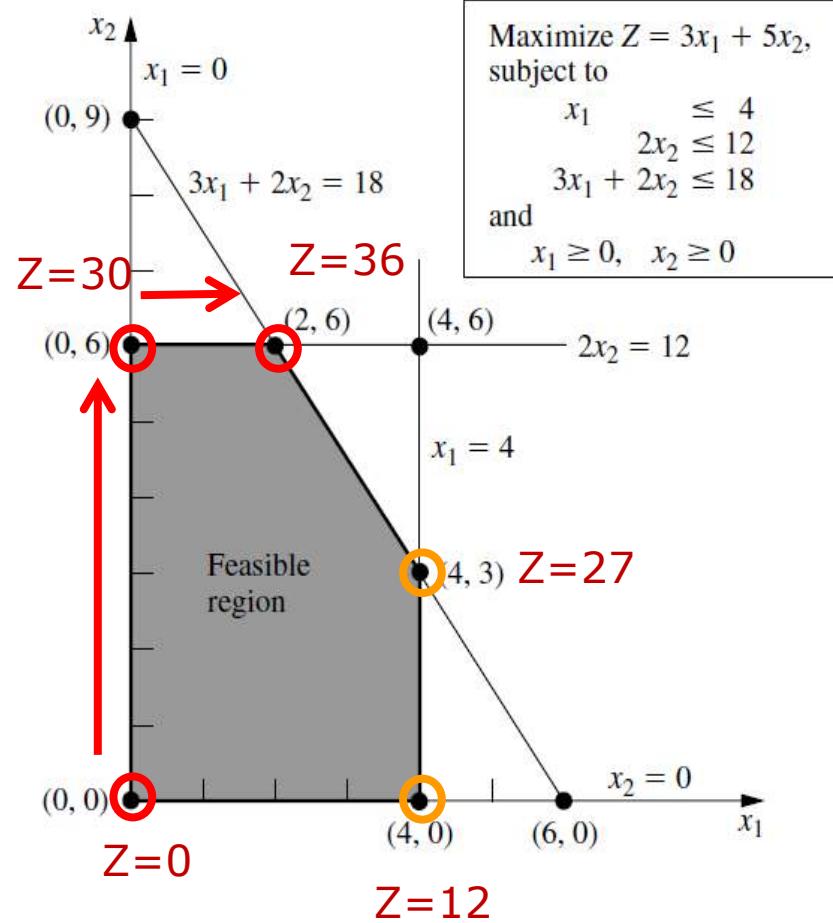
Two cornerpoints are neighbors if they share a line.



Geometry

Idea behind Simplex algorithm:

1. Start in $(x_1, x_2) = (0, 0)$.
2. If there is a neighboring feasible corner point with larger objective then jump to that corner point. Otherwise stop.
3. Jump back to step 2.

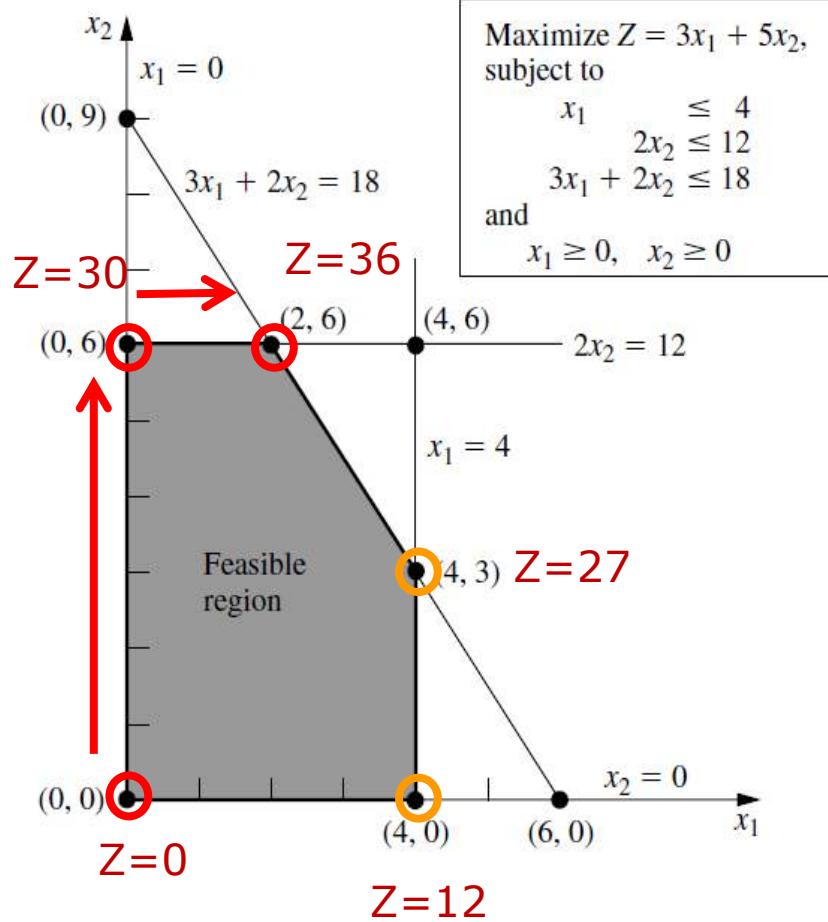


Geometry

Idea behind Simplex algorithm:

1. Start in $(x_1, x_2) = (0, 0)$.
2. If there is a neighboring feasible corner point with larger objective then jump to that corner point. Otherwise stop.
3. Jump back to step 2.

Optimality test: Consider an LP with at least one optimal solution. If a feasible corner point does not have any neighboring solutions that are better with respect to Z then the corner point is optimal.



Augmented form

Wyndor example again:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

We wish to convert " \leq " constraints to " $=$ ". We do this so we can re-use some of the machinery we learned when solving systems of linear equations.

We add "slack variables" for every constraint. For the constraint $x_1 \leq 4$ we add x_3 defined by:

$$x_3 = 4 - x_1$$

We can now rewrite the equality as:

$$x_1 + x_3 = 4 \quad \text{and} \quad x_3 \geq 0$$

LP written in augmented form



Standard form

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

Augmented form

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 + x_3 = 4$$

$$2x_2 + x_4 = 12$$

$$3x_1 + 2x_2 + x_5 = 18$$

and

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

Basic solution

Standard form

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

Augmented form

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 + x_3 = 4$$

$$2x_2 + x_4 = 12$$

$$3x_1 + 2x_2 + x_5 = 18$$

and

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

Augmented solution: original variables + slack variables

$$\text{Eks. } (0,0) \rightarrow (0,0,4,12,18)$$

Basic solution: augmented corner point solution (not necessarily feasible)

basic feasible solution: feasible augmented corner point solution

Basic solution

Augmented form

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 + x_3 = 4$$

$$2x_2 + x_4 = 12$$

$$3x_1 + 2x_2 + x_5 = 18$$

and

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

5 variables, 3 equations.

In general: $5-3=2$ degrees of freedom

In general we can assign any value to two of the variables and then it will be possible to find values for the remaining three variables such that the equations match.

The Simplex algorithm assigns the value 0 to the free variables.

Basic solution

Augmented form

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 + x_3 = 4$$

$$2x_2 + x_4 = 12$$

$$3x_1 + 2x_2 + x_5 = 18$$

and

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

Basic solution

1. Every variable is either denoted a basic or an non-basic variable.
2. Number of basic variables = number of constraints
3. Number of non-basic variables
= (total number of variables) – (number of constraints)
4. Non-basic variables are always assigned the value 0.
5. The value of basic variables is found by removing non-basic variables from the system of equation and solve the remaining system.
6. If all basic variables are ≥ 0 then the solution is feasible.

5 variables, 3 equations.

In general: $5-3=2$ degrees of freedom

In general we can assign any value to two of the variables and then it will be possible to find values for the remaining three variables such that the equations match.

The Simplex algorithm assigns the value 0 to the free variables.

Rewriting the objective function

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 + x_3 = 4$$

$$2x_2 + x_4 = 12$$

$$3x_1 + 2x_2 + x_5 = 18$$

$$\max Z$$

subject to

$$Z - 3x_1 - 5x_2 = 0 \quad (0)$$

$$x_1 + x_3 = 4 \quad (1)$$

$$2x_2 + x_4 = 12 \quad (2)$$

$$3x_1 + 2x_2 + x_5 = 18 \quad (3)$$

and

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$

and

$$x_1, x_2, x_3, x_4, x_5 \geq 0$$



Looks like a system of linear equations...

...but maximization and non-negativity is
“new”

The simplex method – tableau form



TABLE 4.3 Initial system of equations for the Wyndor Glass Co. problem

(a) Algebraic Form	(b) Tabular Form								Right Side	
	Basic Variable	Eq.	Coefficient of:							
			Z	x_1	x_2	x_3	x_4	x_5		
(0) $Z - 3x_1 - 5x_2 = 0$	Z	(0)	1	-3	-5	0	0	0	0	
(1) $x_1 + x_3 = 4$	x_3	(1)	0	1	0	1	0	0	4	
(2) $2x_2 + x_4 = 12$	x_4	(2)	0	0	2	0	1	0	12	
(3) $3x_1 + 2x_2 + x_5 = 18$	x_5	(3)	0	3	2	0	0	1	18	

Initialization: Start by having x_1 and x_2 as non-basic variables. That makes it easy to find the value of the basic variables.

$$x_1 + x_3 = 4 \quad (1)$$

$$2x_2 + x_4 = 12 \quad (2)$$

$$3x_1 + 2x_2 + x_5 = 18 \quad (3)$$

x_1	x_2	x_3	x_4	x_5
0	0	4	12	18

The simplex method



1. Optimality test: no negative numbers in row zero \Rightarrow we are done.
2. Choose incoming basic variable (most negative in row 0).
3. Choose leaving basic variable (min ratio-test).
4. Restore the tableau to proper form with respect to the new basic variable (Gauss operations).
5. Jump back to 1.

The simplex method – 1: optimality test



TABLE 4.3 Initial system of equations for the Wyndor Glass Co. problem

(a) Algebraic Form	(b) Tabular Form								Right Side	
	Basic Variable	Eq.	Coefficient of:							
			Z	x_1	x_2	x_3	x_4	x_5		
(0) $Z - 3x_1 - 5x_2 = 0$	Z	(0)	1	-3	-5	0	0	0	0	
(1) $x_1 + x_3 = 4$	x_3	(1)	0	1	0	1	0	0	4	
(2) $2x_2 + x_4 = 12$	x_4	(2)	0	0	2	0	1	0	12	
(3) $3x_1 + 2x_2 + x_5 = 18$	x_5	(3)	0	3	2	0	0	1	18	

Optimality test:

$$Z - 3x_1 - 5x_2 = 0$$

We can obtain a better solution by increasing either x_1 or x_2 since they both have negative coefficients in row zero.

The simplex method

Step 2: choosing incoming basic variable

TABLE 4.3 Initial system of equations for the Wyndor Glass Co. problem

(a) Algebraic Form	(b) Tabular Form								Right Side	
	Basic Variable	Eq.	Coefficient of:							
			Z	x_1	x_2	x_3	x_4	x_5		
(0) $Z - 3x_1 - 5x_2 = 0$	Z	(0)	1	-3	-5	0	0	0	0	
(1) $x_1 + x_3 = 4$	x_3	(1)	0	1	0	1	0	0	4	
(2) $2x_2 + x_4 = 12$	x_4	(2)	0	0	2	0	1	0	12	
(3) $3x_1 + 2x_2 + x_5 = 18$	x_5	(3)	0	3	2	0	0	1	18	

choosing incoming basic variable:

- which of the two non-basic variables should we increase?
- We chose x_2 since $5 > 3$. If we can increase x_2 by changing the values of the current basic variables then Z is going to increase by 5 per unit we increase x_2 (because all the current basic variables have coefficient 0 in row 0).
- x_2 is the entering basic variable.

Basic Variable	Eq.	Coefficient of:						Right Side	Ratio
		Z	x_1	x_2	x_3	x_4	x_5		
Z	(0)	1	-3	-5	0	0	0	0	
x_3	(1)	0	1	0	1	0	0	4	
x_4	(2)	0	0	2	0	1	0	$12 \rightarrow \frac{12}{2} = 6 \leftarrow \text{minimum}$	
x_5	(3)	0	3	2	0	0	1	$18 \rightarrow \frac{18}{2} = 9$	

- We have found entering variable (pivot column). Mark column with a box.
- We keep $x_1 = 0$ and attempt to increase x_2 .

$$x_1 + x_3 = 4 \quad (1) \qquad x_3 = 4$$

$$2x_2 + x_4 = 12 \quad (2) \qquad x_4 = 12 - 2x_2$$

$$3x_1 + 2x_2 + x_5 = 18 \quad (3) \qquad x_5 = 18 - 2x_2$$

- We can increase x_2 to 6. At that point $x_4 = 0$ and if we increase x_2 anymore then x_4 becomes negative (and all variables are required to be ≥ 0).

- x_4 leaves the basis (leaving variable).

- ”Minimum ratio test”

x_1	x_2	x_3	x_4	x_5
0	6	?	0	?

TABLE 4.5 Simplex tableaux for the Wyndor Glass Co. problem after the first pivot row is divided by the first pivot number

Iteration	Basic Variable	Eq.	Z	Coefficient of:					Right Side
				x_1	x_2	x_3	x_4	x_5	
0	Z	(0)	1	-3	-5	0	0	0	0
	x_3	(1)	0	1	0	1	0	0	4
	x_4	(2)	0	0	2	0	1	0	12
	x_5	(3)	0	3	2	0	0	1	18

We have decided the pivot row. Put a box around this.

Element in both boxes is the *pivot element*.

TABLE 4.6 First two simplex tableaux for the Wyndor Glass Co. problem

Iteration	Basic Variable	Eq.	Z	Coefficient of:					Right Side
				x_1	x_2	x_3	x_4	x_5	
0	Z	(0)	1	-3	-5	0	0	0	0
	x_3	(1)	0	1	0	1	0	0	4
	x_4	(2)	0	0	2	0	1	0	12
	x_5	(3)	0	3	2	0	0	1	18
1	Z	(0)	1						30
	x_3	(1)	0						4
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_5	(3)	0						6

$r_2^{\text{new}} = \frac{1}{2}r_2^{\text{old}}$

We use gaussian elimination to create a new tableau. In this tableau

- The columns of the basic variables contains one "1", the rest of the elements are zero
- The "1" in a column for a basic variable is in the row for which the variable is basic.

If we fail to do so: Optimality test and minimum ratio test in the next iteration is not going to work.

TABLE 4.6 First two simplex tableaux for the Wyndor Glass Co. problem

Iteration	Basic Variable	Eq.	Z	Coefficient of:					Right Side
				x_1	x_2	x_3	x_4	x_5	
0	Z	(0)	1	-3	-5	0	0	0	0
	x_3	(1)	0	1	0	1	0	0	4
	x_4	(2)	0	0	2	0	1	0	12
	x_5	(3)	0	3	2	0	0	1	18
1	Z	(0)	1	-3	0	0	$\frac{5}{2}$	0	30
	x_3	(1)	0	1	0	1	0	0	4
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_5	(3)	0	3	0	0	-1	1	6

We use gaussian elimination to create a new tableau. In this tableau

- The columns of the basic variables contains one "1", the rest of the elements are zero
- The "1" in a column for a basic variable is in the row for which the variable is basic.

If we fail to do so: Optimality test and minimum ratio test in the next iteration is not going to work.

TABLE 4.7 Steps 1 and 2 of iteration 2 for the Wyndor Glass Co. problem

Iteration	Basic Variable	Eq.	Coefficient of:					Right Side	Ratio
			Z	x_1	x_2	x_3	x_4		
1	Z	(0)	1	-3	0	0	$\frac{5}{2}$	0	30
	x_3	(1)	0	1	0	1	0	0	4
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_5	(3)	0	3	0	0	-1	1	$\frac{6}{3} = 2 \leftarrow \text{minimum}$

Optimality test:

$$Z - 3x_1 + \frac{5}{2}x_4 = 30$$

We can improve Z by increasing x_1 (since the coefficient in front of x_1 is negative).

We have to continue.

TABLE 4.8 Complete set of simplex tableaux for the Wyndor Glass Co. problem



Iteration	Basic Variable	Eq.	Coefficient of:					Right Side
			Z	x_1	x_2	x_3	x_4	
0	Z	(0)	1	-3	-5	0	0	0
	x_3	(1)	0	1	0	1	0	4
	x_4	(2)	0	0	2	0	1	12
	x_5	(3)	0	3	2	0	0	18
1	Z	(0)	1	-3	0	0	$\frac{5}{2}$	0
	x_3	(1)	0	1	0	1	0	4
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	6
	x_5	(3)	0	3	0	0	-1	6
2	Z	(0)	1	0	0	0	$\frac{3}{2}$	1
	x_3	(1)	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0
	x_1	(3)	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$

Done!

Assumptions necessary for simplex algorithm to work:

- LP should be on standard form:

$$\max Z = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

subject to

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \leq b_2$$

⋮

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m$$

$$x_1, x_2, \dots, x_n \geq 0$$

- **AND** Right hand side values should be non-negative, that is

$$b_i \geq 0, \forall i = 1, \dots, m$$

Simplex algorithm



- On the DTU inside 42101 group there is a video with another example (under file sharing). The video is in Danish.

Simplex algorithm – special cases



- What if there are two equally good variables when we choose incoming variable in step 2?
- This could for example happen if our objective function was

$$\max Z = 3x_1 + 3x_2$$

- No problem: We just choose one of the variables as incoming. We are going to reach the optimal solution anyway.

Simplex algorithm – special cases

- What if no variable can leave the basis in minimum ratio test in step 3?
- In this case solution is unbounded!

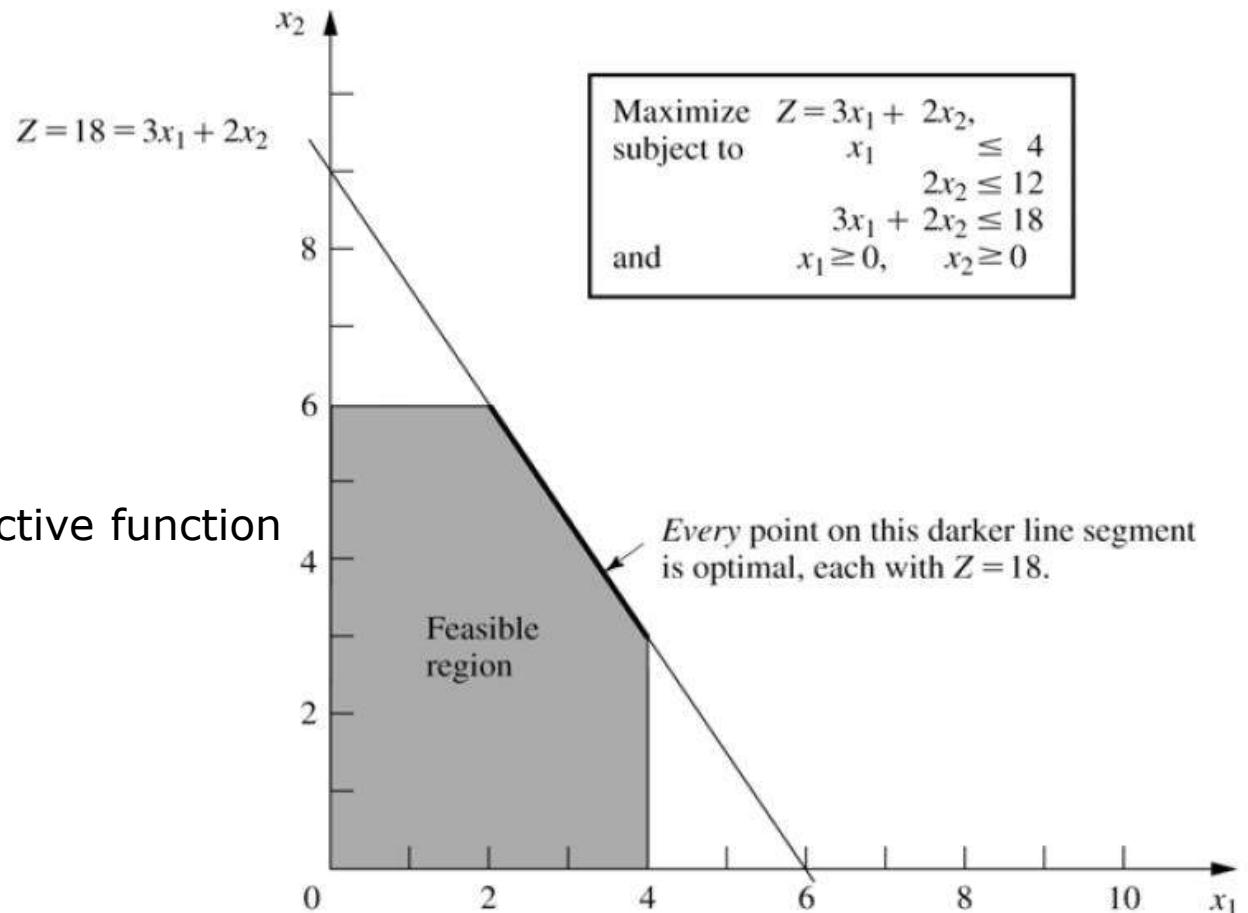
TABLE 4.9 Initial simplex tableau for the Wyndor Glass Co. problem without the last two functional constraints

Basic Variable	Eq.	Coefficient of:			Right Side	Ratio
		Z	x_1	x_2	x_3	
Z	(0)	1	-3	-5	0	0
x_3	(1)	0	1	0	1	4 None

With $x_1 = 0$ and x_2 increasing,
 $x_3 = 4 - 1x_1 - 0x_2 = 4 > 0$.

Simplex algorithm – special cases

- What if there are several optimal solutions?
- If we just want to know one optimal solution we can just do as we "always have done".



Wyndor with a new objective function

Simplex algorithm – special cases

- If we want to know all corner point feasible optimal solutions?
- When simplex ends: Check if any non-basis variable has coefficient 0 in row 0.
- If this is the case then there are multiple optimal CPF solutions.
- Extra pivots give the other optimal solutions.

If we want to know all corner point feasible optimal solutions?

TABLE 4.10 Complete set of simplex tableaux to obtain all optimal BF solutions for the Wyndor Glass Co. problem with $c_2 = 2$

Iteration	Basic Variable	Eq.	Z	Coefficient of:					Right Side	Solution Optimal?
				x_1	x_2	x_3	x_4	x_5		
0	Z	(0)	1	-3	-2	0	0	0	0	No
	x_3	(1)	0	1	0	1	0	0	4	
	x_4	(2)	0	0	2	0	1	0	12	
	x_5	(3)	0	3	2	0	0	1	18	
1	Z	(0)	1	0	-2	3	0	0	12	No
	x_1	(1)	0	1	0	1	0	0	4	
	x_4	(2)	0	0	2	0	1	0	12	
	x_5	(3)	0	0	2	-3	0	1	6	
2	Z	(0)	1	0	0	0	0	1	18	Yes
	x_1	(1)	0	1	0	1	0	0	4	
	x_4	(2)	0	0	0	3	1	-1	6	
	x_2	(3)	0	0	1	$-\frac{3}{2}$	0	$\frac{1}{2}$	3	
Extra	Z	(0)	1	0	0	0	0	1	18	Yes
	x_1	(1)	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	2	
	x_3	(2)	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	2	
	x_2	(3)	0	0	1	0	$\frac{1}{2}$	0	6	

We are done, but x_3 is not in the basis and has coefficient 0 in row 0. This means that there is at least one more optimal CPF solution.

Another non-basic variable with coefficient 0 in row 0.

However, if we pivot this variable in, we would go back to the previous tableau. We have found all optimal CPF solutions

How to handle LPs that are not on the standard form? *Minimization*

- Transform minimization to maximization:

$$\min x \Leftrightarrow \max -x$$

- or

$$\min 5x_1 + 6x_2 \Leftrightarrow \max -5x_1 - 6x_2$$

- In general:

$$\min Z = c_1x_1 + c_2x_2 + \dots + c_nx_n \Leftrightarrow \max -Z = -c_1x_1 - c_2x_2 - \dots - c_nx_n$$

How to handle LPs that are not on the standard form? **Negative variables**

- Assume we have a variable x_j that can take negative values.
- If we know a lower bound L_j for the negative variable then we can define a new variable and substitute:

$$x'_j = x_j - L_j \quad \text{og} \quad x'_j \geq 0$$

- Example. Let's assume

$$x_1 \geq -10$$

in the Wyndor eksemplet. We then introduce a new variable x'_1 and substitute:

$$x'_1 = x_1 + 10, \quad x'_1 \geq 0 \quad \Leftrightarrow \quad x_1 = x'_1 - 10, \quad x'_1 \geq 0$$

Let's also change the objective to

$$\max Z = -3x_1 + 5x_2$$

to make the problem more interesting

$$\begin{aligned} Z &= -3x_1 + 5x_2 \\ x_1 &\leq 4 \\ 2x_2 &\leq 12 \\ 3x_1 + 2x_2 &\leq 18 \\ x_1 &\geq -10, \quad x_2 \geq 0 \end{aligned}$$

→

$$\begin{aligned} Z &= -3(x'_1 - 10) + 5x_2 \\ x'_1 - 10 &\leq 4 \\ 2x_2 &\leq 12 \\ 3(x'_1 - 10) + 2x_2 &\leq 18 \\ x'_1 - 10 &\geq -10, \quad x_2 \geq 0 \end{aligned}$$

→

$$\begin{aligned} Z &= 30 - 3x'_1 + 5x_2 \\ x'_1 &\leq 14 \\ 2x_2 &\leq 12 \\ 3x'_1 + 2x_2 &\leq 48 \\ x'_1 &\geq 0, \quad x_2 \geq 0 \end{aligned}$$

How to handle LPs that are not on the standard form? *Negative variables*

b.v.		eq.		Z		x1'	x2	x3	x4	x5		RHS
Z		0		1		3.00	-5.00	0.00	0.00	0.00		30.00
x3		1		0		1.00	0.00	1.00	0.00	0.00		14.00
x4		2		0		0.00	2.00	0.00	1.00	0.00		12.00
x5		3		0		3.00	2.00	0.00	0.00	1.00		48.00

b.v.		eq.		Z		x1'	x2	x3	x4	x5		RHS
Z		0		1		3.00	0.00	0.00	2.50	0.00		60.00
x3		1		0		1.00	0.00	1.00	0.00	0.00		14.00
x2		2		0		0.00	1.00	0.00	0.50	0.00		6.00
x5		3		0		3.00	0.00	0.00	-1.00	1.00		36.00

Solution:

$$x'_1 = 0, x_2 = 6, Z = 60 \Leftrightarrow x_1 = x'_1 - 10 = -10, x_2 = 6, Z = 60$$

$$\begin{aligned} Z &= -3x_1 + 5x_2 \\ x_1 &\leq 4 \\ 2x_2 &\leq 12 \\ 3x_1 + 2x_2 &\leq 18 \\ x_1 &\geq -10, \quad x_2 \geq 0 \end{aligned}$$



$$\begin{aligned} Z &= -3(x'_1 - 10) + 5x_2 \\ x'_1 - 10 &\leq 4 \\ 2x_2 &\leq 12 \\ 3(x'_1 - 10) + 2x_2 &\leq 18 \\ x'_1 - 10 &\geq -10, \quad x_2 \geq 0 \end{aligned}$$



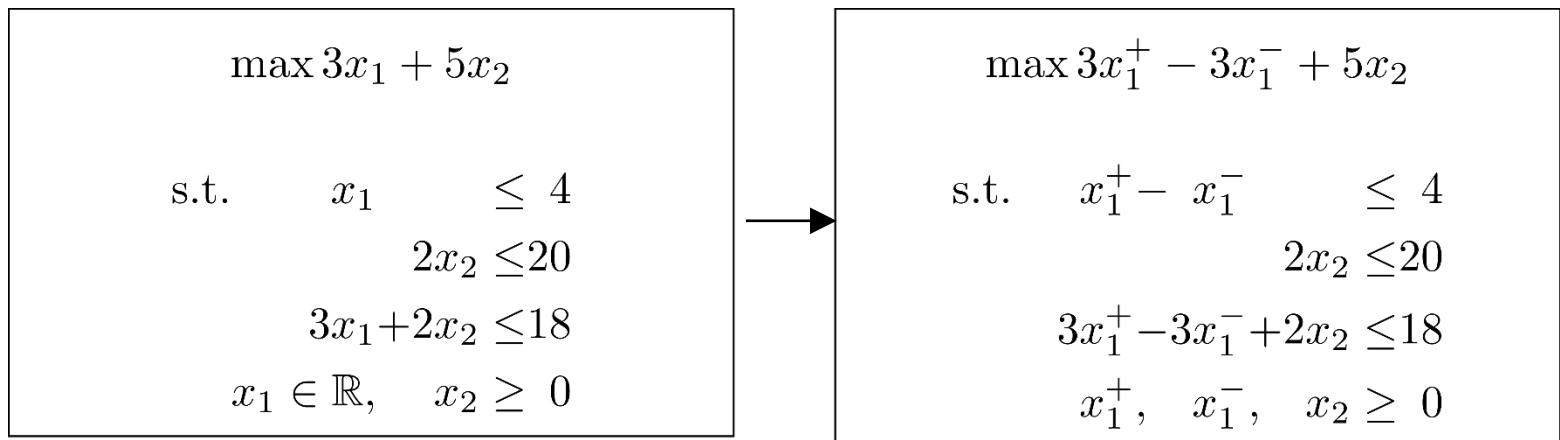
$$\begin{aligned} Z &= 30 - 3x'_1 + 5x_2 \\ x'_1 &\leq 14 \\ 2x_2 &\leq 12 \\ 3x'_1 + 2x_2 &\leq 48 \\ x'_1 &\geq 0, \quad x_2 \geq 0 \end{aligned}$$

How to handle LPs that are not on the standard form? **Negative variables**

- If there is no lower bounds on the negative variable(s) then we need to represent each negative variable by two variables in the LP:
- assume that $x_j \in \mathbb{R}$. Then we rewrite

$$x_j = x_j^+ - x_j^-, \quad x_j^+ \geq 0, \quad x_j^- \geq 0$$

- x_j^+ is the positive part of x_j . x_j^- is the negative part.
- Example:



$$\max 3x_1 + 5x_2$$

s.t.

x_1	≤ 4
$2x_2$	≤ 20
$3x_1 + 2x_2 \leq 18$	
$x_1 \in \mathbb{R}, \quad x_2 \geq 0$	

$$\max 3x_1^+ - 3x_1^- + 5x_2$$

s.t.

$x_1^+ - x_1^-$	≤ 4
$2x_2$	≤ 20
$3x_1^+ - 3x_1^- + 2x_2 \leq 18$	
$x_1^+, \quad x_1^-, \quad x_2 \geq 0$	

b.v.	eq.	z	x1+	x1-	x2	x3	x4	x5	RHS
Z	0	1	-3.00	3.00	-5.00	0.00	0.00	0.00	0.00
x3	1	0	1.00	-1.00	0.00	1.00	0.00	0.00	4.00
x4	2	0	0.00	0.00	2.00	0.00	1.00	0.00	20.00
x5	3	0	3.00	-3.00	2.00	0.00	0.00	1.00	18.00

b.v.	eq.	z	x1+	x1-	x2	x3	x4	x5	RHS
Z	0	1	4.50	-4.50	0.00	0.00	0.00	2.50	45.00
x3	1	0	1.00	-1.00	0.00	1.00	0.00	0.00	4.00
x4	2	0	-3.00	3.00	0.00	0.00	1.00	-1.00	2.00
x2	3	0	1.50	-1.50	1.00	0.00	0.00	0.50	9.00

b.v.	eq.	z	x1+	x1-	x2	x3	x4	x5	RHS
Z	0	1	0.00	0.00	0.00	0.00	1.50	1.00	48.00
x3	1	0	0.00	0.00	0.00	1.00	0.33	-0.33	4.67
x1-	2	0	-1.00	1.00	0.00	0.00	0.33	-0.33	0.67
x2	3	0	0.00	0.00	1.00	0.00	0.50	0.00	10.00

$$\max 3x_1 + 5x_2$$

s.t.

x_1	≤ 4
$2x_2$	≤ 20
$3x_1 + 2x_2$	≤ 18
$x_1 \in \mathbb{R},$	$x_2 \geq 0$

$$\max 3x_1^+ - 3x_1^- + 5x_2$$

s.t.

$x_1^+ - x_1^-$	≤ 4
$2x_2$	≤ 20
$3x_1^+ - 3x_1^- + 2x_2$	≤ 18
$x_1^+, \quad x_1^-, \quad x_2$	≥ 0

b.v.	eq.	Z	x1+	x1-	x2	x3	x4	x5	RHS
Z	0	1	-3.00	3.00	-5.00	0.00	0.00	0.00	0.00
x3	1	0	1.00	-1.00	0.00	1.00	0.00	0.00	4.00
x4	2	0	0.00	0.00	2.00	0.00	1.00	0.00	20.00
x5	3	0	3.00	-3.00	2.00	0.00	0.00	1.00	18.00

b.v.	eq.	Z	$x_1^- = \frac{2}{3}, x_2 = 10 \rightarrow x_1 = -\frac{2}{3}, x_2 = 10$						RHS
Z	0	1							0 45.00
x3	1	0							0 4.00
x4	2	0	-3.00	3.00	0.00	0.00	1.00	-1.00	2.00
x2	3	0	1.50	-1.50	1.00	0.00	0.00	0.50	9.00

b.v.	eq.	Z	x1+	x1-	x2	x3	x4	x5	RHS
Z	0	1	0.00	0.00	0.00	0.00	1.50	1.00	48.00
x3	1	0	0.00	0.00	0.00	1.00	0.33	-0.33	4.67
x1-	2	0	-1.00	1.00	0.00	0.00	0.33	-0.33	0.67
x2	3	0	0.00	0.00	1.00	0.00	0.50	0.00	10.00

How to handle LPs that are not on the standard form?

- How to handle " $=$ " constraints?
- How to handle " \geq " constraints?
- How to handle negative right hand sides of constraints?

More on that next week!

Simplex algorithm

- George Dantzig (1914-2005) invented the Simplex algorithm in 1947.
- Invented several other OR techniques.
- One of the most important algorithms discovered in the last century?
"The Best of the 20th Century: Editors Name Top 10 Algorithms" (SIAM, 2004)
- Important in many follow up OR courses
 E.g.
 - 42114 – integer programming
 - 42115 – network optimization
 - 42136 - Large Scale Optimization using Decomposition
 - 42116 - Implementing OR Solution Methods



Professor George B. Dantzig på Lundtoftesletten i går (t.v.) sammen med sin danske vært, lektor Olli G. B. Madsen fra højskolens institut for matematisk statistik og operationsanalyse.

Han laver indviklet planlægnings-værktøj

En lille, varm mand med en stor, kold hjerne.

Sådan karakteriserede en af de studerende ved Danmarks tekniske Højskole i går dagens gæsteforeleser, professor George B. Dantzig fra Stanford University, USA.

En anden erklærede efter forelæsningen i det fyldte auditorium 11/308, at Dantzig formentlig er den mest overbeviste, personlige protest, der findes, imod den udbredte, filosofiske tese om, at mennesket i virkeligheden slet ikke

er indrettet til rationel forudsigelse.

Professor Dantzig er ikke nogen helt almindelig spåmand. Han har arbejdet med virkelighedsmodeller siden 1940 og bla. 'opfundet' den såkaldte simplex-metode til løsning af lineære programmerings-problemer.

Det er et planlægningsværktøj til bearbejdning af information på en sådan måde, at resultaterne kan trækkes ud over nu-punktet og danne grundlag for beslutninger om, hvordan fremtiden skal se ud.

Professorens budskab til sine

yngre, danske kolleger inden for operationsanalyse og virksomhedsledelse m.v. er, at mennesket ikke længer kan nøjes med at tilrettelægge dagen og vejen og må åske tænke lidt på morgendagen.

Det omgivende samfund er nu så kompliceret, at der må komplicerede planlægningsværktøjer til, hvis vi skal lede udviklingen netop i den retning, vi ønsker det.

Det er de færreste, der kan klare sig gennem tilværelsen ved hjælp af følelser og fornemmelser.

cauchi.

Politikken, 1976

Abstract and concrete models

Concrete Wyndor model

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

- 2 products (glass doors, windows)
- Requires time on 3 plants

	1	2	Capacity (hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

- Profit is per unit and should be multiplied by 1000\$
- We want to plan the production so that our profit is maximized

Abstract and concrete models

Abstract Wyndor model

- We may want to extend the model with more products and more plants
- Or we may want to use the model for another company
- It would be nice to write the model in a more generic way!

- 2 products (glass doors, windows)
- Requires time on 3 plants

	1	2	Capacity (hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

Abstract and concrete models

Abstract Wyndor model

It would be nice to write the model in a more generic way!

- m : number of plants
- n : number of products
- b_i : hours available on plant i per week ($i = 1, \dots, m$)
- p_j : profit per batch of product j ($j = 1, \dots, n$)
- a_{ij} : how many hours we need on plant i when producing one batch of product j

- 2 products (glass doors, windows)
- Requires time on 3 plants

	1	2	Capacity (hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

Abstract and concrete models

Abstract Wyndor model

It would be nice to write the model in a more generic way!

- m : number of plants
- n : number of products
- b_i : hours available on plant i per week ($i = 1, \dots, m$)
- p_j : profit per batch of product j ($j = 1, \dots, n$)
- a_{ij} : how many hours we need on plant i when producing one batch of product j
- x_j : Our decision variables. Indicates how many batches of product j we should make per week.
- The abstract model:

$$\max \sum_{j=1}^n p_j x_j$$

subject to

$$\begin{aligned} \sum_{j=1}^n a_{ij} x_j &\leq b_i \quad \forall i = 1, \dots, m \\ x_j &\geq 0 \quad \forall j = 1, \dots, n \end{aligned}$$

Abstract Wyndor model in Julia (1)

```
using JuMP, GLPK

m = 3          # number of plants
n = 2          # number of products
b = [ 4 12 18] # hours available on plant i per week
p = [ 3 5 ]    # profit per batch of product
a = [ 1 0;
      0 2;
      3 2]

model = Model(with_optimizer(GLPK.Optimizer))
# x[j]: how many batches of product j should we make per week
@variable(model, x[j=1:n] >= 0 )

@objective(model, Max, sum(p[j]*x[j] for j=1:n) )
@constraint(model, [i=1:m], sum(a[i,j]*x[j] for j=1:n) <= b[i] )

print(model)

optimize!(model)

println("Objective value: ", JuMP.objective_value(model))
println("x = ", JuMP.value.(x))
```

Abstract Wyndor model in Julia (2)



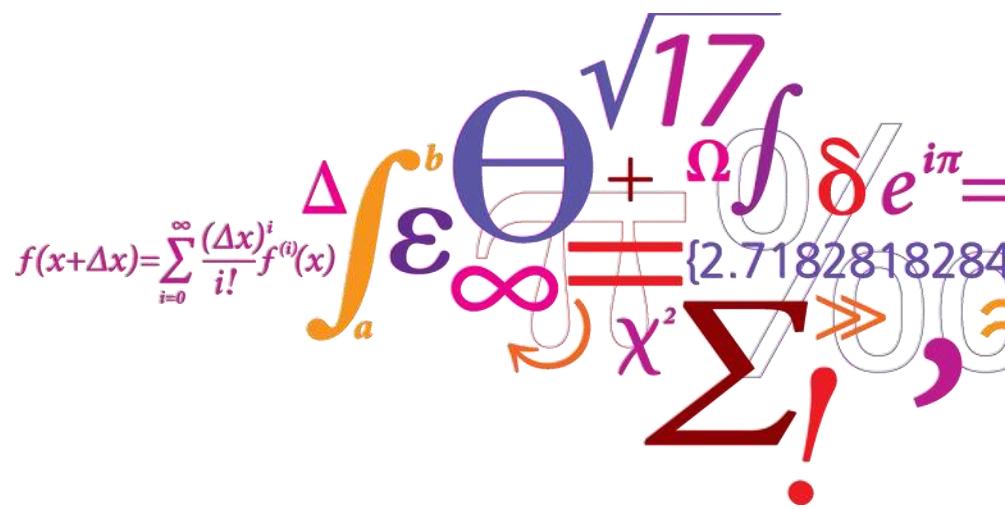
```
using JuMP, GLPK
include("wyndor-data2.jl")  
  
model = Model(with_optimizer(GLPK.Optimizer))  
# x[j]: how many batches of product j should we make per week  
@variable(model, x[j=1:n] >= 0 )  
  
@objective(model, Max, sum(p[j]*x[j] for j=1:n) )  
@constraint(model, [i=1:m], sum(a[i,j]*x[j] for j=1:n) <= b[i] )  
  
print(model)  
  
optimize!(model)  
  
println("Objective value: ", JuMP.objective_value(model))  
println("x = ", JuMP.value.(x))
```



That's it!
Now let's work on the exercises

Two-phase method

Basic graph theory

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$


Simplex algorithm from last week needs:

- LP should be on standard form:

$$\max Z = c_1x_1 + c_2x_2 + \dots + c_nx_n$$

subject to

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \leq b_2$$

⋮

$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m$$

$$x_1, x_2, \dots, x_n \geq 0$$

- **AND** Right hand side values should be non-negative, that is

$$b_i \geq 0, \forall i = 1, \dots, m$$

LP problems that are not on the standard form

Last week we saw how to handle:

- Minimization instead of maximization.
- Variables that are allowed to be negative.

Today we will see how to handle:

- " $=$ " constraints.
- " \geq " constraints.
- Negative right hand sides of constraints.

Handle “=” constraints



- ”=” and ” \geq ” constraints makes it harder to find a feasible start solution for the simplex algorithm
- You can no longer set all the original variables equal to 0.
- Example:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

VS.

and

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

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 = 18$$



and

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

Handle “=”constraints

- For “=”constraints we add an *artificial* variable that measures how much the constraint is violated.
- It is only when all artificial variables are 0 that the solution is feasible to the original problem.

Original form:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 = 18$$

and

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

Augmented form:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 + x_3 = 4$$

$$2x_2 + 1x_4 = 12$$

$$3x_1 + 2x_2 + \bar{x}_5 = 18$$

and

$$x_1, x_2, x_3, x_4, \bar{x}_5 \geq 0$$



- the artificial variabel \bar{x}_5 has to be zero in the final solution otherwise $3x_1 + 2x_2 = 18$ is not satisfied.

Handle “=” and “ \geq ” constraints

- In case of ” \geq ” constraint we introduce both an artificial variable \bar{x}_5 and a *surplus* variable x_6

Original form:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \geq 18$$

and

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

Augmented form:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 + x_3 = 4$$

$$2x_2 + 1x_4 = 12$$

$$3x_1 + 2x_2 + \bar{x}_5 - x_6 = 18$$

and

$$x_1, x_2, x_3, x_4, \bar{x}_5, x_6 \geq 0$$

- The artificial variable \bar{x}_5 must be zero in the final solution. If not, the solution is not feasible.
- The surplus variable can be greater than zero. That just mean that the LHS is larger than the RHS.

Handle “=” and “ \geq ” constraints

- Radiation therapy example

$$\min Z = 0.4x_1 + 0.5x_2$$

subject to

$$0.3x_1 + 0.1x_2 \leq 2.7$$

$$0.5x_1 + 0.5x_2 = 6$$

$$0.6x_1 + 0.4x_2 \geq 6$$

$$x_1, x_2 \geq 0$$

- Slack x_3 , surplus x_5 , artificial \bar{x}_4 and \bar{x}_6

$$\min Z = 0.4x_1 + 0.5x_2$$

subject to

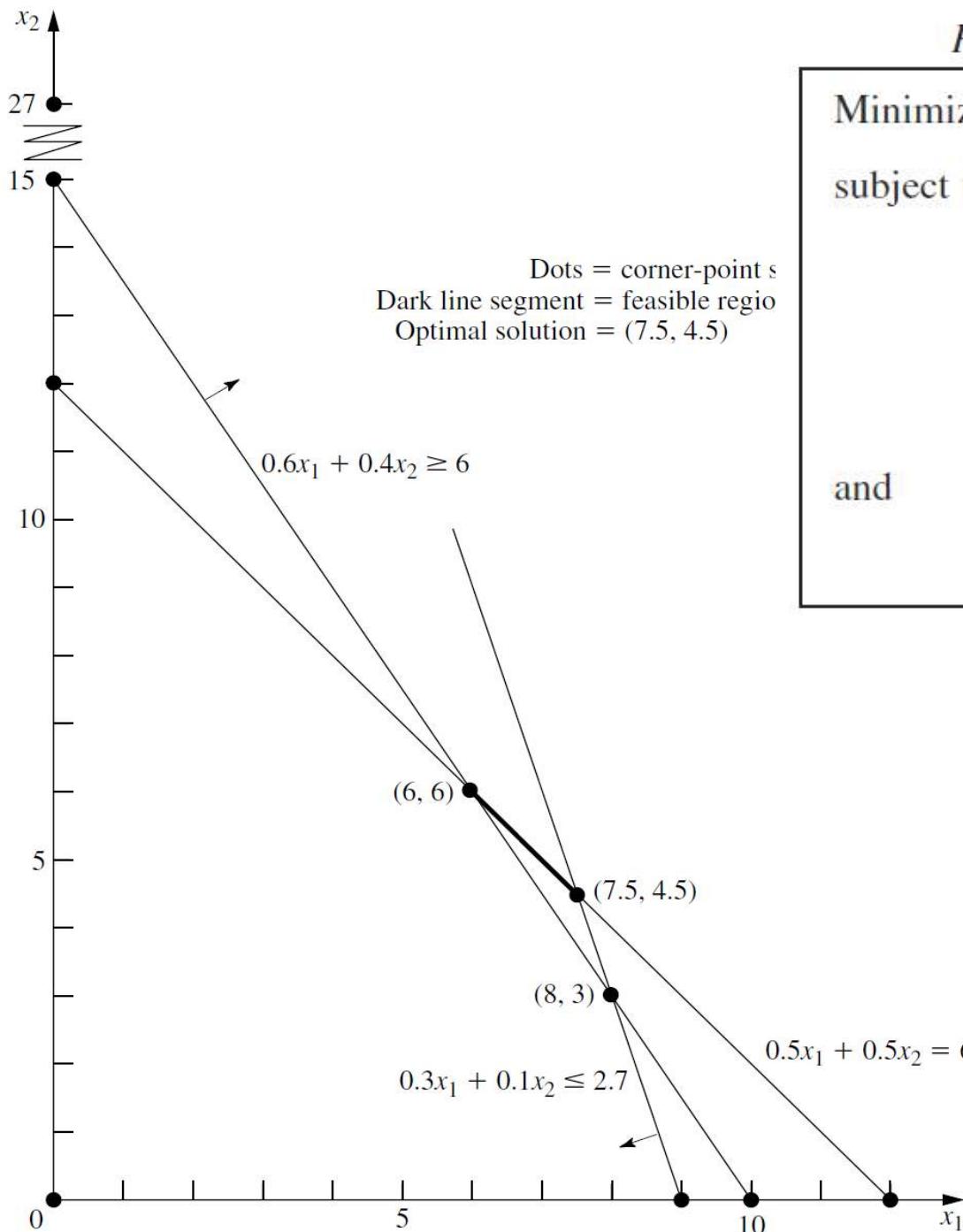
$$0.3x_1 + 0.1x_2 + x_3 = 2.7$$

$$0.5x_1 + 0.5x_2 + \bar{x}_4 = 6$$

$$0.6x_1 + 0.4x_2 - x_5 + \bar{x}_6 = 6$$

$$x_1, x_2, x_3, \bar{x}_4, x_5, \bar{x}_6 \geq 0$$

Radiation Therapy Example



Minimize $Z = 0.4x_1 + 0.5x_2,$
 subject to

$$0.3x_1 + 0.1x_2 \leq 2.7$$

$$0.5x_1 + 0.5x_2 = 6$$

$$0.6x_1 + 0.4x_2 \geq 6$$

and

$$x_1 \geq 0, \quad x_2 \geq 0.$$

$$\min Z = 0.4x_1 + 0.5x_2$$

subject to

$$0.3x_1 + 0.1x_2 + x_3 = 2.7$$

$$0.5x_1 + 0.5x_2 + \bar{x}_4 = 6$$

$$0.6x_1 + 0.4x_2 - x_5 + \bar{x}_6 = 6$$

$$x_1, x_2, x_3, \bar{x}_4, x_5, \bar{x}_6 \geq 0$$

Two-phase method:

Phase 1:	Minimize	$Z = \bar{x}_4 + \bar{x}_6$	(until $\bar{x}_4 = 0, \bar{x}_6 = 0$).
Phase 2:	Minimize	$Z = 0.4x_1 + 0.5x_2$	(with $\bar{x}_4 = 0, \bar{x}_6 = 0$).

Phase 1 Problem (Radiation Therapy Example):

Minimize $Z = \bar{x}_4 + \bar{x}_6,$

subject to

$$0.3x_1 + 0.1x_2 + x_3 = 2.7$$

$$0.5x_1 + 0.5x_2 + \bar{x}_4 = 6$$

$$0.6x_1 + 0.4x_2 - x_5 + \bar{x}_6 = 6$$

and

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad \bar{x}_4 \geq 0, \quad x_5 \geq 0, \quad \bar{x}_6 \geq 0.$$

- Before we can start: transform min to max

$$\max -Z = -\bar{x}_4 - \bar{x}_6$$

$$-Z + \bar{x}_4 + \bar{x}_6 = 0$$

Phase 1 Problem (Radiation Therapy Example):

Minimize $Z = \bar{x}_4 + \bar{x}_6,$

subject to

$$0.3x_1 + 0.1x_2 + x_3 = 2.7$$

$$0.5x_1 + 0.5x_2 + \bar{x}_4 = 6$$

$$0.6x_1 + 0.4x_2 - x_5 + \bar{x}_6 = 6$$

and

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad \bar{x}_4 \geq 0, \quad x_5 \geq 0, \quad \bar{x}_6 \geq 0.$$

- Before we can start: transform min to max

$$\max -Z = -\bar{x}_4 - \bar{x}_6$$

$$-Z + \bar{x}_4 + \bar{x}_6 = 0$$

b.v.	EQ	Z	x_1	x_2	x_3	\bar{x}_4	x_5	\bar{x}_6	RHS
Z	0	-1	0	0	0	1	0	1	0
x_3	1	0	0.3	0.1	1	0	0	0	2.7
x_4	2	0	0.5	0.5	0	1	0	0	6
x_6	3	0	0.6	0.4	0	0	-1	1	6

Phase 1 Problem (Radiation Therapy Example):

Minimize $Z = \bar{x}_4 + \bar{x}_6,$

subject to

$$\begin{aligned} 0.3x_1 + 0.1x_2 + x_3 &= 2.7 \\ 0.5x_1 + 0.5x_2 + \bar{x}_4 &= 6 \\ 0.6x_1 + 0.4x_2 - x_5 + \bar{x}_6 &= 6 \end{aligned}$$

and

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad \bar{x}_4 \geq 0, \quad x_5 \geq 0, \quad \bar{x}_6 \geq 0.$$

- Before we can start: transform min to max

$$\max -Z = -\bar{x}_4 - \bar{x}_6$$

$$-Z + \bar{x}_4 + \bar{x}_6 = 0$$

b.v.	EQ	Z	x_1	x_2	x_3	\bar{x}_4	x_5	\bar{x}_6	RHS
Z	0	-1	0	0	0	1	0	1	0
x_3	1	0	0.3	0.1	1	0	0	0	2.7
x_4	2	0	0.5	0.5	0	1	0	0	6
x_6	3	0	0.6	0.4	0	0	-1	1	6

We want x_3, \bar{x}_4 and \bar{x}_6 as basisvariables. Tableau is not on proper form with respect to that choice of basic solution. We need to "repair" row 0.

Phase 1 Problem (Radiation Therapy Example):

Minimize $Z = \bar{x}_4 + \bar{x}_6,$

subject to

$$\begin{aligned} 0.3x_1 + 0.1x_2 + x_3 &= 2.7 \\ 0.5x_1 + 0.5x_2 + \bar{x}_4 &= 6 \\ 0.6x_1 + 0.4x_2 - x_5 + \bar{x}_6 &= 6 \end{aligned}$$

and

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad \bar{x}_4 \geq 0, \quad x_5 \geq 0, \quad \bar{x}_6 \geq 0.$$

- Before we can start: transform min to max

$$\max -Z = -\bar{x}_4 - \bar{x}_6$$

$$-Z + \bar{x}_4 + \bar{x}_6 = 0$$

b.v.	EQ	Z	x_1	x_2	x_3	\bar{x}_4	x_5	\bar{x}_6	RHS
Z	0	-1	0	0	0	1	0	1	0
x_3	1	0	0.3	0.1	1	0	0	0	2.7
x_4	2	0	0.5	0.5	0	1	0	0	6
x_6	3	0	0.6	0.4	0	0	-1	1	6
Z	0	-1	-0.5	-0.5	0	0	0	1	-6
x_3	1	0	0.3	0.1	1	0	0	0	2.7
x_4	2	0	0.5	0.5	0	1	0	0	6
x_6	3	0	0.6	0.4	0	0	-1	1	6
Z	0	-1	-1.1	-0.9	0	0	1	0	-12
x_3	1	0	0.3	0.1	1	0	0	0	2.7
x_4	2	0	0.5	0.5	0	1	0	0	6
x_6	3	0	0.6	0.4	0	0	-1	1	6

We want x_3, \bar{x}_4 and \bar{x}_6 as basisvariables. Tableau is not on proper form with respect to that choice of basic solution. We need to "repair" row 0.

$$r_0^{new} = r_o^{old} - r_2^{old}$$

$$r_0^{new} = r_o^{old} - r_3^{old}$$

Now tableau is on proper form and we can start the simplex algorithm for phase 1.

TABLE 4.13 Phase 1 of the two-phase method for the radiation therapy example

Iteration	Basic Variable	Eq.	Z	Coefficient of:						Right Side
				x_1	x_2	x_3	\bar{x}_4	x_5	\bar{x}_6	
0	Z	(0)	-1	-1.1	-0.9	0	0	1	0	-12
	x_3	(1)	0	0.3	0.1	1	0	0	0	2.7
	\bar{x}_4	(2)	0	0.5	0.5	0	1	0	0	6
	\bar{x}_6	(3)	0	0.6	0.4	0	0	-1	1	6
1	Z	(0)	-1	0	$-\frac{16}{30}$	$\frac{11}{3}$	0	1	0	-2.1
	x_1	(1)	0	1	$\frac{1}{3}$	$\frac{10}{3}$	0	0	0	9
	\bar{x}_4	(2)	0	0	$\frac{1}{3}$	$-\frac{5}{3}$	1	0	0	1.5
	\bar{x}_6	(3)	0	0	0.2	-2	0	-1	1	0.6
2	Z	(0)	-1	0	0	$-\frac{5}{3}$	0	$-\frac{5}{3}$	$\frac{8}{3}$	-0.5
	x_1	(1)	0	1	0	$\frac{20}{3}$	0	$\frac{5}{3}$	$-\frac{5}{3}$	8
	\bar{x}_4	(2)	0	0	0	$\frac{5}{3}$	1	$\frac{5}{3}$	$-\frac{5}{3}$	0.5
	x_2	(3)	0	0	1	-10	0	-5	5	3
3	Z	(0)	-1	0	0	0	1	0	1	0
	x_1	(1)	0	1	0	0	-4	-5	5	6
	x_3	(2)	0	0	0	1	$\frac{3}{5}$	1	-1	0.3
	x_2	(3)	0	0	1	0	6	5	-5	6

Ready for phase 2

Phase 2 Problem (Radiation Therapy Example):

Minimize $Z = 0.4x_1 + 0.5x_2,$

subject to

$$0.3x_1 + 0.1x_2 + x_3 = 2.7$$

$$0.5x_1 + 0.5x_2 = 6$$

$$0.6x_1 + 0.4x_2 - x_5 = 6$$

and

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0, \quad x_5 \geq 0.$$

$$\min Z = 0.4x_1 + 0.5x_2 = \max -Z = -0.4x_1 - 0.5x_2$$

In our tableau we are going to use:

$$-Z + 0.4x_1 + 0.5x_2 = 0$$

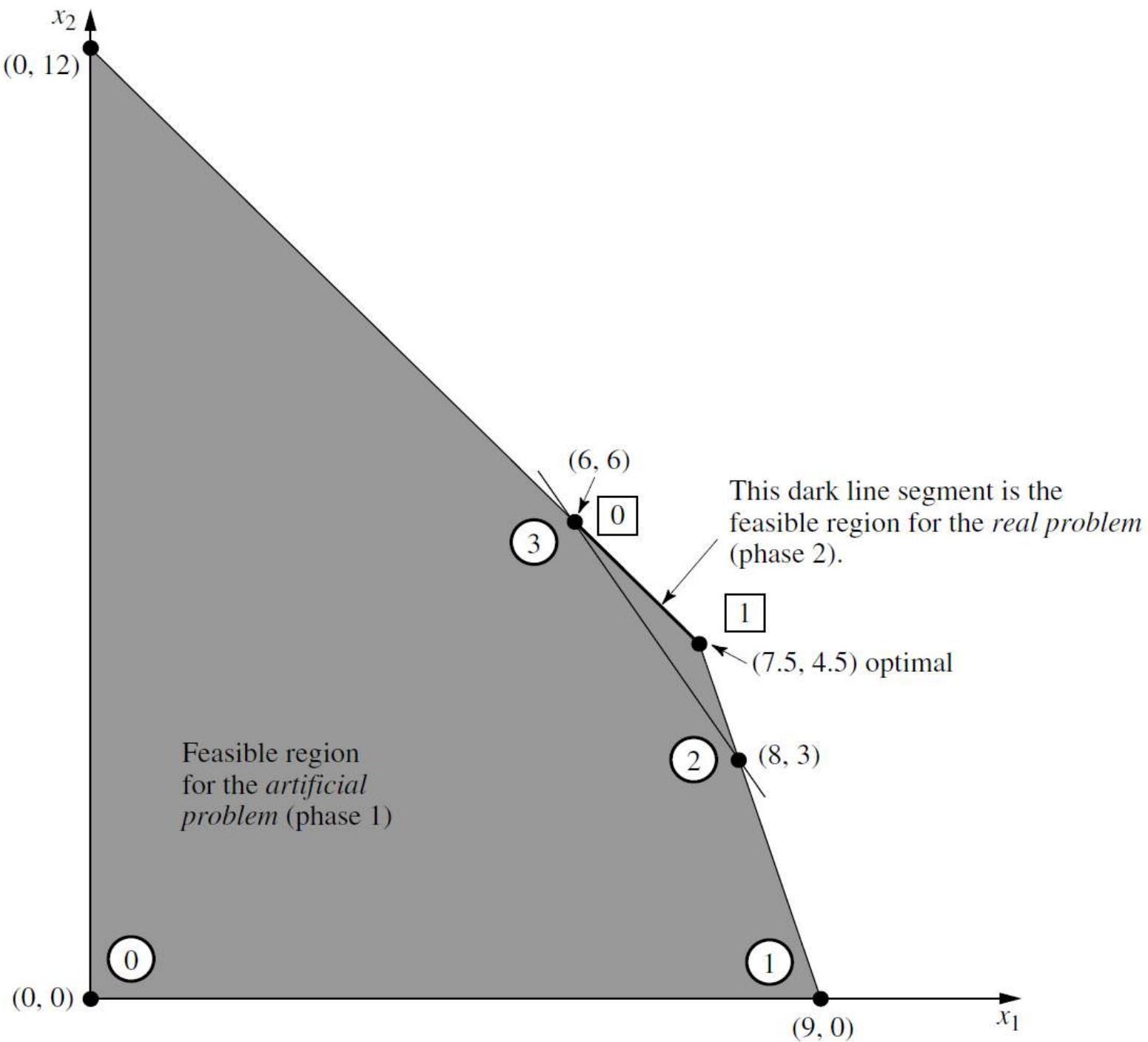
TABLE 4.14 Preparing to begin phase 2 for the radiation therapy example

	Basic Variable	Eq.	Z	Coefficient of:						Right Side
				x_1	x_2	x_3	\bar{x}_4	x_5	\bar{x}_6	
Final Phase 1 tableau	Z	(0)	-1	0	0	0	1	0	1	0
	x_1	(1)	0	1	0	0	-4	-5	5	6
	x_3	(2)	0	0	0	1	$\frac{3}{5}$	1	-1	0.3
	x_2	(3)	0	0	1	0	6	5	-5	6
Drop \bar{x}_4 and \bar{x}_6	Z	(0)	-1	0	0	0	0	0	0	0
	x_1	(1)	0	1	0	0	0	-5	6	6
	x_3	(2)	0	0	0	1	0	1	0	0.3
	x_2	(3)	0	0	1	0	0	5	6	6
Substitute phase 2 objective function	Z	(0)	-1	0.4	0.5	0	0	0	0	0
	x_1	(1)	0	1	0	0	0	-5	6	6
	x_3	(2)	0	0	0	1	0	1	0	0.3
	x_2	(3)	0	0	1	0	0	5	6	6
Restore proper form from Gaussian elimination	Z	(0)	-1	0	0	0	0	-0.5	0	-5.4
	x_1	(1)	0	1	0	0	0	-5	6	6
	x_3	(2)	0	0	0	1	0	1	0	0.3
	x_2	(3)	0	0	1	0	0	5	6	6

$$r_0^{new} = r_0^{old} - 0.4r_1^{old} - 0.5r_3^{old}$$

TABLE 4.15 Phase 2 of the two-phase method for the radiation therapy example

Iteration	Basic Variable	Eq.	Z	Coefficient of:				Right Side
				x_1	x_2	x_3	x_5	
0	Z	(0)	-1	0	0	0	-0.5	-5.4
	x_1	(1)	0	1	0	0	-5	6
	x_3	(2)	0	0	0	1	1	0.3
	x_2	(3)	0	0	1	0	5	6
1	Z	(0)	-1	0	0	0.5	0	-5.25
	x_1	(1)	0	1	0	5	0	7.5
	x_5	(2)	0	0	0	1	1	0.3
	x_2	(3)	0	0	1	-5	0	4.5



Summary 1/2

Translate to augmented form

- \leq constraint: add slack variable
- $=$ constraint: add artificial variable
- \geq constraint: add surplus variabel and artificial variables

If we did not add any artificial variables: Use normal simplex. Otherwise, use 2-phase Simplex.

Thanks to Morten Bondorf Gerdes for suggesting this summary!

Summary 2/2

Normal simplex:

1. Optimality test: no negative numbers in row zero \Rightarrow we are done.
2. Choose incoming basic variable (most negative in row 0).
3. Choose leaving basic variable (min ratio-test).
4. Restore the tableau to proper form with respect to the new basic variable (Gauss operations).
5. Jump back to 1.

2-phase simplex:

- Phase 1: Setup tableau that minimizes artificial variables. Put tableau in proper form (work on row 0).
- Phase 1: Solve LP using normal simplex
- Phase 1: If all artificial variables reaches zero then go to phase 2. Otherwise the LP has no feasible solutions.
- Phase 2: Use last tableau from phase 1. Remove all artificial variables.
- Phase 2: Insert the original objective function.
- Phase 2: Put tableau in proper form (work on row 0).
- Phase 2: Solve LP using normal simplex

Thanks to Morten Bondorf Gerdes for suggesting this summary!

2-phase method



- Video with another 2-phase example on our DTU inside group (file sharing/videos). The video is in Danish.

No feasible solutions?



What happens if there are no feasible solutions to the LP?

Answer: First phase ends with a solution where one or more artificial variables have value greater than 0.

Handle negative right hand side?



Easy: multiply whole constraint by -1. Examples:

$$\begin{array}{ll} x_1 + x_2 \geq -10 & -x_1 - x_2 \leq 10 \\ 4x_1 - 3x_2 = -6 & \Leftrightarrow -4x_1 + 3x_2 = 6 \\ -2x_1 + x_2 \leq -2 & 2x_1 - x_2 \geq 2 \end{array}$$

Handle negative right hand side?



Easy: multiply whole constraint by -1. Examples:

$$\begin{array}{ll} x_1 + x_2 \geq -10 & -x_1 - x_2 \leq 10 \\ 4x_1 - 3x_2 = -6 & \Leftrightarrow -4x_1 + 3x_2 = 6 \\ -2x_1 + x_2 \leq -2 & 2x_1 - x_2 \geq 2 \end{array}$$

That was the last piece for the puzzle.
Now we know how to solve any LP!

Simplex can solve large LPs fast!



name	variables	constraints	Non-zeros		Iterations	Time (s)
25FV47	1571	821	10400		2282	0.16
CRE-B	72447	9648	256095		20620	0.90
OSA-60	232966	10280	1397793		3703	0.78
KEN-18	154699	105127	358171		99423	9.77

Abstract and concrete models

Concrete Wyndor model

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

- 2 products (glass doors, windows)
- Requires time on 3 plants

	1	2	Capacity (hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

- Profit is per unit and should be multiplied by 1000\$
- We want to plan the production so that our profit is maximized

Abstract and concrete models

Abstract Wyndor model

- We may want to extend the model with more products and more plants
- Or we may want to use the model for another company
- It would be nice to write the model in a more generic way!

- 2 products (glass doors, windows)
- Requires time on 3 plants

	1	2	Capacity (hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

Abstract and concrete models

Abstract Wyndor model

It would be nice to write the model in a more generic way!

- m : number of plants
- n : number of products
- b_i : hours available on plant i per week ($i = 1, \dots, m$)
- p_j : profit per batch of product j ($j = 1, \dots, n$)
- a_{ij} : how many hours we need on plant i when producing one batch of product j

- 2 products (glass doors, windows)
- Requires time on 3 plants

	1	2	Capacity (hours/week)
1	1	0	4
2	0	2	12
3	3	2	18
profit	3	5	

Abstract and concrete models

Abstract Wyndor model

It would be nice to write the model in a more generic way!

- m : number of plants
- n : number of products
- b_i : hours available on plant i per week ($i = 1, \dots, m$)
- p_j : profit per batch of product j ($j = 1, \dots, n$)
- a_{ij} : how many hours we need on plant i when producing one batch of product j
- x_j : Our decision variables. Indicates how many batches of product j we should make per week.
- The abstract model:

$$\max \sum_{j=1}^n p_j x_j$$

subject to

$$\begin{aligned} \sum_{j=1}^n a_{ij} x_j &\leq b_i \quad \forall i = 1, \dots, m \\ x_j &\geq 0 \quad \forall j = 1, \dots, n \end{aligned}$$

Abstract Wyndor model in Julia (1)

```
using JuMP, GLPK

m = 3          # number of plants
n = 2          # number of products
b = [ 4 12 18] # hours available on plant i per week
p = [ 3 5 ]    # profit per batch of product
a = [ 1 0;
      0 2;
      3 2]

model = Model(with_optimizer(GLPK.Optimizer))
# x[j]: how many batches of product j should we make per week
@variable(model, x[j=1:n] >= 0 )

@objective(model, Max, sum(p[j]*x[j] for j=1:n) )
@constraint(model, [i=1:m], sum(a[i,j]*x[j] for j=1:n) <= b[i] )

print(model)

optimize!(model)

println("Objective value: ", JuMP.objective_value(model))
println("x = ", JuMP.value.(x))
```

Abstract Wyndor model in Julia (2)

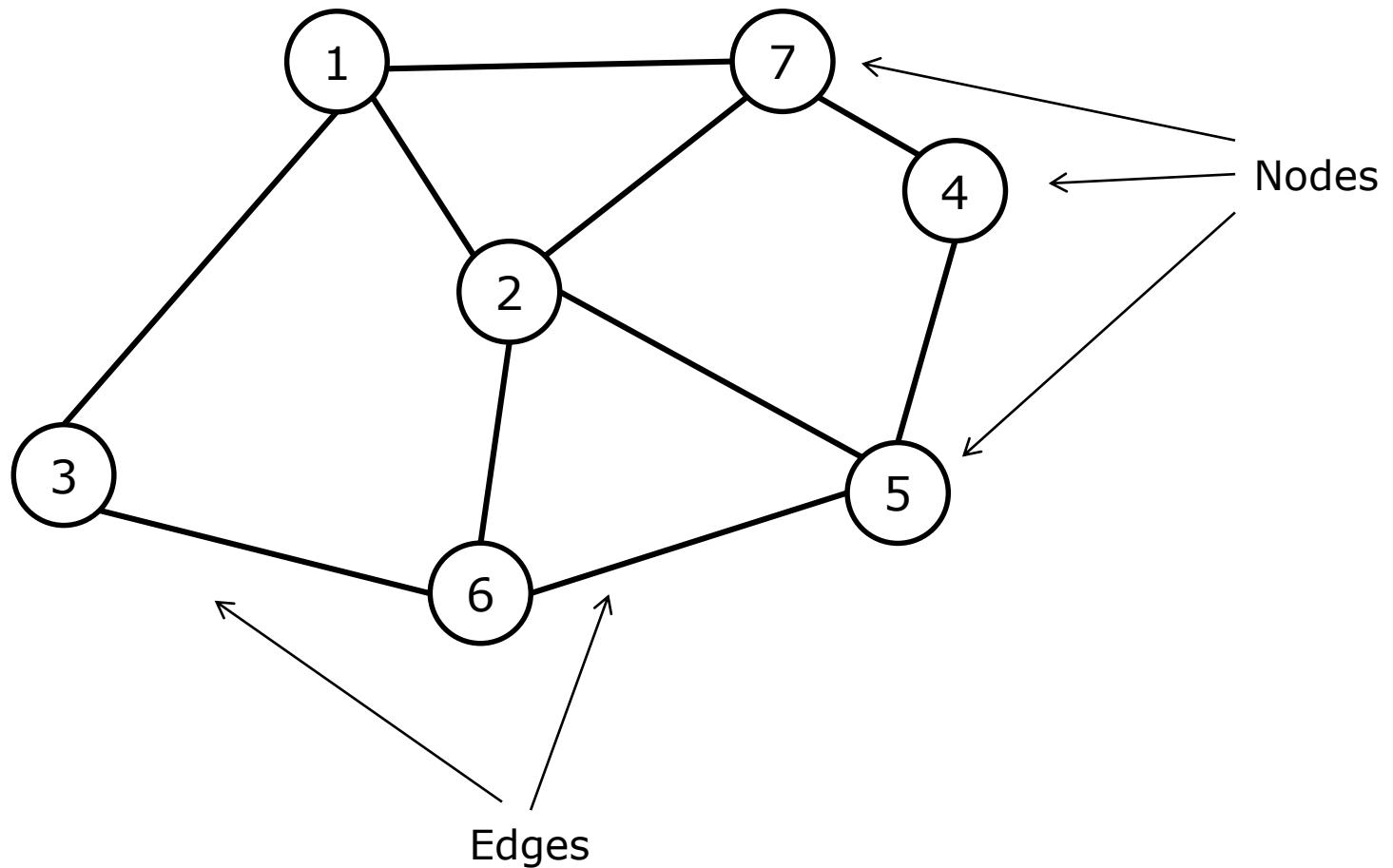
```
using JuMP, GLPK
include("wyndor-data2.jl")  
  
model = Model(with_optimizer(GLPK.Optimizer))  
#  $x[j]$ : how many batches of product  $j$  should we make per week  
@variable(model, x[j=1:n] >= 0 )  
  
@objective(model, Max, sum(p[j]*x[j] for j=1:n) )  
@constraint(model, [i=1:m], sum(a[i,j]*x[j] for j=1:n) <= b[i] )  
  
print(model)  
  
optimize!(model)  
  
println("Objective value: ", JuMP.objective_value(model))  
println("x = ", JuMP.value.(x))
```



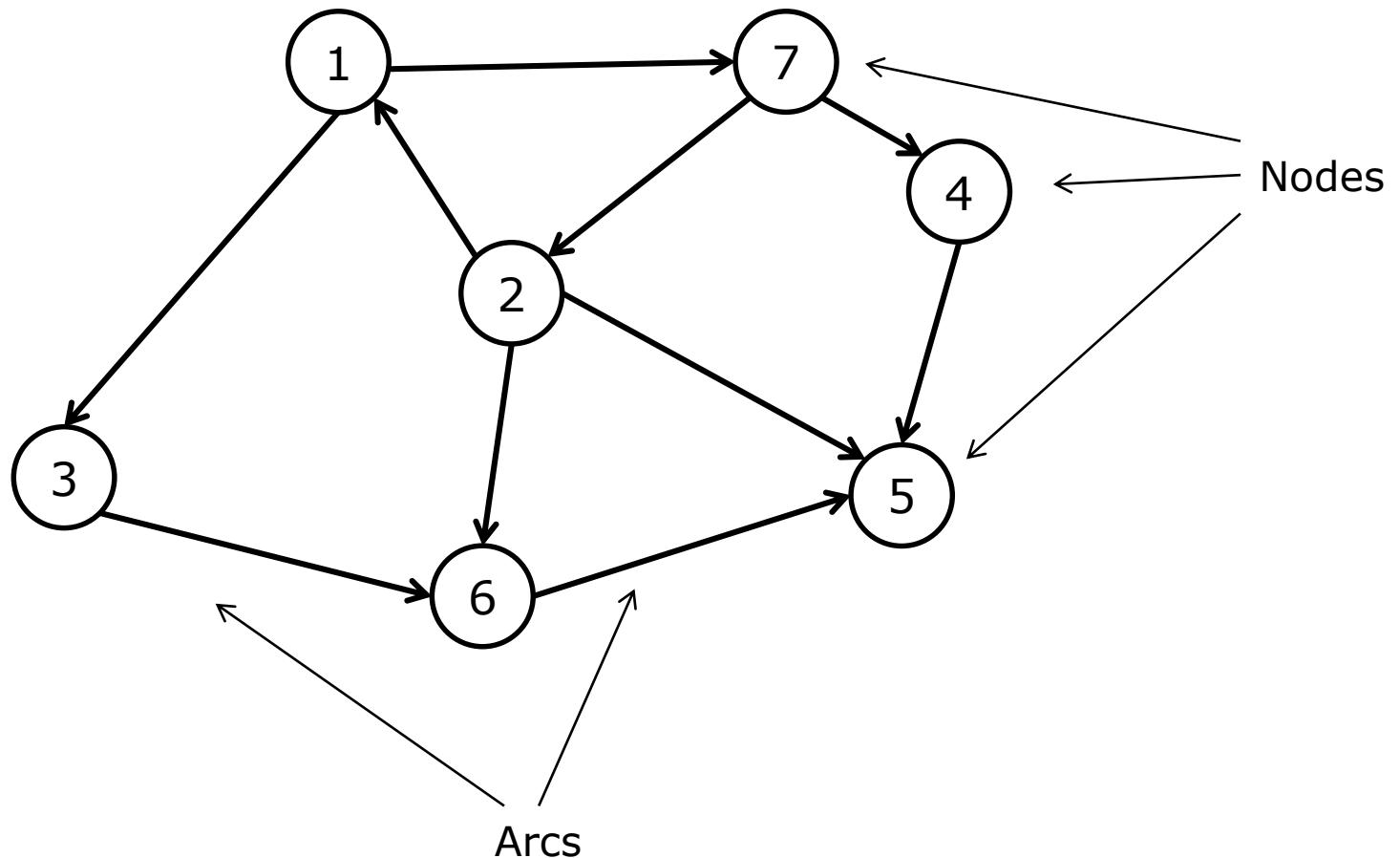
Basic graph theory



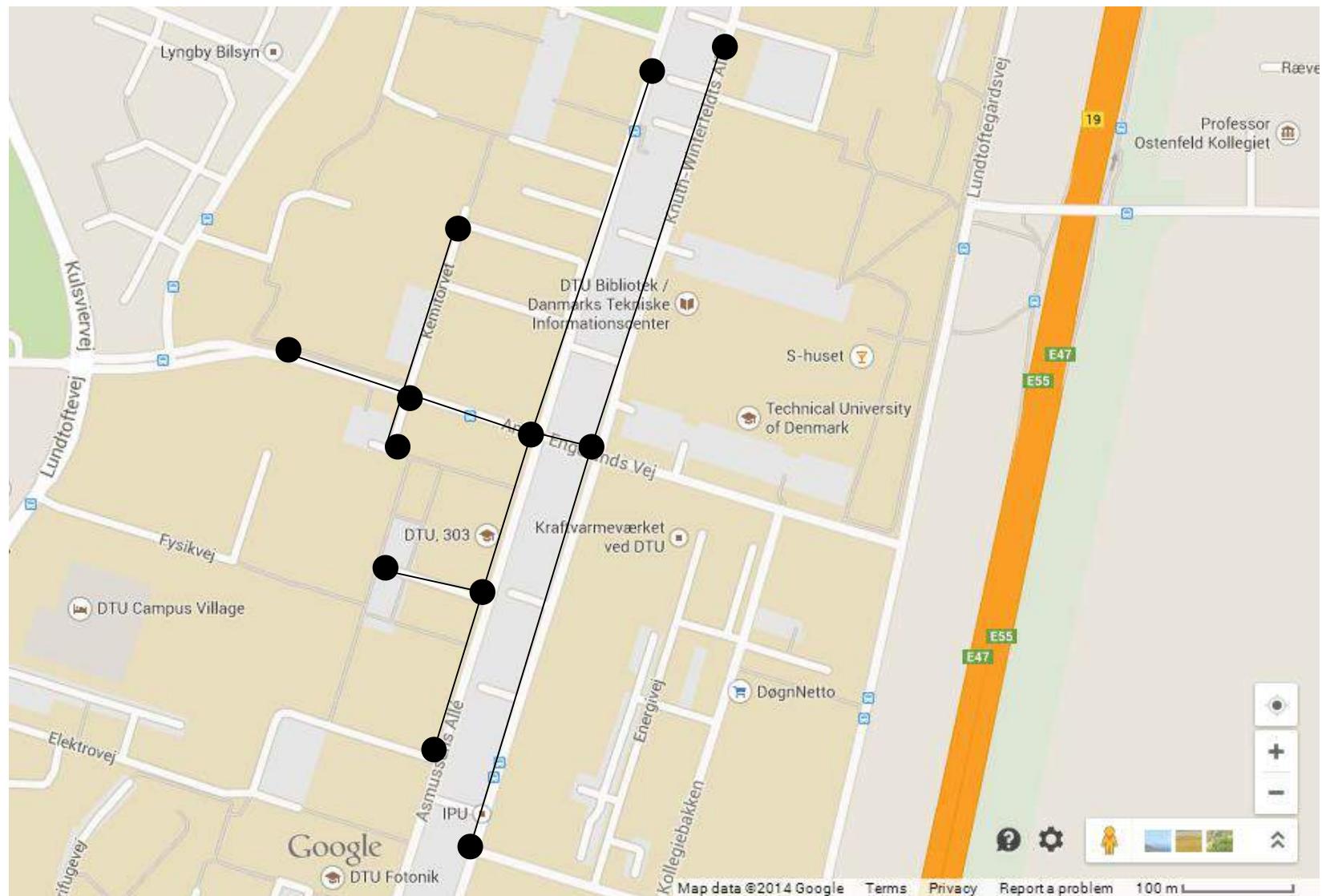
Undirected graphs (networks)



Directed graphs (networks)



Examples of graphs- Road networks



Examples of graphs - Sewer systems

- Can we transport the water away quickly enough?

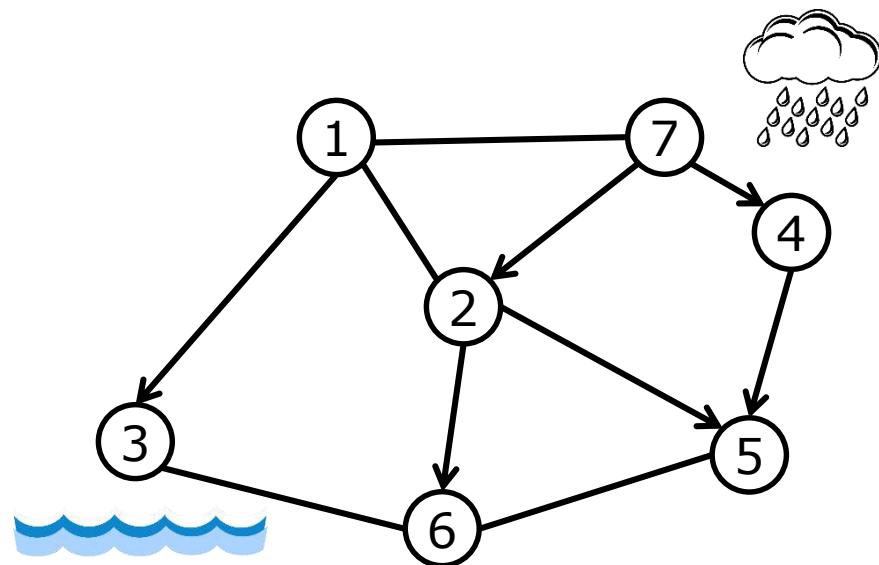
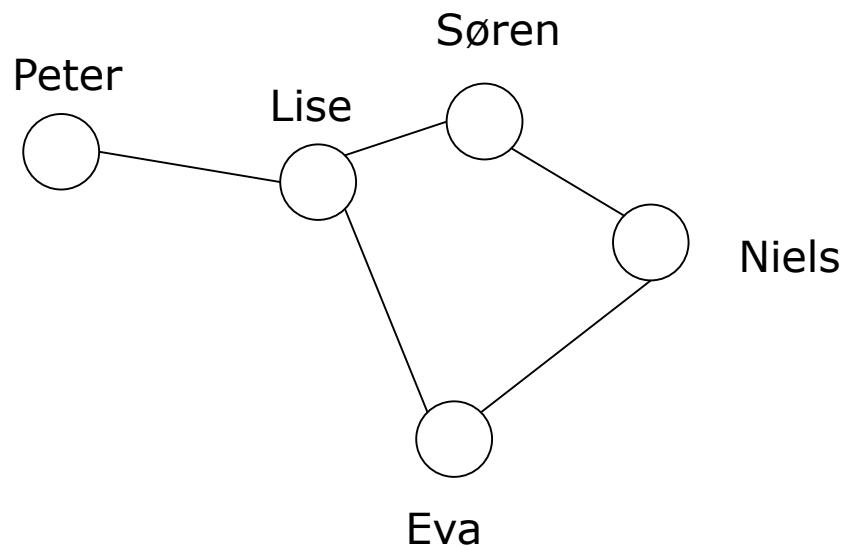


Foto: Lisa Risager
License: Attribution-ShareAlike 2.0 Generic
Istedgade July 2011

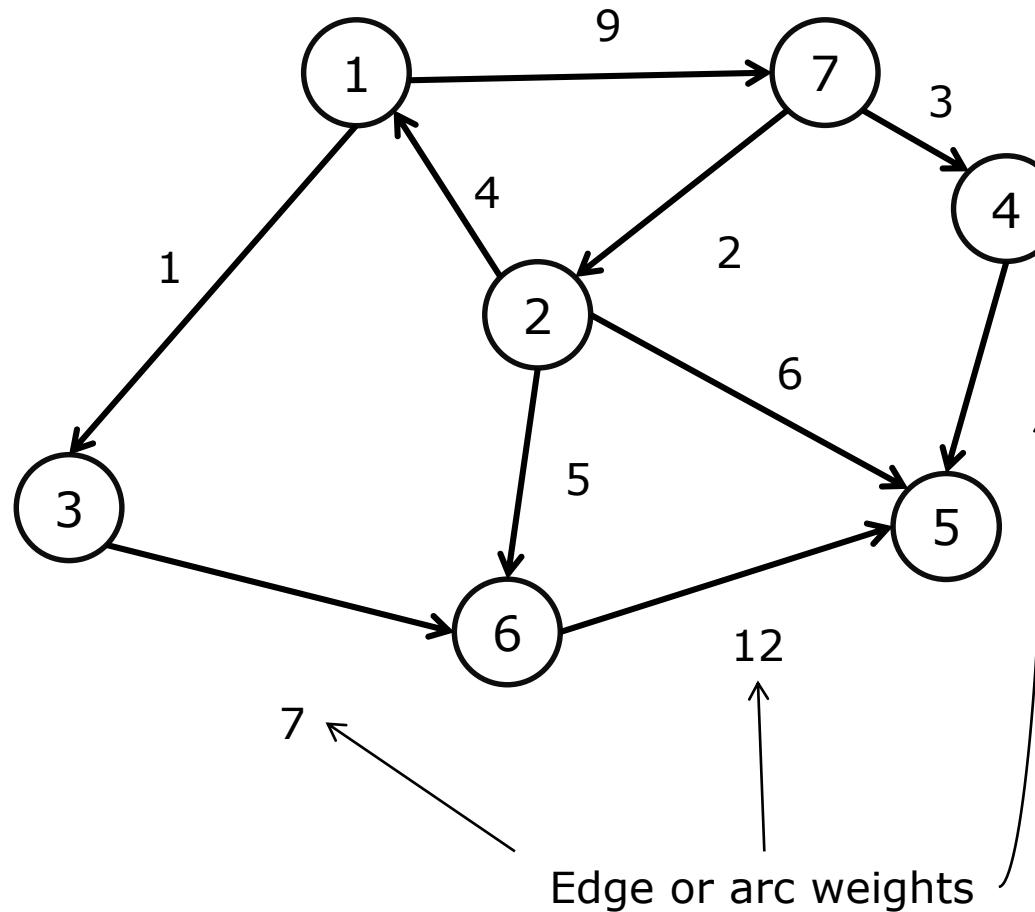
Examples of graphs- social networks



- Who knows who?
- Who follows who?

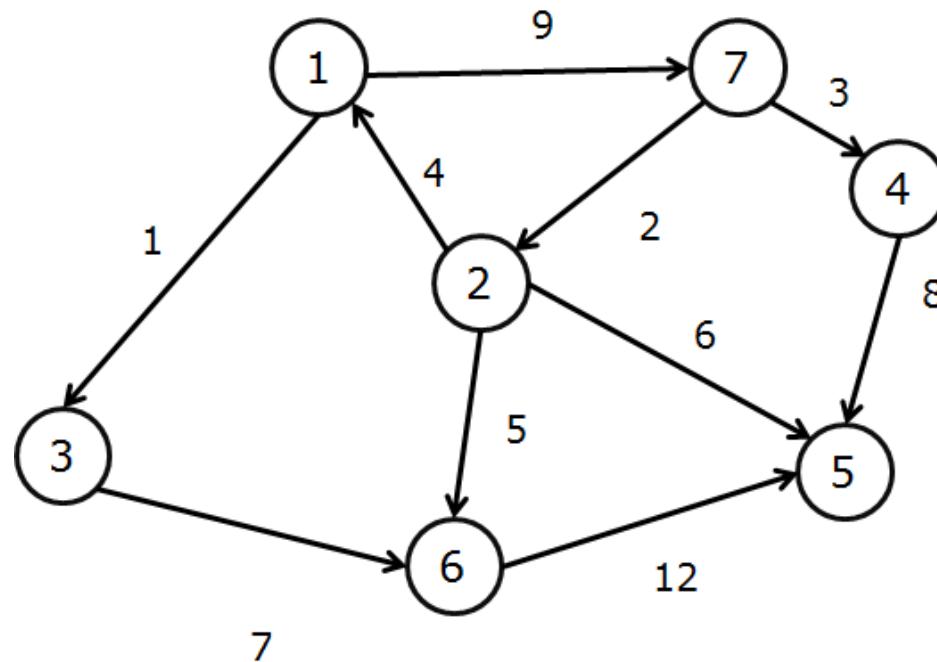


graphs (networks)

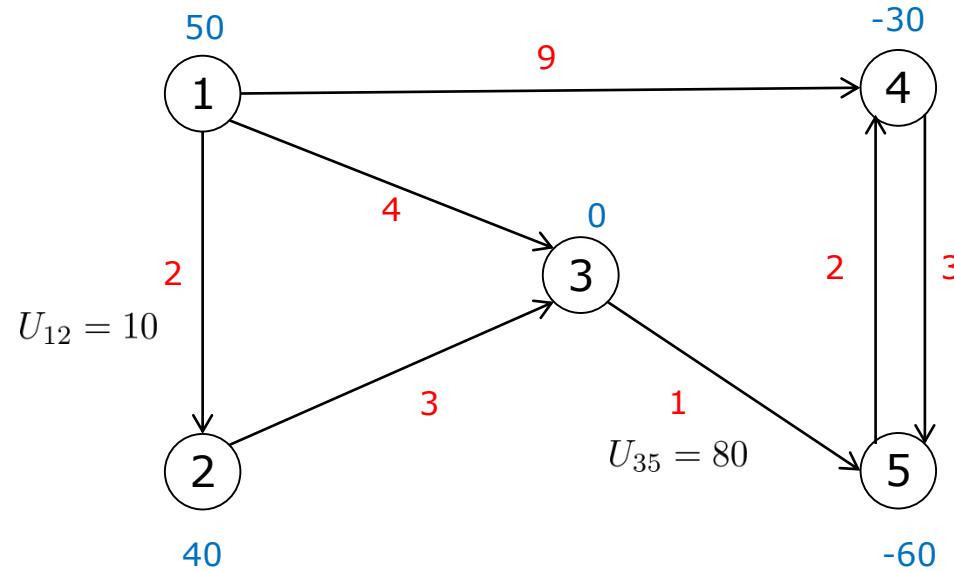


(Could represent arc capacity, cost, distance, etc.)

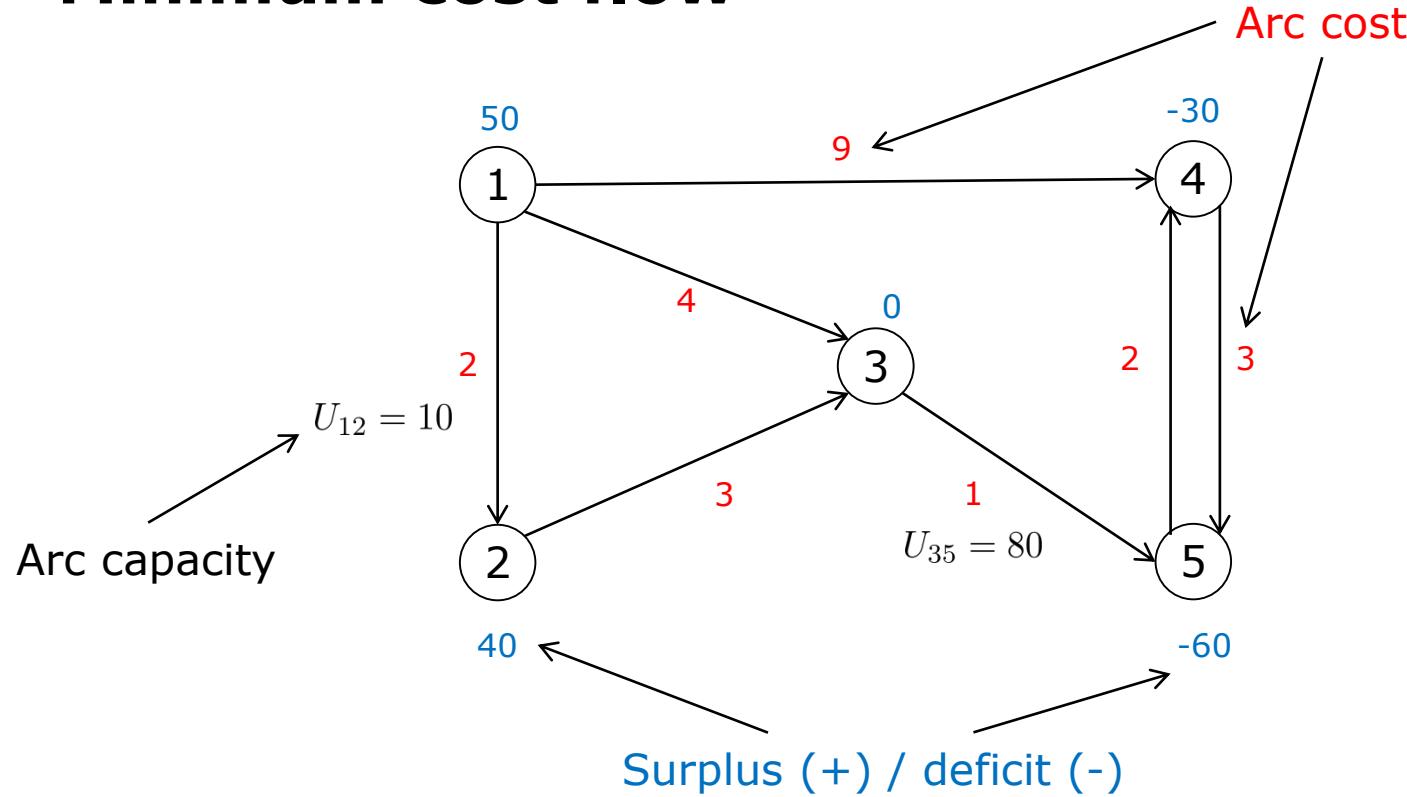
Nodes	Arcs	Flow
Intersections	Roads	Vehicles
Switching points	Wires/channels	Messages/Data
Pumping stations	Pipes	Fluids



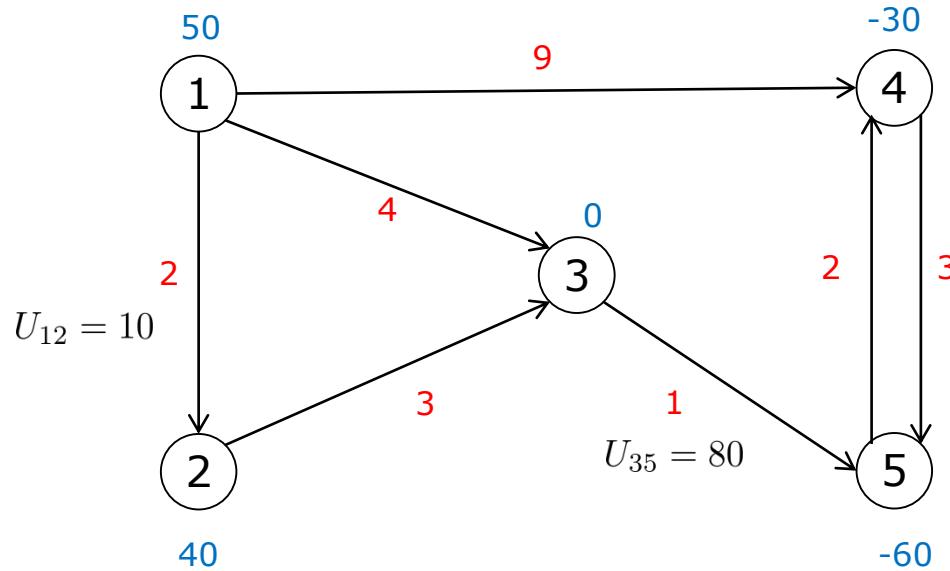
Minimum cost flow – an LP/graph example



Minimum cost flow



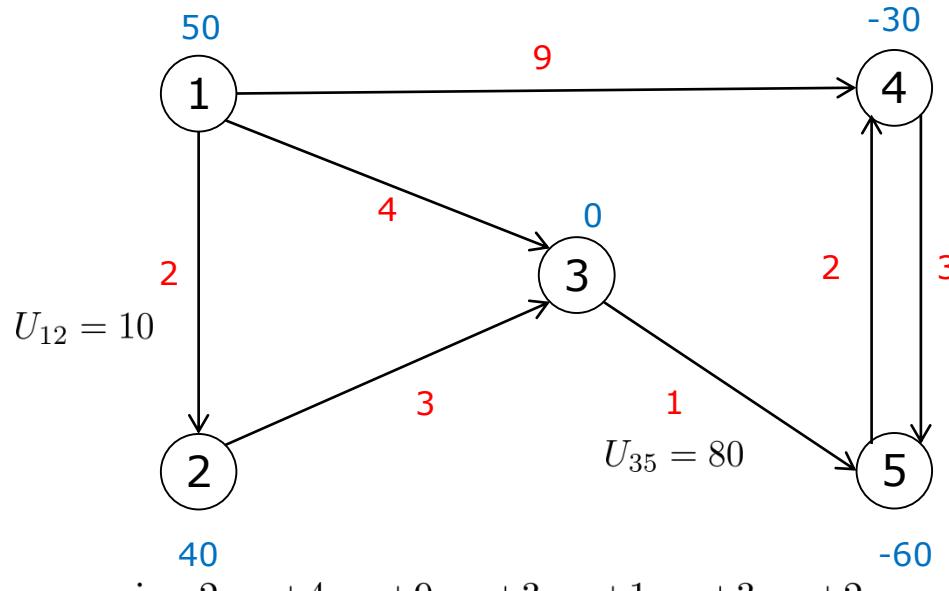
Minimum cost flow



Decision variables:

x_{ij} : how many units we send through the arc from node i to node j .

Minimum cost flow



$$\min 2x_{12} + 4x_{13} + 9x_{14} + 3x_{23} + 1x_{35} + 3x_{45} + 2x_{54}$$

subject to

$$x_{12} + x_{13} + x_{14} = 50$$

$$- x_{12} + x_{23} = 40$$

$$- x_{13} - x_{23} + x_{35} = 0$$

$$- x_{14} + x_{45} - x_{54} = -30$$

$$- x_{35} - x_{45} + x_{54} = -60$$

$$x_{12} \leq 10$$

$$x_{35} \leq 80$$

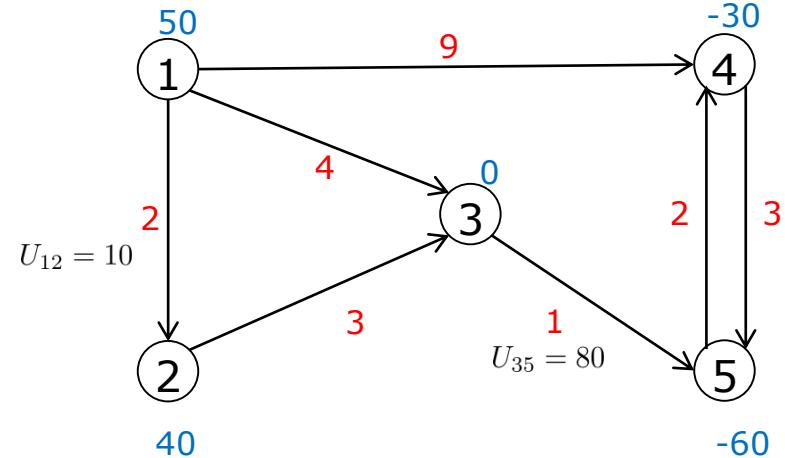
$$x_{12}, x_{13}, x_{14}, x_{23}, x_{35}, x_{45}, x_{54} \geq 0$$

Minimum cost flow : Generic/abstract model

- m : number of nodes in the network
- n : number of arcs in the network
- (i, j) the arc going from i to j
- c_{ij} : cost for sending one unit through arc (i, j)
- u_{ij} : capacity of arc (i, j)
- b_i =surplus / deficit at node i
- A : set of all arcs in the network

Decision variables:

- x_{ij} : The amount of goods to send through arc (i, j)



Minimum cost flow : Generic/abstract model

- m : number of nodes in the network
- n : number of arcs in the network
- (i, j) the arc going from i to j
- c_{ij} : cost for sending one unit through arc (i, j)
- u_{ij} : capacity of arc (i, j)
- b_i =surplus / deficit at node i
- A : set of all arcs in the network

Decision variables:

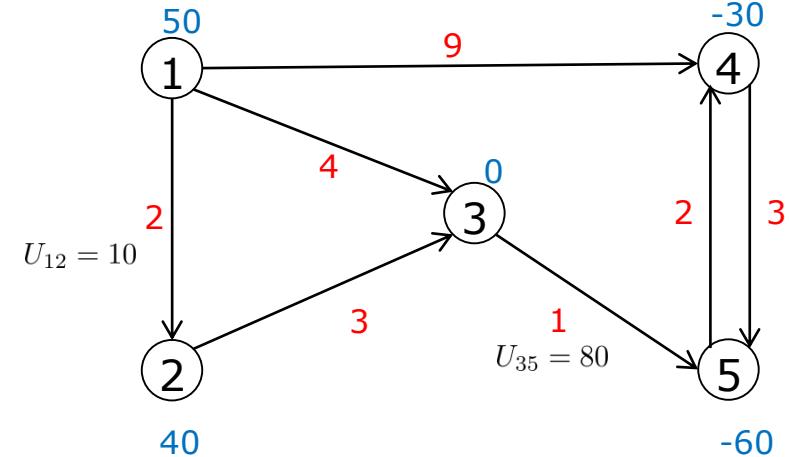
- x_{ij} : The amount of goods to send through arc (i, j)

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

subject to

$$\sum_{(i,j) \in A: i=k} x_{ij} - \sum_{(i,j) \in A: j=k} x_{ij} = b_k \quad \forall k = 1, \dots, m$$

$$0 \leq x_{ij} \leq u_{ij} \quad \forall (i, j) \in A$$



Minimum cost flow : Julia model

```
module test
using JuMP, GLPK

mutable struct Arc
    from::Int64
    to::Int64
    cost::Int64
    UB::Int64
end

m = 5
# Here we define the 7 arcs
arcs = [Arc(1,2,2,10), Arc(1,3,4,0), Arc(1,4,9,0), Arc(2,3,3,0), Arc(3,5,1,80),
        Arc(4,5,3,0), Arc(5,4,2,0)]
demands = [50, 40, 0, -30, -60]

model = Model(with_optimizer(GLPK.Optimizer))
@variable(model, x[arcs] >= 0 )
@objective(model, Min, sum(a.cost*x[a] for a in arcs) )
@constraint(model, [i=1:m], sum(x[a] for a in arcs if a.from==i)
                           - sum(x[a] for a in arcs if a.to==i) == demands[i] )
for a in arcs
    if a.UB > 0
        @constraint(model, x[a] <= a.UB )
    end
end

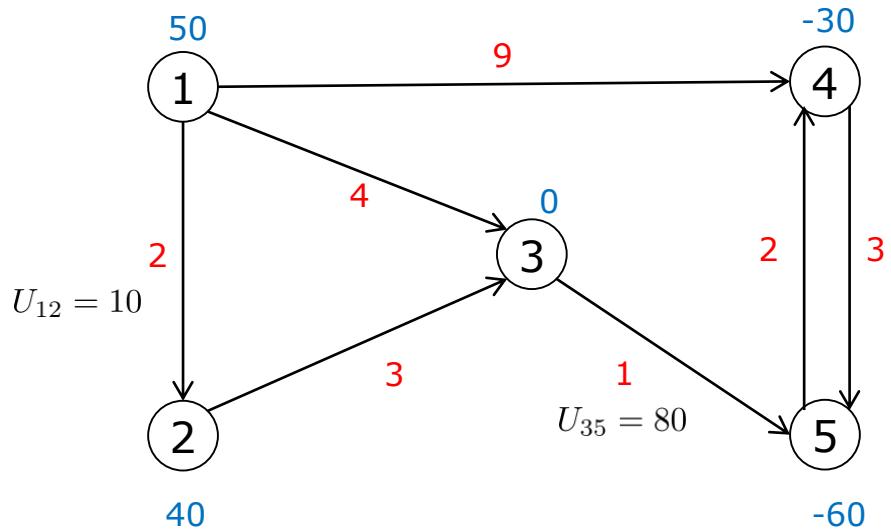
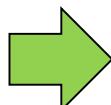
optimize!(model)

println("Objective value: ", JuMP.objective_value(model))
for a in arcs
    println("Flow on arc (",a.from,",",a.to,") is ", JuMP.value.(x[a]))
end

end
```

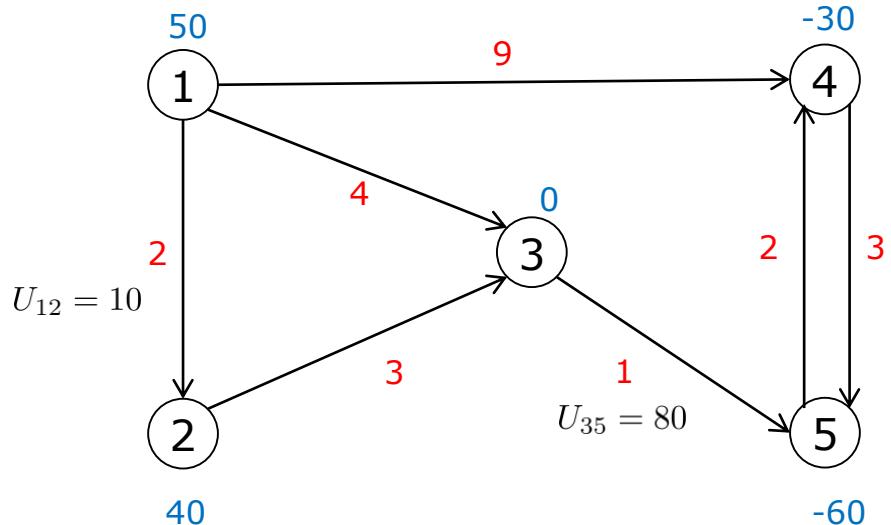
Minimum cost flow : Generic/abstract model

An “**instance**” of the Minimum cost flow problem.

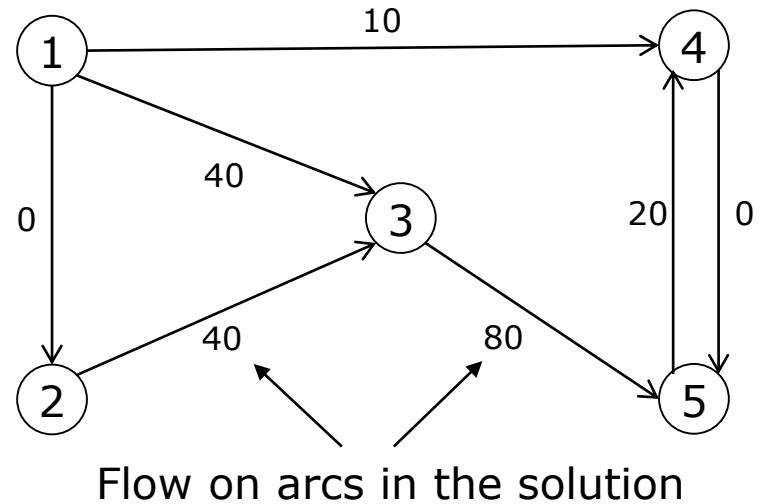
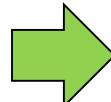


Minimum cost flow : Generic/abstract model

An “**instance**” of the Minimum cost flow problem.



An optimal solution to the instance above
(objective value is 490)



Flow on arcs in the solution

Now it is exercise time!

Lecture 4

Linear programming

Matrix form

Chapter 5

See appendix 4 in our book to get a refresher on matrix computations

A collage of mathematical symbols including integrals, summation, infinity, and various Greek letters like Delta, Theta, Omega, Epsilon, and Chi.

Standard form

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

Matrix form

$$\max Z = [\begin{array}{cc} 3 & 5 \end{array}] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

subject to

$$\begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

and

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Matrix form, in general:

subject to

$$\max \mathbf{c}\mathbf{x}$$

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

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

$$\max \mathbf{c}x$$

Matrix form

subject to

$$Ax \leq b$$

$$x \geq 0$$

$$\max[c_1, c_2, \dots, c_n] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

subject to

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \leq \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$
$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\max c_1 x_1 + c_2 x_2 + \dots + c_n x_n$$

subject to

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1$$
$$a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \leq b_2$$
$$\vdots \leq \vdots$$
$$a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m$$
$$x_1, x_2, \dots, x_n \geq 0$$

Matrix form

Augmented form. Slack variables:

$$x_s = \begin{bmatrix} x_{n+1} \\ x_{n+2} \\ \vdots \\ x_{n+m} \end{bmatrix}$$

Constraint now like this:

$$[A, I] \begin{bmatrix} x \\ x_s \end{bmatrix} = b \quad \text{og} \quad \begin{bmatrix} x \\ x_s \end{bmatrix} \geq 0$$

Matrix form – Wyndor example

Wyndor example:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

$$x_1 \geq 0, \quad x_2 \geq 0$$

$$[A, I] \begin{bmatrix} x \\ x_s \end{bmatrix} = b \quad \text{og} \quad \begin{bmatrix} x \\ x_s \end{bmatrix} \geq 0$$

$$\mathbf{c} = [3, 5], \quad [\mathbf{A}, \mathbf{I}] = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{x}_s = \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

Example: write on matrix form.

Maximize $Z = 4x_1 + 3x_2 + 6x_3,$

subject to

$$3x_1 + x_2 + 3x_3 \leq 30$$

$$2x_1 + 2x_2 + 3x_3 \leq 40$$

and

$$x_1 \geq 0, \quad x_2 \geq 0, \quad x_3 \geq 0.$$

Basic solutions



m =number of functional constraints

Every basic solution has m basic variables, the other variables are non-basic.
Non-basic variables are set to 0.

We find the value of the basic variables by solving a system of linear equations with m constraints and m variables

A feasible basic solution is a basic solution with all basic variables ≥ 0 .

Matrix form

Constraints in general:

$$[A, I] \begin{bmatrix} x \\ x_s \end{bmatrix} = b$$

Wyndor example

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

● ● ●

”We find the value of the basic variables by solving a system of linear equations with m constraints and m variables ” If x_3, x_4, x_5 are basic variables then we need to solve:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} \Rightarrow \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$




 $Bx_B = b \Rightarrow x_B = B^{-1}b$

Matrix form

Constraints in general:

$$[A, I] \begin{bmatrix} x \\ x_s \end{bmatrix} = b$$

Wyndor example

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \\ \bullet & \bullet & \bullet & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

”We find the value of the basic variables by solving a system of linear equations with m constraints and m variables” If x_1, x_2, x_4 are basic variables then we need to solve:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 3 & 2 & 0 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} \Rightarrow \begin{bmatrix} x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 3 & 2 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 6 \end{bmatrix}$$


 $Bx_B = b \Rightarrow x_B = B^{-1}b$

Matrix form

$$\max Z = [c, \mathbf{0}] \begin{bmatrix} x \\ x_s \end{bmatrix}$$

$$[A, I] \begin{bmatrix} x \\ x_s \end{bmatrix} = b$$

$$\begin{bmatrix} x \\ x_s \end{bmatrix} \geq \mathbf{0}$$

For every choice of basic variables x_B we compute their value:

$$x_B = B^{-1}b$$

Since non-basic variables are forced to zero we can compute the objective value corresponding to the choice of basic variables:

$$Z = c_B x_B = c_B B^{-1}b$$

Where c_B are the coefficients from $[c, \mathbf{0}]$ corresponding to the basic variables.

Wyndor example

$$\max Z = [\begin{array}{ccccc} 3 & 5 & 0 & 0 & 0 \end{array}] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$

● ●
 ● ● ●
 ●

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix}$$

● ●
 ● ● ●
 ●

We saw that:

$$x_B = B^{-1}b \quad \text{og} \quad Z = c_B x_B = c_B B^{-1}b$$

Example 1: If x_3, x_4 og x_5 are basic variables then:

$$x_B = B^{-1}b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} \quad \text{and} \quad Z = c_B x_B = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = 0$$

Example 2: If x_1, x_2 and x_4 are basic variables then:

$$x_B = B^{-1}b = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 3 & 2 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 4 \\ 12 \\ 18 \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \\ 6 \end{bmatrix} \quad \text{and} \quad Z = c_B x_B = \begin{bmatrix} 3 & 5 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 6 \end{bmatrix} = 27$$

TABLE 4.8 Complete set of simplex tableaux for the Wyndor Glass Co. problem

$\max Z = 3x_1 + 5x_2$
 subject to
 $x_1 \leq 4$
 $2x_2 \leq 12$
 $3x_1 + 2x_2 \leq 18$
 and
 $x_1 \geq 0, x_2 \geq 0$

Iteration	Basic Variable	Eq.	Coefficient of:						Right Side
			Z	x_1	x_2	x_3	x_4	x_5	
0	Z	(0)	1	-3	-5	0	0	0	0
	x_3	(1)	0	1	0	1	0	0	4
	x_4	(2)	0	0	2	0	1	0	12
	x_5	(3)	0	3	2	0	0	1	18
1	Z	(0)	1	-3	0	0	$\frac{5}{2}$	0	30
	x_3	(1)	0	1	0	1	0	0	4
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_5	(3)	0	3	0	0	-1	1	6
2	Z	(0)	1	0	0	0	$\frac{3}{2}$	1	36
	x_3	(1)	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	2
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_1	(3)	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	2

Start:

$$\begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{Z} \\ \mathbf{X} \\ \mathbf{X}_S \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{b} \end{bmatrix}.$$

Matrix form

Start:

$$\begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{I} \end{bmatrix} \begin{bmatrix} Z \\ \mathbf{x} \\ \mathbf{x}_s \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}.$$

We can compute tableau for every choice of basis variables. Choice of basis variables gives B and c_B :

right hand side:

$$\begin{bmatrix} Z \\ \mathbf{x}_B \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{c}_B \mathbf{B}^{-1} \\ \mathbf{0} & \mathbf{B}^{-1} \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_B \mathbf{B}^{-1} \mathbf{b} \\ \mathbf{B}^{-1} \mathbf{b} \end{bmatrix}$$

Left hand side :

$$\begin{bmatrix} 1 & \mathbf{c}_B \mathbf{B}^{-1} \\ \mathbf{0} & \mathbf{B}^{-1} \end{bmatrix} \begin{bmatrix} 1 & -\mathbf{c} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{c}_B \mathbf{B}^{-1} \mathbf{A} - \mathbf{c} & \mathbf{c}_B \mathbf{B}^{-1} \\ \mathbf{0} & \mathbf{B}^{-1} \mathbf{A} & \mathbf{B}^{-1} \end{bmatrix}$$

Complete
tableau:

$$\begin{bmatrix} 1 & \mathbf{c}_B \mathbf{B}^{-1} \mathbf{A} - \mathbf{c} & \mathbf{c}_B \mathbf{B}^{-1} \\ \mathbf{0} & \mathbf{B}^{-1} \mathbf{A} & \mathbf{B}^{-1} \end{bmatrix} \begin{bmatrix} Z \\ \mathbf{x} \\ \mathbf{x}_s \end{bmatrix} = \begin{bmatrix} \mathbf{c}_B \mathbf{B}^{-1} \mathbf{b} \\ \mathbf{B}^{-1} \mathbf{b} \end{bmatrix}$$

TABLE 4.8 Complete set of simplex tableaux for the Wyndor Glass Co. problem

$\max Z = 3x_1 + 5x_2$
 subject to
 $x_1 \leq 4$
 $2x_2 \leq 12$
 $3x_1 + 2x_2 \leq 18$
 and
 $x_1 \geq 0, x_2 \geq 0$

Iteration	Basic Variable	Eq.	Coefficient of:						Right Side
			Z	x_1	x_2	x_3	x_4	x_5	
0	Z	(0)	1	-3	-5	0	0	0	0
	x_3	(1)	0	1	0	1	0	0	4
	x_4	(2)	0	0	2	0	1	0	12
	x_5	(3)	0	3	2	0	0	1	18
1	Z	(0)	1	-3	0	0	$\frac{5}{2}$	0	30
	x_3	(1)	0	1	0	1	0	0	4
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_5	(3)	0	3	0	0	-1	1	6
2	Z	(0)	1	0	0	0	$\frac{3}{2}$	1	36
	x_3	(1)	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	2
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_1	(3)	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	2

Entire tableau:

$$\begin{bmatrix} 1 & \mathbf{c}_B \mathbf{B}^{-1} \mathbf{A} - \mathbf{c} & \mathbf{c}_B \mathbf{B}^{-1} \\ \mathbf{0} & \mathbf{B}^{-1} \mathbf{A} & \mathbf{B}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{Z} \\ \mathbf{x} \\ \mathbf{x}_s \end{bmatrix} = \begin{bmatrix} \mathbf{c}_B \mathbf{B}^{-1} \mathbf{b} \\ \mathbf{B}^{-1} \mathbf{b} \end{bmatrix}$$

A fundamental insight

Start:

$$\left[\begin{array}{c|c|c} -c & 0 & 0 \\ A & I & b \end{array} \right]$$

We can compute the tableau for any choice of basis variables (basisvariable defines B and c_B)

$$\left[\begin{array}{c|c|c} c_B B^{-1} A - c & c_B B^{-1} & c_B B^{-1} b \\ B^{-1} A & B^{-1} & B^{-1} b \end{array} \right]$$

In the final iteration we define

$$y^* = c_B B^{-1}$$

$$S^* = B^{-1}$$

Fundamental insight: The last tableau can be found just using y^* , S^* and the information from the initial tableau.

$$\left[\begin{array}{c|c|c} y^* A - c & y^* & y^* b \\ S^* A & S^* & S^* b \end{array} \right]$$

TABLE 4.8 Complete set of simplex tableaux for the Wyndor Glass Co. problem

		$\max Z = 3x_1 + 5x_2$
		subject to
		$x_1 \leq 4$
		$2x_2 \leq 12$
		$3x_1 + 2x_2 \leq 18$
		and
		$x_1 \geq 0, x_2 \geq 0$

Iteration	Basic Variable	Eq.	Coefficient of:						Right Side
			Z	x_1	x_2	x_3	x_4	x_5	
0	Z	(0)	1	-3	-5	0	0	0	0
	x_3	(1)	0	1	0	1	0	0	4
	x_4	(2)	0	0	2	0	1	0	12
	x_5	(3)	0	3	2	0	0	1	18
1	Z	(0)	1	-3	0	0	$\frac{5}{2}$	0	30
	x_3	(1)	0	1	0	1	0	0	4
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_5	(3)	0	3	0	0	-1	1	6
2	Z	(0)	1	0	0	0	$\frac{3}{2}$	1	36
	x_3	(1)	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	2
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_1	(3)	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	2

Fundamental insight: The last tableau can be found just using y^* , S^* and the information from the initial tableau.

$$\left[\begin{array}{cc|cc} y^* A - c & & y^* & y^* b \\ S^* A & & S^* & S^* b \end{array} \right]$$

Normal simplex:

1. Optimality test: no negative numbers in row zero \Rightarrow we are done.
2. Choose incoming basic variable (most negative in row 0).
3. Choose leaving basic variable (min ratio-test).
4. Restore the tableau to proper form with respect to the new basic variable (Gauss operations).
5. Jump back to 1.

Revised simplex = simplex algorithm on matrix form

Simplex algorithm on matrix form

1. Optimality test: no negative numbers in row zero \Rightarrow we are done.
2. Choose incoming basic variable (most negative in row 0).
3. Choose leaving basic variable (min ratio-test).
4. Compute new tableau using:

$$\left[\begin{array}{c|cc|c} c_B B^{-1} A - c & c_B B^{-1} & c_B B^{-1} b \\ B^{-1} A & B^{-1} & B^{-1} b \end{array} \right]$$

5. Jump back to 1

- Let's try on the Wyndor example
- Matrix computations in Julia:
- Create a matrix:
 - $A = [1 \ 0; 0 \ 2; 3 \ 2]$
 - $B = [0 \ 2 \ 0; 0 \ 0 \ 1; 1 \ 2 \ 3]$
- Invert a matrix:
 - $\text{Binv} = \text{inv}(B)$
- Multiply two matrices:
 - $\text{res} = \text{Binv} * A$
- Fractions in Julia:
 - $M = [1 // 2 \ 2//3; 4 \ 4//9]$

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \\ 3 & 2 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ 0 & 2 & 3 \end{bmatrix}$$

$$M = \begin{bmatrix} \frac{1}{2} & \frac{2}{3} \\ 4 & \frac{4}{9} \end{bmatrix}$$

Order of basic variables decides order in B and c_B



$$[A, I] = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \end{bmatrix} \quad [c, 0] = [\ 3 \ 5 \ 0 \ 0 \ 0 \]$$

● ● ● ● ●

Iteration	Basic Variable	Eq.	Z	Coefficient of:					Right Side
				x_1	x_2	x_3	x_4	x_5	
0	Z	(0)	1	-3	-5	0	0	0	0
	x_3	(1)	0	1	0	1	0	0	4
	x_4	(2)	0	0	2	0	1	0	12
	x_5	(3)	0	3	2	0	0	1	18
1	Z	(0)	1	-3	0	0	$\frac{5}{2}$	0	30
	x_3	(1)	0	1	0	1	0	0	4
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_5	(3)	0	3	0	0	-1	1	6

$$x_B = \begin{bmatrix} x_3 \\ x_2 \\ x_5 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 2 & 1 \end{bmatrix}, c_B = [\ 0 \ 5 \ 0 \]$$

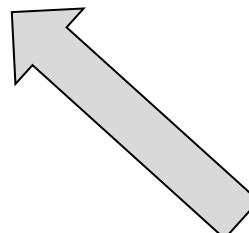
● ● ● ● ●

Revised simplex = simplex algorithm on matrix form

Simplex algorithm on matrix form

1. Optimality test: no negative numbers in row zero \Rightarrow we are done.
2. Choose incoming basic variable (most negative in row 0).
3. Choose leaving basic variable (min ratio-test).
4. Compute new tableau using:

$$\left[\begin{array}{c|cc} c_B B^{-1} A - c & c_B B^{-1} & c_B B^{-1} b \\ B^{-1} A & B^{-1} & B^{-1} b \end{array} \right]$$



5. Jump back to 1

Optimization: We do not need to compute all of $B^{-1}A$. We just a single column from this matrix (correpsonding to the incoming variable).

- Important for computer implementation.
- Not important when computing by hand (if you have a computer to do the matrix computations).

Avoid computing B^{-1}

- We need to compute B^{-1} in every iteration. This takes time.
- We can compute B_{new}^{-1} from B_{old}^{-1} (B^{-1} from the previous iteration):

Avoid computing B^{-1}

- x_k : incoming basic variable.
- a'_{ik} : Coefficient of variable x_k in the i th constraint of the current tableau.
- r : Constraint # for variable leaving the basis.
- E : Identity matrix apart from column r that is replaced by:

$$\eta = \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_m \end{bmatrix}, \text{ hvor } \eta_i = \begin{cases} -\frac{a'_{ik}}{a'_{rk}} & \text{if } i \neq r \\ -\frac{1}{a'_{rk}} & \text{if } i = r \end{cases}$$

- $B_{new}^{-1} = E B_{old}^{-1}$ (so we avoid inversion and does a multiplication instead)
- The matrix simplex method along with methods like the above are used in commercial LP solvers.

Why simplex method on matrix form?

- Basically the same algorithm as normal simplex
- Not a lot easier for hand computations.
- Easier for the computer
 - We do not have to compute entire tableau in each iteration
 - Matrix operations are highly optimized
- Matrix form gives us more insight into how Simplex is working. We will build on this in the next lecture.

Two drawbacks of the simplex algorithm

- Degeneracy.
- Worst case exponential running time.

Degenerate basic solution

m =number of functional constraints

Every basic solution has m basic variables, the other variables are non-basic.
Non-basic variables are set to 0.

We find the value of the basic variables by solving a system of linear equations with m constraints and m variables

A feasible basic solution is a basic solution with all basic variables ≥ 0 .
A feasible basic solution is **degenerate** if one of the basic variables has value 0.

Example of degenerate basic solutions

Wyndor with an extra constraint:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

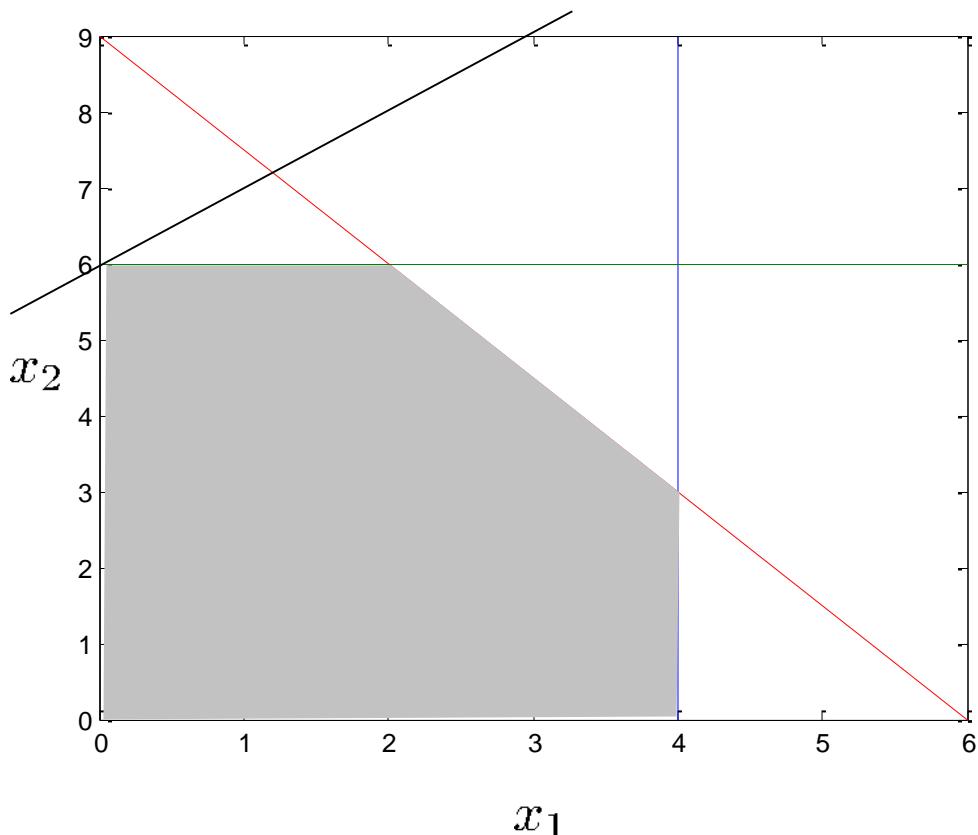
$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$-x_1 + x_2 \leq 6$$

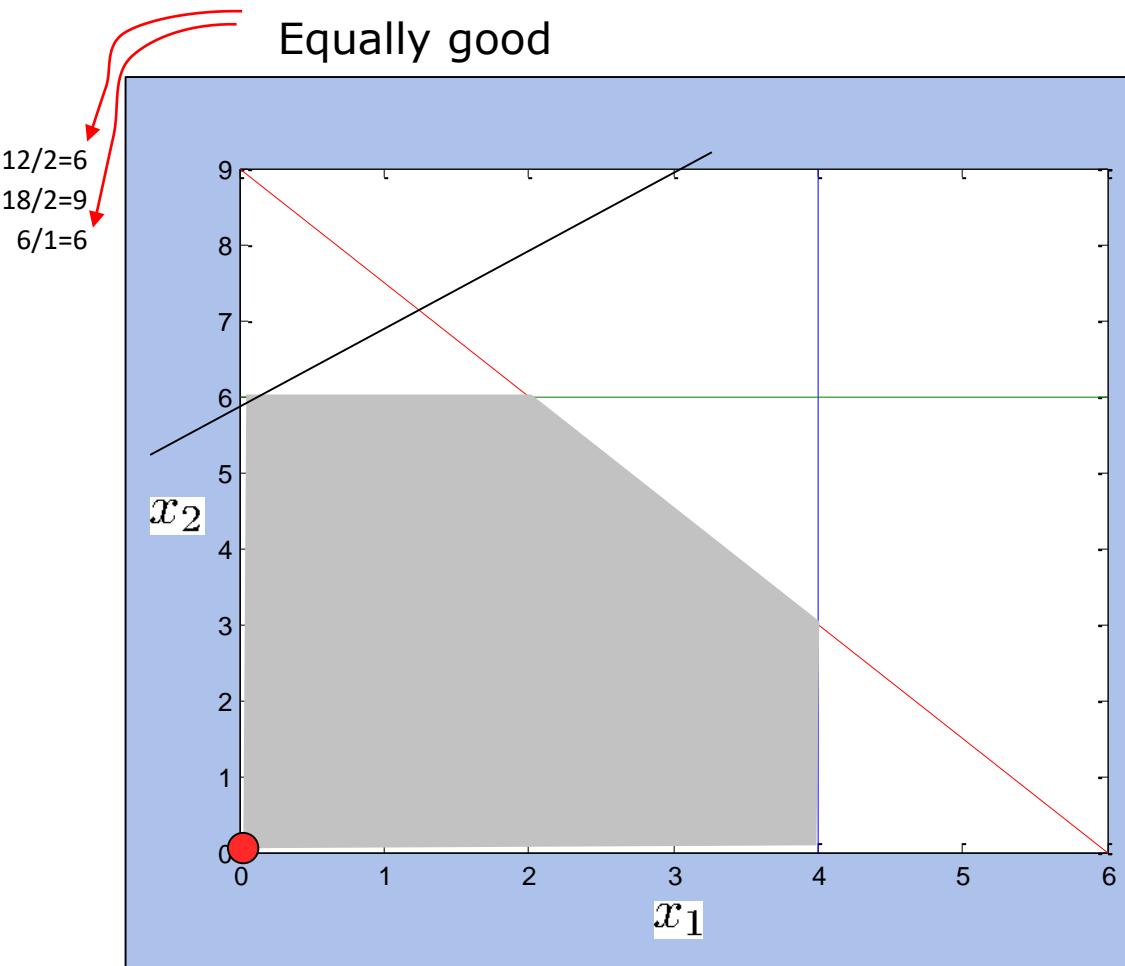
and

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



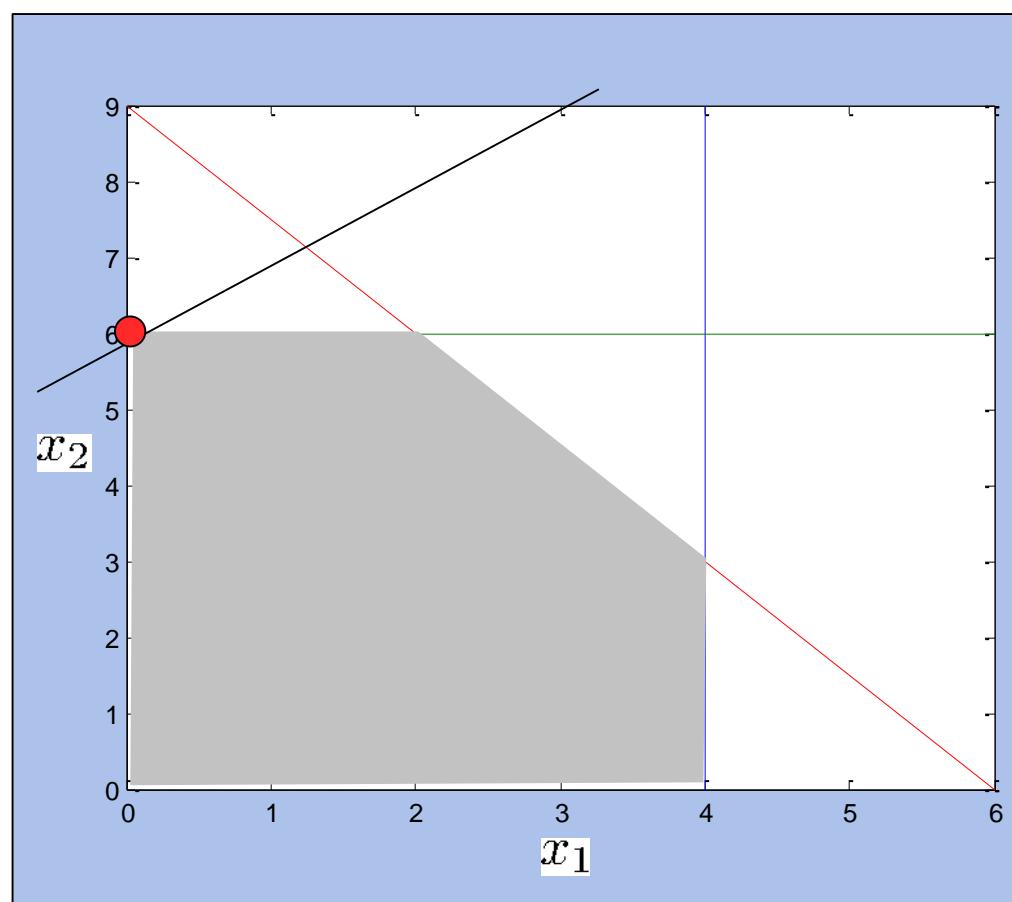
Example of degenerate basic solutions

B.v.	EQ	Z	x1	x2	x3	x4	x5	x6	RHS
Z	0	1	-3	-5	0	0	0	0	0
x3	1	0	1	0	1	0	0	0	4
x4	2	0	0	2	0	1	0	0	12
x5	3	0	3	2	0	0	1	0	18
x6	4	0	-1	1	0	0	0	1	6



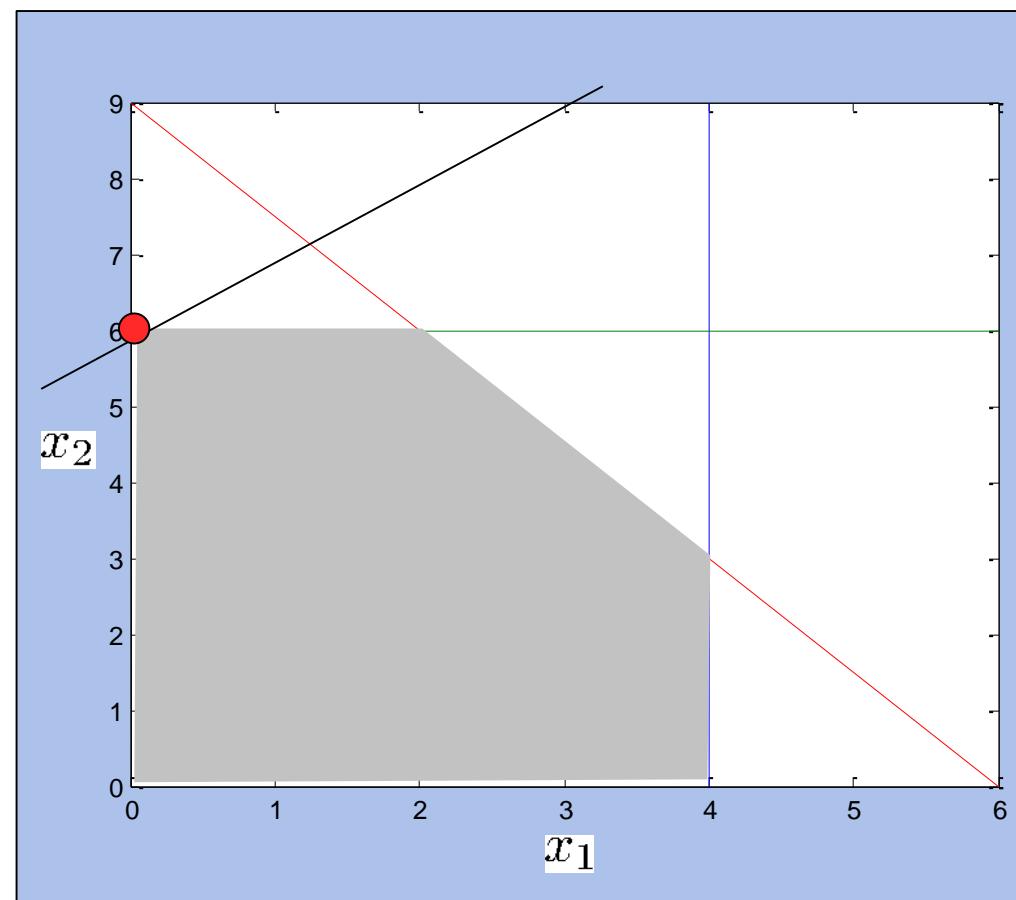
Example of degenerate basic solutions

B.v.	EQ	Z	x1	x2	x3	x4	x5	x6	RHS
Z	0	1	-3	-5	0	0	0	0	0
x3	1	0	1	0	1	0	0	0	4
x4	2	0	0	2	0	1	0	0	12
x5	3	0	3	2	0	0	1	0	18
x6	4	0	-1	1	0	0	0	1	6
Z	0	1	-8	0	0	0	0	5	30
x3	1	0	1	0	1	0	0	0	4
x4	2	0	2	0	0	1	0	-2	0
x5	3	0	5	0	0	0	1	-2	6
x2	4	0	-1	1	0	0	0	1	6



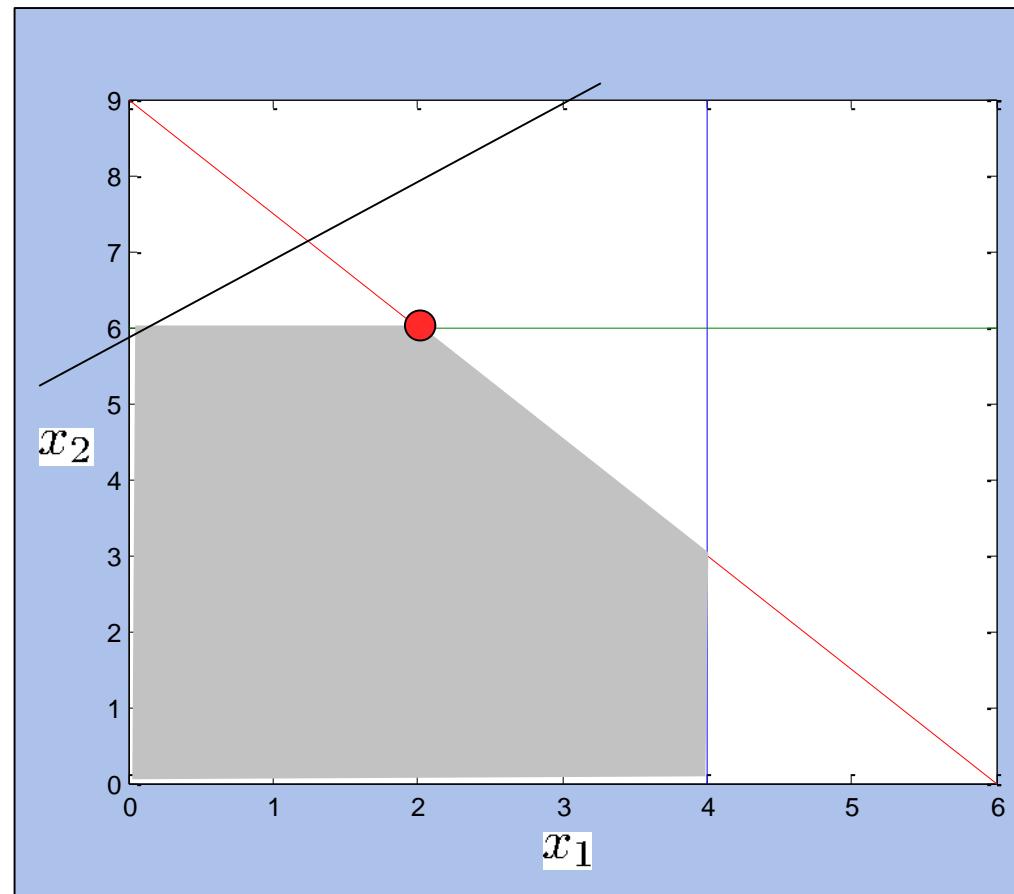
Example of degenerate basic solutions

B.v.	EQ	Z	x1	x2	x3	x4	x5	x6	RHS
Z	0	1	-3	-5	0	0	0	0	0
x3	1	0	1	0	1	0	0	0	4
x4	2	0	0	2	0	1	0	0	12
x5	3	0	3	2	0	0	1	0	18
x6	4	0	-1	1	0	0	0	1	6
Z	0	1	-8	0	0	0	0	5	30
x3	1	0	1	0	1	0	0	0	4
x4	2	0	2	0	0	1	0	-2	0
x5	3	0	5	0	0	0	1	-2	6
x2	4	0	-1	1	0	0	0	1	6
Z	0	1	0	0	0	4	0	-3	30
x3	1	0	0	0	1	-0.5	0	1	4
x1	2	0	1	0	0	0.5	0	-1	0
x5	3	0	0	0	0	-2.5	1	3	6
x2	4	0	0	1	0	0.5	0	0	6



Example of degenerate basic solutions

B.v.	EQ	Z	x1	x2	x3	x4	x5	x6	RHS
Z	0	1	-3	-5	0	0	0	0	0
x3	1	0	1	0	1	0	0	0	4
x4	2	0	0	2	0	1	0	0	12
x5	3	0	3	2	0	0	1	0	18
x6	4	0	-1	1	0	0	0	1	6
Z	0	1	-8	0	0	0	0	5	30
x3	1	0	1	0	1	0	0	0	4
x4	2	0	2	0	0	1	0	-2	0
x5	3	0	5	0	0	0	1	-2	6
x2	4	0	-1	1	0	0	0	1	6
Z	0	1	0	0	0	4	0	-3	30
x3	1	0	0	0	1	-0.5	0	1	4
x1	2	0	1	0	0	0.5	0	-1	0
x5	3	0	0	0	0	-2.5	1	3	6
x2	4	0	0	1	0	0.5	0	0	6
Z	0	1	0	0	0	1.5	1	0	36
x3	1	0	0	0	1	0.33	-0.3	0	2
x1	2	0	1	0	0	-0.3	0.33	0	2
x6	3	0	0	0	0	-0.8	0.33	1	2
x2	4	0	0	1	0	0.5	0	0	6



Bland's rule

In the worst case the simplex algorithm can cycle between different basic feasible solutions that all represent the same corner point. This can go on for ever.

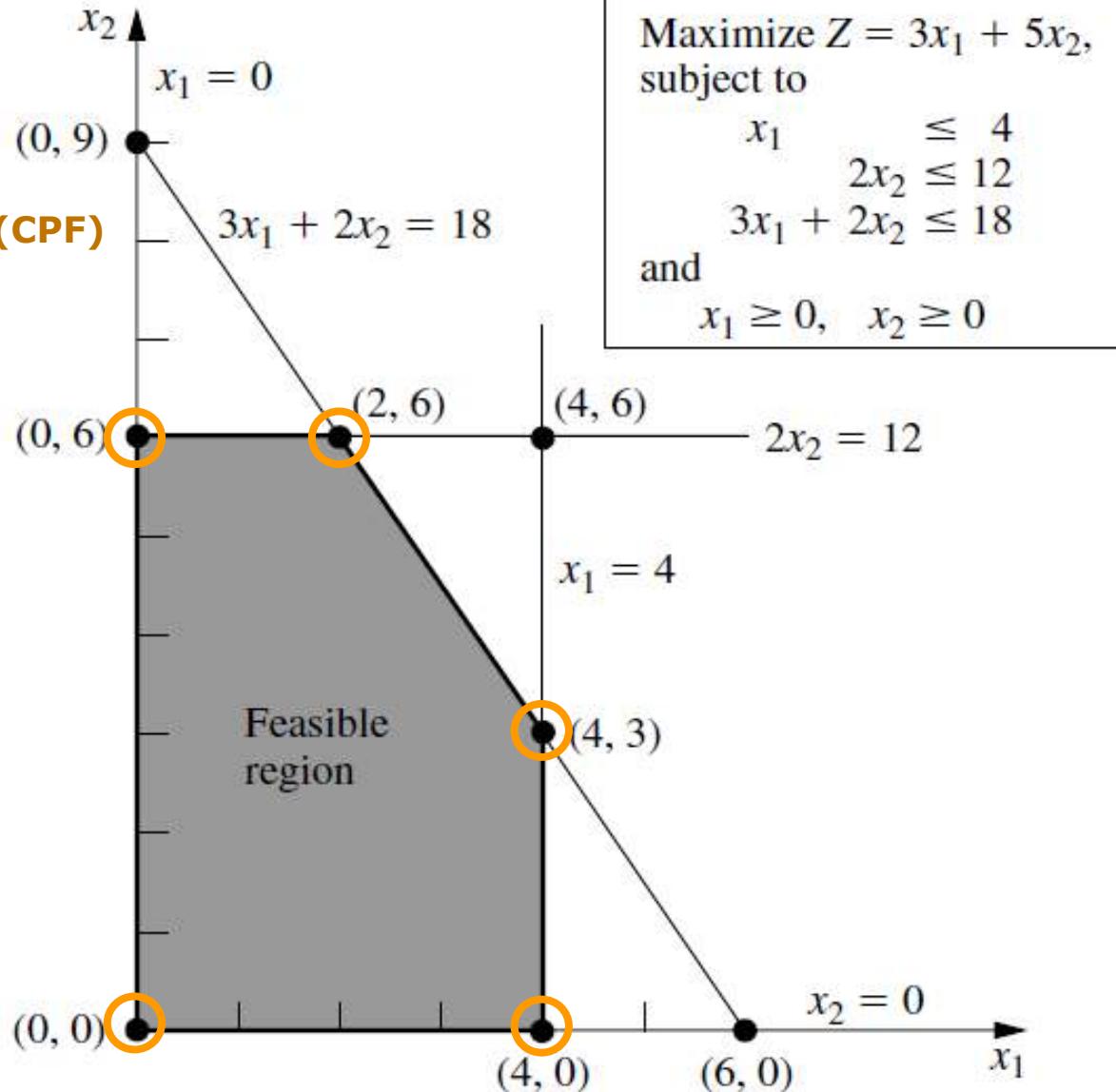
It is not going to be a problem in the instances we solve by hand.

Bland's rule was designed to avoid this

- **Variable to enter the basis:** Look at all the variables that has a negative coefficient in row zero. Choose the one with lowest **index**.
- **Variable to leave the basis:** If two or more variables are equally good according to the minimum ratio test, then choose the one with lowest **index**. Otherwise proceed as in the standard Simplex algorithm.

•Complexity of the simplex method

Corner point feasible solutions (CPF)



Simplex algorithm jumps from CPF to CPF until the optimal CPF is reached.

Complexity of the simplex method

- There is a finite (but large) number of corner points.
- n : number of variables.
- m : number of functional constraints
- $n + m$: total number of constraints(functional and non-negativity).
- Upper bound for number of feasible corner points: In how many ways can we pick n out of the $n + m$ constraints.

$$\binom{m+n}{n} = \frac{(m+n)!}{m!n!}$$

- Example: if $m = 50, n = 50$
- Example: Upper bound on number of corner points:

$$\frac{100!}{(50!)^2} \approx 10^{29}$$

Complexity of the simplex method

Klee and Minty (1975)

max

$$2^{n-1}x_1 + 2^{n-2}x_2 + \dots + 2x_{n-1} + 1x_n$$

hvor

$$1x_1 + \dots + \dots + \dots + \leq 5$$

$$4x_1 + 1x_2 + \dots + \dots + \leq 5^2$$

$$8x_1 + 4x_2 + 1x_3 + \dots + \leq 5^3$$

$$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$$

$$2^n x_1 + 2^{n-1} x_2 + \dots + 4x_{n-1} + 1x_n \leq 5^n$$

$$x_i \geq 0, i = 1, \dots, n$$

- n variables, n constraints, 2^n corner points
- If simplex starts in $(0, 0, \dots, 0)$ it visits all corner points
- Optimal solution $(0, 0, \dots, 5^n)$. $Z^* = 5^n$

Klee Minty ($n = 2$)

$$\max 2x_1 + x_2$$

subject to

$$x_1 \leq 5$$

$$4x_1 + x_2 \leq 25$$

$$x_1, x_2 \geq 0$$

b.v.	eq.	z	x1	x2	x3	x4	RHS
z	0	1	-2.00	-1.00	0.00	0.00	0.00
x3	1	0	1.00	0.00	1.00	0.00	5.00
x4	2	0	4.00	1.00	0.00	1.00	25.00

b.v.	eq.	z	x1	x2	x3	x4	RHS
z	0	1	0.00	-1.00	2.00	0.00	10.00
x1	1	0	1.00	0.00	1.00	0.00	5.00
x4	2	0	0.00	1.00	-4.00	1.00	5.00

b.v.	eq.	z	x1	x2	x3	x4	RHS
z	0	1	0.00	0.00	-2.00	1.00	15.00
x1	1	0	1.00	0.00	1.00	0.00	5.00
x2	2	0	0.00	1.00	-4.00	1.00	5.00

b.v.	eq.	z	x1	x2	x3	x4	RHS
z	0	1	2.00	0.00	0.00	1.00	25.00
x3	1	0	1.00	0.00	1.00	0.00	5.00
x2	2	0	4.00	1.00	0.00	1.00	25.00

Klee Minty ($n = 3$)

$$\max 4x_1 + 2x_2 + x_3$$

subject to

$$x_1 \leq 5$$

$$4x_1 + x_2 \leq 25$$

$$8x_1 + 4x_2 + x_3 \leq 125$$

$$x_1, x_2, x_3 \geq 0$$

8 iterations. for $n = 4$ we would need 16 iterations and so on.

b.v.		eq.		z		x1	x2	x3	x4	x5	x6		RHS
z		0		1		-4.00	-2.00	-1.00	0.00	0.00	0.00		0.00
x4		1		0		1.00	0.00	0.00	1.00	0.00	0.00		5.00
x5		2		0		4.00	1.00	0.00	0.00	1.00	0.00		25.00
x6		3		0		8.00	4.00	1.00	0.00	0.00	1.00		125.00
b.v.		eq.		z		x1	x2	x3	x4	x5	x6		RHS
z		0		1		0.00	-2.00	-1.00	4.00	0.00	0.00		20.00
x1		1		0		1.00	0.00	0.00	1.00	0.00	0.00		5.00
x5		2		0		0.00	1.00	0.00	-4.00	1.00	0.00		5.00
x6		3		0		0.00	4.00	1.00	-8.00	0.00	1.00		85.00
b.v.		eq.		z		x1	x2	x3	x4	x5	x6		RHS
z		0		1		0.00	0.00	-1.00	-4.00	2.00	0.00		30.00
x1		1		0		1.00	0.00	0.00	1.00	0.00	0.00		5.00
x2		2		0		0.00	1.00	0.00	-4.00	1.00	0.00		5.00
x6		3		0		0.00	0.00	1.00	8.00	-4.00	1.00		65.00
b.v.		eq.		z		x1	x2	x3	x4	x5	x6		RHS
z		0		1		4.00	0.00	-1.00	0.00	2.00	0.00		50.00
x4		1		0		1.00	0.00	0.00	1.00	0.00	0.00		5.00
x2		2		0		4.00	1.00	0.00	0.00	1.00	0.00		25.00
x6		3		0		-8.00	0.00	1.00	0.00	-4.00	1.00		25.00
b.v.		eq.		z		x1	x2	x3	x4	x5	x6		RHS
z		0		1		-4.00	0.00	0.00	0.00	-2.00	1.00		75.00
x4		1		0		1.00	0.00	0.00	1.00	0.00	0.00		5.00
x2		2		0		4.00	1.00	0.00	0.00	1.00	0.00		25.00
x3		3		0		-8.00	0.00	1.00	0.00	-4.00	1.00		25.00
b.v.		eq.		z		x1	x2	x3	x4	x5	x6		RHS
z		0		1		0.00	0.00	0.00	4.00	-2.00	1.00		95.00
x1		1		0		1.00	0.00	0.00	1.00	0.00	0.00		5.00
x2		2		0		0.00	1.00	0.00	-4.00	1.00	0.00		5.00
x3		3		0		0.00	0.00	1.00	8.00	-4.00	1.00		65.00
b.v.		eq.		z		x1	x2	x3	x4	x5	x6		RHS
z		0		1		0.00	2.00	0.00	-4.00	0.00	1.00		105.00
x1		1		0		1.00	0.00	0.00	1.00	0.00	0.00		5.00
x5		2		0		0.00	1.00	0.00	-4.00	1.00	0.00		5.00
x3		3		0		0.00	4.00	1.00	-8.00	0.00	1.00		85.00
b.v.		eq.		z		x1	x2	x3	x4	x5	x6		RHS
z		0		1		4.00	2.00	0.00	0.00	0.00	1.00		125.00
x4		1		0		1.00	0.00	0.00	1.00	0.00	0.00		5.00
x5		2		0		4.00	1.00	0.00	0.00	1.00	0.00		25.00
x3		3		0		8.00	4.00	1.00	0.00	0.00	1.00		125.00

Complexity of the simplex method

- As we just saw: There are examples with n constraints and n variables that needs 2^n iterations.
- That's not good: if that was the performance to expect for any LP then we could not solve very large LPs.
- Luckily it turns out that simplex in practise is much much faster. Worst case (almost?) only occur on constructed examples.
- There are other methods (not simplex based) that can approximate the optimal solution to any LP in polynomial time (much faster than 2^n)

Complexity of the simplex method

- As we just saw: There are examples with n constraints and n variables that needs 2^n iterations.
- That's not good: if that was the performance to expect for any LP then we could not solve very large LPs.
- Luckily it turns out that simplex in practise is much much faster. Worst case (almost?) only occur on constructed examples.
- There are other methods (not simplex based) that can approximate the optimal solution to any LP in polynomial time (much faster than 2^n)

name	variables	constraints	Non-zeros		Iterations	Time (s)
CRE-B	72447	9648	256095		20620	0.90
OSA-60	232966	10280	1397793		3703	0.78
KEN-18	154699	105127	358171		99423	9.77

Project

- Handin Sunday the 13th of October
 - Consider using Lyx (www.lyx.org) or Latex to write the report. Once you learned to use one of these tools it is a convenient way of writing math. There are tutorial on YouTube.
- Explain solution methods. I am not too interested in the results, but I would like to see that you have understood the material. Make sure to show this in the report.
- Work in groups of 2-3. Write to me if you do not have a group.
- Hand in on DTU inside (PDF file). Write the name and study number of each participant and use DTU inside's functionality for adding multiple authors.
- You are allowed to talk to other groups about the assignment but the same text, figures, models, tables and so on must not be present in more than one hand-in. Likewise, you are **NOT** allowed to copy text, figures, models, tables , etc. from handins from earlier years, solutions, documents from the internet etc.
- Violations are reported to DTU
- You need to pass the project to attend the exam.
- No questions in first exercise hour (Do normal exercises)
- I answer questions in the second exercise hour.
- Beer prizes for the best solutions!

Now it is exercise time!

Duality and sensitivity analysis

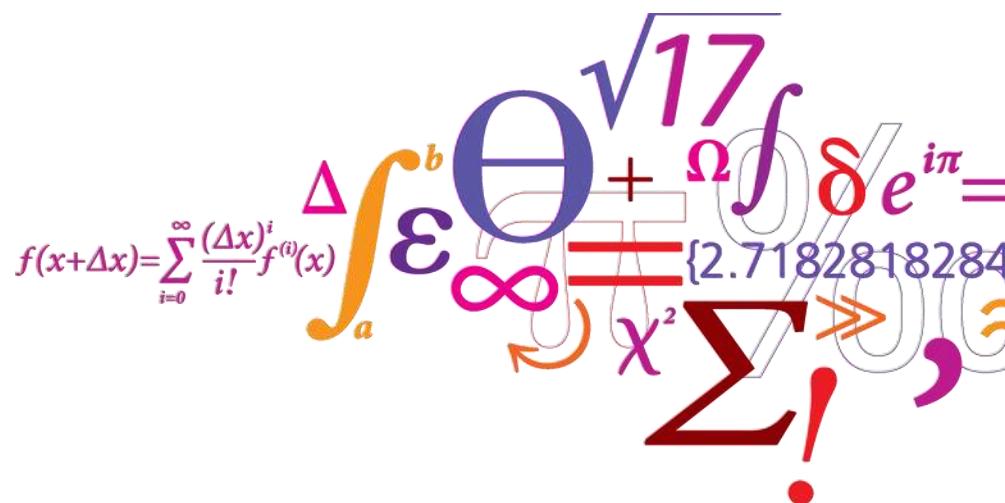
6.1. Duality

6.2. Economic interpretation

6.3. Primal-Dual relations

6.4. Primal not on standard form

7.1-7.2. Sensitivity analysis

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$


6.1 Duality - motivation

Wyndor example:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

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

Can we easily find an upper bound for the objective value.

- Multiply third constraint by 3:

$$9x_1 + 6x_2 \leq 54 \text{ so } Z = 3x_1 + 5x_2 \leq 9x_1 + 6x_2 \leq 54$$

- Multiply first constraint by 3 and second by 5/2 and add together:

$$3x_1 + 5x_2 \leq 3 \cdot 4 + \frac{5}{2} \cdot 12 = 42 \text{ so } Z \leq 42$$

- How good of an upper bound can we find in this way?

Wyndor example:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4 \quad y_1$$

$$2x_2 \leq 12 \quad y_2$$

$$3x_1 + 2x_2 \leq 18 \quad y_3$$

$$x_1 \geq 0, \quad x_2 \geq 0$$

Let's be systematic. We multiply each of the constraints by y_1, y_2, y_3 and add together

$$y_1(x_1) + 2y_2(x_2) + y_3(3x_1 + 2x_2) \leq 4y_1 + 12y_2 + 18y_3$$

$$(y_1 + 3y_3)x_1 + (2y_2 + 2y_3)x_2 \leq 4y_1 + 12y_2 + 18y_3$$



Coefficient for x_1 should be ≥ 3 , coefficient for x_2 should be ≥ 5 and we wish to make the right hand side as small as possible.

$$\min 4y_1 + 12y_2 + 18y_3$$

subject to

$$\bullet \quad y_1 + 3y_3 \geq 3$$

$$\bullet \quad 2y_2 + 2y_3 \geq 5$$

$$y_1, y_2, y_3 \geq 0$$

The dual problem

6.1 Duality

Primal Problem

$$\text{Maximize} \quad Z = \sum_{j=1}^n c_j x_j,$$

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i, \quad \text{for } i = 1, 2, \dots, m$$

and

$$x_j \geq 0, \quad \text{for } j = 1, 2, \dots, n.$$

Dual Problem

$$\text{Minimize} \quad W = \sum_{i=1}^m b_i y_i,$$

subject to

$$\sum_{i=1}^m a_{ij} y_i \geq c_j, \quad \text{for } j = 1, 2, \dots, n$$

and

$$y_i \geq 0, \quad \text{for } i = 1, 2, \dots, m.$$

Max -> Min

Primal coefficients for the objective -> Right hand side coefficients for dual

Right hand side coefficients for primal -> dual coefficients for the objective

m constraints and n variables -> n constraints and m variables in dual problem

6.1 Duality

Primal Problem

$$\text{Maximize} \quad Z = \sum_{j=1}^n c_j x_j,$$

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i, \quad \text{for } i = 1, 2, \dots, m$$

and

$$x_j \geq 0, \quad \text{for } j = 1, 2, \dots, n.$$

Dual Problem

$$\text{Minimize} \quad W = \sum_{i=1}^m b_i y_i,$$

subject to

$$\sum_{i=1}^m a_{ij} y_i \geq c_j, \quad \text{for } j = 1, 2, \dots, n$$

and

$$y_i \geq 0, \quad \text{for } i = 1, 2, \dots, m.$$

- When you want to dualise a problem that is on the standard form just use the boxes above
- What we did on the first two slides was to show where the dual problem comes from. To complex to do this every time we need to construct the dual problem
- Dual to the dual problem = the primal problem!

6.1 Duality

Primal Problem

$$\text{Maximize} \quad Z = \sum_{j=1}^n c_j x_j,$$

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i, \quad \text{for } i = 1, 2, \dots, m$$

and

$$x_j \geq 0, \quad \text{for } j = 1, 2, \dots, n.$$

Dual Problem

$$\text{Minimize} \quad W = \sum_{i=1}^m b_i y_i,$$

subject to

$$\sum_{i=1}^m a_{ij} y_i \geq c_j, \quad \text{for } j = 1, 2, \dots, n$$

and

$$y_i \geq 0, \quad \text{for } i = 1, 2, \dots, m.$$

- Written in matrix form:

$$\text{Maximize} \quad Z = \mathbf{c}\mathbf{x},$$

subject to

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

and

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

$$\text{Minimize} \quad \mathbf{W} = \mathbf{y}\mathbf{b},$$

subject to

$$\mathbf{y}\mathbf{A} \geq \mathbf{c}$$

and

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

- y and c are row vectors (x and b are column vectors).

Some simple lemmas



Lemma 1 Let α and β be two column vectors with n elements. Let $y \geq 0$ be a row vector with n elements. If

$$\alpha \leq \beta$$

then

$$y\alpha \leq y\beta$$

Lemma 2 Let α and β be two **row** vectors with n elements. Let $x \geq 0$ be a **column** vector with n elements. If

$$\alpha \leq \beta$$

then

$$\alpha x \leq \beta x$$

Weak duality (I)

Given an LP (the primal) and its dual

Primal Problem

$$\text{Maximize} \quad Z = \mathbf{c}\mathbf{x},$$

subject to

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

and

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

Dual Problem

$$\text{Minimize} \quad W = \mathbf{y}\mathbf{b},$$

subject to

$$\mathbf{y}\mathbf{A} \geq \mathbf{c}$$

and

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

For every feasible solution x to the primal and every feasible solution y to the dual the weak duality theorem ensures:

$$cx \leq yb$$

Weak duality (II)



Given an LP (the primal) and its dual

Primal Problem

$$\text{Maximize} \quad Z = \mathbf{c}\mathbf{x},$$

subject to

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

and

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

Dual Problem

$$\text{Minimize} \quad W = \mathbf{y}\mathbf{b},$$

subject to

$$\mathbf{y}\mathbf{A} \geq \mathbf{c}$$

and

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

For every feasible solution x to the primal and every feasible solution y to the dual the weak duality theorem ensures:

$$cx \leq yb$$

Proof:

From dual feasible and primal feasible: $c \leq yA$ and $x \geq 0$. From lemma 2: $cx \leq yAx$

From primal feasible and dual feasible: $Ax \leq b$ and $y \geq 0$. From lemma 1: $yAx \leq yb$

Altogether: $cx \leq yAx \leq yb$ or just: $cx \leq yb$ \square

Weak duality (III)

Given an LP (the primal) and its dual

Primal Problem

$$\text{Maximize} \quad Z = \mathbf{c}\mathbf{x},$$

subject to

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

and

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

Dual Problem

$$\text{Minimize} \quad \mathbf{W} = \mathbf{y}\mathbf{b},$$

subject to

$$\mathbf{y}\mathbf{A} \geq \mathbf{c}$$

and

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

For every feasible solution x to the primal and every feasible solution y to the dual the weak duality theorem ensures:

$$cx \leq yb$$

Consequence of weak duality: If x is primal feasible, y is dual feasible and $cx = by$ then x is optimal for the primal and y is optimal for the dual.

Strong duality

For every feasible LP with a finite optimal primal solution x^* there exist a solution y^* to the dual problem such that

$$cx^* = y^*b$$

Proof We use fundamental insight from last week, so let's recap that.

Dual Problem

Minimize	$\mathbf{W} = \mathbf{y}\mathbf{b},$
subject to	
$\mathbf{y}\mathbf{A} \geq \mathbf{c}$	
and	
$\mathbf{y} \geq \mathbf{0}.$	

A fundamental insight (from last week)

Start:

$$\left[\begin{array}{c|c|c} -c & 0 & 0 \\ A & I & b \end{array} \right]$$

We can compute the tableau for any choice of basis variables (basisvariable defines B and c_B)

$$\left[\begin{array}{c|c|c} c_B B^{-1} A - c & c_B B^{-1} & c_B B^{-1} b \\ B^{-1} A & B^{-1} & B^{-1} b \end{array} \right]$$

In the final iteration we define

$$y^* = c_B B^{-1}$$

$$S^* = B^{-1}$$

Fundamental insight: The last tableau can be found just using y^* , S^* and the information from the initial tableau.

$$\left[\begin{array}{c|c|c} y^* A - c & y^* & y^* b \\ S^* A & S^* & S^* b \end{array} \right]$$

TABLE 4.8 Complete set of simplex tableaux for the Wyndor Glass Co. problem

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

Iteration	Basic Variable	Eq.	Coefficient of:					Right Side
			Z	x_1	x_2	x_3	x_4	
0	Z	(0)	1	-3	-5	0	0	0
	x_3	(1)	0	1	0	1	0	4
	x_4	(2)	0	0	2	0	1	0
	x_5	(3)	0	3	2	0	0	18
1	Z	(0)	1	-3	0	0	$\frac{5}{2}$	0
	x_3	(1)	0	1	0	1	0	4
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	6
	x_5	(3)	0	3	0	0	-1	6
2	Z	(0)	1	0	0	0	$\frac{3}{2}$	1
	x_3	(1)	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0
	x_1	(3)	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$

we will use $y^* = c_B B^{-1}$ as our solution to the dual and check if it is feasible and optimal.

$$\left[\begin{array}{c|c|c} y^* A - c & y^* & y^* b \\ S^* A & S^* & S^* b \end{array} \right]$$

Strong duality (II)

For every feasible LP with a finite optimal primal solution x^* there exist a solution y^* to the dual problem such that

$$cx^* = y^*b$$

Proof (using fundamental insight from last week) use $y^* = c_B B^{-1}$

$$y^*A - c \geq 0$$

$$y^* \geq 0$$

$$Z = y^*b = cx^*$$

Fundamental
insight:

Z	x	x_s	
1	$-c$	0	0

1	$y^*A - c$	y^*	y^*b
	S^*A	S^*	S^*b

First tableau

Last tableau

Dual Problem

Minimize $\mathbf{W} = \mathbf{yb}$,
subject to
 $\mathbf{yA} \geq \mathbf{c}$
and
 $\mathbf{y} \geq \mathbf{0}$.

Strong duality (III)

Z x x_s

1	$-c$	0	0
	A	I	b

First tableau

1	$y^*A - c$	y^*	y^*b
	S^*A	S^*	S^*b

Last tableau

- When we have solved the primal problem we also get the optimal solution to the dual "for free"
- The solution to the dual problem: $y^* = c_B B^{-1}$ is read in row 0 in the final tableau under the slack variables.

TABLE 4.8 Complete set of simplex tableaux for the Wyndor Glass Co. problem

Iteration	Basic Variable	Eq.	Coefficient of:						Right Side
			Z	x_1	x_2	x_3	x_4	x_5	
0	Z	(0)	1	-3	-5	0	0	0	0
	x_3	(1)	0	1	0	1	0	0	4
	x_4	(2)	0	0	2	0	1	0	12
	x_5	(3)	0	3	2	0	0	1	18
1	Z	(0)	1	-3	0	0	$\frac{5}{2}$	0	30
	x_3	(1)	0	1	0	1	0	0	4
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_5	(3)	0	3	0	0	-1	1	6
2	Z	(0)	1	0	0	0	$\frac{3}{2}$	1	36
	x_3	(1)	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	2
	x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
	x_1	(3)	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	2

Strong duality (IV)

Primal		Dual
Optimal solution	\Leftrightarrow	Optimal solution
Unbounded solution	\Rightarrow	No feasible solution
No feasible solution	\Rightarrow	No feasible solution or unbounded solution
$\begin{aligned} & \max x_1 \\ \text{subject to} \\ & -x_1 \leq 2 \\ & x_1 \geq 0 \end{aligned}$	\Rightarrow	$\begin{aligned} & \min 2y_1 \\ \text{subject to} \\ & -y_1 \geq 1 \\ & y_1 \geq 0 \end{aligned}$
Unbounded		No Feasible solution
$\begin{aligned} & \max x_1 + x_2 \\ \text{subject to} \\ & x_1 - x_2 \leq -1 \\ & -x_1 + x_2 \leq -1 \\ & x_1, x_2 \geq 0 \end{aligned}$	\Rightarrow	$\begin{aligned} & \min -y_1 - y_2 \\ \text{subject to} \\ & y_1 - y_2 \geq 1 \\ & -y_2 + y_2 \geq 1 \\ & y_1, y_2 \geq 0 \end{aligned}$
No Feasible solution		No Feasible solution

Economical interpretation of dual variables

y_i : dual variable, “shadow price”, “marginal price”

The dual variable y_i gives the value per unit of resource i . If the primal problem

$$\max cx$$

subject to

$$\begin{aligned} Ax &\leq b \\ x &\geq 0 \end{aligned}$$

has optimal objective value z^* . Then

$$\max cx$$

subject to

$$\begin{aligned} Ax &\leq b + \epsilon \\ x &\geq 0 \end{aligned}$$

has optimal objective $z^* + y\epsilon$. The change ϵ must be so small that the optimal basis is not changed (more on that later).

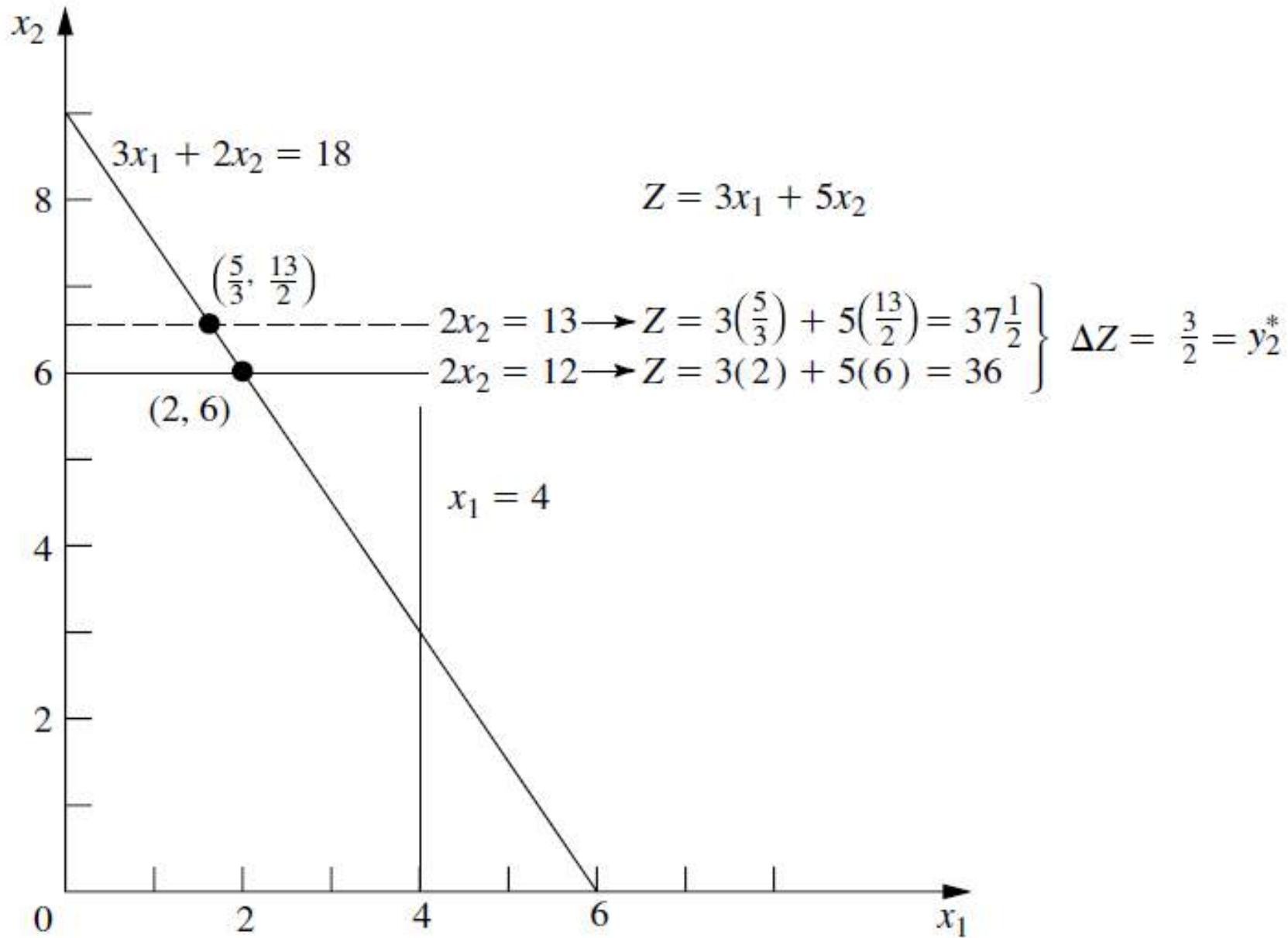
Fundamental insight:
First tableau

Z	x	x_s	
1	$-c$	0	0
	A	I	b
1	$y^*A - c$	y^*	y^*b
	S^*A	S^*	S^*b

Last tableau

TABLE 4.8 Complete set of simplex tableaux for the Wyndor Glass Co. problem

Basic		Coefficient of:					Right Side
		Z	x_1	x_2	x_3	x_4	
	$\max Z = 3x_1 + 5x_2$	1	-3	-5	0	0	0
	subject to	0	1	0	1	0	4
	$x_1 \leq 4$	0	0	2	0	1	12
		0	3	2	0	0	18
	$2x_2 \leq 12$	1.5					
	$3x_1 + 2x_2 \leq 18$	1	1	-3	0	$\frac{5}{2}$	30
		0	1	0	1	0	4
and		0	0	1	0	$\frac{1}{2}$	6
	$x_1 \geq 0, x_2 \geq 0$	0	3	0	0	-1	6
	Z	(0)	1	0	0	$\frac{3}{2}$	36
2	x_3	(1)	0	0	0	1	$-\frac{1}{3}$
	x_2	(2)	0	0	1	0	$\frac{1}{2}$
	x_1	(3)	0	1	0	0	$-\frac{1}{3}$



Complementary slackness

Primal Problem

$$\text{Maximize} \quad Z = \sum_{j=1}^n c_j x_j,$$

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i, \quad \text{for } i = 1, 2, \dots, m$$

and

$$x_j \geq 0, \quad \text{for } j = 1, 2, \dots, n.$$

Dual Problem

$$\text{Minimize} \quad W = \sum_{i=1}^m b_i y_i,$$

subject to

$$\sum_{i=1}^m a_{ij} y_i \geq c_j, \quad \text{for } j = 1, 2, \dots, n$$

and

$$y_i \geq 0, \quad \text{for } i = 1, 2, \dots, m.$$

Assume that x is a feasible solution to the primal problem and y is a feasible solution to the dual problem. Then, a necessary and sufficient condition for x and y being optimal are:

$$\sum_{j=1}^n a_{ij} x_j = b_i \text{ or } y_i = 0 \quad \forall i = 1 \dots m \tag{1}$$

and

$$\sum_{i=1}^m a_{ij} y_i = c_j \text{ or } x_j = 0 \quad \forall j = 1 \dots n \tag{2}$$

Complementary slackness

Assume that x is a feasible solution to the primal problem and y is a feasible solution to the dual problem. Then, a necessary and sufficient condition for x and y being optimal are:

$$\sum_{j=1}^n a_{ij}x_j = b_i \text{ or } y_i = 0 \quad \forall i = 1 \dots m \quad (1)$$

and

$$\sum_{i=1}^m a_{ij}y_i = c_j \text{ or } x_j = 0 \quad \forall j = 1 \dots n \quad (2)$$

If we can guess feasible solutions x and y and show that they satisfy (1) and (2) then we know that x and y are optimal.

On the other hand the theorem states that all optimale solutions satisfy (1) and (2).

Economical interpretation of Complementary slackness

$$\sum_{j=1}^n a_{ij}x_j < b_i \Rightarrow y_i = 0$$

If resource i is not fully used then we cannot earn more by increasing the amount of resource i (so $y_i = 0$)

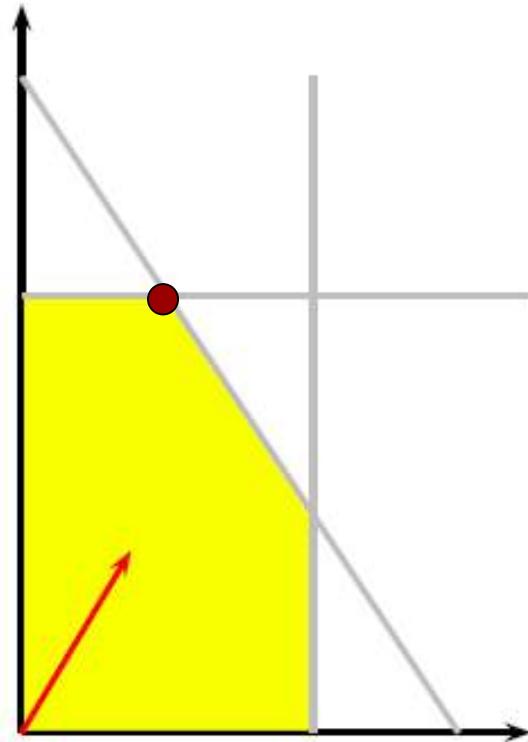
$$\sum_{i=1}^m a_{ij}y_i > c_j \Rightarrow x_j = 0$$

If the cost of producing item j is larger than the price of the item then we should not produce item j (so $x_j = 0$)

Example: Wyndor

$$\begin{aligned} \max \quad & 3x_1 + 5x_2 \\ \text{hvor } & x_1 \leq 4 \quad (y_1) \\ & 2x_2 \leq 12 \quad (y_2) \\ & 3x_1 + 2x_2 \leq 18 \quad (y_3) \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$x^* = (2, 6), y^* = (0, \frac{3}{2}, 1)$$



- Resource 1 is not fully used so $y_1 = 0$
- $y_2 > 0$ so resource 2 is fully used
- $y_3 > 0$ so resource 3 is fully used

Dual for Non-standard form

Slow way: Turn the problem into standard form and then construct the dual

Nonstandard Form	Equivalent Standard Form
Minimize Z	Maximize $(-Z)$
$\sum_{j=1}^n a_{ij}x_j \geq b_i$	$-\sum_{j=1}^n a_{ij}x_j \leq -b_i$
$\sum_{j=1}^n a_{ij}x_j = b_i$	$\sum_{j=1}^n a_{ij}x_j \leq b_i$ and $-\sum_{j=1}^n a_{ij}x_j \leq -b_i$
x_j unconstrained in sign	$x_j^+ - x_j^-$, $x_j^+ \geq 0$, $x_j^- \geq 0$

(table 6.12 in the book)

Example LP:

$$\begin{aligned} & \max 3x_1 + 2x_2 + 5x_3 \\ \text{subject to } & 5x_1 + 3x_2 + x_3 = -8 \quad (y_1) \\ & 4x_1 + 2x_2 + 8x_3 \leq 23 \quad (y_2) \\ & 6x_1 + 7x_2 + 3x_3 \geq 1 \quad (y_3) \\ & x_1 \leq 4 \quad (y_4) \\ & x_3 \geq 0 \\ & x_1, x_2 \in \mathbb{R} \end{aligned}$$

Dual for Non-standard form

Fast way: Use the Sensible, Odd, Bizarre (SOB) rule:

Label	Primal Problem (or Dual Problem)	Dual Problem (or Primal Problem)
	Maximize Z (or W)	Minimize W (or Z)
Sensible	Constraint i :	Variable y_i (or x_i):
Odd	\leq form \leftarrow	$y_i \geq 0$ \rightarrow
Bizarre	$=$ form \leftarrow	Unconstrained \rightarrow
	\geq form \leftarrow	$y'_i \leq 0$ \rightarrow
Sensible	Variable x_j (or y_j):	Constraint j :
Odd	$x_j \geq 0$ \leftarrow	\geq form \rightarrow
Bizarre	Unconstrained \leftarrow	$=$ form \rightarrow
	$x'_j \leq 0$ \leftarrow	\leq form \rightarrow

$$\max 3x_1 + 2x_2 + 5x_3$$

$$\text{s.t. } 5x_1 + 3x_2 + x_3 = -8 \quad (y_1)$$

$$4x_1 + 2x_2 + 8x_3 \leq 23 \quad (y_2)$$

$$6x_1 + 7x_2 + 3x_3 \geq 1 \quad (y_3)$$

$$x_1 \leq 4 \quad (y_4)$$

$$x_3 \geq 0$$

$$x_1, x_2 \in \mathbb{R}$$

$$\min -8y_1 + 23y_2 + y_3 + 4y_4$$

$$\text{s.t. } 5y_1 + 4y_2 + 6y_3 + y_4 = 3 \quad (x_1)$$

$$3y_1 + 2y_2 + 7y_3 = 2 \quad (x_2)$$

$$y_1 + 8y_2 + 3y_3 \geq 5 \quad (x_3)$$

$$y_1 \in \mathbb{R}$$

$$y_2, y_4 \geq 0$$

$$y_3 \leq 0$$

Constructing dual using SOB: "cook book"

Primal problem

$$\begin{aligned} & \max 3x_1 + 2x_2 + 5x_3 \\ \text{subject to } & 5x_1 + 3x_2 + x_3 = -8 \quad (y_1) \\ & 4x_1 + 2x_2 + 8x_3 \leq 23 \quad (y_2) \\ & 6x_1 + 7x_2 + 3x_3 \geq 1 \quad (y_3) \\ & x_1 \leq 4 \quad (y_4) \\ & x_3 \geq 0 \\ & x_1, x_2 \in \mathbb{R} \end{aligned}$$

1. Start by writing the objective function of the dual problem'
 - Objective coefficients are found as the coefficients on the right hand side of the primal
 - "max" in primal => "min" in dual
 - "min" in primal => "max" in dual

Dual problem

$$\min -8y_1 + 23y_2 + y_3 + 4y_4$$

Constructing dual using SOB: "cook book"

Primal problem

$$\begin{aligned} & \max 3x_1 + 2x_2 + 5x_3 \\ \text{subject to } & 5x_1 + 3x_2 + x_3 = -8 \quad (y_1) \\ & 4x_1 + 2x_2 + 8x_3 \leq 23 \quad (y_2) \\ & 6x_1 + 7x_2 + 3x_3 \geq 1 \quad (y_3) \\ & x_1 \leq 4 \quad (y_4) \\ & x_3 \geq 0 \\ & x_1, x_2 \in \mathbb{R} \end{aligned}$$

2. Write constraints for the dual problem. Each variable in the primal results in one constraint in the dual.
- Coefficients of dual constraints are find in the columns of the A matrix for the primal problem
 - Right hand side of constraint is found in the objective function of the primal problem
 - $\leq, =, \geq$ is decided using the domain of the variable (SOB regel)

Dual problem

$$\begin{aligned} & \min -8y_1 + 23y_2 + y_3 + 4y_4 \\ \text{subject to } & 5y_1 + 4y_2 + 6y_3 + y_4 = 3 \quad (x_1) \end{aligned}$$

Constructing dual using SOB: "cook book"

Primal problem

$$\begin{aligned} & \max 3x_1 + 2x_2 + 5x_3 \\ \text{subject to } & 5x_1 - 3x_2 + x_3 = -8 \quad (y_1) \\ & 4x_1 - 2x_2 + 8x_3 \leq 23 \quad (y_2) \\ & 6x_1 - 7x_2 + 3x_3 \geq 1 \quad (y_3) \\ & x_1 \leq 4 \quad (y_4) \\ & x_3 \geq 0 \\ & x_1, x_2 \in \mathbb{R} \end{aligned}$$

2. Write constraints for the dual problem. Each variable in the primal results in one constraint in the dual.

- Coefficients of dual constraints are found in the columns of the A matrix for the primal problem
- Right hand side of constraint is found in the objective function of the primal problem
- $\leq, =, \geq$ is decided using the domain of the variable (SOB regel)

Dual problem

$$\begin{aligned} & \min -8y_1 + 23y_2 + y_3 + 4y_4 \\ \text{subject to } & 5y_1 + 4y_2 + 6y_3 + y_4 = 3 \quad (x_1) \\ & 3y_1 + 2y_2 + 7y_3 = 2 \quad (x_2) \\ & y_1 + 8y_2 + 3y_3 \geq 5 \quad (x_3) \end{aligned}$$

Constructing dual using SOB: "cook book"

Primal problem

$$\begin{aligned} & \max 3x_1 + 2x_2 + 5x_3 \\ \text{subject to } & 5x_1 + 3x_2 + x_3 = -8 \quad (y_1) \\ & 4x_1 + 2x_2 + 8x_3 \leq 23 \quad (y_2) \\ & 6x_1 + 7x_2 + 3x_3 \geq 1 \quad (y_3) \\ & x_1 \leq 4 \quad (y_4) \\ & x_3 \geq 0 \\ x_1, x_2 & \in \mathbb{R} \end{aligned}$$

2. Write constraints for the dual problem. Each variable in the primal results in one constraint in the dual.

- Coefficients of dual constraints are found in the columns of the A matrix for the primal problem
- Right hand side of constraint is found in the objective function of the primal problem
- $\leq, =, \geq$ is decided using the domain of the variable (SOB regel)

Dual problem

$$\begin{aligned} & \min -8y_1 + 23y_2 + y_3 + 4y_4 \\ \text{subject to } & 5y_1 + 4y_2 + 6y_3 + y_4 = 3 \quad (x_1) \\ & 3y_1 + 2y_2 + 7y_3 = 2 \quad (x_2) \\ & y_1 + 8y_2 + 3y_3 \geq 5 \quad (x_3) \end{aligned}$$

Constructing dual using SOB: "cook book"

Primal problem

$$\begin{aligned} & \max 3x_1 + 2x_2 + 5x_3 \\ \text{subject to } & 5x_1 + 3x_2 + x_3 = -8 \quad (y_1) \\ & 4x_1 + 2x_2 + 8x_3 \leq 23 \quad (y_2) \\ & 6x_1 + 7x_2 + 3x_3 \geq 1 \quad (y_3) \\ & x_1 \leq 4 \quad (y_4) \\ & x_3 \geq 0 \\ & x_1, x_2 \in \mathbb{R} \end{aligned}$$

3. Write up domains of dual variables.

- Use constraint types in the primal problem and SOB rule
 - If the primal problem is a maximization problem:
 - “=” => $\in \mathbb{R}$, “ \geq ” => “ ≤ 0 ”, “ \leq ” => “ ≥ 0 ”

Dual problem

$$\begin{aligned} & \min -8y_1 + 23y_2 + y_3 + 4y_4 \\ \text{subject to } & 5y_1 + 4y_2 + 6y_3 + y_4 = 3 \quad (x_1) \\ & 3y_1 + 2y_2 + 7y_3 = 2 \quad (x_2) \\ & y_1 + 8y_2 + 3y_3 \geq 5 \quad (x_3) \\ & y_1 \in \mathbb{R} \\ & y_2, y_4 \geq 0 \\ & y_3 \leq 0 \end{aligned}$$

Constructing dual using SOB: Summary

1. Start by writing the objective function of the dual problem'
 - Objective coefficients are found as the coefficients on the right hand side of the primal
 - "max" in primal => "min" in dual
 - "min" in primal => "max" in dual
2. Write constraints for the dual problem. Each variable in the primal results in one constraint in the dual.
 - Coefficients of dual constraints are find in the columns of the A matrix for the primal problem
 - Right hand side of constraint is found in the objective function of the primal problem
 - $\leq, =, \geq$ is decided using the domain of the variable (SOB regel)
3. Write up domains of dual variables.
 - Use constraint types in the primal problem and SOB regel
 - If the primal problem is a maximization problem:
 - " $=$ " => $\in \mathbb{R}$, " \geq " => " ≤ 0 ", " \leq " => " ≥ 0 "

Radiation Therapy example



Primal problem:

$$\min Z = 0.4x_1 + 0.5x_2$$

subject to

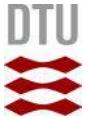
$$0.3x_1 + 0.1x_2 \leq 2.7$$

$$0.5x_1 + 0.5x_2 = 6$$

$$0.6x_1 + 0.4x_2 \geq 6$$

$$x_1, x_2 \geq 0$$

Radiation Therapy example



Primal problem:

$$\min Z = 0.4x_1 + 0.5x_2$$

subject to

$$0.3x_1 + 0.1x_2 \leq 2.7$$

$$0.5x_1 + 0.5x_2 = 6$$

$$0.6x_1 + 0.4x_2 \geq 6$$

$$x_1, x_2 \geq 0$$

Dual problem:

$$\max 2.7y_1 + 6y_2 + 6y_3$$

subject to

$$0.3y_1 + 0.5y_2 + 0.6y_3 \leq 0.4$$

$$0.1y_1 + 0.5y_2 + 0.4y_3 \leq 0.5$$

$$y_1 \leq 1$$

$$y_2 \in \mathbb{R}$$

$$y_3 \geq 0$$

Sensitivity analysis: Changes on the right hand side (1)

$$\max 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \geq 0$$

- Dual variables can be used to estimate what impact a change in the right hand side has on the objective value.
- This prediction is only valid as long as the change is so "small" that the optimal basis is not changed.
- How do we decide how much we can change the right hand side without changing the basis?

Sensitivity analysis: Changes on the right hand side (2)

Start tableau:

$-c$	0	0
A	I	b

Last tableau:

$y^* A - c$	y^*	$y^* b$
$S^* A$	S^*	$S^* b$

- Tableau is feasible and optimal if the following is true:
 - $y^* A - c \geq 0$
 - $y^* \geq 0$
 - $S^* b \geq 0$
- Change of right hand side (b) only changes $S^* b$

Sensitivity analysis: Changes on the right hand side (2)

Start tableau:

$-c$	0	0
A	I	b

Last tableau:

$y^*A - c$	y^*	y^*b
S^*A	S^*	S^*b

- Tableau is feasible and optimal if the following is true:
 - $y^*A - c \geq 0$
 - $y^* \geq 0$
 - $S^*b \geq 0$
- Change of right hand side (b) only changes S^*b
- We need to ensure that $S^*(b + \Delta) = S^*b + S^*\Delta \geq 0$ where Δ is the change to the right hand side. From this inequality we can decide the interval for a feasible change.

Sensitivity analysis: Changes on the right hand side (3)

$$\max 3x_1 + 5x_2$$



subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \geq 0$$

- Ensure $S^*(b + \Delta) = S^*b + S^*\Delta \geq 0$ where Δ is the change to the right hand side.
- Example: Let's consider a change to the right hand side of the second constraint.
- Ensure

$$S^*b + S^* \begin{bmatrix} 0 \\ \Delta \\ 0 \end{bmatrix} \geq 0$$

- S^* and S^*b is read in the final tableau.

Sensitivity analysis: Changes on the right hand side (4)

$$\max 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \geq 0$$

$$\left[\begin{array}{c|c|c} y^* A - c & y^* & y^* b \\ S^* A & S^* & S^* b \end{array} \right]$$

Basic Variable	Eq.	Z	Coefficient of:					Right Side
			x_1	x_2	x_3	x_4	x_5	
Z	(0)	1	-3	-5	0	0	0	0
x_3	(1)	0	1	0	1	0	0	4
x_4	(2)	0	0	2	0	1	0	12
x_5	(3)	0	3	2	0	0	1	18
Z	(0)	1	-3	0	0	$\frac{5}{2}$	0	30
x_3	(1)	0	1	0	1	0	0	4
x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
x_5	(3)	0	3	0	0	-1	1	6
Z	(0)	1	0	0	0	$\frac{3}{2}$	1	36
x_3	(1)	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	2
x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
x_1	(3)	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	2

S^* (or B^{-1})

$S^* b$

Sensitivity analysis: Changes on the right hand side (5)

- Ensure

$$S^*b + S^* \begin{bmatrix} 0 \\ \Delta \\ 0 \end{bmatrix} \geq 0$$

$$\Leftrightarrow \begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 & \frac{1}{3} & -\frac{1}{3} \\ 0 & \frac{1}{2} & 0 \\ 0 & -\frac{1}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} 0 \\ \Delta \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix} + \begin{bmatrix} \frac{1}{3}\Delta \\ \frac{1}{2}\Delta \\ -\frac{1}{3}\Delta \end{bmatrix} \geq 0$$

- Thus, Δ should satisfy:

$$2 + \frac{1}{3}\Delta \geq 0 \Leftrightarrow \frac{1}{3}\Delta \geq -2 \Leftrightarrow \Delta \geq -6$$

$$6 + \frac{1}{2}\Delta \geq 0 \Leftrightarrow \frac{1}{2}\Delta \geq -6 \Leftrightarrow \Delta \geq -12$$

$$2 - \frac{1}{3}\Delta \geq 0 \Leftrightarrow \frac{1}{3}\Delta \leq 2 \Leftrightarrow \Delta \leq 6$$

- First and last inequality are tightest so interval for Δ is:

$$-6 \leq \Delta \leq 6$$

- Right hand side in second constraint was 12. This right hand side can be adjusted in the interval $[12 - 6; 12 + 6] = [6; 18]$ without changing the optimal basic solution. In this interval we can "trust" the shadow price.
- The dual variable for the second constraint is $\frac{3}{2}$. If we change the right hand side to 18 the objective is increasing to $36 + 6 \cdot \frac{3}{2} = 45$.

$$\max 3x_1 + 5x_2$$

subject to

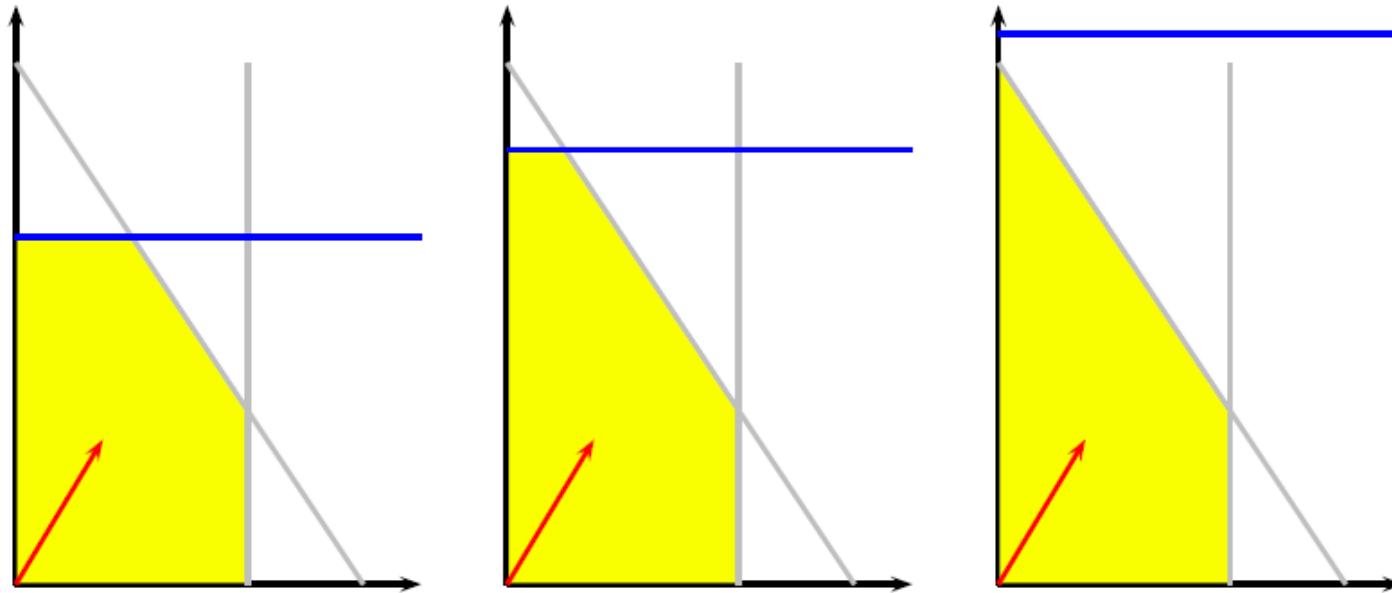
$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \geq 0$$

Sensitivity analysis: Changes on the right hand side (6)



Sensitivity analysis: Changes to the objective function (1)

$$\max 3x_1 + 5x_2$$

subject to

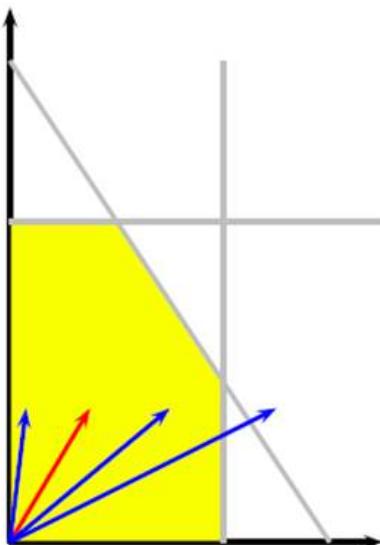
$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \geq 0$$

What happens if we change c_1 (objective coefficient for x_1)?



Sensitivity analysis: Changes to the objective function (2)

Start tableau:

$-c$	0	0
A	I	b

Last tableau:

$y^*A - c$	y^*	y^*b
S^*A	S^*	S^*b

- Tableau is feasible and optimal if the following is true:
 - $y^*A - c \geq 0$
 - $y^* \geq 0$
 - $S^*b \geq 0$
- Changes to the objective function (c) only impacts $y^*A - c$ and $y^* = C_B B^{-1}$

Sensitivity analysis: Changes to the objective function (3)

Basic Variable	Eq.	Coefficient of:						Right Side
		Z	x_1	x_2	x_3	x_4	x_5	
Z	(0)	1	-3	-5	0	0	0	0
x_3	(1)	0	1	0	1	0	0	4
x_4	(2)	0	0	2	0	1	0	12
x_5	(3)	0	3	2	0	0	1	18

Z	(0)	1	-3	0	0	$\frac{5}{2}$	0	30
x_3	(1)	0	1	0	1	0	0	4
x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
x_5	(3)	0	3	0	0	-1	1	6

Z	(0)	1	0	0	0	$\frac{3}{2}$	1	36
x_3	(1)	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	2
x_2	(2)	0	0	1	0	$\frac{1}{2}$	0	6
x_1	(3)	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	2

Sensitivity analysis: Changes to the objective function (4)

We start from the final tableau

B.v.	EQ	Z	x1	x2	x3	x4	x5	RHS
Z	0	1	0	0	0	3/2	1	36
x3	1	0	0	0	1	1/3	-1/3	2
x2	2	0	0	1	0	1/2	0	6
x1	3	0	1	0	0	-1/3	1/3	2

We add an extra row to row zero that accounts for the Δ contribution.

Remember

$$y^*A - (c + \Delta) \geq 0$$

B.v.	EQ	Z	x1	x2	x3	x4	x5	RHS
Z	0	1	0	0	0	3/2	1	36
			-1	0	0	0	0	0 Δ
x3	1	0	0	0	1	1/3	-1/3	2
x2	2	0	0	1	0	1/2	0	6
x1	3	0	1	0	0	-1/3	1/3	2

The tableau is not on proper form if $\Delta \neq 0$

B.v.	EQ	Z	x1	x2	x3	x4	x5	RHS
Z	0	1	0	0	0	3/2	1	36
			0	0	0	-1/3	1/3	2 Δ
x3	1	0	0	0	1	1/3	-1/3	2
x2	2	0	0	1	0	1/2	0	6
x1	3	0	1	0	0	-1/3	1/3	2

We restore the proper form by adding row 3 to the Δ row

Sensitivity analysis: Changes to the objective function (5)

Last tableau from previous slide:

B.v.	EQ	Z	x1	x2	x3	x4	x5	RHS
Z	0	1	0	0	0	3/2	1	36
			0	0	0	-1/3	1/3	2 Δ
x3	1	0	0	0	1	1/3	-1/3	2
x2	2	0	0	1	0	1/2	0	6
x1	3	0	1	0	0	-1/3	1/3	2

- Row 0 must be ≥ 0 to keep the tableau optimal.
- We must ensure:

$$\frac{3}{2} - \frac{1}{3}\Delta \geq 0 \Leftrightarrow \frac{1}{3}\Delta \leq \frac{3}{2} \Leftrightarrow \Delta \leq \frac{9}{2}$$

$$1 + \frac{1}{3}\Delta \geq 0 \Leftrightarrow \frac{1}{3}\Delta \geq -1 \Leftrightarrow \Delta \geq -3$$

- Conclusion: Δ must be in the interval $[-3; \frac{9}{2}]$ in order for the tableau to stay optimal.
- c_1 should be in the interval $3 + [-3; \frac{9}{2}] = [0; \frac{15}{2}]$.
- When c_1 is changed by Δ the objective value is changed to $36 + 2\Delta$ as long as $\Delta \in [-3; \frac{9}{2}]$

$$\begin{aligned} & \max 3x_1 + 5x_2 \\ \text{subject to} \\ & x_1 \leq 4 \\ & 2x_2 \leq 12 \\ & 3x_1 + 2x_2 \leq 18 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Sensitivity analysis: Changes to the objective function (6)

What if we change the objective coefficient of a non-basic variable?

Let's use x_4 as an example.

Add Δ row as before

In this case the tableau is in the proper form right away.

We just need to ensure that row zero is ≥ 0

B.v.	EQ	Z	x1	x2	x3	x4	x5	RHS
Z	0	1	0	0	0	3/2	1	36
x3	1	0	0	0	1	1/3	-1/3	2
x2	2	0	0	1	0	1/2	0	6
x1	3	0	1	0	0	-1/3	1/3	2

B.v.	EQ	Z	x1	x2	x3	x4	x5	RHS
Z	0	1	0	0	0	3/2	1	36
			0	0	0	-1	0	0 Δ
x3	1	0	0	0	1	1/3	-1/3	2
x2	2	0	0	1	0	1/2	0	6
x1	3	0	1	0	0	-1/3	1/3	2

We get:

$$\frac{3}{2} - \Delta \geq 0 \Leftrightarrow \Delta \leq \frac{3}{2}$$

Sensitivity analysis: New decision variable ... after we have solved LP

$$\begin{aligned} \max \quad & 3x_1 + 5x_2 + 4x_e \\ \text{hvor } & x_1 + 2x_e \leq 4 \\ & 2x_2 + 3x_e \leq 12 \\ & 3x_1 + 2x_2 + x_e \leq 18 \\ & x_1, x_2, x_e \geq 0 \end{aligned}$$



We have added a new decision variable
to Wyndor example.

TABLE 4.8 Complete set of simplex tableaux for the Wyndor Glass Co. problem

We assume that the old basis solution still is optimal.

We compute the entry in row 0 for the new variable

- if entry in row zero is ≥ 0 : The new variable cannot improve the current solution
- if entry in row zero is < 0 : The new variable can improve the current solution. We have to pivot new variable into the basis (we continue computation from the current solution).

		Coefficient of:					Right Side	
		Z	x_1	x_2	x_3	x_4	x_5	
		1	-3	-5	0	0	0	0
		0	1	0	1	0	0	4
		0	0	2	0	1	0	12
		0	3	2	0	0	1	18
		1	-3	0	0	$\frac{5}{2}$	0	30
		0	1	0	1	0	0	4
		0	0	1	0	$\frac{1}{2}$	0	6
		0	3	0	0	-1	1	6
2	Z	(0)	1	0	0	$\frac{3}{2}$	1	36
	x_3	(1)	0	0	0	$\frac{1}{3}$	$-\frac{1}{3}$	2
	x_2	(2)	0	0	1	$\frac{1}{2}$	0	6
	x_1	(3)	0	1	0	$-\frac{1}{3}$	$\frac{1}{3}$	2

Sensitivity analysis: New decision variable ... after we have solved LP

$$\begin{aligned} \max \quad & 3x_1 + 5x_2 + 4x_e \\ \text{hvor} \quad & x_1 + 2x_e \leq 4 \\ & 2x_2 + 3x_e \leq 12 \\ & 3x_1 + 2x_2 + x_e \leq 18 \\ & x_1, x_2, x_e \geq 0 \end{aligned}$$

The coefficient for x_e in row 0 becomes

$$y^* A - c_e = \left[\begin{array}{ccc} 0 & \frac{3}{2} & 1 \end{array} \right] \left[\begin{array}{c} 2 \\ 3 \\ 1 \end{array} \right] - 4 = \frac{11}{2} - 4 > 0$$

Old solution is still optimal, so $x_e^* = 0$

Summary

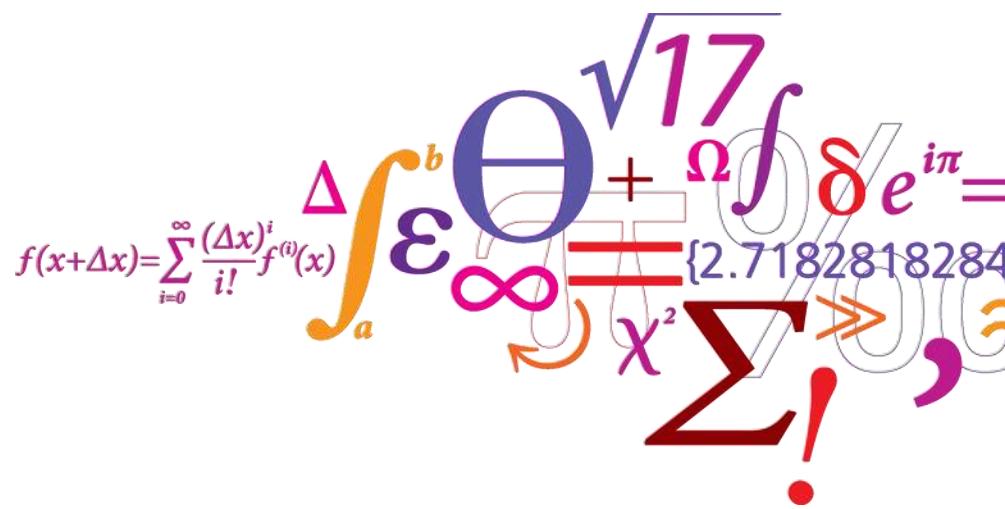
Some applications of duality (and the fundamental insight)

- If a problem has many constraints and few variables, then it may be faster to solve the dual problem.
- Sometimes it is easier to find a feasible start solution to the dual problem and one can avoid the 2-phase method.
- If we can guess a feasible primal solution x and a feasible dual solution y
 - If $cx = yb$ then we have found an optimal pair of solutions.
 - If $|cx - yb|$ is sufficiently small then we may accept the primal and dual solution.
- Sensitivity analysis: How does the optimal solution change if we make small changes to the LP?
- Can be used for constructing algorithms for special cases of linear programming (more on this later).

Exercise time!

Lecture 06

Integer programming

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$

$$\int_a^b \Theta + \Omega \int \delta e^{i\pi} =$$
$$\infty - \{2.7182818284$$
$$\Sigma \gg,$$
$$\chi^2$$
$$\sum!$$

Integer programming - intro

Linear programming (LP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \geq 0 \quad \forall j \in \{1, \dots, n\}$$

Integer programming - intro

Linear programming (LP):

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

subject to

$$\begin{aligned} \sum_{j=1}^n a_{ij} x_j &\leq b_i \quad \forall i \in \{1, \dots, m\} \\ x_j &\geq 0 \quad \forall j \in \{1, \dots, n\} \end{aligned}$$

Integer programming (IP):

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

subject to

$$\begin{aligned} \sum_{j=1}^n a_{ij} x_j &\leq b_i \quad \forall i \in \{1, \dots, m\} \\ x_j &\in \mathbb{Z}_+ \quad \forall j \in \{1, \dots, n\} \end{aligned}$$

In Operations Research:

$$\mathbb{Z}_+ = \{0, 1, 2, 3, \dots\}$$

$$\mathbb{Z} = \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$

Integer programming - intro

Integer programming (IP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \in \mathbb{Z}_+ \quad \forall j \in \{1, \dots, n\}$$

Mixed integer programming (MIP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$\begin{aligned} x_j &\in \mathbb{Z}_+ & \forall j \in S_1 \\ x_j &\geq 0 & \forall j \in S_2 \end{aligned}$$

hvor $S_1 \cap S_2 = \emptyset$ og $S_1 \cup S_2 = \{1, \dots, n\}$

Integer programming - intro

Integer programming (IP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \in \mathbb{Z}_+ \quad \forall j \in \{1, \dots, n\}$$

Binary integer programming (BIP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \in \{0, 1\} \quad \forall j \in S_1$$

Wyndor problem with integer variables

Decision variables:

x_1 = production of product 1 (units/week)

x_2 = production af product 2 (units/week)

(LP)

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

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

$$Z^* = 36, x_1^* = 2, x_2^* = 6$$

(IP)

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \in \mathbb{Z}_+$$

$$Z^* = 36, x_1^* = 2, x_2^* = 6$$

Wyndor problem with integer variables

Decision variables:

x_1 = production of product 1 (units/week)

x_2 = production af product 2 (units/week)

(LP)

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 19 \quad \leftarrow$$

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

$$Z^* = 37, x_1^* = 2\frac{1}{3}, x_2^* = 6$$



(IP)

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 19 \quad \leftarrow$$

$$x_1, x_2 \in \mathbb{Z}_+$$

$$Z^* = 36, x_1^* = 2, x_2^* = 6$$

Wyndor problem (LP)

$$\max Z = 3x_1 + 5x_2$$

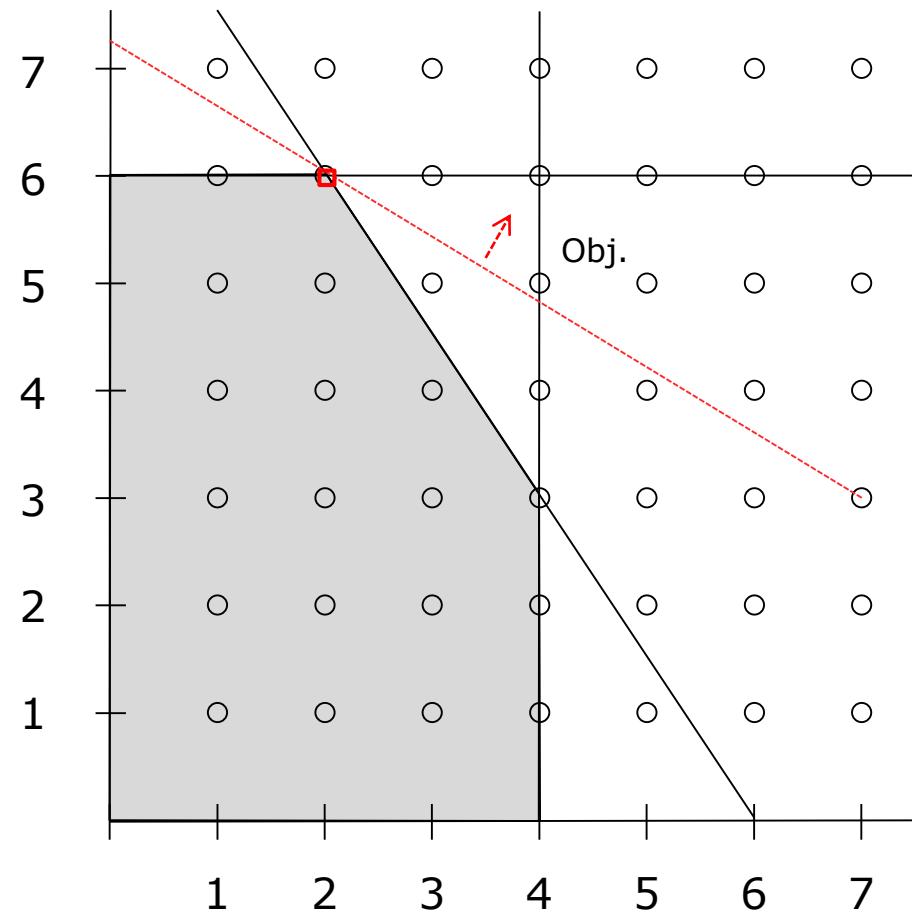
subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

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



Wyndor problem (IP)

$$\max Z = 3x_1 + 5x_2$$

subject to

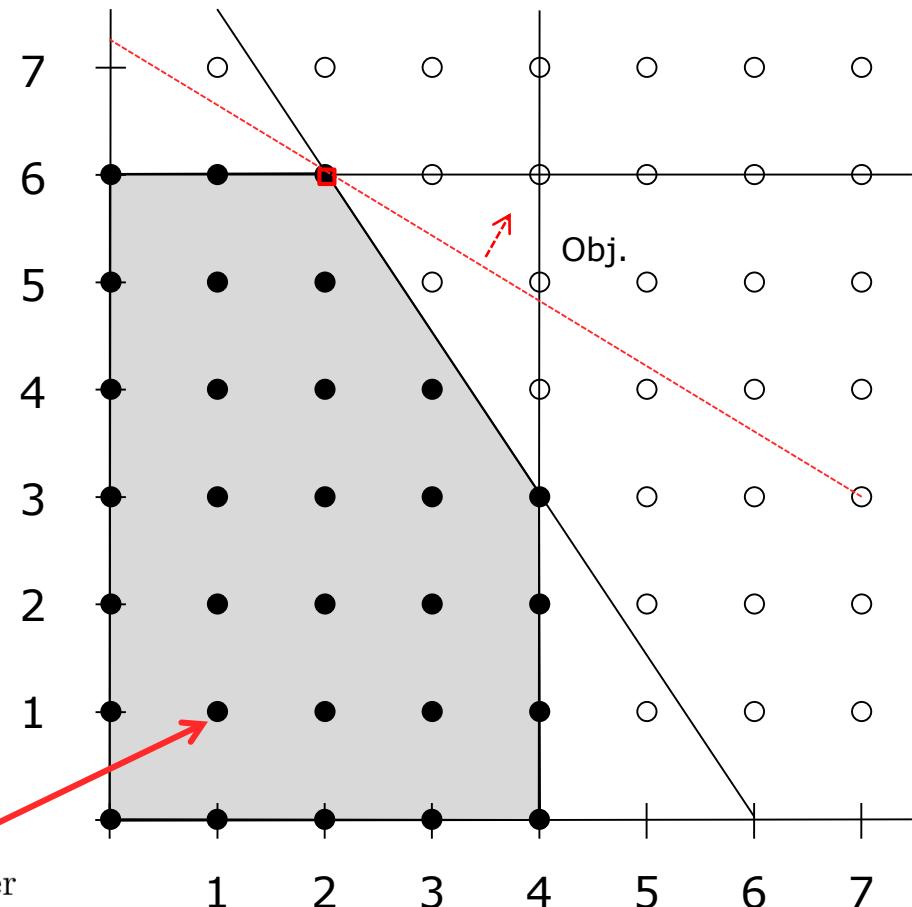
$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \in \mathbb{Z}_+$$

The black dots are the feasible integer solutions



Wyndor problem (IP)

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 19$$

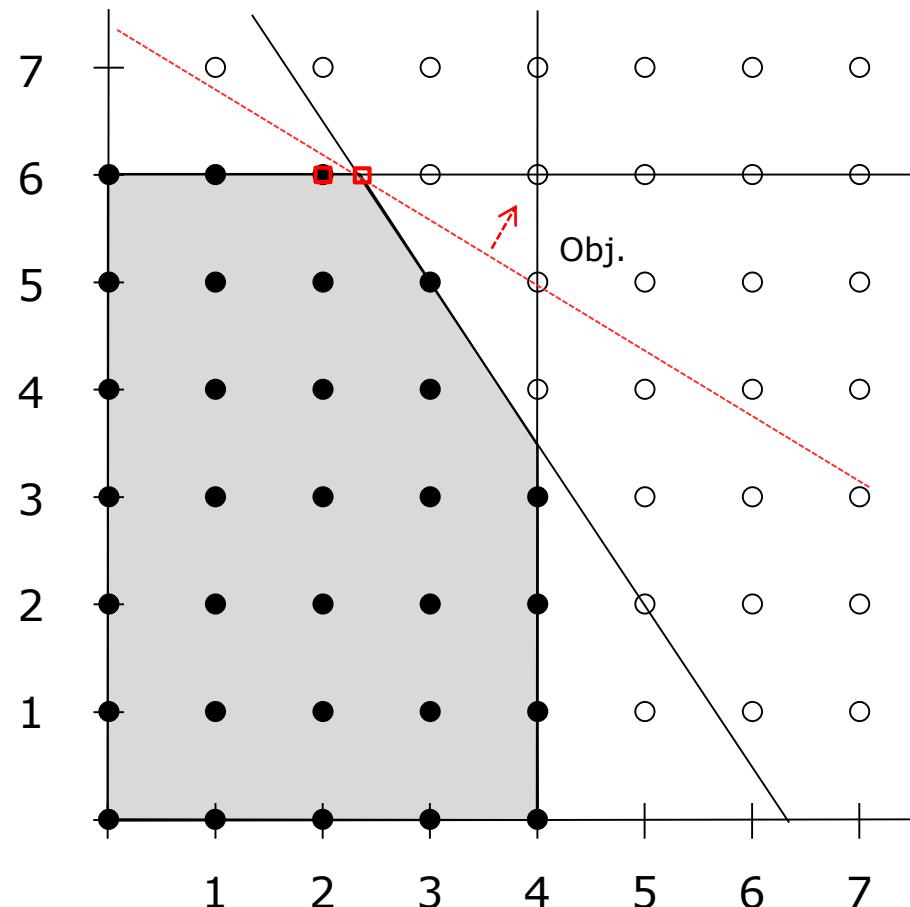
$$x_1, x_2 \in \mathbb{Z}_+$$

Optimal LP solution moved to

$$Z^* = 37, x_1^* = 2\frac{1}{3}, x_2^* = 6$$

Optimal IP solution "stuck at"

$$Z^* = 36, x_1^* = 2, x_2^* = 6$$



Wyndor with integer variables in Julia

```
using JuMP
using GLPK

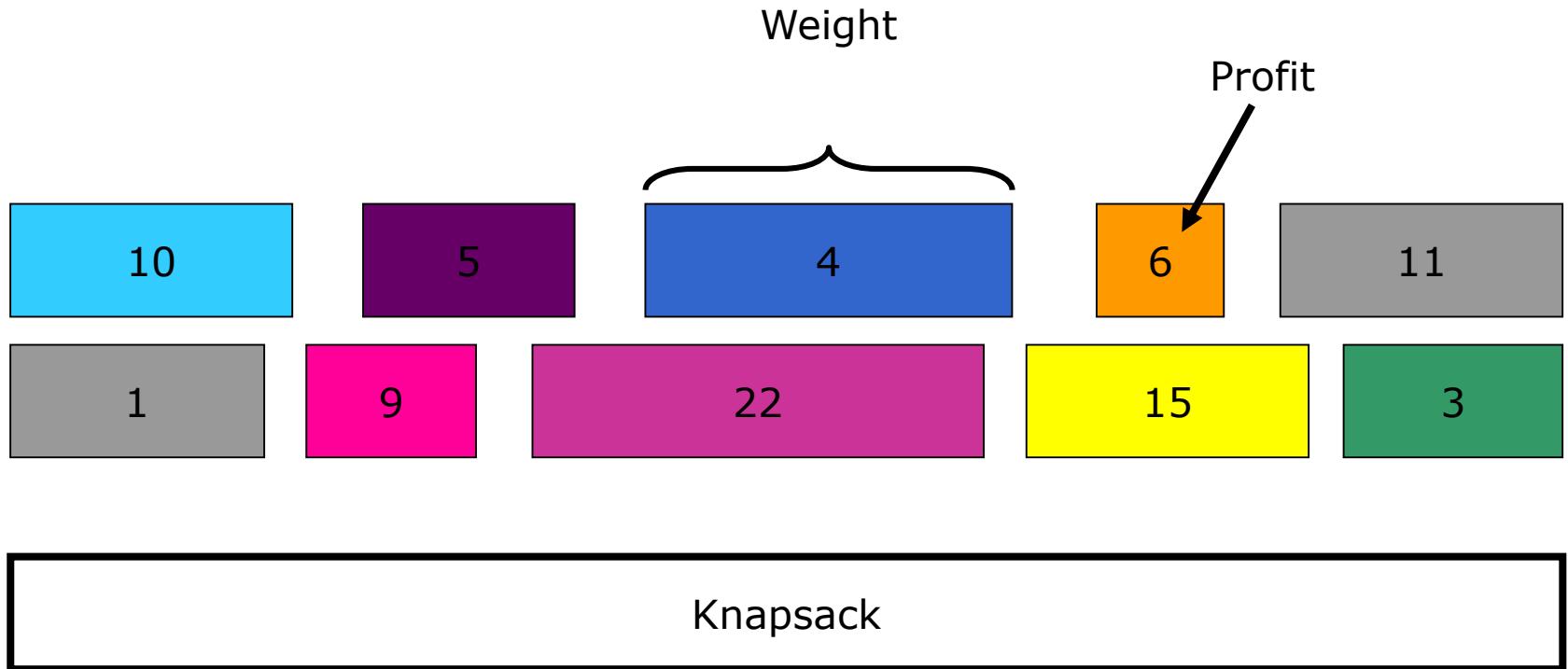
m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x1 >= 0, Int )
@variable(m, x2 >= 0, Int ) ←

@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 19 )

optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("x1 = ", JuMP.value(x1))
println("x2 = ", JuMP.value(x2))
```

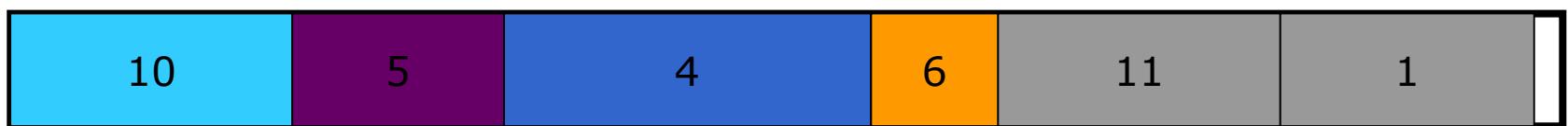
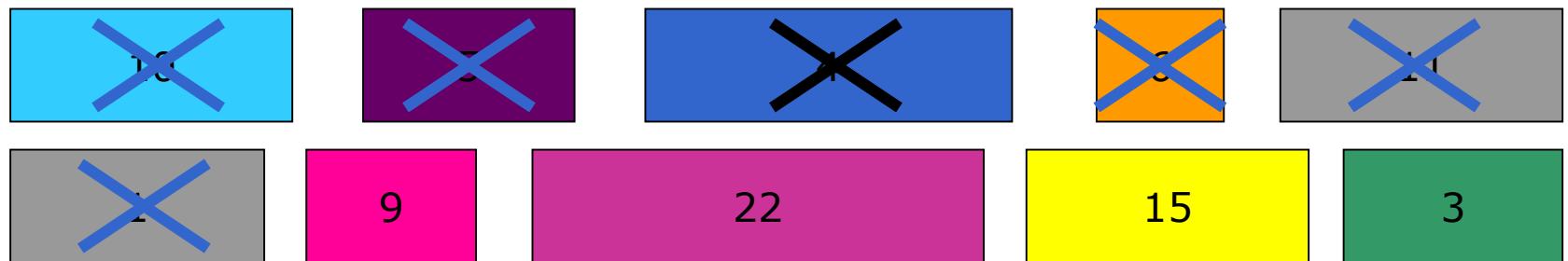
Knapsack problem



maximize profit of the selected items. Total weight of chosen items cannot exceed the capacity of the knapsack.

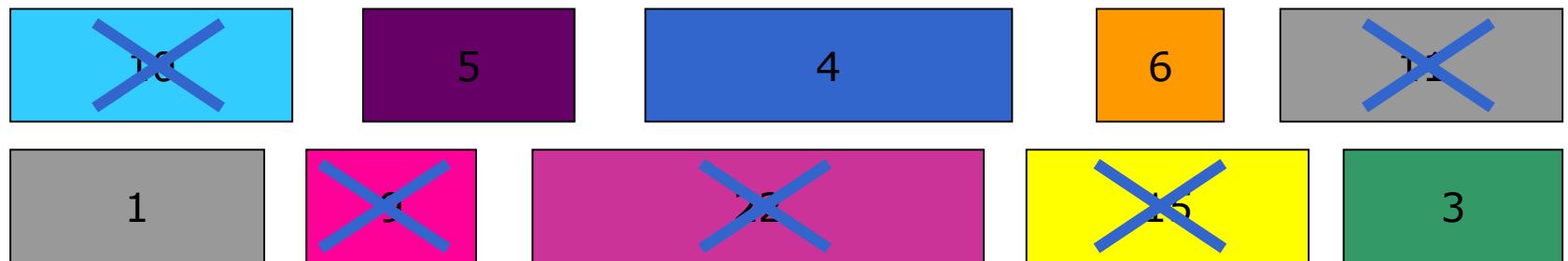
Knapsack problem

A feasible solution:



Knapsack problem

A better solution:



Solution from
previous slide



$$\text{Profit: } 10+5+4+6+11+1 = 37$$



$$\text{Profit: } 22+15+11+10+9 = 67$$

Knapsack problem

Parameters and sets:

n : number of items.

$N = \{1, \dots, n\}$: set of all items.

p_j : profit of selecting item j ($j \in N$).

w_j : weight of item j ($j \in N$).

c : knapsack capacity.

Decision variables:

x_j : 1 if item j is added to the knapsack, 0 otherwise.

The model:

$$\max \sum_{j \in N} p_j x_j$$

subject to

$$\sum_{j \in N} w_j x_j \leq c$$

$$x_j \in \{0, 1\} \quad \forall j \in N$$

Example



Knapsack

Item	1	2	3	4	5	6	7	8	9	10
Profit	10	5	4	6	11	1	9	22	15	3
Weight	158	118	205	71	158	142	95	252	158	122

Kapacitet: 866

$$\max 10x_1 + 5x_2 + 4x_3 + 6x_4 + 11x_5 + 1x_6 + 9x_7 + 22x_8 + 15x_9 + 3x_{10}$$

subject to

$$158x_1 + 118x_2 + 205x_3 + 71x_4 + 158x_5 + 142x_6 + 95x_7 + 252x_8 + 158x_9 + 122x_{10} \leq 866$$
$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10} \in \{0, 1\}$$

Example

```
using JuMP, GLPK

n=10
p=[10, 5, 4, 6, 11, 1, 9, 22, 15, 3]
w=[158, 118, 205, 71, 158, 142, 95, 252, 158, 122]
c=866

m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x[1:n], Bin )

@objective(m, Max, sum(p[j] * x[j] for j=1:n) )
@constraint(m, sum(w[j] * x[j] for j=1:n) <= c)

optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("x = ", JuMP.value.(x))
```

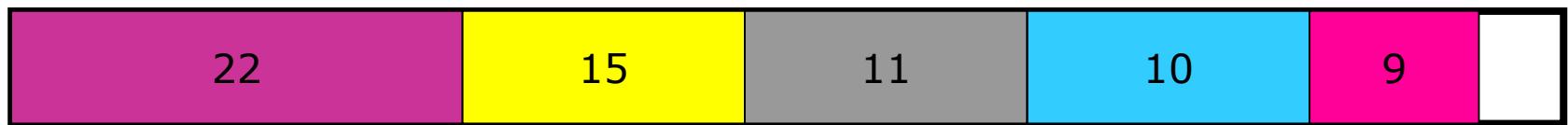
```
Objective value: 68.0
x = [0.0, 1.0, 0.0, 1.0, 1.0, 0.0, 1.0, 1.0, 1.0, 0.0]
```

Example

Earlier solutions:

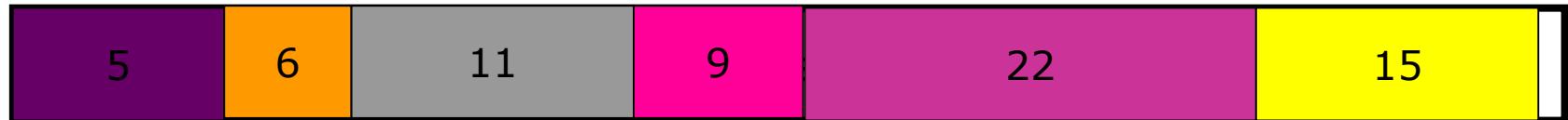


Profit: $10+5+4+6+11+1 = 37$



Profit: $22+15+11+10+9 = 67$

Optimal solution:



Profit: 68

Knapsack problem

Parameters and sets:

n : number of items.

$N = \{1, \dots, n\}$: set of all items.

p_j : profit of selecting item j ($j \in N$).

w_j : weight of item j ($j \in N$).

c : knapsack capacity.

Decision variables:

x_j : 1 if item j is added to the knapsack, 0 otherwise.

The model:

$$\max \sum_{j \in N} p_j x_j$$

subject to

$$\sum_{j \in N} w_j x_j \leq c$$

$$x_j \in \{0, 1\} \quad \forall j \in N$$

What if there was many of each item?

Knapsack problem

Parameters and sets:

n : number of items.

$N = \{1, \dots, n\}$: set of all items.

p_j : profit of selecting item j ($j \in N$).

w_j : weight of item j ($j \in N$).

c : knapsack capacity.

Decision variables:

x_j : 1 if item j is added to the knapsack, 0 otherwise.

The model:

$$\max \sum_{j \in N} p_j x_j$$

subject to

$$\sum_{j \in N} w_j x_j \leq c$$

$$x_j \in \mathbb{Z}_+ \quad \forall j \in N$$

model if there are many of each item

Parameters and sets:

n : number of items.

$N = \{1, \dots, n\}$: set of all items.

p_j : profit of selecting item j ($j \in N$).

w_j : weight of item j ($j \in N$).

c : knapsack capacity.

Decision variables:

x_j : 1 if item j is added to the knapsack, 0 otherwise.

The model:

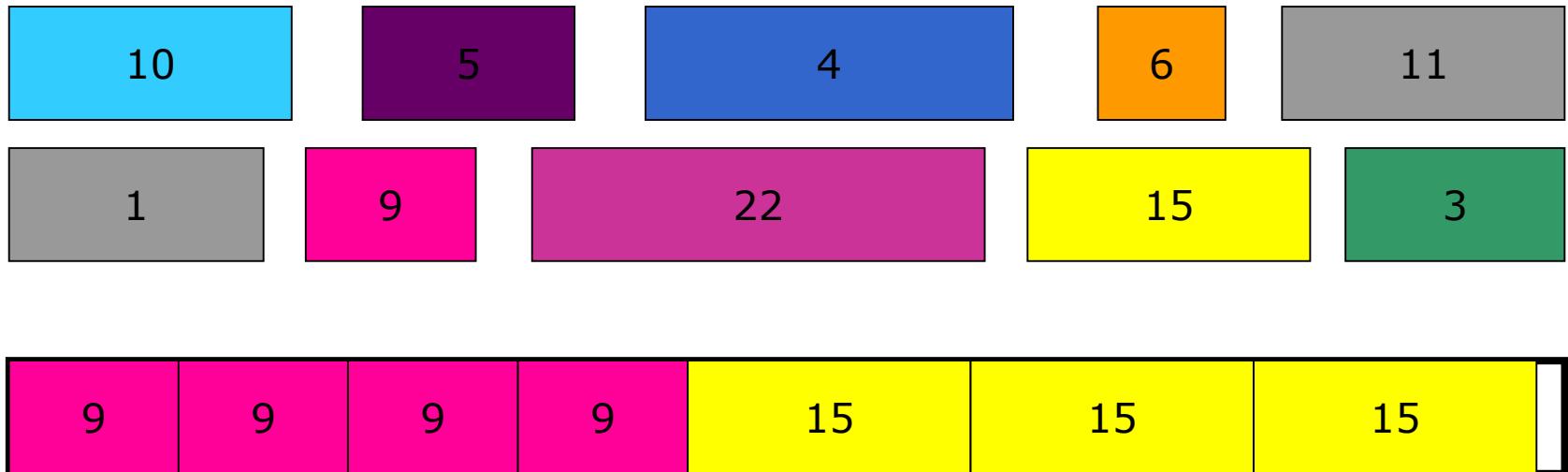
$$\max \sum_{j \in N} p_j x_j$$

subject to

$$\sum_{j \in N} w_j x_j \leq c$$

$$x_j \in \mathbb{Z}_+ \quad \forall j \in N$$

model if there are many of each item



Optimal objective value: $4*9+3*15=81$

"Startup cost" - Wyndor

- Consider wyndor example again. Assume it cost us 2000\$ to start a production of product 1 and 24000\$ to start producing the second product ("startup cost")
- Once startup cost is paid we can produce as many units as the other constraints allow. Start up cost is only paid once
- New decision variable $y_j \in \{0, 1\}$ indicates if startup cost has been paid for product j (1 = yes, 0=no).
- M is a big number

"Startup cost" - Wyndor

Decision variables:

- $x_j \geq 0$ production of product $j = 1, 2$ (units/week)
 $y_j \in \{0, 1\}$ if start up cost for product $j = 1, 2$ has been paid

$$\max Z = 3x_1 + 5x_2 - 1y_1 - 24y_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1 \leq My_1$$

$$x_2 \leq My_2$$

$$x_1, x_2 \geq 0$$

$$y_1, y_2 \in \{0, 1\}$$

What is the lowest value can we assign to M ?

$$Z^* = 11, x_1^* = 4, x_2^* = 0$$

"Startup cost" - Wyndor

Decision variables:

- $x_j \geq 0$ production of product $j = 1, 2$ (units/week)
 $y_j \in \{0, 1\}$ if start up cost for product $j = 1, 2$ has been paid

$$\max Z = 3x_1 + 5x_2 - 1y_1 - 24y_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1 \leq M y_1$$

$$x_2 \leq M y_2$$

$$x_1, x_2 \geq 0$$

$$y_1, y_2 \in \{0, 1\}$$

What is the lowest value can we assign to M ?

Answer: $M = 6$

$$Z^* = 11, x_1^* = 4, x_2^* = 0$$

“Chosing between alternatives” - Wyndor

- What if we only were allowed to produce one of the two products?
- Idea: Introduce a binary variable y . If $y = 1$ we are only allowed to make product 1. $y = 0$ we are only allowed to make product 2.

“Chosing between alternatives” - Wyndor

Decision variables:

$x_j \geq 0$ production of product $j = 1, 2$ (units/week)

$y \in \{0, 1\}$ $y = 1$: we can only produce product 1. $y = 0$: we can only produce product 2.

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1 \leq M y$$

$$x_2 \leq M(1 - y)$$

$$x_1, x_2 \geq 0$$

$$y \in \{0, 1\}$$

$$Z^* = 30, x_1^* = 0, x_2^* = 6, y = 0$$

“Chosing between alternatives” - Wyndor

Decision variables:

$x_j \geq 0$ production of product $j = 1, 2$ (units/week)

$y \in \{0, 1\}$ $y = 1$: we can only produce product 1. $y = 0$: we can only produce product 2.

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1 \leq M y$$

$$x_2 \leq M(1 - y)$$

$$x_1, x_2 \geq 0$$

$$y \in \{0, 1\}$$

What is the lowest value can we assign to M ?

Answer: $M = 6$

$$Z^* = 30, x_1^* = 0, x_2^* = 6, y = 0$$

“Equally spaced alternatives” - Wyndor

- What if we only wanted solutions where x_1 is even and x_2 is odd?

“Equally spaced alternatives” - Wyndor

What if we only wanted solutions where x_1 is even and x_2 is odd?

Decision variables:

$x_j \geq 0$ production of product $j = 1, 2$ (units/week)

$y_j \in \mathbb{Z}_+$ helper variable

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1 = 2y_1$$

$$x_2 = 2y_2 + 1$$

$$x_1, x_2 \geq 0$$

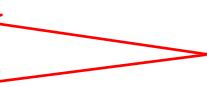
$$y_1, y_2 \in \mathbb{Z}_+$$

$$Z^* = 31, x_1^* = 2, x_2^* = 5, y_1 = 1, y_2 = 2$$

Activate / deactivate constraints - Wyndor

$$\max Z = 3x_1 + 5x_2$$

subject to

$$\begin{array}{ll} x_1 & \leq 4 \\ 2x_2 & \leq 12 \\ 3x_1 + 2x_2 & \leq 18 \\ x_1, x_2 & \in \mathbb{Z}_+ \end{array}$$


Only one of these two constraints need to be satisfied. How can we model that?

Activate / deactivate constraints - Wyndor

$$\max Z = 3x_1 + 5x_2$$

subject to

$$\begin{array}{ll} x_1 & \leq 4 \\ 2x_2 & \leq 12 \\ 3x_1 + 2x_2 & \leq 18 \\ x_1, x_2 & \in \mathbb{Z}_+ \end{array}$$


Only one of these two constraints need to be satisfied. How can we model that?

Add binary decision variable y . If $y = 1$ then the second constraint is activated, if $y = 0$ the the hird constraint is activated.

$$\max Z = 3x_1 + 5x_2$$

subject to

$$\begin{array}{ll} x_1 & \leq 4 \\ 2x_2 & \leq 12 + M(1 - y) \\ 3x_1 + 2x_2 & \leq 18 + My \\ x_1, x_2 & \geq 0 \\ y & \in \{0, 1\} \end{array}$$

Which value should we assign to M ?

Solution: $Z^* = 45, x_1^* = 0, x_2^* = 9, y^* = 0$

Activate / deactivate constraints - Wyndor

$$\max Z = 3x_1 + 5x_2$$

subject to

$$\begin{array}{ll} x_1 & \leq 4 \\ 2x_2 & \leq 12 \\ 3x_1 + 2x_2 & \leq 18 \\ x_1, x_2 & \in \mathbb{Z}_+ \end{array}$$

Only one of these two constraints need to be satisfied. How can we model that?

Add binary decision variable y . If $y = 1$ then the second constraint is activated, if $y = 0$ the the hird constraint is activated.

$$\max Z = 3x_1 + 5x_2$$

subject to

$$\begin{array}{ll} x_1 & \leq 4 \\ 2x_2 & \leq 12 + M(1 - y) \\ 3x_1 + 2x_2 & \leq 18 + My \\ x_1, x_2 & \geq 0 \\ y & \in \{0, 1\} \end{array}$$

Which value should we assign to M ?
Answer: $M = 6$

Solution: $Z^* = 45, x_1^* = 0, x_2^* = 9, y^* = 0$

Non linearities - Wyndor

We would like to solve the following problem, but objective is not linear.

$$\max Z = 3x_1 + 5x_2 - 0.5x_2^2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \in \mathbb{Z}_+$$

Non linearities - Wyndor

We would like to solve the following problem, but objective is not linear.

$$\max Z = 3x_1 + 5x_2 - 0.5x_2^2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \in \mathbb{Z}_+$$

Idea:

- There are only 7 feasible values for x_2 : 0,1,2,3,4,5,6
- We introduce a binary variable $y_j \in \{0, 1\}$ for $j=0,1,\dots,6$
- if $x_2=k$ then we want $y_k = 1$ and $y_j = 0$ for $j \neq k$, $j \in \{0, 1, \dots, 6\}$
- For example, if $x_2 = 5$ then $y_5 = 1$ and $y_j = 0$ for $j \in \{0, 1, 2, 3, 4, 6\}$

Non linearities - Wyndor

Original objective

$$\max Z = 3x_1 + 5x_2 - 0.5x_2^2$$

Equivalent IP model

$$\max Z = 3x_1 + 5x_2 - 0.5(0y_0 + 1y_1 + 4y_2 + 9y_3 + 16y_4 + 25y_5 + 36y_6)$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$y_0 + y_1 + y_2 + y_3 + y_4 + y_5 + y_6 = 1$$

$$0y_0 + 1y_1 + 2y_2 + 3y_3 + 4y_4 + 5y_5 + 6y_6 = x_2$$

$$x_1, x_2 \in \mathbb{Z}_+$$

$$y_0, y_1, y_2, y_3, y_4, y_5, y_6 \in \{0, 1\}$$

Non linearities - Wyndor

Original objective

$$\max Z = 3x_1 + 5x_2 - 0.5x_2^2$$

Equivalent IP model

$$\max Z = 3x_1 + 5x_2 - 0.5(0y_0 + 1y_1 + 4y_2 + 9y_3 + 16y_4 + 25y_5 + 36y_6)$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$y_0 + y_1 + y_2 + y_3 + y_4 + y_5 + y_6 = 1$$

$$0y_0 + 1y_1 + 2y_2 + 3y_3 + 4y_4 + 5y_5 + 6y_6 = x_2$$

$$x_1, x_2 \in \mathbb{Z}_+$$

$$y_0, y_1, y_2, y_3, y_4, y_5, y_6 \in \{0, 1\}$$

$$Z^* = 22.5, x_1^* = 4, x_2^* = 3, y_3^* = 1 \text{ (all other variables are zero)}$$

```
using JuMP, GLPK

m = Model(with_optimizer(GLPK.Optimizer))

@variable(m, y[0:6], Bin)
@variable(m, x[1:2] >= 0, Int)

@objective(m, Max, 3x[1]+5x[2] -0.5(sum(j^2 * y[j] for j=0:6)) )
@constraint(m, x[1] <= 4)
@constraint(m, 2x[2] <= 12)
@constraint(m, 3x[1] + 2x[2] <= 18)
@constraint(m, sum(j * y[j] for j=0:6) == x[2] )
@constraint(m, sum(y[j] for j=0:6) == 1)

optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("y = ", JuMP.value.(y))
println("x1 = ", JuMP.value(x[1]))
println("x2 = ", JuMP.value(x[2]))
```

Solution:

```
println("Objective value: ", JuMP.objective_value(m))
println("y = ", JuMP.value.(y))
println("x1 = ", JuMP.value(x[1]))
println("x2 = ", JuMP.value(x[2]))
```

```
Objective value: 22.5
y = 1-dimensional DenseAxisArray{Float64,1,...} with index sets:
    Dimension 1, 0:6
And data, a 7-element Array{Float64,1}:
0.0
0.0
0.0
1.0
0.0
0.0
0.0
x1 = 4.0
x2 = 3.0
```

“ugly” output because y-index runs from 0:

```
@variable(m, y[0:6], Bin)
```

Relation between integer programming and linear programming



Integer programming (IP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \in \mathbb{Z}^+ \quad \forall j \in \{1, \dots, n\}$$

Linear programming (LP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \geq 0 \quad \forall j \in \{1, \dots, n\}$$

What is the relation between Z_{IP} and Z_{LP} ?

1. $Z_{LP} = Z_{IP}$?
2. $Z_{LP} \leq Z_{IP}$?
3. $Z_{LP} \geq Z_{IP}$?
4. There is no simple relation between Z_{IP} and Z_{LP} ?

Relation between integer programming and linear programming



Integer programming (IP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \in \mathbb{Z}^+ \quad \forall j \in \{1, \dots, n\}$$

Linear programming (LP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \geq 0 \quad \forall j \in \{1, \dots, n\}$$

What is the relation between Z_{IP} and Z_{LP} ?

1. $Z_{LP} = Z_{IP}$?

2. $Z_{LP} \leq Z_{IP}$?

3. $Z_{LP} \geq Z_{IP}$?

Correct answer

4. There is no simple relation between Z_{IP} and Z_{LP} ?

Relation between integer programming and linear programming

Integer programming (IP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \in \mathbb{Z}^+ \quad \forall j \in \{1, \dots, n\}$$

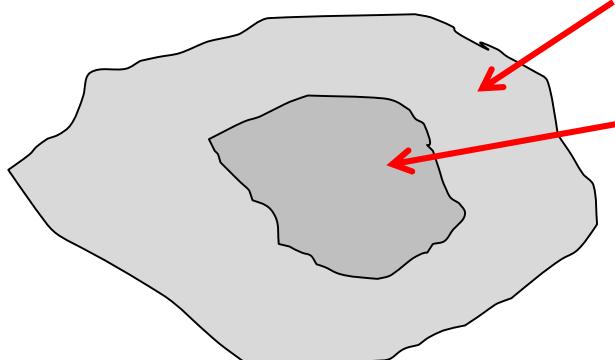
Linear programming (LP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \geq 0 \quad \forall j \in \{1, \dots, n\}$$



Set of feasible solutions to the LP
(drawn in an abstract way)

Set of feasible solutions to the IP

Relation between integer programming and linear programming

Integer programming (IP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \in \mathbb{Z}^+ \quad \forall j \in \{1, \dots, n\}$$

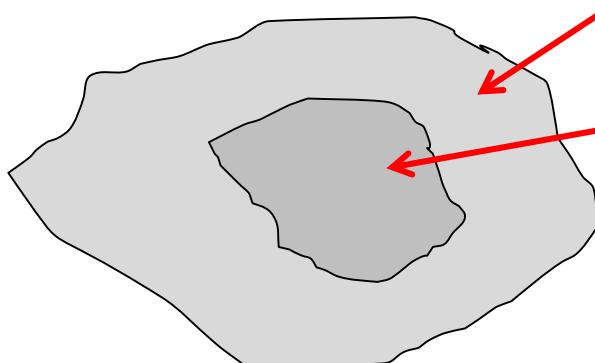
Linear programming (LP):

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

subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i \quad \forall i \in \{1, \dots, m\}$$

$$x_j \geq 0 \quad \forall j \in \{1, \dots, n\}$$



Set of feasible solutions to the LP
(drawn in an abstract way)

Set of feasible solutions to the IP

If $A \subseteq B$ then maximizing over B is going to give the same or a better solution as maximizing over A

Recall what we saw for the modified Wyndor problem:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 19$$

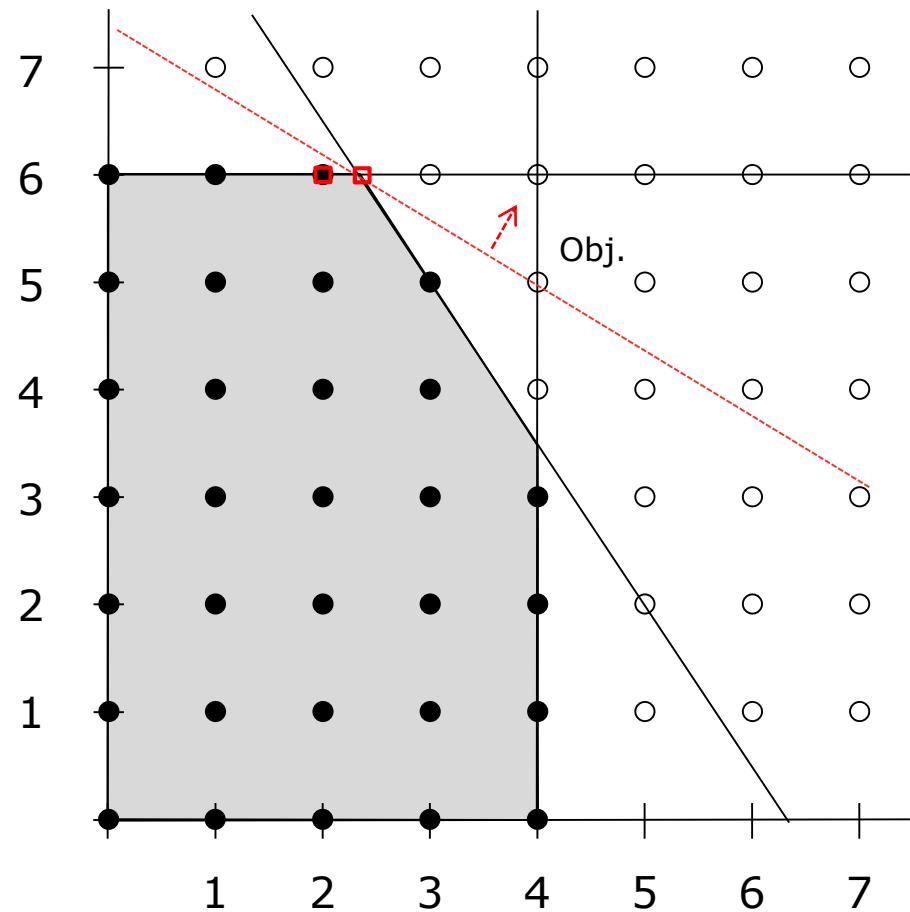
$$x_1, x_2 \in \mathbb{Z}^+$$

Optimal LP solution moved to

$$Z_{LP}^* = 37, x_1^* = 2\frac{1}{3}, x_2^* = 6$$

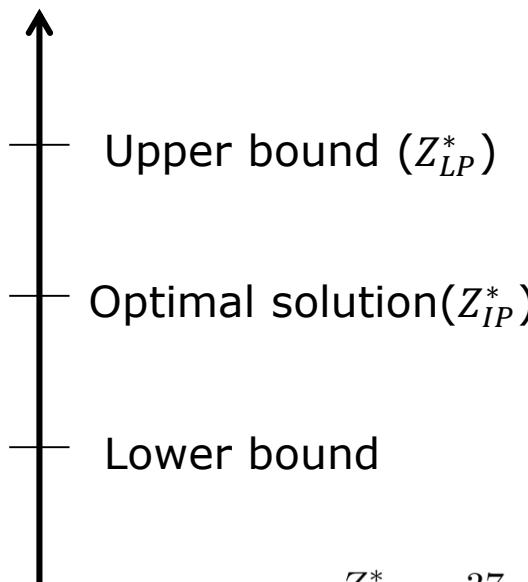
Optimal IP solution "stuck at"

$$Z_{IP}^* = 36, x_1^* = 2, x_2^* = 6$$



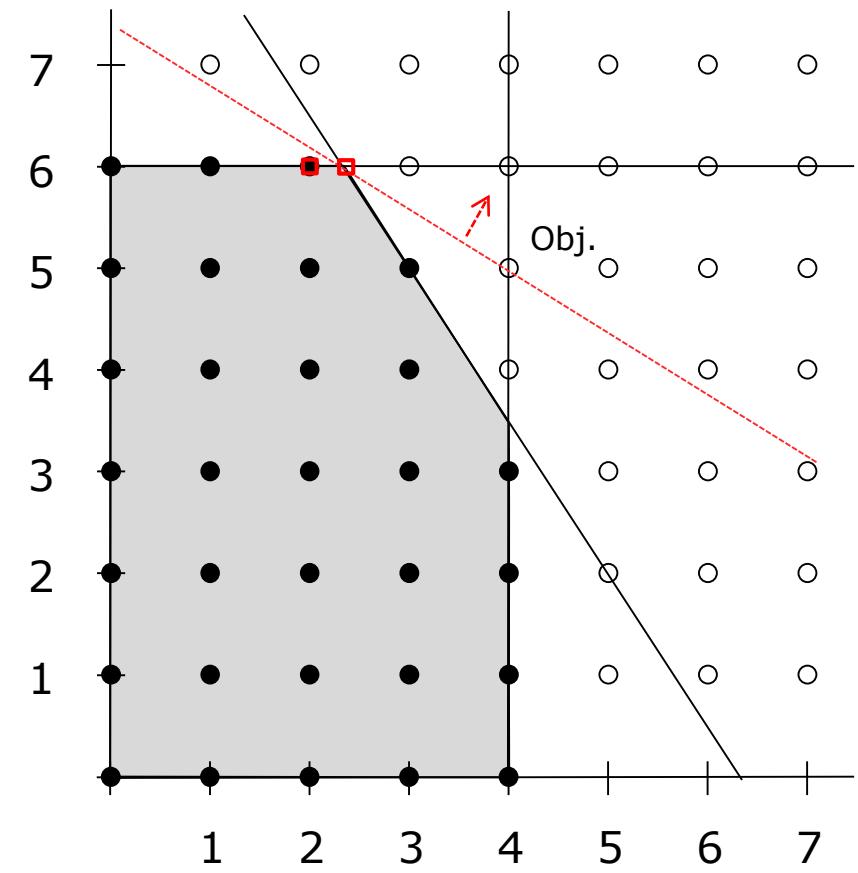
Upper and lower bounds on objective value (maximization)

- Integer programming problem with maximize objective function.
- Upper bound: Solve LP where all integer requirements have been removed (*LP relaxation*).
- Lower bound: Every feasible solution to the integer programming problem gives a lower bound.



$$Z_{LP}^* = 37, x_1^* = 2\frac{1}{3}, x_2^* = 6$$

$$Z_{IP}^* = 36, x_1^* = 2, x_2^* = 6$$



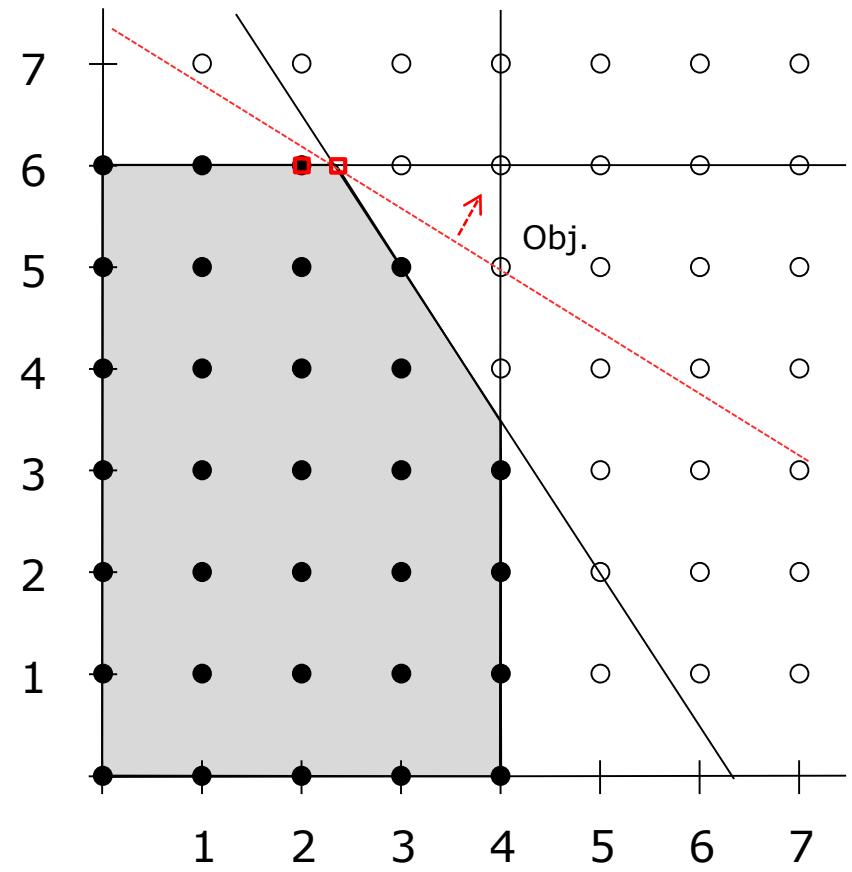
How to find a feasible integer solution

- One could try rounding all fractional variables in the LP solution to the nearest integer
- In the modified wyndor examples this works well.

$$Z_{LP}^* = 37, x_1^* = 2\frac{1}{3}, x_2^* = 6$$

$$x_1^{round} = 2, x_2^{round} = 6, Z^{round} = 36$$

- We happened to find the optimal integer solution.



How to find a feasible integer solution

- One could try rounding all fractional variables in the LP relaxation to the nearest integer.

- Consider

$$\max 3x_1 + 5x_2$$

subject to

$$\frac{1}{3}x_1 + x_2 \leq 4\frac{2}{3}$$

$$4x_1 + x_2 \leq 16$$

$$x_1, x_2 \in \mathbb{Z}_+$$

- Now we get

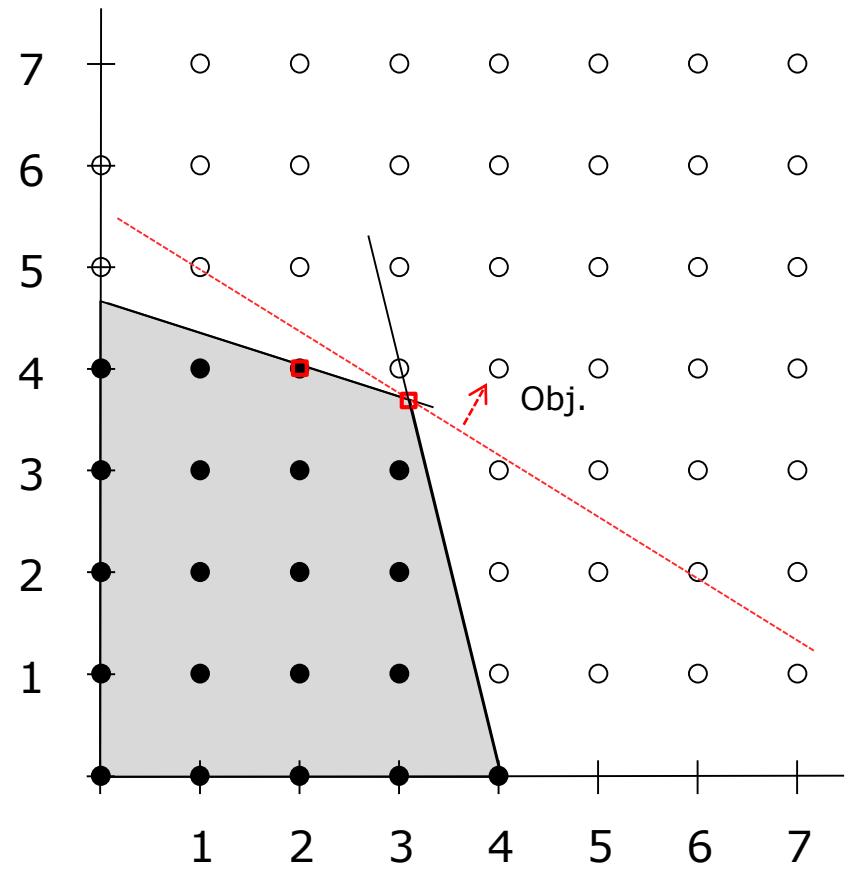
$$Z_{IP}^* = 26, x_1^* = 2, x_2^* = 4$$

$$Z_{LP}^* = \frac{302}{11} \approx 27.45, x_1^* = \frac{34}{11} \approx 3.09, x_2^* = \frac{40}{11} \approx 3.63$$

Unfortunately, the rounded solution

$$x_1^{round} = 3, x_2^{round} = 4$$

is infeasible



How to find a feasible integer solution

$$\max 3x_1 + 5x_2$$

subject to

$$\frac{1}{3}x_1 + x_2 \leq 4\frac{2}{3}$$

$$4x_1 + x_2 \leq 16$$

$$x_1, x_2 \in \mathbb{Z}_+$$

- $x_1^{LP} = \frac{34}{11} \approx 3.09, x_2^{LP} = \frac{40}{11} \approx 3.63$

One could try combinations of rounding up/down all variables with fractional values in the LP relaxation

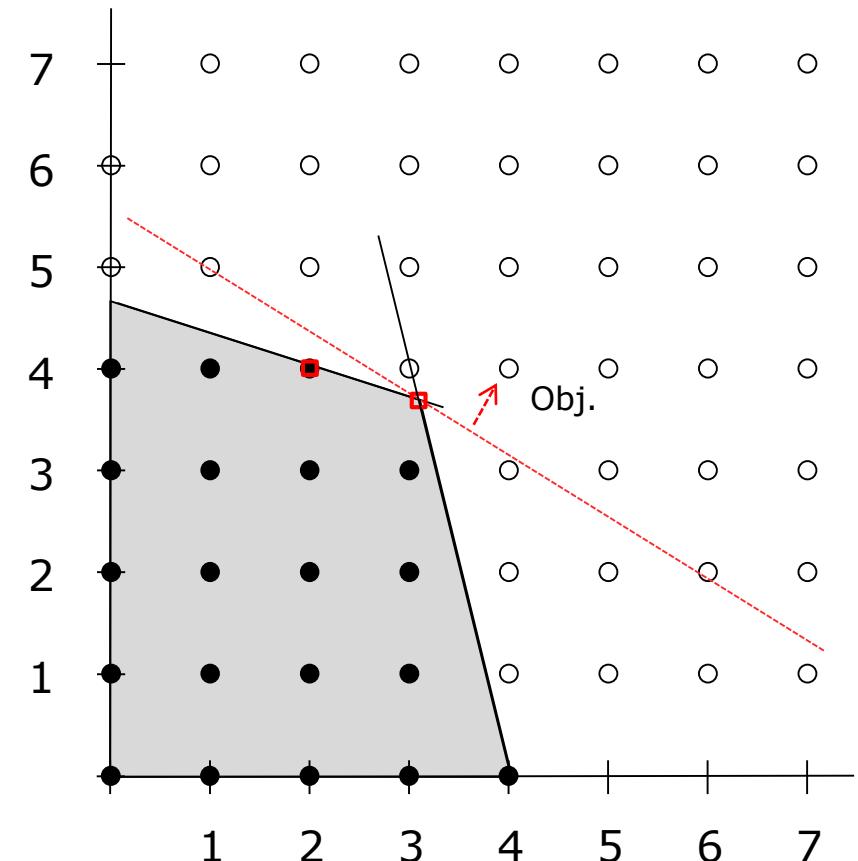
$$(x_1, x_2) = (\lfloor 3.09 \rfloor, \lfloor 3.63 \rfloor) = (3, 3)$$

$$(x_1, x_2) = (\lfloor 3.09 \rfloor, \lceil 3.63 \rceil) = (3, 4)$$

$$(x_1, x_2) = (\lceil 3.09 \rceil, \lfloor 3.63 \rfloor) = (4, 3)$$

$$(x_1, x_2) = (\lceil 3.09 \rceil, \lceil 3.63 \rceil) = (4, 4)$$

- Only one was feasible



How to find a feasible integer solution

$$\max 3x_1 + 5x_2$$

subject to

$$\frac{1}{3}x_1 + x_2 \leq 4\frac{2}{3}$$

$$4x_1 + x_2 \leq 16$$

$$x_1, x_2 \in \mathbb{Z}_+$$

- $x_1^{LP} = \frac{34}{11} \approx 3.09, x_2^{LP} = \frac{40}{11} \approx 3.63$

One could try combinations of rounding up/down all variables with fractional values in the LP relaxation

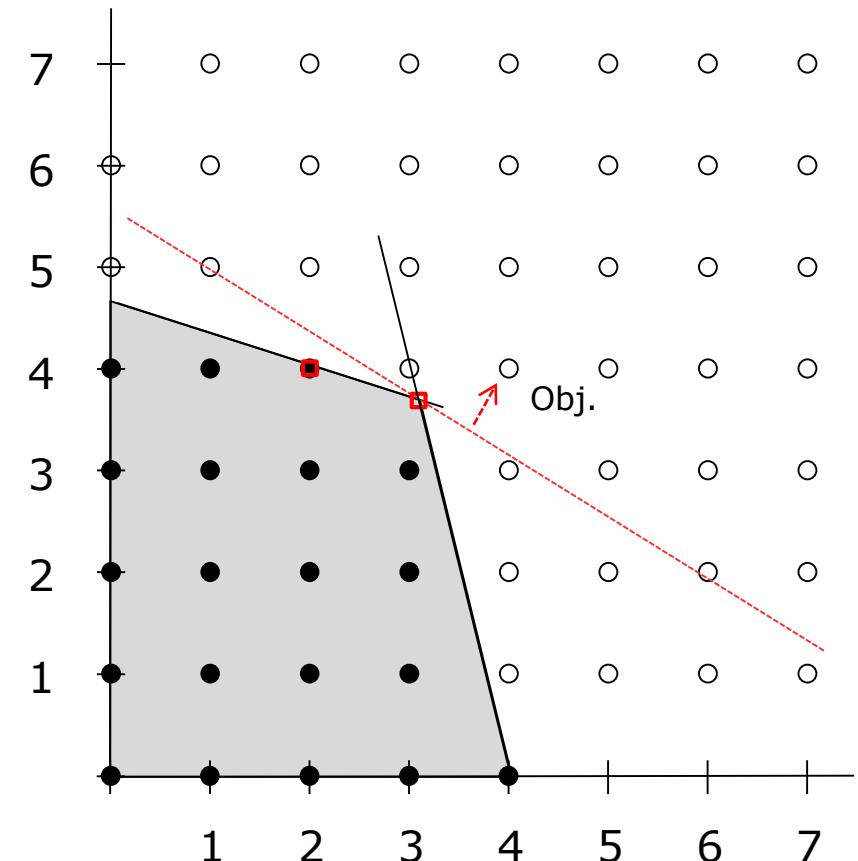
$$(x_1, x_2) = (\lfloor 3.09 \rfloor, \lfloor 3.63 \rfloor) = (3, 3)$$

$$(x_1, x_2) = (\lfloor 3.09 \rfloor, \lceil 3.63 \rceil) = (3, 4)$$

$$(x_1, x_2) = (\lceil 3.09 \rceil, \lfloor 3.63 \rfloor) = (4, 3)$$

$$(x_1, x_2) = (\lceil 3.09 \rceil, \lceil 3.63 \rceil) = (4, 4)$$

- Only one was feasible
- In general we would have to try 2^k rounding combinations where k is the number of variables that takes fractional value in the LP relaxation.
- 2^k quickly grows very large.



How to find a feasible integer solution

$$\max x_2$$

subject to

$$-7x_1 + x_2 \leq 0$$

$$7x_1 + x_2 \leq 7$$

$$x_1, x_2 \in \mathbb{Z}_+$$

- $x_1^{LP} = \frac{1}{2}, x_2^{LP} = \frac{7}{2}$

One could try combinations of rounding up/down all variables with fractional values in the LP relaxation

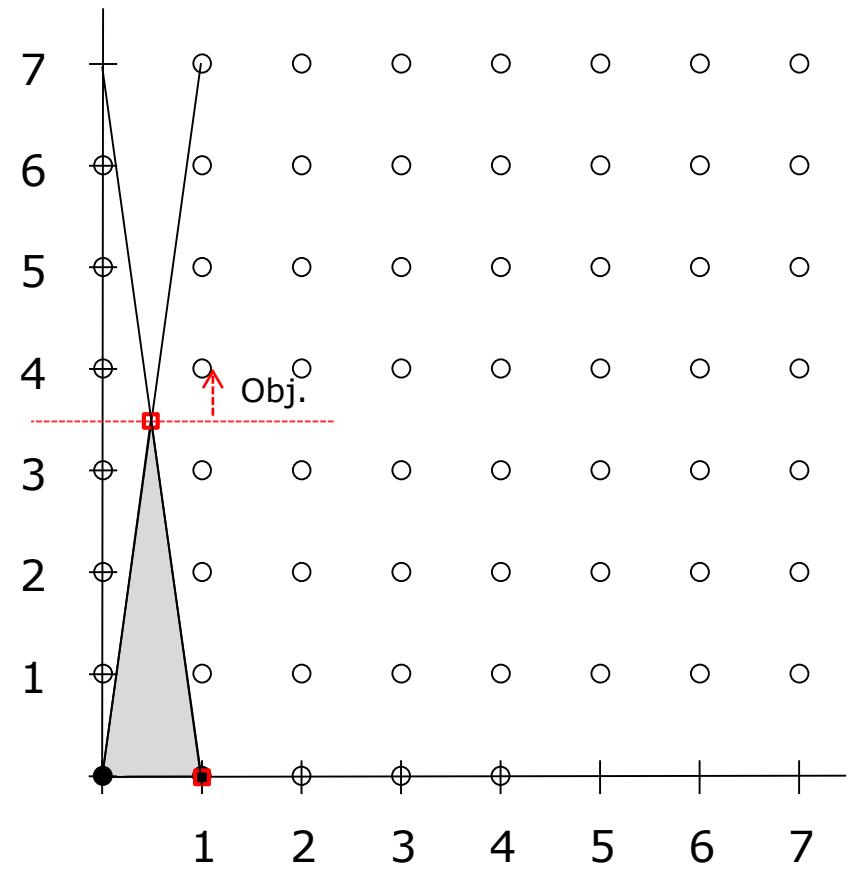
$$(x_1, x_2) = (\lfloor 0.5 \rfloor, \lceil 3.5 \rceil) = (0, 3)$$

$$(x_1, x_2) = (\lfloor 0.5 \rfloor, \lceil 3.5 \rceil) = (0, 4)$$

$$(x_1, x_2) = (\lceil 0.5 \rceil, \lfloor 3.5 \rfloor) = (1, 3)$$

$$(x_1, x_2) = (\lceil 0.5 \rceil, \lfloor 3.5 \rfloor) = (1, 4)$$

- None are feasible.
- Clearly, rounding is not enough to give us feasible integer solutions.



How to find a feasible integer solution?

- Later in the course we will see the *branch-and-bound* method.
- This method will always find the optimal solution (subject to some assumptions)
- The solvers in Julia use an advanced branch-and-bound method to solve integer programming problems.
- Unfortunately, branch-and-bound can be very time-consuming.

Solving Integer programming problems

- Solving LPs is "easy". We can solve (almost) all real-life problems within a reasonable amount of time (seconds, minutes, hours).
- Solving IPs is difficult. For some real-life problems it is (currently) not possible to find the optimal solution within a reasonable amount of time [[NP-hard problem](#)].

Solving Integer programming problems

- Solving LPs is "easy". We can solve (almost) all real-life problems within a reasonable amount of time (seconds, minutes, hours).
- Solving IPs is difficult. For some real-life problems it is (currently) not possible to find the optimal solution within a reasonable amount of time [[NP-hard problem](#)].
- Sometimes we can do much better than the standard solvers by designing an algorithm specialized for a specific type of IP.
- The difficulty of solving IPs does not stop us from using these types of models. There are simply too many nice applications!

Facility location

Parameters	
f_i	cost of opening facility $i \in I$
c_{ij}	cost of serving customer $j \in J$ using facility $i \in I$
Decision variables	
$y_i \in \{0, 1\}$	1 if facility $i \in I$ is opened. Zero otherwise
$x_{ij} \in \{0, 1\}$	1 if customer j is served by facility i . 0 otherwise.

$$\min \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i$$

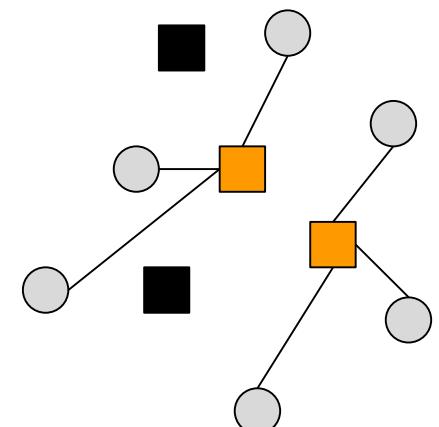
subject to

$$\sum_{i \in I} x_{ij} = 1 \quad \forall j \in J$$

$$x_{ij} \leq y_i \quad \forall i \in I, j \in J$$

$$x_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J$$

$$y_i \in \{0, 1\} \quad \forall i \in I$$



Exercise time!

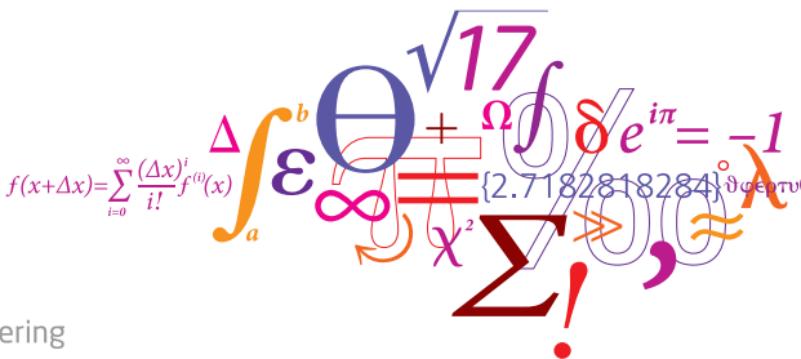
Maybe generalized set partitioning example

- Could be VRP with a bunch of predefined routes?
- or berth allocation?
- or aircraft stand allocation?
- or packing rectangles in a larger rectangle? Could make a simple instance they could try to solve by hand? With some extra restrictions? Some kind of balance constraint?

Modelling with Integer Variables II

Richard M. Lusby

DTU Management



Today's Agenda

- ① Modelling with binary variables
- ② Example formulations
- ③ Solve simple integer programs in Julia

Learning Objectives

At the end of today's lecture you should be able to

- ① Utilize auxiliary binary variables when modelling integer programs
- ② Model different logical constructions with integer linear programming
- ③ Know what the set covering/partitioning problem is
- ④ Solve simple integer programs in Julia

Innovative Use of Binary Variables (12.3)

Either-Or Constraints I

- Consider the case when a choice can be made between two constraints
- At most one of the two **must** hold
- Arises in situations where there might be a choice as to which of two resources is used for a certain purpose
- Also typically arises when enforcing sequencing requirements

$$3x_1 + 2x_2 \leq 18$$

$$x_1 + 4x_2 \leq 16$$

How do we make sure at most one of the two constraints must hold?

Innovative Use of Binary Variables (12.3)

Either-Or Constraints I

- Consider the case when a choice can be made between two constraints
- At most one of the two **must** hold
- Arises in situations where there might be a choice as to which of two resources is used for a certain purpose
- Also typically arises when enforcing sequencing requirements

$$3x_1 + 2x_2 \leq 18$$

$$x_1 + 4x_2 \leq 16$$

How do we make sure at most one of the two constraints must hold?

Innovative Use of Binary Variables (12.3)

Either-Or Constraints II

- Assuming M is large enough, we effectively have the two cases

$$3x_1 + 2x_2 \leq 18$$

$$x_1 + 4x_2 \leq 16 + M$$

$$3x_1 + 2x_2 \leq 18 + M$$

$$x_1 + 4x_2 \leq 16$$

- Adding a **large** constant term to the right hand side eliminates the constraint
- An **auxiliary binary variable** y then selects between the two alternatives
- The resulting formulation is the following

$$3x_1 + 2x_2 \leq 18 + M \cdot y$$

$$x_1 + 4x_2 \leq 16 + M \cdot (1 - y)$$

- The binary variable y is equal to one if the second constraint is enforced

Could we model this differently?

Innovative Use of Binary Variables (12.3)

Either-Or Constraints II

- Assuming M is large enough, we effectively have the two cases

$$3x_1 + 2x_2 \leq 18$$

$$x_1 + 4x_2 \leq 16 + M$$

$$3x_1 + 2x_2 \leq 18 + M$$

$$x_1 + 4x_2 \leq 16$$

- Adding a **large** constant term to the right hand side eliminates the constraint
- An **auxiliary binary variable** y then selects between the two alternatives
- The resulting formulation is the following

$$3x_1 + 2x_2 \leq 18 + M \cdot y$$

$$x_1 + 4x_2 \leq 16 + M \cdot (1 - y)$$

- The binary variable y is equal to one if the second constraint is enforced

Could we model this differently?

Innovative Use of Binary Variables (12.3)

K Out of N Constraints Must Hold

- Assume we have a model with N possible constraints
- Assume at most K of the constraints must hold ($K < N$)
- Identify the combination of K constraints that permits the best objective value
- Generalization of Either-Or Constraints to an arbitrary number of constraints

$$f_1(x_1, x_2, \dots, x_n) \leq d_1$$

$$f_2(x_1, x_2, \dots, x_n) \leq d_2$$

 \vdots

$$f_N(x_1, x_2, \dots, x_n) \leq d_N$$

$$f_1(x_1, x_2, \dots, x_n) \leq d_1 + M \cdot y_1$$

$$f_2(x_1, x_2, \dots, x_n) \leq d_2 + M \cdot y_2$$

 \vdots

$$f_N(x_1, x_2, \dots, x_n) \leq d_N + M \cdot y_N$$

- **Auxiliary binary variable** y_i controls the deactivation of constraint i

$$\sum_{i=1}^N y_i \leq N - K$$

Innovative Use of Binary Variables (12.3)

K Out of *N* Constraints Must Hold

- Assume we have a model with N possible constraints
- Assume at most K of the constraints must hold ($K < N$)
- Identify the combination of K constraints that permits the best objective value
- Generalization of Either-Or Constraints to an arbitrary number of constraints

$$f_1(x_1, x_2, \dots, x_n) \leq d_1$$

$$f_2(x_1, x_2, \dots, x_n) \leq d_2$$

 \vdots

$$f_N(x_1, x_2, \dots, x_n) \leq d_N$$

$$f_1(x_1, x_2, \dots, x_n) \leq d_1 + M \cdot y_1$$

$$f_2(x_1, x_2, \dots, x_n) \leq d_2 + M \cdot y_2$$

 \vdots

$$f_N(x_1, x_2, \dots, x_n) \leq d_N + M \cdot y_N$$

- **Auxiliary binary variable** y_i controls the deactivation of constraint i

$$\sum_{i=1}^N y_i \leq N - K$$

Innovative Use of Binary Variables (12.3)

K Out of *N* Constraints Must Hold

- Assume we have a model with N possible constraints
- Assume at most K of the constraints must hold ($K < N$)
- Identify the combination of K constraints that permits the best objective value
- Generalization of Either-Or Constraints to an arbitrary number of constraints

$$f_1(x_1, x_2, \dots, x_n) \leq d_1$$

$$f_2(x_1, x_2, \dots, x_n) \leq d_2$$

 \vdots

$$f_N(x_1, x_2, \dots, x_n) \leq d_N$$

$$f_1(x_1, x_2, \dots, x_n) \leq d_1 + M \cdot y_1$$

$$f_2(x_1, x_2, \dots, x_n) \leq d_2 + M \cdot y_2$$

 \vdots

$$f_N(x_1, x_2, \dots, x_n) \leq d_N + M \cdot y_N$$

- **Auxiliary binary variable** y_i controls the deactivation of constraint i

$$\sum_{i=1}^N y_i \leq N - K$$

Innovative Use of Binary Variables (12.3)

Functions with N Possible Values

- Consider the situation where a given function can take one of N values
- This requirement can be generally stated as follows:

$$f(x_1, x_2, \dots, x_n) = d_1 \quad \text{or} \quad d_2 \quad \text{or} \quad \dots \quad \text{or} \quad d_N$$

- This situation could arise when a constraint has a number of possible right hand side values, or when a decision variable can assume one of N given values

$$f(x_1, x_2, \dots, x_n) = \sum_{j=1}^n a_j x_j$$

- Using **auxiliary binary variables** y we can reformulate this as follows

$$f_1(x_1, x_2, \dots, x_n) = \sum_{i=1}^N d_i y_i$$

$$\sum_{i=1}^N y_i = 1$$

Innovative Use of Binary Variables (12.3)

Functions with N Possible Values - Example

- Recall the Wyndor Glass Company's problem
- Originally 18 hours were available for production at Plant 3
- Now suppose production at Plant 3 should be restricted to 6, 12, or 18 hours

Original Model:

$$3x_1 + 2x_2 \leq 18$$

Revised Model:

$$3x_1 + 2x_2 = 6y_1 + 12y_2 + 18y_3$$

$$y_1 + y_2 + y_3 = 1$$

$$y_i \in \{0, 1\} \quad \text{for } i = 1, 2, 3$$

- One constraint in the original model is replaced by two constraints and three auxiliary binary variables

Innovative Use of Binary Variables (12.3)

Functions with N Possible Values - Example

- Recall the Wyndor Glass Company's problem
- Originally 18 hours were available for production at Plant 3
- Now suppose production at Plant 3 should be restricted to 6, 12, or 18 hours

Original Model:

$$3x_1 + 2x_2 \leq 18$$

Revised Model:

$$3x_1 + 2x_2 = 6y_1 + 12y_2 + 18y_3$$

$$y_1 + y_2 + y_3 = 1$$

$$y_i \in \{0, 1\} \quad \text{for } i = 1, 2, 3$$

- One constraint in the original model is replaced by two constraints and three **auxiliary binary variables**

Innovative Use of Binary Variables (12.3)

Binary Representation of General Integer Variables

- Suppose we have defined an integer variable x with the following bounds

$$0 \leq x \leq u$$

- If we define N to be the integer such that $2^N \leq u < 2^{N+1}$, then the **binary representation** of x is

$$x = \sum_{i=0}^N 2^i y_i$$

- Each y_i is an **auxiliary binary variable**
- Substituting this binary representation for each of the general integer variables will reduce the entire problem to a **binary integer program**
- More effective algorithms typically exist for binary integer programs than general integer programs

Innovative Use of Binary Variables (12.3)

Binary Representation of General Integer Variables - Example

- Suppose we have two non-negative general integer variables x_1 and x_2 and that the constraint set includes

$$x_1 \leq 5$$

$$2x_1 + 3x_2 \leq 30$$

- The constraints imply that $0 \leq x_1 \leq 5$ and that $0 \leq x_2 \leq 10$
- Therefore $N = 2$ for x_1 and $N = 3$ for x_2 , and the binary representations are:

$$x_1 = y_0 + 2y_1 + 4y_2$$

$$x_2 = y_3 + 2y_4 + 4y_5 + 8y_6$$

- After substitution, the two constraints become

$$y_0 + 2y_1 + 4y_2 \leq 5$$

$$2y_0 + 4y_1 + 8y_2 + 3y_3 + 6y_4 + 12y_5 + 24y_6 \leq 30$$

Innovative Use of Binary Variables (12.3)

Key take aways

- Auxiliary binary variables can be used (in combination) to model combinatorial relationships in terms of questions that must be answered yes or no
- Such variables are not really decision variables, but are rather variables that help model the problem as an integer program
- Adding a significant number of auxiliary binary variables might result in a **computationally intractable model**

Example 1 (12.4)

Making Decisions with Continuous Variables

Good Products Company Case

The Research and Development Division of the company has developed three possible new products. However, to avoid undue diversification of the company's product line, management has imposed the following restriction.

Restriction 1:

From the three possible new products, **at most two** can be produced.

Each of these products can be produced in either of the two plants. For administrative reasons, management has imposed a second restriction in this regard.

Restriction 2:

Just one of the two plants can be the sole producer of the new products

Example 1 (12.4)

Making Decisions with Continuous Variables

Data

	Production Time Per Unit			
	Product 1	Product 2	Product 3	Available Time
Plant 1	3 hours	4 hours	2 hours	30 hours
Plant 2	4 hours	6 hours	2 hours	40 hours
Unit Profit	5	7	3	(thousands of dollars)
Sales Potential	7	5	9	(units per week)

Example 1 (12.4)

A First Model

$$\begin{aligned} \text{Maximize: } & Z = 5x_1 + 7x_2 + 3x_3 \\ \text{s.t. } & 3x_1 + 4x_2 + 2x_3 \leq 30 \\ & 4x_1 + 6x_2 + 2x_3 \leq 40 \\ & x_1 \leq 7 \\ & x_2 \leq 5 \\ & x_3 \leq 9 \\ & x_1, x_2, x_3 \geq 0 \end{aligned}$$

Example 1 (12.4)

First Restriction

- The first restriction states that at most two of x_1, x_2 , and x_3 can be positive
- Control the value of each continuous variable x_i by an auxiliary binary y_i
- We introduce a sufficiently large constant term M and define the constraints

$$x_1 \leq M \cdot y_1$$

$$x_2 \leq M \cdot y_2$$

$$x_3 \leq M \cdot y_3$$

- To ensure at most two of x_1, x_2 , and x_3 are positive we also add

$$y_1 + y_2 + y_3 \leq 2$$

- Setting y_i to zero ensure $x_i = 0$, while setting y_i to one ensures $x_i \leq M$

Example 1 (12.4)

Second Restriction

- The second restriction required us to choose which plant to produce at
- Introduce an auxiliary binary variable y_4 to ensure that there is **either** production at Plant 1 **or** at Plant 2
- Introduce a sufficiently large constant M to help deactivate constraints
- Update the constraints as follows:

Original Model:

$$3x_1 + 4x_2 + 2x_3 \leq 30$$

$$4x_1 + 6x_2 + 2x_3 \leq 40$$

Revised Model:

$$3x_1 + 4x_2 + 2x_3 \leq 30 + M \cdot y_4$$

$$4x_1 + 6x_2 + 2x_3 \leq 40 + M \cdot (1 - y_4)$$

$$y_4 \in \{0, 1\}$$

- Setting $y_4 = 1$ ensures that the capacity constraint at Plant 2 is enforced

Example 1 (12.4)

Full Model

$$\text{Maximize: } Z = 5x_1 + 7x_2 + 3x_3$$

$$\text{s.t. } 3x_1 + 4x_2 + 2x_3 \leq 30 + M \cdot y_4$$

$$4x_1 + 6x_2 + 2x_3 \leq 40 + M \cdot (1 - y_4)$$

$$x_1 \leq 7$$

$$x_2 \leq 5$$

$$x_3 \leq 9$$

$$x_1 \leq M \cdot y_1$$

$$x_2 \leq M \cdot y_2$$

$$x_3 \leq M \cdot y_3$$

$$y_1 + y_2 + y_3 \leq 2$$

$$x_i \geq 0 \quad \text{for } i = 1, 2, 3$$

$$y_i \in \{0, 1\} \quad \text{for } i = 1, 2, 3, 4$$

Example 1 (12.4)

Julia

```
1  using JuMP
2  using GLPK
3
4  m = Model(with_optimizer(GLPK.Optimizer))
5
6  bigM=30
7
8  VarUB = [7.6 5.0 9.0]
9
10 @variable(m, x[1:3])
11 @variable(m, y[1:4], Bin)
12
13 @objective(m, Max, 5*x[1]+7*x[2]+3*x[3])
14
15 @constraint(m, 3*x[1]+4*x[2]+2*x[3] <= 30+bigM*y[4])
16 @constraint(m, 4*x[1]+6*x[2]+2*x[3] <= 40+bigM*(1-y[4]))
17 @constraint(m, [i=1:3], 0<=x[i]<=VarUB[i])
18 @constraint(m, [i=1:3], x[i]<=bigM*y[i])
19 @constraint(m, sum(y[1] for i=1:3) == 2)
20
21 optimize!(m)
22
23 println("Objective Value: ", JuMP.objective_value(m))
24 println("x1 = ", JuMP.value(x[1]))
25 println("x2 = ", JuMP.value(x[2]))
26 println("x3 = ", JuMP.value(x[3]))
```

Example 2 (12.4)

Violating Proportionality

The SUPERSUDS CORPORATION is developing marketing plans for next year's new products. For three of these products, the decision has been made to purchase five TV spots for commercials on television. The table below shows the impact of allocating 0,1,2, or 3 spots to each product. Impact is measured in terms of profit (millions of dollars) resulting from the spots.

Data

No. of Spots	Profit Product		
	1	2	3
0	0	0	0
1	1	0	-1
2	3	2	2
3	3	3	4

How many slots would you assign to each product?

Example 2 (12.4)

Violating Proportionality

The SUPERSUDS CORPORATION is developing marketing plans for next year's new products. For three of these products, the decision has been made to purchase five TV spots for commercials on television. The table below shows the impact of allocating 0,1,2, or 3 spots to each product. Impact is measured in terms of profit (millions of dollars) resulting from the spots.

Data

No. of Spots	Profit Product		
	1	2	3
0	0	0	0
1	1	0	-1
2	3	2	2
3	3	3	4

How many slots would you assign to each product?

Example 2 (12.4)

Violating Proportionality - Formulation 1

- We might be tempted to define decision variables x_1, x_2 , and x_3 to count the number of slots assigned to each product
- However, we **cannot** write a **linear** objective function with these variables
- Define **auxiliary binary variables** y_{ij} for each positive integer value of $x_i = j$

$$y_{ij} = \begin{cases} 1 & \text{if } x_i = j \text{ for } i = 1, 2, 3 \text{ and } j = 1, 2, 3 \\ 0 & \text{otherwise,} \end{cases}$$

Example 2 (12.4)

Violating Proportionality - Formulation 1

Maximize: $y_{11} + 3y_{12} + 3y_{13} + 0y_{21} + 2y_{22} + 3y_{23} - y_{31} + 2y_{32} + 4y_{33}$

s.t. $y_{11} + y_{12} + y_{13} \leq 1$

$$y_{21} + y_{22} + y_{23} \leq 1$$

$$y_{31} + y_{32} + y_{33} \leq 1$$

$$y_{11} + 2y_{12} + 3y_{13} + y_{21} + 2y_{22} + 3y_{23} + y_{31} + 2y_{32} + 3y_{33} = 5$$

$$y_{ij} \in \{0, 1\} \quad \text{for } i = 1, 2, 3 \text{ and } j = 1, 2, 3$$

Optimal solution: $y_{12} = 1, y_{33} = 1$

Example 2 (12.4)

Violating Proportionality - Formulation 2

- Alternatively we can model the problem with the following y_{ij} variables

$$y_{ij} = \begin{cases} 1 & \text{if } x_i \geq j \text{ for } i = 1, 2, 3 \text{ and } j = 1, 2, 3 \\ 0 & \text{otherwise,} \end{cases}$$

- For example, $x_i = 3 \Rightarrow y_{i1} = 1, y_{i2} = 1, y_{i3} = 1$
- And so, $x_i = y_{i1} + y_{i2} + y_{i3}$
- We need to enforce relationships between y_{ij} variables
- For example, we cannot set $y_{i2} = 1$ without also setting $y_{i1} = 1$ and we cannot set $y_{i3} = 1$ without also setting $y_{i2} = 1$
- The following constraints are therefore necessary for $i = 1, 2, 3$

$$y_{i1} \geq y_{i2}$$

$$y_{i2} \geq y_{i3}$$

Example 2 (12.4)

Violating Proportionality - Formulation 2

- The definition of the objective function also changes with the new variables
- For product i , the y_{ij} variables model incremental changes in the value of x_i
- The incremental changes in objective value for product 1 are 1, 2, and 0
- The objective therefore becomes:

$$\text{Maximize: } y_{11} + 2y_{12} + 0y_{13} + 0y_{21} + 2y_{22} + y_{23} - y_{31} + 3y_{32} + 2y_{33}$$

Example 2

Violating Proportionality - Formulation 2

Maximize: $y_{11} + 2y_{12} + 0y_{13} + 0y_{21} + 2y_{22} + y_{23} - y_{31} + 3y_{32} + 2y_{33}$

s.t.

$$y_{12} - y_{11} \leq 0$$

$$y_{13} - y_{12} \leq 0$$

$$y_{22} - y_{21} \leq 0$$

$$y_{23} - y_{22} \leq 0$$

$$y_{32} - y_{31} \leq 0$$

$$y_{33} - y_{32} \leq 0$$

$$y_{11} + y_{12} + y_{13} + y_{21} + y_{22} + y_{23} + y_{31} + y_{32} + y_{33} = 5$$

$$y_{ij} \in \{0, 1\} \quad \text{for } i = 1, 2, 3 \text{ and } j = 1, 2, 3$$

Optimal solution: $y_{11} = 1, y_{12} = 1, y_{31} = 1, y_{32} = 1, y_{33} = 1$

Example 3 (12.4)

Crew Scheduling

Data

Flight	Feasible Flight Sequences											
	1	2	3	4	5	6	7	8	9	10	11	12
San Francisco to LA	1		1		1				1			
San Francisco to Denver		1		1			1				1	
San Francisco to Seattle			1		1			1				1
LA to Chicago				1		1		1		1		1
LA to San Francisco	1				1				1		1	
Chicago to Denver				1	1				1			
Chicago to Seattle						1	1		1		1	1
Denver to San Francisco	1		1	1					1			
Denver to Chicago					1		1			1		
Seattle to San Francisco			1			1	1					1
Seattle to LA					1			1	1	1	1	1
Cost \$1,000s	2	3	4	6	7	5	7	8	9	9	8	9

Example 3 (12.4)

Crew Scheduling

- There are three crews based in San Francisco
- Each needs to be assigned a sequence of flights
- There are 12 sequences available
- Each sequence contains a subset of 12 flights that can feasibly be flown
- All flights must be **covered** by **at least** one of the three sequences
- A flight can be covered more than once (one crew fly as passengers)
- Find the three flight sequences that minimize the total crew cost

Example 3 (12.4)

Set Covering Problem

- Define the following binary variables

$$x_j = \begin{cases} 1 & \text{if sequence } j \text{ is assigned a crew } j = 1, 2, 3, \dots, 12 \\ 0 & \text{otherwise} \end{cases}$$

- The objective is then:

$$\text{Minimize: } 2x_1 + 3x_2 + 4x_3 + 6x_4 + 7x_5 + 5x_6 + 7x_7 + 8x_8 + 9x_9 + 9x_{10} + 8x_{11} + 9x_{12}$$

- We need to ensure that each flight is covered at least once
- For example, the following would ensure we cover the last flight

$$x_6 + x_9 + x_{10} + x_{11} + x_{12} \geq 1$$

Example 3 (12.4)

Mathematical Formulation

Minimize: $2x_1 + 3x_2 + 4x_3 + 6x_4 + 7x_5 + 5x_6 + 7x_7 + 8x_8 + 9x_9 + 9x_{10} + 8x_{11} + 9x_{12}$

s.t.

$$\begin{aligned} & x_1 + x_4 + x_7 + x_{10} \geq 1 \\ & x_2 + x_5 + x_8 + x_9 \geq 1 \\ & x_3 + x_6 + x_9 + x_{12} \geq 1 \\ & x_4 + x_7 + x_9 + x_{10} + x_{12} \geq 1 \\ & x_1 + x_6 + x_{10} + x_{11} \geq 1 \\ & x_4 + x_5 + x_9 \geq 1 \\ & x_7 + x_8 + x_{10} + x_{11} + x_{12} \geq 1 \\ & x_2 + x_4 + x_5 + x_9 \geq 1 \\ & x_5 + x_8 + x_{11} \geq 1 \\ & x_3 + x_7 + x_8 + x_{12} \geq 1 \\ & x_6 + x_9 + x_{10} + x_{11} + x_{12} \geq 1 \\ & \sum_{j=1}^{12} x_j = 3 \\ & x_j \in \{0, 1\} \text{ for } j = 1, 2, \dots, 12 \end{aligned}$$

Example 3 (12.4)

Some comments

- Optimal solution is $x_1 = 1$, $x_5 = 1$, and $x_{12} = 1$ with a total cost of \$18,000
- This example introduces a broader class of problems **set covering problems**
- Given a set of elements and a set of subsets of the elements, find the minimal number of subsets needed to cover each element at least once. To ensure an element, say i , is covered, we need in general the following constraint:

$$\sum_{j \in S_i} x_j \geq 1$$

where S_i is the set of subsets that contain element i

- In a **set partitioning** problem, each element must be covered **exactly** once

$$\sum_{j \in S_i} x_j = 1$$

- In a **set packing** problem, each element must be covered **at most** once

$$\sum_{j \in S_i} x_j \leq 1$$

Example 3 (12.4)

Julia

```
1  using JuMP
2  using GLPK
3
4  m = Model(with_optimizer(GLPK.Optimizer))
5
6  n = 12
7
8  @variable(m, x[1:12], Bin)
9
10 cost = [2 3 4 6 7 5 7 8 9 9 8 9]
11
12 A = [1 0 0 1 0 0 1 0 0 1 0 0;
13      0 1 0 0 1 0 0 1 0 0 1 0;
14      0 0 1 0 0 1 0 0 1 0 0 1;
15      0 0 0 1 0 0 1 0 1 1 0 1;
16      1 0 0 0 0 1 0 0 0 1 1 0;
17      0 0 0 1 1 0 0 0 1 0 0 0;
18      0 0 0 0 0 0 1 1 0 1 1 1;
19      0 1 0 1 1 0 0 0 1 0 0 0;
20      0 0 0 0 1 0 0 1 0 0 1 0;
21      0 0 1 0 0 0 1 1 0 0 0 1;
22      0 0 0 0 0 1 0 0 1 1 1 1;]
23
24 @objective(m, Min, sum(cost[i]*x[i] for i=1:n))
25
26 @constraint(m, [j=1:size(A)[1]], sum(A[j,i]*x[i] for i=1:n) >= 1)
27 @constraint(m, sum(x[i] for i=1:n) == 3)
28
29 optimize!(m)
30
31 println("Objective Value: ", JuMP.objective_value(m))
32 println("Variable values ", JuMP.value.(x))
```

Learning Objectives

At the end of today's lecture you should be able to

- ① Utilize auxiliary binary variables when modelling integer programs
- ② Model different logical constructions with integer linear programming
- ③ Know what the set covering/partitioning/packing problem is
- ④ Solve simple integer programs in Julia

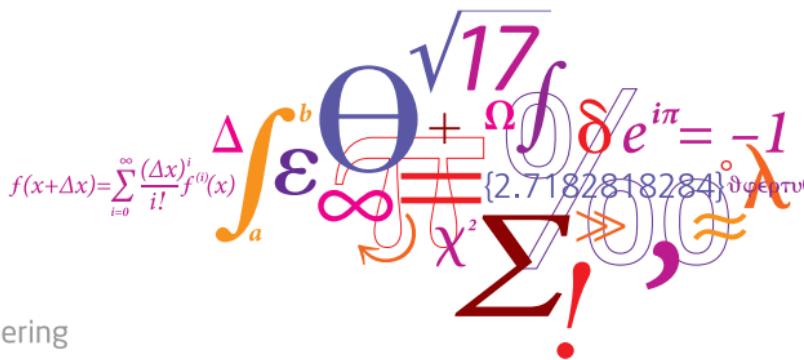
Richard Lusby
Associate Professor, DTU Management Engineering

Building 424, Room 208 rmlu@dtu.dk
2800 Kgs. Lyngby, Denmark +45 4525 3084 phone

Modelling Integer Variables III

Richard M. Lusby

DTU Management



Today's Agenda

- ① Heuristic Approaches
- ② Greedy Constructive Algorithms
- ③ Three examples

Learning Objectives

At the end of today's lecture you should be able to

- ① Understand and explain the difference between Heuristic and Exact Approaches
- ② Know what a Greedy Constructive Algorithm is
- ③ Design a Greedy Constructive Algorithm
- ④ Give examples of when a Greedy Algorithm fails/succeeds

Heuristic Approaches

- Solving integer programs is not easy
- For a problem with n binary variables, there are as many as 2^n possible solutions
- Exponential growth in the number of solutions as n increases
- Exhaustive enumeration is not possible
- **Exact** approaches will find the optimal solution (and a certificate of optimality)
 - The focus of Lecture 11
- **Heuristic** approaches are rules of thumb used to find feasible solutions,
- In general, heuristics do not provide a certificate of optimality (or even quality)
- Heuristics should provide **good** solutions **quickly**

Constructive Algorithm

- Constructive algorithms typically begin with all variables **unassigned**
- Consist of a sequence of **iterations**
- At each iteration one variable is selected and is assigned a **feasible value**
- Feasibility means consistent with **previous** values
- How does one identify the next unassigned variable and set the variable's value?
- Common procedures achieve this in a **greedy** or **myopic** fashion

Greedy Constructive Algorithm

- We take the best **here and now** decision
- Maximize (or minimize) our immediate **profit (cost)**
- Based on a partial solution greedily select the next variable
- Very short sighted (we do not look ahead further than the current step)
- Difficult to assess the full implications of a decision taken now
- We do not **back track** or revise decisions
- Greedy algorithms can be difficult to design
- In some cases will return an **optimal** solution

Greedy Constructive Algorithm

NASA Capital Budgeting Example

The U.S. Space agency, NASA, must decide how to divide its limited budgets among fourteen competing missions for the years 2018-2042. Each mission requires a certain investment (billions of dollars) and has an associated value (a weighted sum reflecting intellectual gains, direct benefit to life on earth etc). Certain missions are incompatible (Missions 4 & 5, Missions 11& 14, and Missions 8 & 9) while Missions 4-7 depend on Mission 3 (i.e, each can only be chosen if Mission 3 is also chosen). Similarly Mission 11 depends on Mission 2.

NASA Capital Budgeting Example

Data

j	Mission	Budget Requirements (\$ billion)					Value
		2018- 2022	2023- 2027	2028- 2032	2033- 2037	2038- 2042	
1	Communications Sat.	6	-	-	-	-	200
2	Orbital Microwave	2	3	-	-	-	3
3	Io Lander	3	5	-	-	-	20
4	Uranus Probe	-	-	-	-	10	50
5	Uranus Probe	-	5	8	-	-	70
6	Mercury Probe	-	-	1	8	4	20
7	Saturn Probe	1	8	-	-	-	5
8	Infrared Imaging	-	-	-	5	-	10
9	Ground-based SETI	4	5	-	-	-	200
10	Large Orbital Structs	-	8	4	-	-	150
11	Color Imaging	-	-	2	7	-	18
12	Medical Technology	5	7	-	-	-	8
13	Polar Orbital Platform	-	1	4	1	1	300
14	Geosynchronous SETI	-	4	5	3	3	185
Budget		10	12	14	14	14	

NASA Capital Budgeting Example

A Binary Integer Program - Variables and Objective

- Define the following binary decision variables

$$x_j = \begin{cases} 1 & \text{if Mission } j \text{ is selected } j = 1, 2, 3, \dots, 14 \\ 0 & \text{otherwise} \end{cases}$$

- This yields the following objective function

Maximize: $200x_1 + 3x_2 + 20x_3 + 50x_4 + 70x_5 + 20x_6 + 5x_7 + 10x_8 + 200x_9 + 150x_{10} + 18x_{11} + 8x_{12} + 300x_{13} + 185x_{14}$

NASA Capital Budgeting Example

A Binary Integer Program - Constraints

$$6x_1 + 2x_2 + 3x_3 + 1x_7 + 4x_9 + 5x_{12} \leq 10$$

$$3x_2 + 5x_3 + 5x_5 + 8x_7 + 5x_9 + 8x_{10} + 7x_{12} + 1x_{13} + 4x_{14} \leq 12$$

$$8x_5 + 1x_6 + 4x_{10} + 2x_{11} + 4x_{13} + 5x_{14} \leq 14$$

$$8x_6 + 5x_8 + 7x_{11} + 1x_{13} + 3x_{14} \leq 14$$

$$10x_4 + 4x_6 + 1x_{13} + 3x_{14} \leq 14$$

$$x_4 + x_5 \leq 1$$

$$x_8 + x_9 \leq 1$$

$$x_{11} + x_{14} \leq 1$$

$$x_{11} \leq x_2$$

$$x_4 \leq x_3$$

$$x_5 \leq x_3$$

$$x_6 \leq x_3$$

$$x_7 \leq x_3$$

$$x_j \in \{0, 1\} \text{ for } i = 1, 2, \dots, 14$$

NASA Capital Budgeting Example

A Binary Integer Program

- Problem is known is essentially a **multidimensional knapsack** problem
- Has five **budget** constraints
 - One for each of the five planning periods
- Contains three **packing** constraints to model **mutually** exclusive choices
- Contains five constraints to model the **dependencies** between missions
- Optimal solution:

$$x_1 = 1, x_3 = 1, x_4 = 1, x_8 = 1, x_{13} = 1, x_{14} = 1, \text{ Obj Value: } 765$$

- We run Missions **1, 3, 4, 8, 13, and 14** yielding a total value of **765**

Julia Implementation

```
1  using JuMP
2  using GLPK
3
4  m = Model(with_optimizer(GLPK.Optimizer))
5
6  profit = [200 3 20 50 70 20 5 10 200 150 18 8 300 185]
7  weights = [6 0 0 0 0; 2 3 0 0 0; 3 5 0 0 0; 0 0 0 0 10; 0 5 8 0 0; 0 0 1 8 4;
8      1 8 0 0 0; 0 0 8 5 0; 4 5 0 0 0; 0 8 4 0 0; 0 0 2 7 0; 5 7 0 0 0;
9      0 1 4 1 1; 0 4 5 3 3]
10
11 budgets = [10 12 14 14 14]
12 n=14
13 b=5
14
15 @variable(m, x[1:n], Bin)
16 @objective(m, Max, sum(profit[i]*x[i] for i=1:n))
17 @constraint(m, {j=1:b}, sum(weights[i,j]*x[i] for i=1:n) ≤ budgets[j])
18 @constraint(m, x[4]+x[5]≤1)
19 @constraint(m, x[8]+x[11]≤1)
20 @constraint(m, x[9]+x[14]≤1)
21 @constraint(m, x[11]≤x[2])
22 @constraint(m, x[4]≤x[3])
23 @constraint(m, x[5]≤x[3])
24 @constraint(m, x[6]≤x[3])
25 @constraint(m, x[7]≤x[3])
26
27 optimize!(m)
28
29 println("Objective Value: ", JuMP.objective_value(m))
30 println("Variable values: ", JuMP.value.(x))
```

NASA Capital Budgeting Example

- Construct a solution by successively adding missions until constraints prevent further inclusions
- We prefer missions with **high value**
- We will also consider the constraints
 - One high valued mission might consume too many resources and **block** additional missions from being included
- Selecting a mission will secure its value but naturally impact future choices
- Dependencies between missions change a mission's implicit value
- Missions will be selected based on comparing mission specific **ratios** that attempt to capture the trade-off between the objective and constraint considerations

NASA Capital Budgeting Example

Mission Ratios

$$r_j = \frac{\text{Mission } j \text{ value} + \text{Allowance for enabled successor values}}{\sum_{i=1}^8 \text{Fraction of remaining amount of constraint } i \text{ consumed by Mission } j}$$

$$r_j = \frac{c_j + \sum_{\text{free } k \text{ preceded by } j} \left(\frac{c_k}{2} \right)}{\sum_{i=1}^8 \left(\frac{a_{ij}}{b_i^{(t)}} \right)}$$

where

c_j = objective coefficient for Mission j

a_{ij} = coefficient for Mission j in the i th constraint

$b_i^{(t)}$ = right hand side remaining in the i th constraint at iteration t

NASA Capital Budgeting Example

Mission Ratios

- The numerator tries to capture the immediate value of the mission, and the potential value its dependent missions
- Half of the value of all enabled successors is **arbitrarily** added
- The denominator sums the fractions of remaining resources that a mission would consume if selected
- We favor missions with a high ratio
- Such missions provide a good return per unit invested

At every iteration t of the greedy heuristic we will choose among the unassigned missions for which all predecessors have been scheduled in the partial solution and fix the value of x_j . If there remain available resources for the mission x_j is assigned the value one, otherwise we set it to zero.

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 0

- Initially all variables are unassigned

$$\mathbf{x}^{(0)} = [\#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#]$$

- All $b_i^{(0)}$ equal the initial right hand side values
- Ratios for Missions 1 & 2 can be computed as follows

$$r_1 = \frac{200}{\frac{6}{10}} = 333.33$$

$$r_2 = \frac{3 + \frac{18}{2}}{\frac{2}{10} + \frac{3}{12}} = 26.57$$

- Similar calculations yield the following:

$$r_3 = 129.07, r_4 = 29.17, r_5 = 35.21, r_6 = 21.54, r_7 = 6.52, r_8 = 7.37, r_9 = 110.09, \\ r_{10} = 157.05, r_{11} = 10.96, r_{12} = 7.38, r_{13} = 586.05, r_{14} = 87.30$$

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 0

- Initially all variables are unassigned

$$\mathbf{x}^{(0)} = [\#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#]$$

- All $b_i^{(0)}$ equal the initial right hand side values
- Ratios for Missions 1 & 2 can be computed as follows

$$r_1 = \frac{200}{\frac{6}{10}} = 333.33$$

$$r_2 = \frac{3 + \frac{18}{2}}{\frac{2}{10} + \frac{3}{12}} = 26.57$$

- Similar calculations yield the following:

$$r_3 = 129.07, r_4 = 29.17, r_5 = 35.21, r_6 = 21.54, r_7 = 6.52, r_8 = 7.37, r_9 = 110.09, \\ r_{10} = 157.05, r_{11} = 10.96, r_{12} = 7.38, \textcolor{red}{r_{13} = 586.05}, r_{14} = 87.30$$

- Fix $x_{13}=1$

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 1

- **One variable value** is now fixed

$$\boldsymbol{x}^{(1)} = [\#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \mathbf{1}, \#]$$

- Right hand sides of the constraints are updated to reflect partial solution

$$b_1^{(1)} = 10, b_2^{(1)} = 11, b_3^{(1)} = 10, b_4^{(1)} = 13, b_5^{(1)} = 13, \\ b_6^{(1)} = 1, b_7^{(1)} = 1, b_8^{(1)} = 1$$

- The updated mission ratios are:

$$r_1 = 333.33, r_2 = 25.38, r_3 = 122.59, r_4 = 28.26, r_5 = 31.05, r_6 = 19.55, \\ r_7 = 6.05, r_8 = 7.22, r_9 = 107.84, r_{10} = 133.06, r_{11} = 10.35, r_{12} = 7.04, \\ r_{13} = \text{N/A}, r_{14} = 79.56$$

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 1

- One variable value is now fixed

$$\boldsymbol{x}^{(1)} = [\#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \mathbf{1}, \#]$$

- Right hand sides of the constraints are updated to reflect partial solution

$$b_1^{(1)} = 10, b_2^{(1)} = 11, b_3^{(1)} = 10, b_4^{(1)} = 13, b_5^{(1)} = 13, \\ b_6^{(1)} = 1, b_7^{(1)} = 1, b_8^{(1)} = 1$$

- The updated mission ratios are:

$$r_1 = 333.33, r_2 = 25.38, r_3 = 122.59, r_4 = 28.26, r_5 = 31.05, r_6 = 19.55, \\ r_7 = 6.05, r_8 = 7.22, r_9 = 107.84, r_{10} = 133.06, r_{11} = 10.35, r_{12} = 7.04, \\ r_{13} = \text{N/A}, r_{14} = 79.56$$

- Fix $x_1=1$

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 2

- **Two variable values** are now fixed

$$\boldsymbol{x}^{(2)} = [\mathbf{1}, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \mathbf{1}, \#]$$

- Right hand sides of the constraints are updated to reflect partial solution

$$b_1^{(2)} = 4, b_2^{(2)} = 11, b_3^{(2)} = 10, b_4^{(2)} = 13, b_5^{(2)} = 13, \\ b_6^{(2)} = 1, b_7^{(2)} = 1, b_8^{(2)} = 1$$

- The updated mission ratios are

$$\begin{aligned} & N/A, r_2 = 15.53, r_3 = 76.79, r_4 = 28.26, r_5 = 31.05, r_6 = 19.55, \\ & r_7 = 5.12, r_8 = 7.22, r_9 = 81.48, r_{10} = 133.06, r_{11} = 10.35, r_{12} = 4.24, \\ & r_{13} = N/A, r_{14} = 79.56 \end{aligned}$$

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 2

- Two variable values are now fixed

$$\boldsymbol{x}^{(2)} = [1, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#, \#]$$

- Right hand sides of the constraints are updated to reflect partial solution

$$b_1^{(2)} = 4, b_2^{(2)} = 11, b_3^{(2)} = 10, b_4^{(2)} = 13, b_5^{(2)} = 13, \\ b_6^{(2)} = 1, b_7^{(2)} = 1, b_8^{(2)} = 1$$

- The updated mission ratios are

$$\text{N/A}, r_2 = 15.53, r_3 = 76.79, r_4 = 28.26, r_5 = 31.05, r_6 = 19.55, \\ r_7 = 5.12, r_8 = 7.22, r_9 = 81.48, \textcolor{red}{r_{10} = 133.06}, r_{11} = 10.35, r_{12} = 4.24, \\ r_{13} = \text{N/A}, r_{14} = 79.56$$

- Fix $x_{10}=1$

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 3

- Three variable values are now fixed

$$\boldsymbol{x}^{(3)} = [1, \#, \#, \#, \#, \#, \#, \#, \#, \#, \mathbf{1}, \#, \#, \mathbf{1}, \#]$$

- Right hand sides of the constraints are updated to reflect partial solution

$$b_1^{(3)} = 4, b_2^{(3)} = 3, b_3^{(3)} = 6, b_4^{(3)} = 13, b_5^{(3)} = 13, \\ b_6^{(3)} = 1, b_7^{(3)} = 1, b_8^{(3)} = 1$$

- The updated mission ratios are

$$\begin{aligned} & N/A, r_2 = 8.00, r_3 = 38.28, r_4 = 28.26, r_5 = 17.50, r_6 = 18.35, \\ & r_7 = 1.71, r_8 = 7.22, r_9 = 54.54, r_{10} = N/A, r_{11} = 9.61, r_{12} = 2.23, \\ & r_{13} = N/A, r_{14} = 50.99 \end{aligned}$$

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 3

- Three variable values are now fixed

$$\boldsymbol{x}^{(3)} = [1, \#, \#, \#, \#, \#, \#, \#, \#, \boldsymbol{1}, \#, \#, \boldsymbol{1}, \#]$$

- Right hand sides of the constraints are updated to reflect partial solution

$$\begin{aligned} b_1^{(3)} &= 4, & b_2^{(3)} &= 3, & b_3^{(3)} &= 6, & b_4^{(3)} &= 13, & b_5^{(3)} &= 13, \\ b_6^{(3)} &= 1, & b_7^{(3)} &= 1, & b_8^{(3)} &= 1 \end{aligned}$$

- The updated mission ratios are

$$\begin{aligned} N/A, r_2 &= 8.00, & r_3 &= 38.28, & r_4 &= 28.26, & r_5 &= 17.50, & r_6 &= 18.35, \\ r_7 &= 1.71, & r_8 &= 7.22, & r_9 &= 54.54, & r_{10} &= N/A, & r_{11} &= 9.61, & r_{12} &= 2.23, \\ r_{13} &= N/A, & r_{14} &= 50.99 \end{aligned}$$

- Fix $x_9=0$ (**violates** constraint 2)

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 9

- Nine variable values are now fixed

$$x^{(9)} = [1, \#, 0, 0, 0, \#, \#, 0, 1, \#, \#, 1, 0]$$

- Right hand sides of the constraints are updated to reflect partial solution

$$\begin{aligned} b_1^{(9)} &= 4, & b_2^{(9)} &= 3, & b_3^{(9)} &= 6, & b_4^{(9)} &= 13, & b_5^{(9)} &= 13, \\ b_6^{(9)} &= 1, & b_7^{(9)} &= 1, & b_8^{(9)} &= 1 \end{aligned}$$

- The updated mission ratios are

$$\begin{aligned} N/A, r_2 &= 8.00, & r_3 &= N/A, & r_4 &= N/A, & r_5 &= N/A, & r_6 &= N/A, \\ r_7 &= 1.71, & r_8 &= 7.22, & r_9 &= N/A, & r_{10} &= N/A, & r_{11} &= 9.61, & r_{12} &= 2.23, \\ r_{13} &= N/A, & r_{14} &= N/A \end{aligned}$$

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 9

- Nine variable values are now fixed

$$x^{(9)} = [1, \#, 0, 0, 0, \#, \#, 0, 1, \#, \#, 1, 0]$$

- Right hand sides of the constraints are updated to reflect partial solution

$$\begin{aligned} b_1^{(9)} &= 4, & b_2^{(9)} &= 3, & b_3^{(9)} &= 6, & b_4^{(9)} &= 13, & b_5^{(9)} &= 13, \\ b_6^{(9)} &= 1, & b_7^{(9)} &= 1, & b_8^{(9)} &= 1 \end{aligned}$$

- The updated mission ratios are

$$\begin{aligned} N/A, & r_2 = 8.00, & r_3 = N/A, & r_4 = N/A, & r_5 = N/A, & r_6 = N/A, \\ r_7 &= 1.71, & r_8 &= 7.22, & r_9 = N/A, & r_{10} = N/A, & r_{11} = 9.61, & r_{12} = 2.23, \\ r_{13} &= N/A, & r_{14} &= N/A \end{aligned}$$

- Fix $x_2=1$ (select **second best** mission ratio)

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 10

- **Ten variable values** are now fixed

$$\boldsymbol{x}^{(10)} = [1, 1, 0, 0, 0, \#, \#, 0, 1, \#, \#, 1, 0]$$

- Right hand sides of the constraints are updated to reflect partial solution

$$b_1^{(10)} = 2, b_2^{(10)} = 0, b_3^{(10)} = 6, b_4^{(10)} = 13, b_5^{(10)} = 13, \\ b_6^{(10)} = 1, b_7^{(10)} = 1, b_8^{(10)} = 1$$

- The updated mission ratios are

$$N/A, N/A, r_3 = N/A, r_4 = N/A, r_5 = N/A, r_6 = N/A,$$

$$r_7 = 0.0000, r_8 = 7.22, r_9 = N/A, r_{10} = N/A, r_{11} = 9.62, r_{12} = 0.0000,$$

$$r_{13} = N/A, r_{14} = N/A$$

NASA Capital Budgeting Example

Greedy Heuristic - Iteration 10

- **Ten variable values** are now fixed

$$x^{(10)} = [1, 1, 0, 0, 0, \#, \#, 0, 1, \#, \#, 1, 0]$$

- Right hand sides of the constraints are updated to reflect partial solution

$$\begin{aligned} b_1^{(10)} &= 2, & b_2^{(10)} &= 0, & b_3^{(10)} &= 6, & b_4^{(10)} &= 13, & b_5^{(10)} &= 13, \\ b_6^{(10)} &= 1, & b_7^{(10)} &= 1, & b_8^{(10)} &= 1 \end{aligned}$$

- The updated mission ratios are

$$N/A, N/A, r_3 = N/A, r_4 = N/A, r_5 = N/A, r_6 = N/A,$$

$$r_7 = 0.0000, r_8 = 7.22, r_9 = N/A, r_{10} = N/A, r_{11} = 9.62, r_{12} = 0.0000,$$

$$r_{13} = N/A, r_{14} = N/A$$

- Fix $x_{11}=1$

NASA Capital Budgeting Example

Greedy Heuristic - Final Solution

- Greedy heuristic terminates with a solution value of 675

$$\mathbf{x} = [1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0]$$

- Recall the optimal solution with value 765 is

$$\mathbf{x} = [1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0]$$

- The greedy algorithm clearly returns a **suboptimal** solution
- Are there occasions when a greedy algorithm returns an **optimal** solution?

The Activity Selection Problem

Example

- Assume we a given set of activities $A = \{a_1, a_2, \dots, a_n\}$
- Each activity $a \in A$ has a known start s_a and finish time f_a with

$$0 \leq s_a < f_a < \infty$$

- **Compatible** activities do not overlap in time, i.e., for any two activities i and j

$$\text{either } f_j \leq s_i \text{ or } f_i \leq s_j$$

- Find the **maximal number** of compatible activities that can selected
- An approach which greedily selects compatible activities in order of increasing completion time **yields the optimal solution**
- Applications in machine scheduling where one wishes to perform the greatest number of tasks

The Activity Selection Problem

Example

Consider the following activities

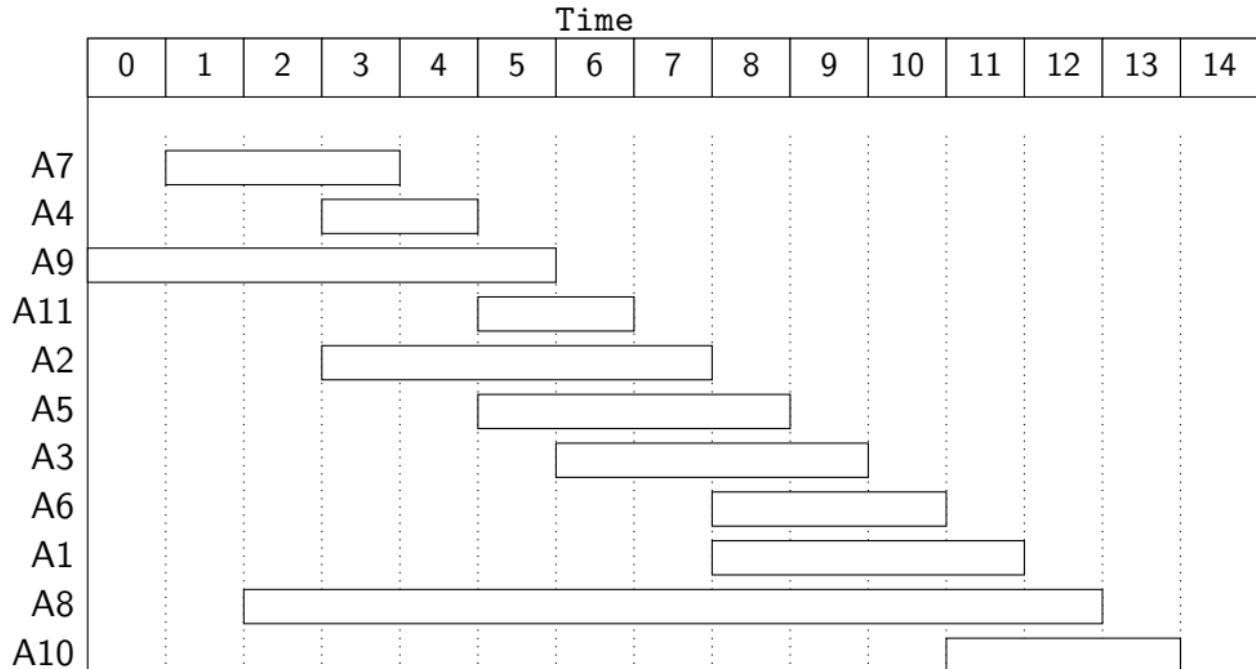
Activity	1	2	3	4	5	6	7	8	9	10	11
Start	8	3	6	3	5	8	1	2	0	11	5
End	12	8	10	5	9	11	4	13	6	14	7

Sorted by finish time

Activity	7	4	9	11	2	5	3	6	1	8	10
Start	1	3	0	5	3	5	6	8	8	2	11
End	4	5	6	7	8	9	10	11	12	13	14

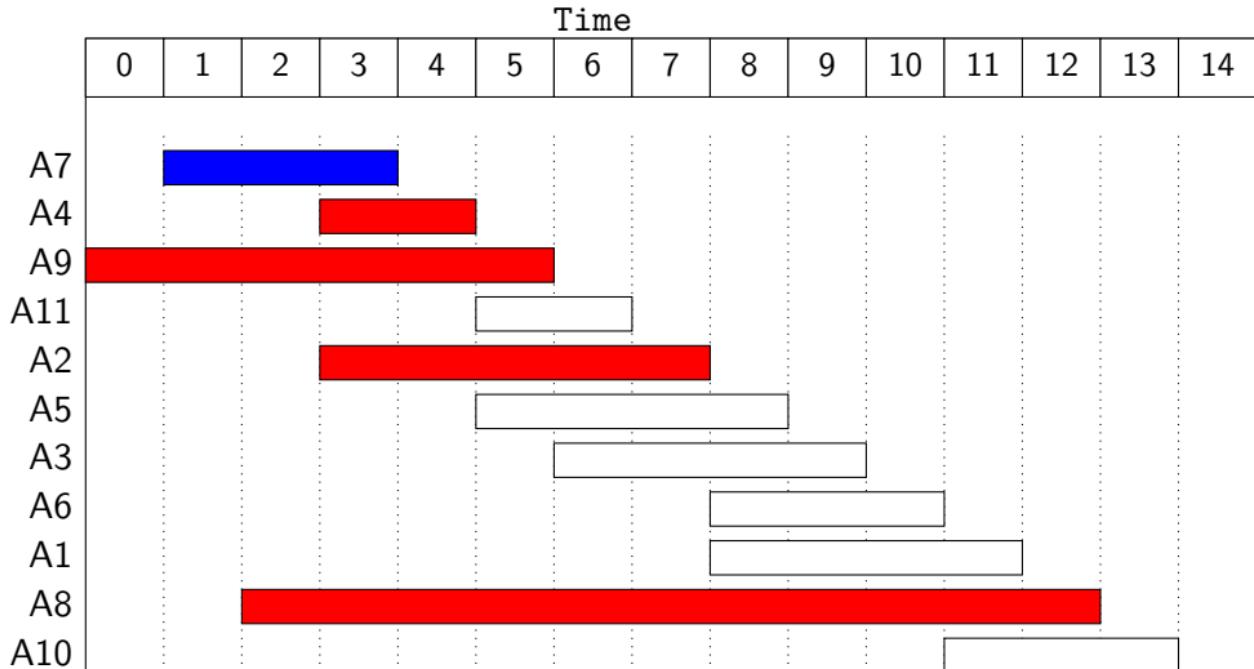
The Activity Selection Problem

In Action



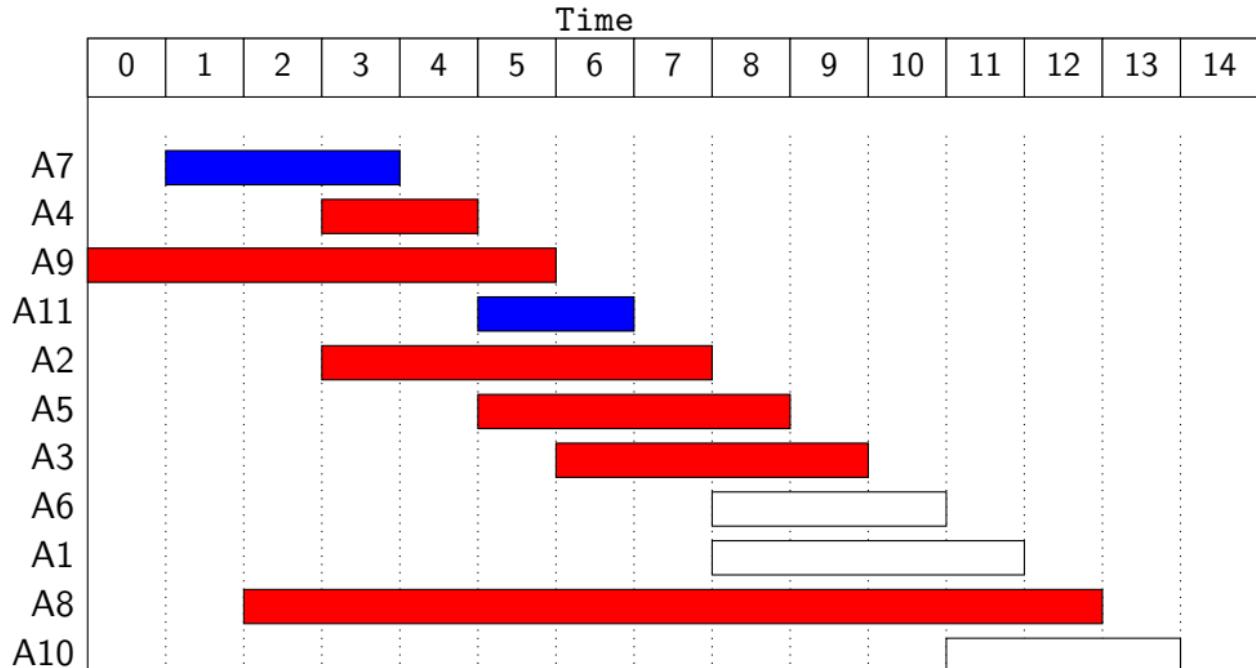
The Activity Selection Problem

In Action



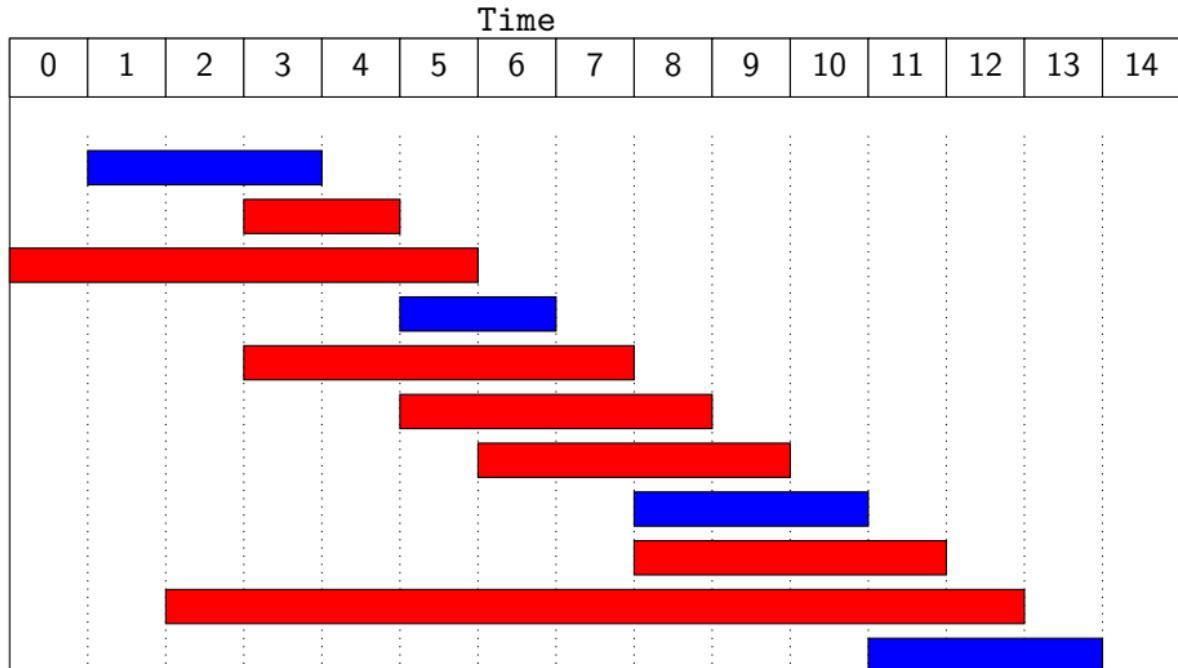
The Activity Selection Problem

In Action



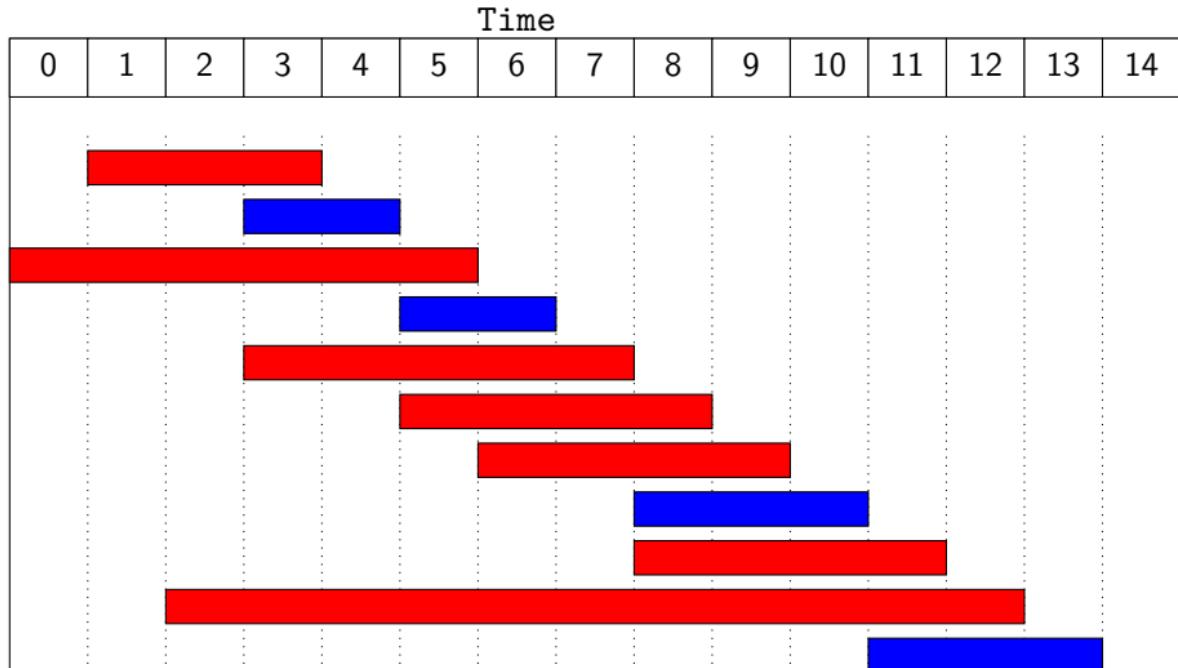
The Activity Selection Problem

In Action



The Activity Selection Problem

In Action



The Activity Selection Problem

Observations

- The greedy approach returns a solution with the **four** activities $\{a_7, a_{11}, a_6, a_{10}\}$
- Another solution that is not returned by the algorithm having **four** activities is the set $\{a_4, a_{11}, a_6, a_{10}\}$
- How do we know four activities is the optimal number?
- Intuitively, scheduling activities in order of increasing finish time seems like a smart thing to do as it leaves the most time for remaining activities
- That is, the greedy choice maximizes the amount of unscheduled time remaining
- We have to **prove** that it is optimal

The Activity Selection Problem

Greedy Algorithm Optimality Proof: Part 1

- Given a set of activities $A = \{a_1, a_2, \dots, a_n\}$ ordered by finish time
- Suppose $S \subset A$ is an optimal solution
- If $a_1 \in S$, we are done
- If $a_1 \notin S$:
 - Let the first activity in S be a_k
 - Make a new solution $S \setminus \{a_k\} \cup \{a_1\}$ by removing a_k using a_1 instead
 - This is valid solution ($f_{a_1} < f_{a_k}$) of maximal size $|S \setminus \{a_k\} \cup \{a_1\}| = |S|$
- There **always** exists an optimal solution that begins with a greedy choice

The Activity Selection Problem

Greedy Algorithm Optimality Proof: Part 2

- S is an optimal solution for A containing $a_1 \Rightarrow S' = S \setminus \{a_1\}$ is an optimal solution for $A' = \{a_j \in A : s_{a_j} \geq f_{a_1}\}$. That is the problem reduces to finding an optimal solution for the activities that are compatible with a_1 .
- Suppose we have a solution S'' to A' such that $|S''| > |S'| = |S| - 1$
- Then $S''' = S'' \cup a_1$ would be a solution to A
- This leads to a contradiction since $|S'''| > |S|$
- Correctness follows by induction on size of S

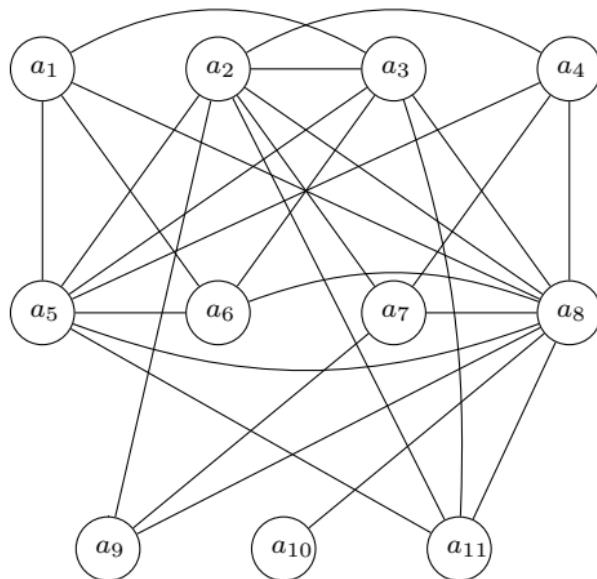
The Activity Selection Problem

Comments

- At each iteration we solve a problem of the same format as the original having one fewer dimensions
- The problem has what is called **the optimal substructure property**
- The optimal solution contains within it the optimal solution to subproblems
- The optimal solution can therefore be obtained by making a sequence of **locally optimal** decisions to ultimately obtain a **globally** optimally solution
- These are two important properties central to the success of greedy algorithms

The Activity Selection Problem

A Graphical Representation



The Activity Selection Problem

A Binary Integer Program

- Define the following binary decision variables

$$x_j = \begin{cases} 1 & \text{if activity } j \text{ is selected } j = 1, 2, 3, \dots, 11 \\ 0 & \text{otherwise} \end{cases}$$

- This yields the following objective function

$$\text{Maximize: } \sum_{j=1}^{11} x_j$$

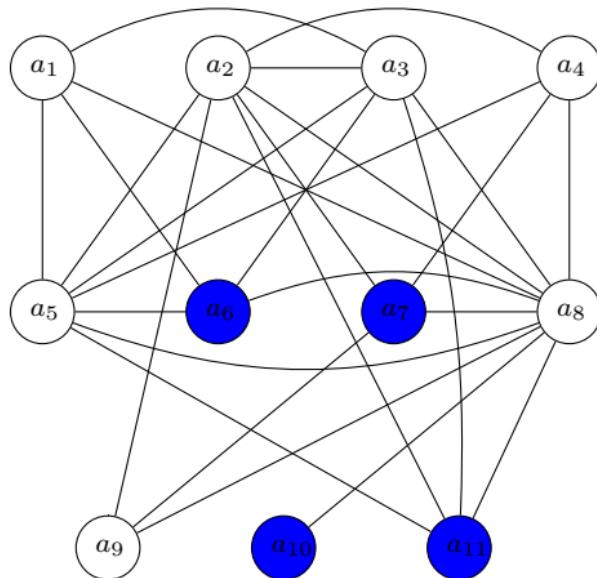
- The following constraints must be satisfied

$$x_i + x_j \leq 1 \text{ for all incompatible activities } i \text{ and } j$$
$$x_j \in \{0, 1\}$$

- Looking for the largest set of nodes in the graph such that any pair of nodes in the solution are not incident on the same edge

The Activity Selection Problem

Binary Integer Program Solution



Julia Implementation

```
1  using JuMP
2  using GLPK
3
4  struct Activity
5      start::Int32
6      finish::Int32
7  end
8
9  activities = [Activity(8,12), Activity(3,8),Activity(6,10),Activity(3,5),
10    Activity(5,9),Activity(8,11),Activity(1,4), Activity(2,13),
11    Activity(0,6), Activity(11,14),Activity(5,7)]
12
13 m = Model(with_optimizer(GLPK.Optimizer))
14
15 # number of activities
16 n=length(activities)
17 @variable(m, x[activities], Bin)
18 @objective(m, Max, sum(x[a] for a in activities))
19
20 for i=1:n-1
21     for j=i+1:n
22         ai = activities[i]
23         aj = activities[j]
24         if(!(ai.finish <= aj.start || aj.finish <= ai.start))
25             @constraint(m, x[ai]+x[aj]<=1)
26         end
27     end
28 end
29
30 optimize!(m)
31
32 println("Objective Value: ", JuMP.objective_value(m))
33 println("Variable values: ", JuMP.value.(x))
```

Assigning People to Tasks

- Four tasks need doing and you have four available employees
- The time in hours for each person to do each task is known
- You must find the best assignment of people to tasks, i.e., find the **assignment** that minimizes the time necessary to complete all tasks

Person.\Task	1	2	3	4
1	8	6	5	7
2	6	5	3	4
3	7	8	4	6
4	6	7	5	6

Assigning People to Tasks

- Four tasks need doing and you have four available employees
- The time in hours for each person to do each task is known
- You must find the best assignment of people to tasks, i.e., find the **assignment** that minimizes the time necessary to complete all tasks

Person.\Task	1	2	3	4
1	8	6	5	7
2	6	5	3	4
3	7	8	4	6
4	6	7	5	6

What would you do?

Greedy Approach

- Sequentially assign tasks by choosing the person-task assignment that gives the shortest duration
- Break ties arbitrarily
- This gives the following:
 - ① Person two gets task three
 - ② Person four gets task one
 - ③ Person one gets job two
 - ④ Person three gets job four
- Solution has an objective value of 21
- Is this optimal?

Mathematical Modelling

- Can we formulate this problem mathematically?
- Aim is to **assign** people to tasks
- Introduce a set of **binary decision variables**

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ is assigned task } j, \text{ for } i, j = 1, 2, 3, 4 \\ 0 & \text{otherwise,} \end{cases}$$

- Define the **objective value**

$$\begin{aligned} \text{Minimize: } & 8x_{11} + 6x_{12} + 5x_{13} + 7x_{14} \\ & + 6x_{21} + 5x_{22} + 3x_{23} + 4x_{24} \\ & + 7x_{31} + 8x_{32} + 4x_{33} + 6x_{34} \\ & + 6x_{41} + 7x_{42} + 5x_{43} + 6x_{44} \end{aligned}$$

- Formulate task assignment **constraint** (one for each)

$$x_{11} + x_{21} + x_{31} + x_{41} = 1$$

- Formulate person assignment **constraint** (one for each)

$$x_{11} + x_{12} + x_{13} + x_{14} = 1$$

Julia Implementation

```
1  using JuMP
2  using GLPK
3
4
5  m = Model(with_optimizer(GLPK.Optimizer))
6
7  cost = [8 6 5 7;
8      6 5 3 4;
9      7 8 4 6;
10     6 7 5 6]
11
12 n=4
13
14 @variable(m, x[1:n,1:n], Bin)
15 @objective(m, Min, sum(cost[i,j]*x[i,j] for i=1:n for j=1:n))
16
17 @constraint(m, [i=1:n], sum(x[i,j] for j=1:n) ==1)
18 @constraint(m, [j=1:n], sum(x[i,j] for i=1:n) ==1)
19
20 optimize!(m)
21
22 println("Objective Value: ", JuMP.objective_value(m))
23 println("Variable values: ", JuMP.value.(x))
```

Comments

- Optimal solution has a value of 20
- Person 1 gets task 2, person 2 task 4, person 3 task 3, person 4 task 1
- Greedy approach lacks foresight

The General Problem

- This type of problem is known as an **assignment problem**
- Identify the best assignment
- Ensure each **task** is assigned exactly one **agent**
- Ensure each **agent** is assigned exactly one **task**

$$\begin{aligned} \text{Maximize: } & \sum_{i=1}^n \sum_{j=1}^n \rho_{ij} x_{ij} \\ \text{s.t. } & \sum_{i=1}^n x_{ij} = 1 \quad \forall j = 1, 2, \dots, n \\ & \sum_{j=1}^n x_{ij} = 1 \quad \forall i = 1, 2, \dots, n \\ & x_{ij} \in \{0, 1\} \quad \forall i, j = 1, 2, \dots, n \end{aligned}$$

The General Problem

- This type of problem is known as an **assignment problem**
- Identify the best assignment
- Ensure each **task** is assigned exactly one **agent**
- Ensure each **agent** is assigned exactly one **task**

$$\begin{aligned} \text{Maximize: } & \sum_{i=1}^n \sum_{j=1}^m \rho_{ij} x_{ij} \\ \text{s.t. } & \sum_{i=1}^n x_{ij} = 1 \quad \forall j = 1, 2, \dots, m \\ & \sum_{j=1}^m w_{ij} x_{ij} \leq b_i \quad \forall i = 1, 2, \dots, n \\ & x_{ij} \in \{0, 1\} \quad \forall i = 1, 2, \dots, n, j = 1, 2, \dots, m \end{aligned}$$

- In the **generalized assignment problem** the number of tasks and agents may vary, an agent may be assigned multiple tasks, and each agent has an available **budget**

Learning Objectives

At the end of today's lecture you should be able to

- ① Understand and explain the difference between Heuristic and Exact Approaches
- ② Know what a Greedy Constructive Algorithm is
- ③ Design a Greedy Constructive Algorithm
- ④ Give examples of when a Greedy Algorithm fails/succeeds

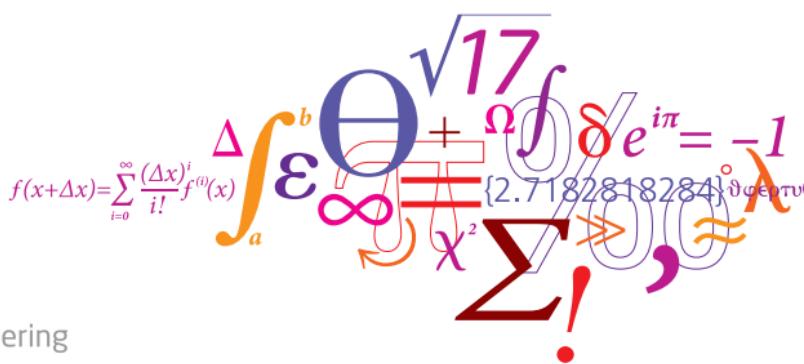
Richard Lusby
Associate Professor, DTU Management Engineering

Building 358, Room 145a rmlu@dtu.dk
2800 Kgs. Lyngby, Denmark +45 4525 3084 phone

Modelling with Integer Variables IV

Richard M. Lusby

DTU Management



Today's Agenda

- ① The Travelling Salesman Problem (TSP)
- ② Solution neighbourhoods
- ③ Improvement Heuristics
- ④ An example (TSP)

Learning Objectives

At the end of today's lecture you should be able to

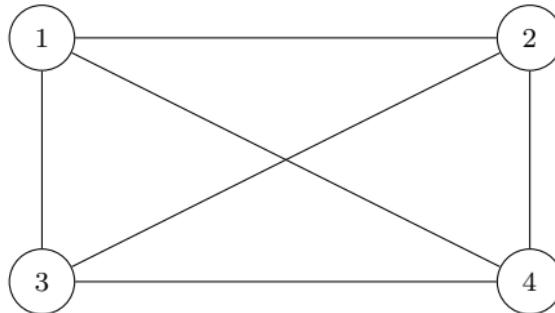
- ① Formulate the Travelling Salesman Problem (TSP) using Integer Programming
- ② Understand the concept of solution neighbourhood
- ③ Explain what an improvement heuristic is
- ④ Implement the 2-opt and 3-opt exchange for the TSP

The Travelling Salesman Problem

- Probably the most well known problem in the Operations Research world
- The TSP asks the following question: “Given a list of cities and the distances between each pair of cities, what is the shortest possible tour that visits each city exactly once and returns to the origin city?”
- Difficult combinatorial optimization problem
- Has many applications
 - Vehicle Routing Problems, **Circuit Board Design**, Genome Sequencing, ...

The Travelling Salesman Problem

- Typically modelled as an undirected weighted graph
- **Vertices** model “cities”, **edges** denote connections between “cities”
- The weight of an edge gives the “distance”
- The graph is typically complete
- Comes an **assymmetric** and **symmetric** variants
- The symmetric version has half the number of possible solutions



Where are we at?



TSP Example: Circuit Board Design

- A practical example of the TSP
- Circuit boards have a number of small holes through which chips and other components are wired
- Realistic instances have hundreds of such small holes
- Efficient manufacturing requires that these holes are drilled as quickly as possible
- Entails identifying a routing through the holes with the minimum total distance



Example

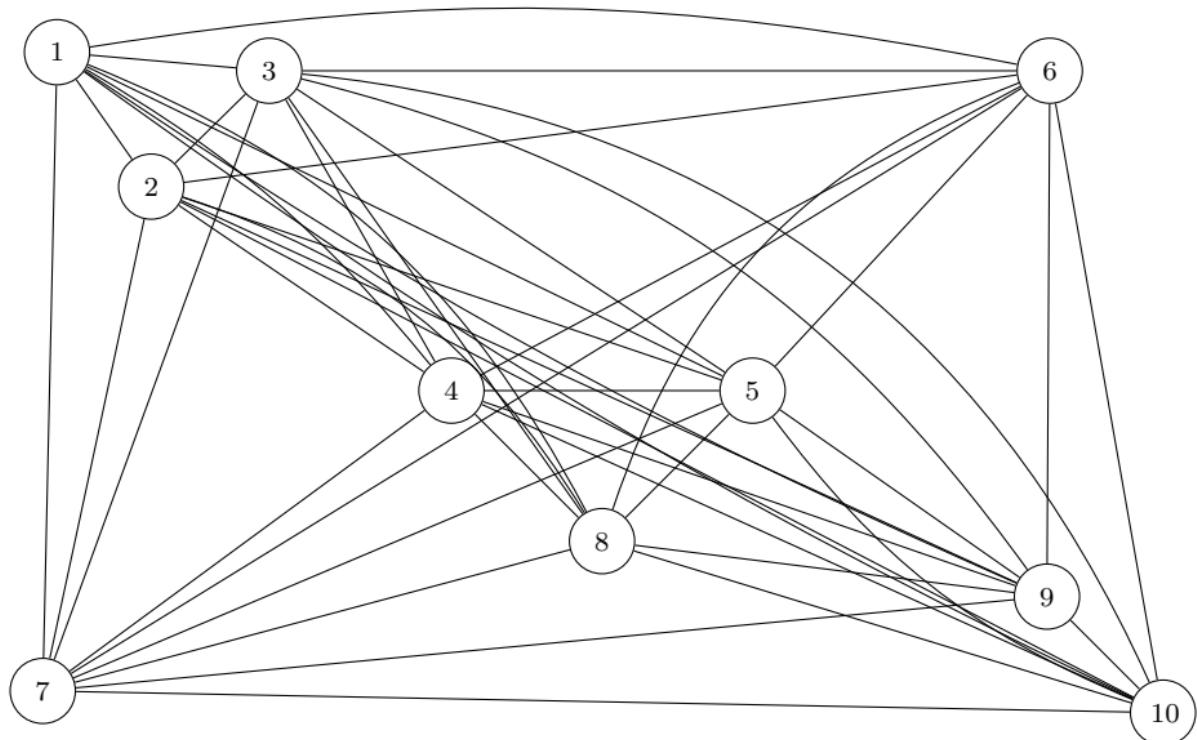
Circuit Board Design - 10 holes

Data: Euclidean Distances between hole pairs (mm)

		Holes									
		1	2	3	4	5	6	7	8	9	10
Holes	1	-	3.6	5.1	10.0	15.3	20.0	16.0	14.2	23.0	26.4
	2	3.6	-	3.6	6.4	12.1	18.1	13.2	10.6	19.7	23.0
	3	5.1	3.6	-	7.1	10.6	15.0	15.8	10.8	18.4	21.9
	4	10.0	6.4	7.1	-	7.0	15.7	10.0	4.2	13.9	17.0
	5	15.3	12.1	10.6	7.0	-	9.9	15.3	5.0	7.8	11.3
	6	20.0	18.1	15.0	15.7	9.9	-	25.0	14.9	12.0	15.0
	7	16.0	13.2	15.8	10.0	15.3	25.0	-	10.3	19.2	21.0
	8	14.2	10.6	10.8	4.2	5.0	14.9	10.3	-	10.2	13.0
	9	23.0	19.7	18.4	13.9	7.8	12.0	19.2	10.2	-	3.6
	10	26.4	23.0	21.9	17.0	11.3	15.0	21.0	13.0	3.6	-

Circuit Board Design

Full Network



Circuit Board Design

- The problem can be modelled as a **Symmetric TSP**
- How many possible solutions are there?

Circuit Board Design

- The problem can be modelled as a **Symmetric TSP**
- How many possible solutions are there?

$$\frac{n \cdot (n - 1)!}{2}$$

Circuit Board Design

- The problem can be modelled as a **Symmetric TSP**
- How many possible solutions are there?

$$\frac{n \cdot (n - 1)!}{2}$$

- A problem with 100 holes has:

Circuit Board Design

- The problem can be modelled as a **Symmetric TSP**
- How many possible solutions are there?

$$\frac{n \cdot (n - 1)!}{2}$$

- A problem with 100 holes has:

$$\frac{100 \cdot (100 - 1)!}{2} = 4.6663 \times 10^{157} \text{ possible solutions}$$

(compare with the number of grains of sand on the planet $\approx 7.5 \times 10^{18}$)

Circuit Board Design

- The problem can be modelled as a **Symmetric TSP**
- How many possible solutions are there?

$$\frac{n \cdot (n - 1)!}{2}$$

- A problem with 100 holes has:

$$\frac{100 \cdot (100 - 1)!}{2} = 4.6663 \times 10^{157} \text{ possible solutions}$$

(compare with the number of grains of sand on the planet $\approx 7.5 \times 10^{18}$)

- Enumeration is certainly not an option for large instances

Circuit Board Design

- The problem can be modelled as a **Symmetric TSP**
- How many possible solutions are there?

$$\frac{n \cdot (n - 1)!}{2}$$

- A problem with 100 holes has:

$$\frac{100 \cdot (100 - 1)!}{2} = 4.6663 \times 10^{157} \text{ possible solutions}$$

(compare with the number of grains of sand on the planet $\approx 7.5 \times 10^{18}$)

- Enumeration is certainly not an option for large instances
- Heuristic approaches sacrifice optimality for speed

The Travelling Salesman Problem

A Mathematical Formulation for the assymmetric case

- Define the following binary decision variables

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

- We have the following objective function

$$\sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$$

- We also have the following constraints

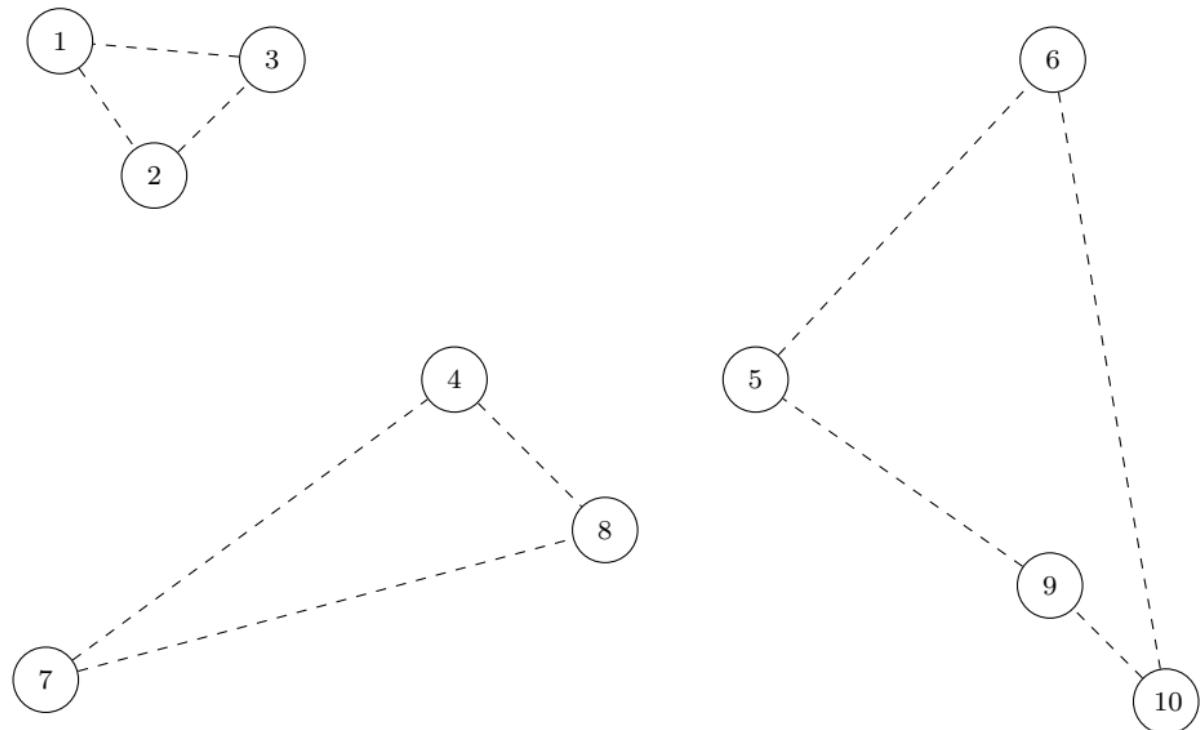
$$\sum_{i=1}^n x_{ij} = 1 \quad \forall j = 1, 2, \dots, n$$

$$\sum_{j=1}^n x_{ij} = 1 \quad \forall i = 1, 2, \dots, n$$

$$x_{ij} \in \{0, 1\} \quad \forall i = 1, 2, \dots, n, \quad \forall j = 1, 2, \dots, n$$

Circuit Board Design

Optimal Solution: 72.6mm



The Travelling Salesman Problem

A Revised Mathematical Formulation for the assymmetric case

- Update the decision variables

$$x_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is used in the solution} \\ 0 & \text{otherwise} \end{cases}$$

$$u_i \geq 0, \text{ the position of node } i \text{ in the solution}$$

- Update the constraints

$$\sum_{i=1}^n x_{ij} = 1 \quad \forall j = 1, 2, \dots, n$$

$$\sum_{j=1}^n x_{ij} = 1 \quad \forall i = 1, 2, \dots, n$$

$$x_{ij} \in \{0, 1\} \quad \forall i = 1, 2, \dots, n, \quad \forall j = 1, 2, \dots, n$$

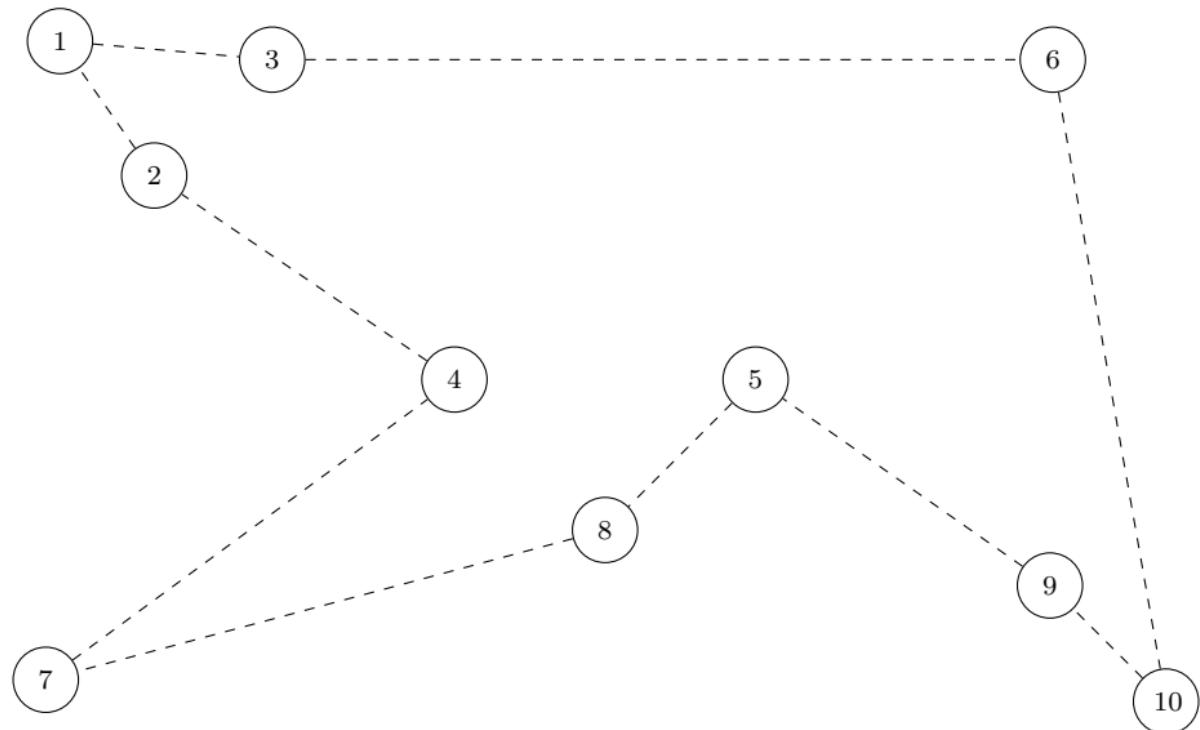
$$u_i - u_j + 1 \leq (n - 1) \cdot (1 - x_{ij}) \quad \forall i \neq 1, \forall j \neq 1$$

$$u_1 = 1$$

$$2 \leq u_i \leq n \quad \forall i = 1, 2, \dots, n$$

Circuit Board Design

Optimal Solution - 81.8mm



Julia Code

```
1  using JuMP
2  using GLPK
3
4  m = Model(with_optimizer(GLPK.Optimizer))
5  n=10
6  cost  = [0 3.6 5.1 10.0 15.3 20.0 16.0 14.2 23.0 26.4;
7           3.6 0 3.6 6.4 12.1 18.1 13.2 10.6 19.7 23.0;
8           5.1 3.6 0 7.1 10.6 15.0 15.8 10.8 18.4 21.9;
9           10.0 6.4 7.1 0 7.0 15.7 10.0 4.2 13.9 17.0;
10          15.3 12.1 10.6 7.0 0 9.9 15.3 5.0 7.8 11.3;
11          20.0 18.1 15.0 15.7 9.9 0 25.0 14.9 12.0 15.0;
12          16.0 13.2 15.8 10.0 15.3 25.0 0 10.3 19.2 21.0;
13          14.2 10.6 10.8 4.2 5.0 14.9 10.3 0 10.2 13.0;
14          23.0 19.7 18.4 13.9 7.8 12.0 19.2 10.2 0 3.6;
15          26.4 23.0 21.9 17.0 11.3 15.0 21.0 13.0 3.6 0]
```

Julia Code

```
17 #variable definition
18 @variable(m, x[1:n,1:n], Bin)
19 @variable(m, u[1:n])
20 #objective function
21 @objective(m, Min, sum(cost[i, j] * x[i,j] for i = 1:n for j = 1:n))
22 # constraints - flow out of node
23 @constraint(m, [i=1:n], sum(x[i,j]) for j=1:n) == 1)
24 # constraints - flow into node
25 @constraint(m, [j=1:n], sum(x[i,j] for i=1:n) == 1)
26 # constraints - node order
27 for i=1:n, j=1:n
28     if i != 1 && j!=1
29         @constraint(m, u[i]-u[j]+(n-1)*x[i,j] <= (n-2))
30     end
31 end
32 # constraints - bounds on u
33 @constraint(m, u[1]==1)
34 @constraint(m, [i=2:n], u[i]>=2)
35 @constraint(m, [i=2:n], u[i]<=10)
36
37 optimize!(m)
38
39 println("Objective Value: ", JuMP.objective_value(m))
40 for i=1:n, j=1:n
41     if JuMP.value(x[i,j]) > 1-1e-6
42         println("Edge ", i, "-", j, " ", JuMP.value(x[i,j]))
43     end
44 end
```

Greedy Constructive Algorithm

Greedy Solution - Nearest Neighbour

- Solving large combinatorial optimization problems with mathematical programming techniques can be extremely time consuming, if not impractical
- One of many different construction heuristics proposed for the TSP
- Starts at a randomly chosen vertex and iteratively builds a full **tour** by selecting the closest vertex to the last visited vertex
- Very short sighted
- **Quickly** yields a solution
- Solution quality is often adversely affected by the **lack** of choice in the final iterations and the approach is unlikely to find the optimal tour

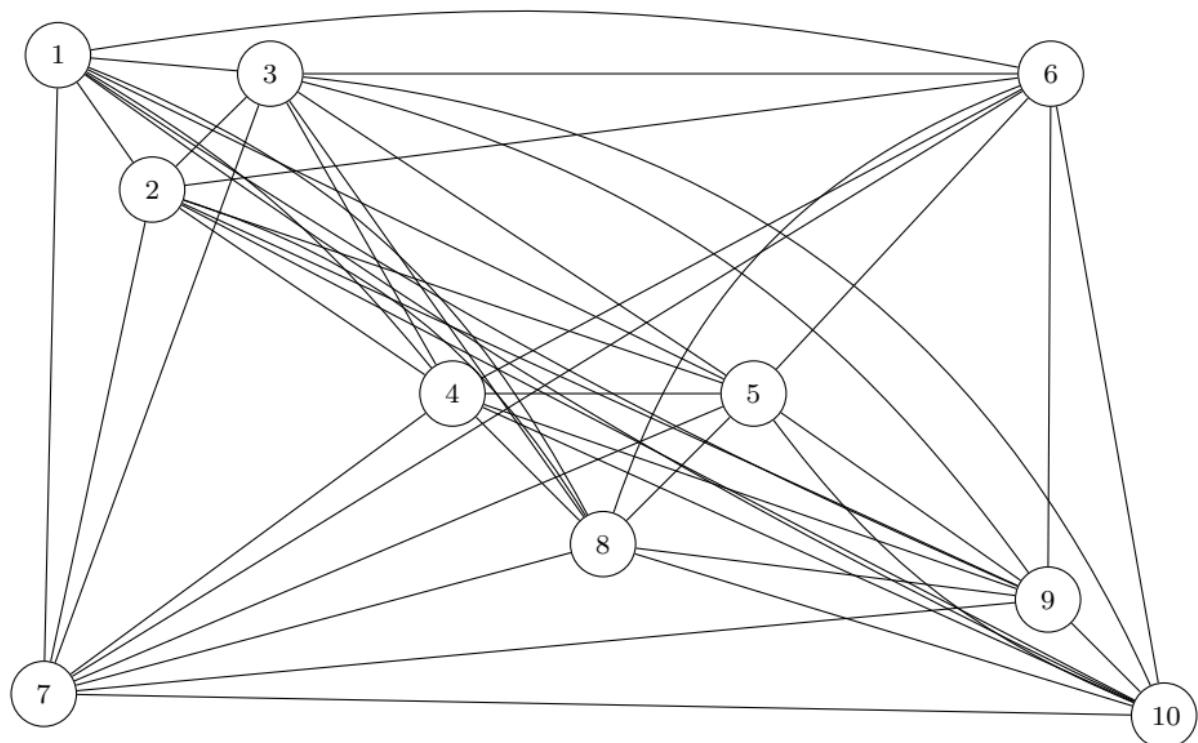
Greedy Constructive Algorithm

The Algorithm

- ① **Initialize** the solution with a city chosen at random
- ② Given a partial solution of k cities $P = (v_1, v_2, \dots, v_k)$, **find** the city v_{k+1} that is closest to v_k and which is not in the partial solution
- ③ **Augment** the solution with city v_{k+1}
- ④ If $|P| = n$, then **Stop**, else **go to** Step 2.

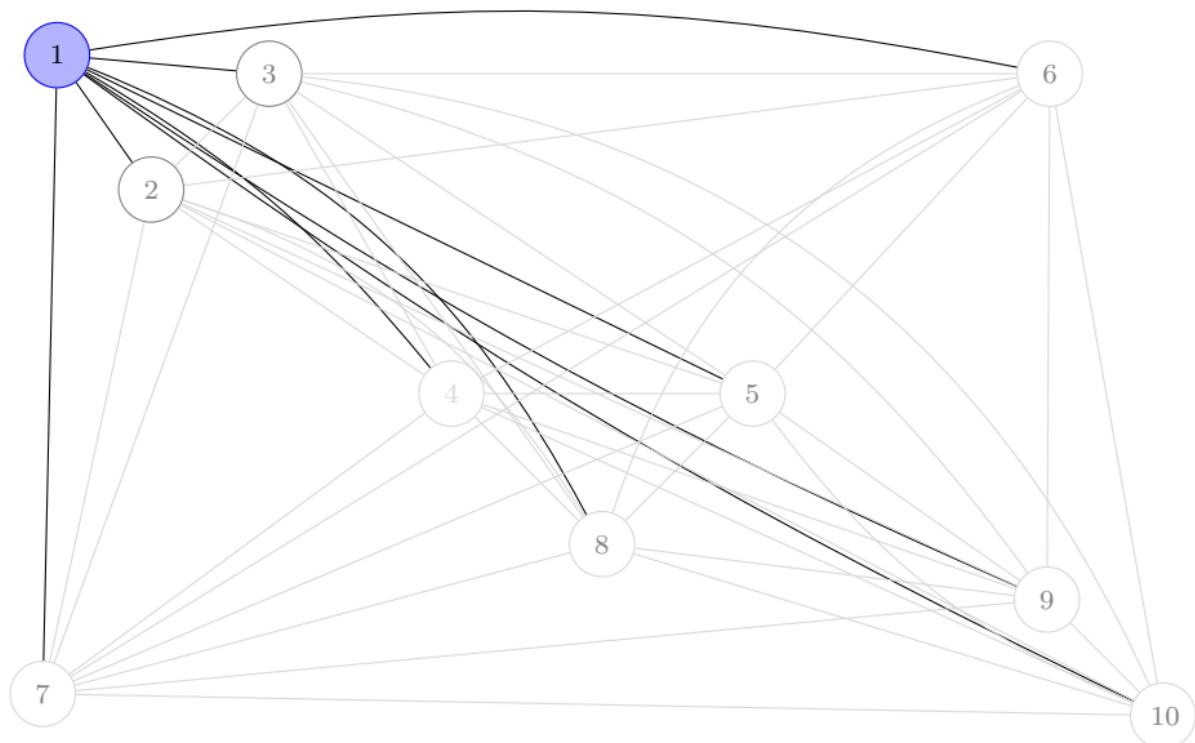
Circuit Board Design

Walk through



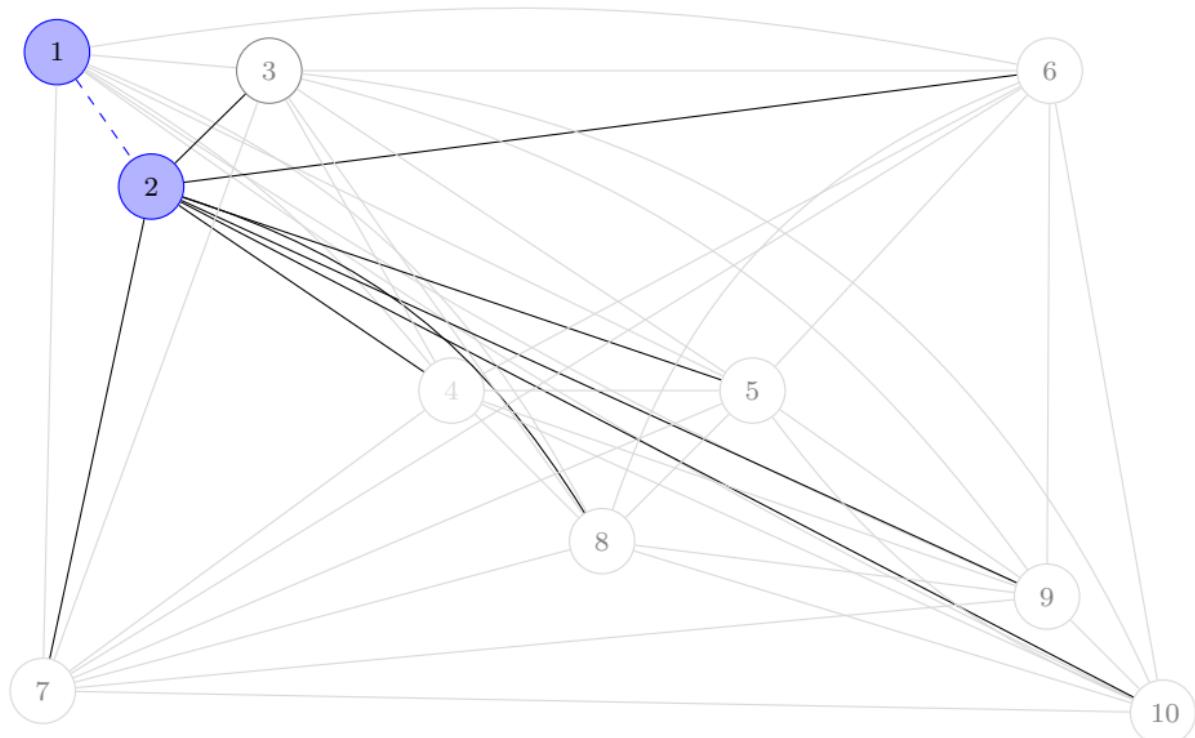
Circuit Board Design

Walk through



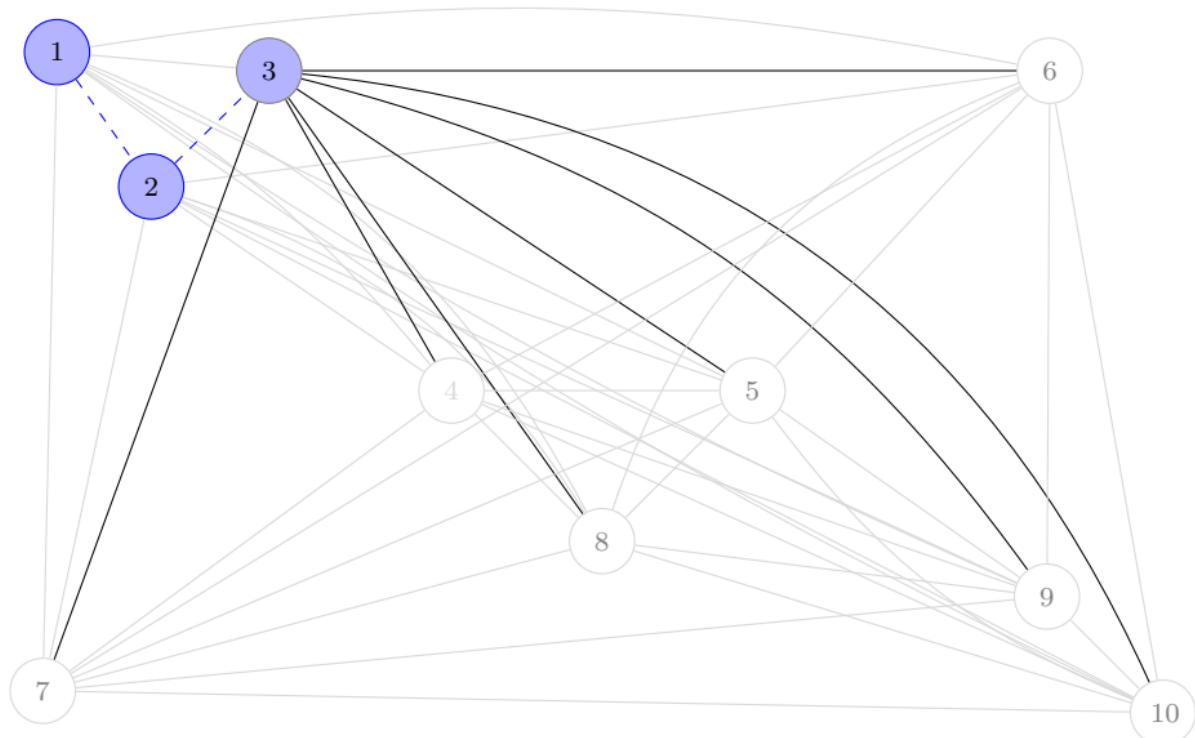
Circuit Board Design

Walk through



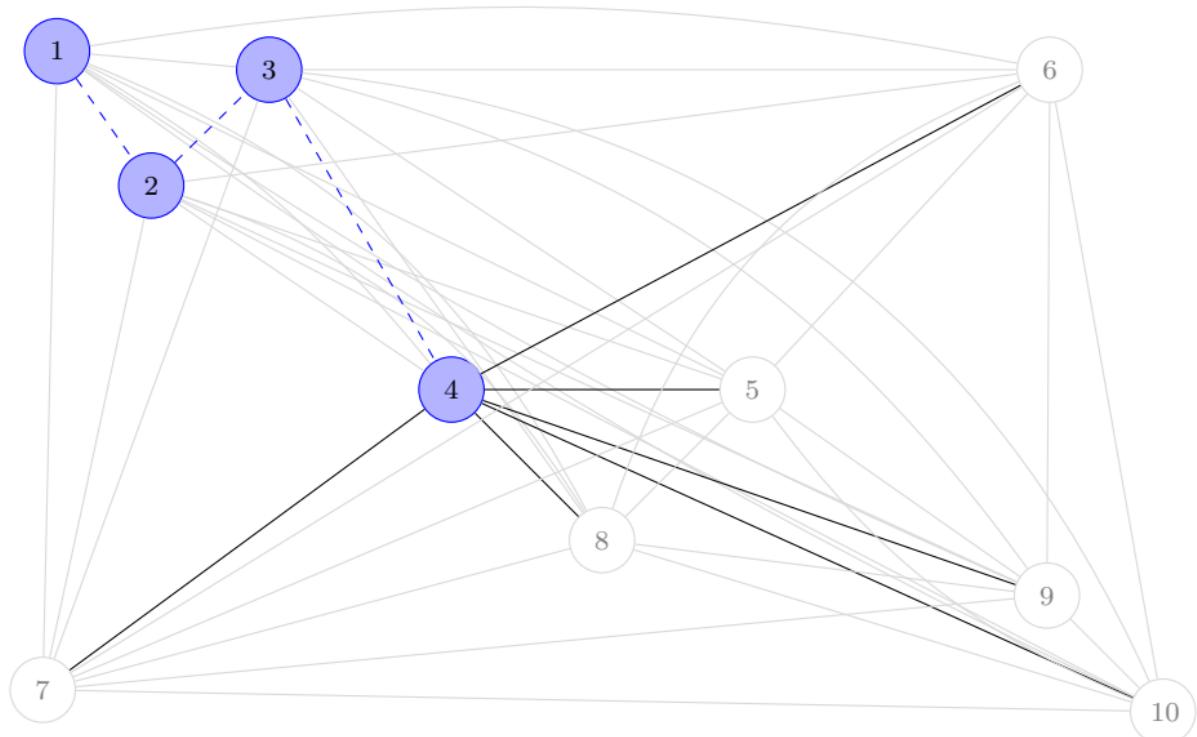
Circuit Board Design

Walk through



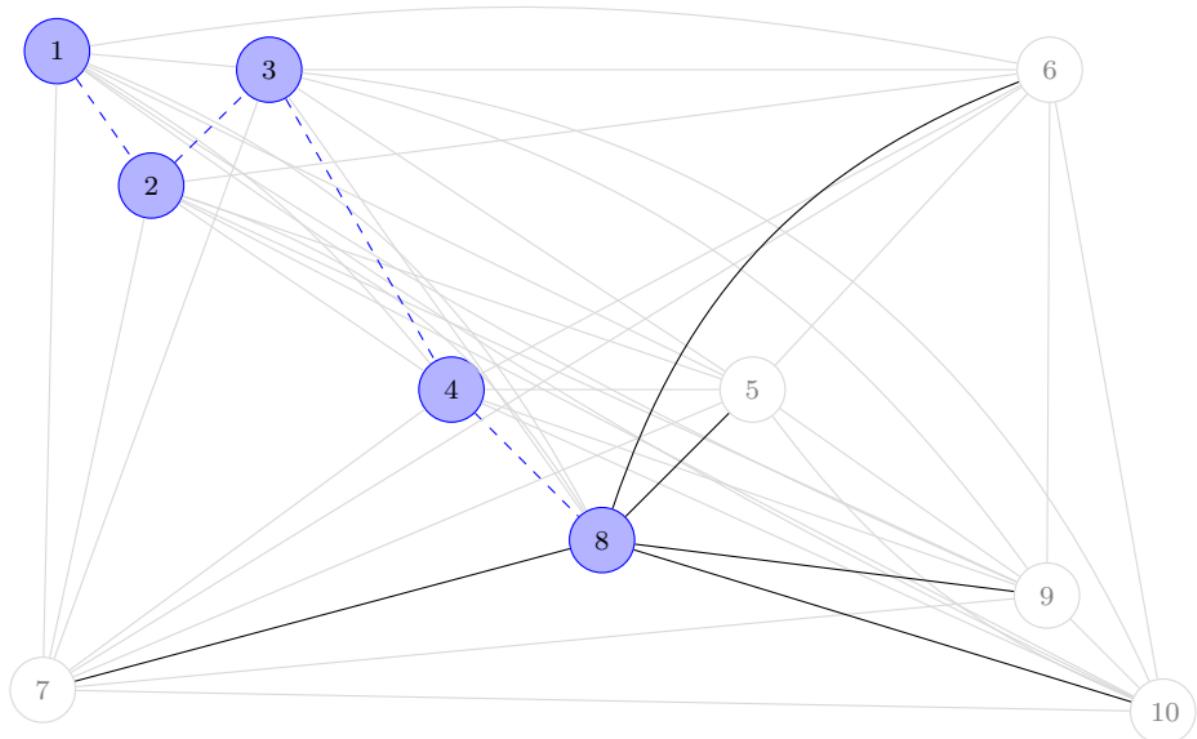
Circuit Board Design

Walk through



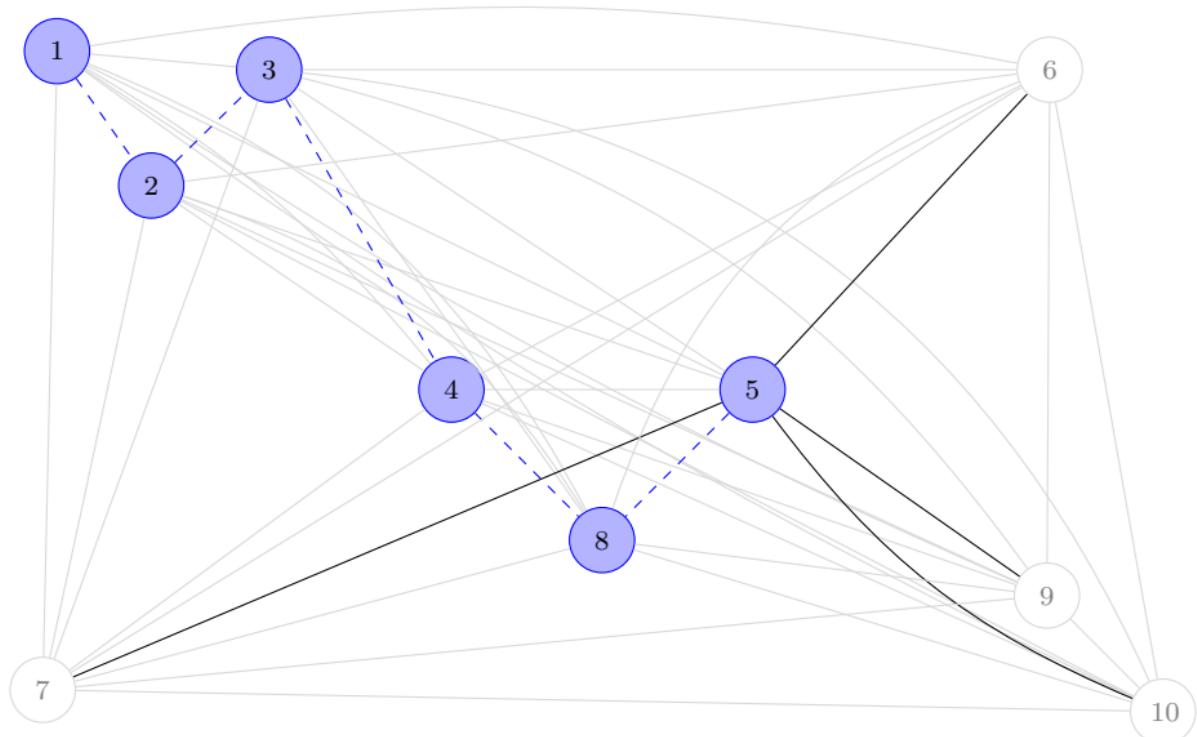
Circuit Board Design

Walk through



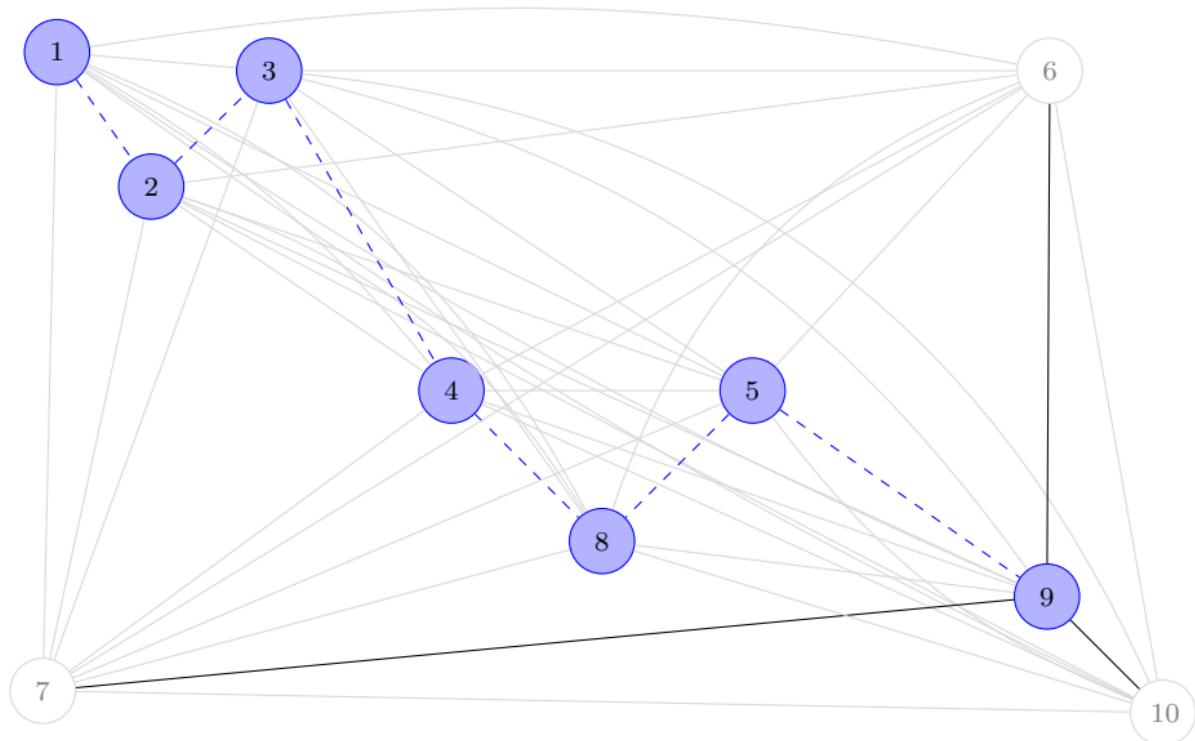
Circuit Board Design

Walk through



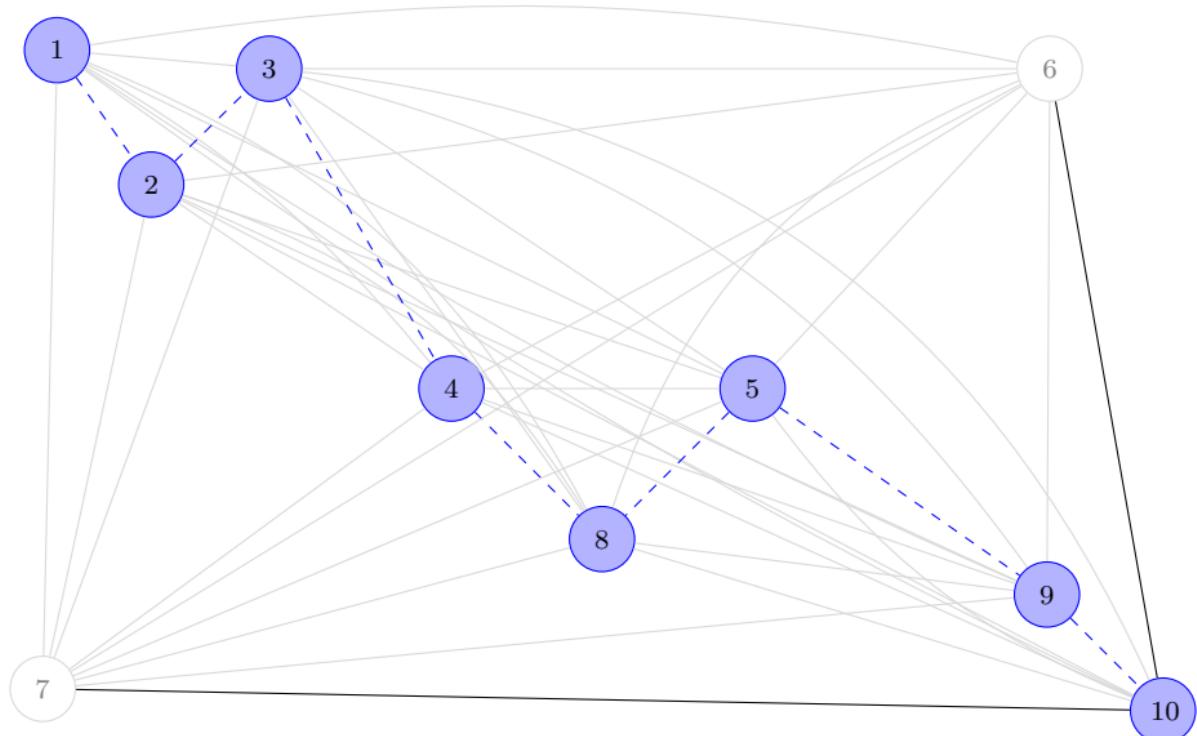
Circuit Board Design

Walk through



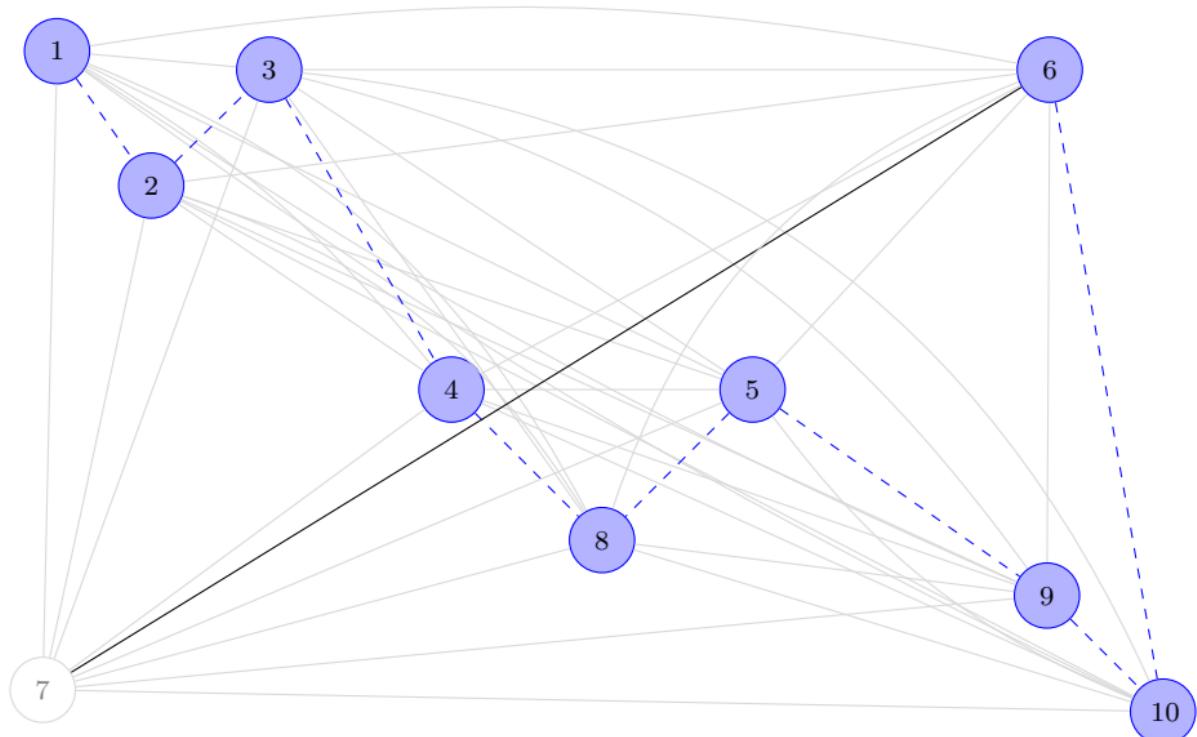
Circuit Board Design

Walk through



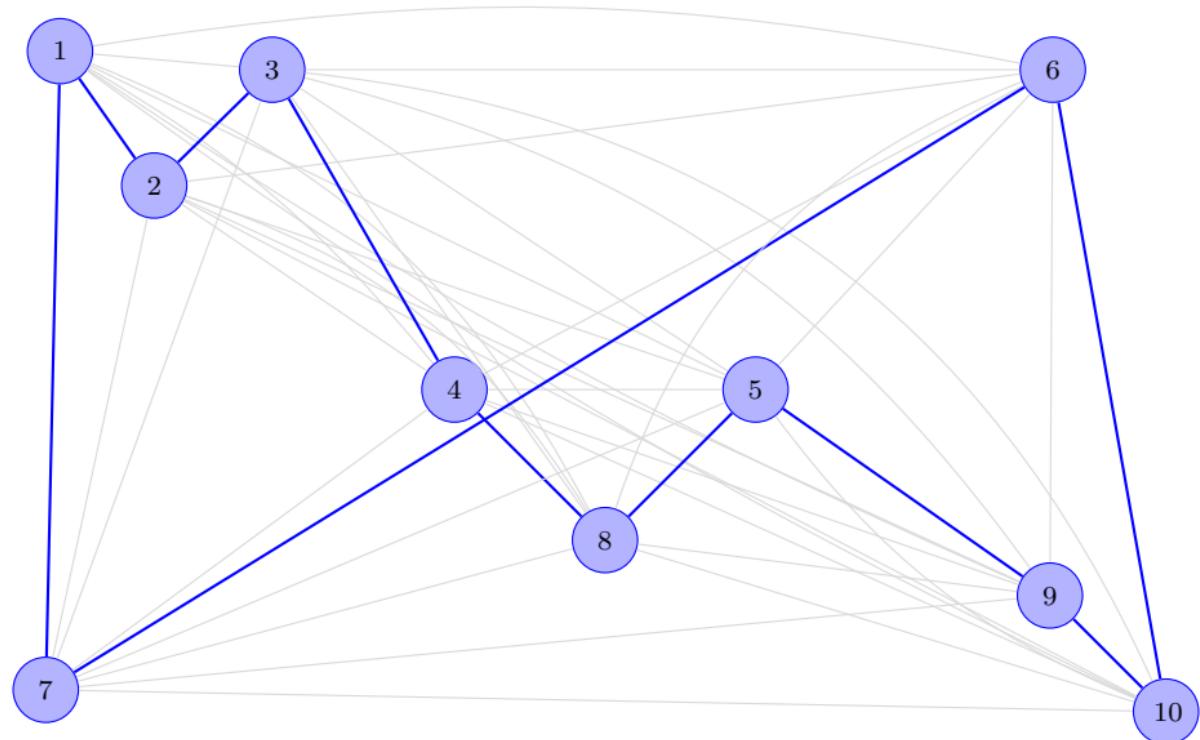
Circuit Board Design

Walk through



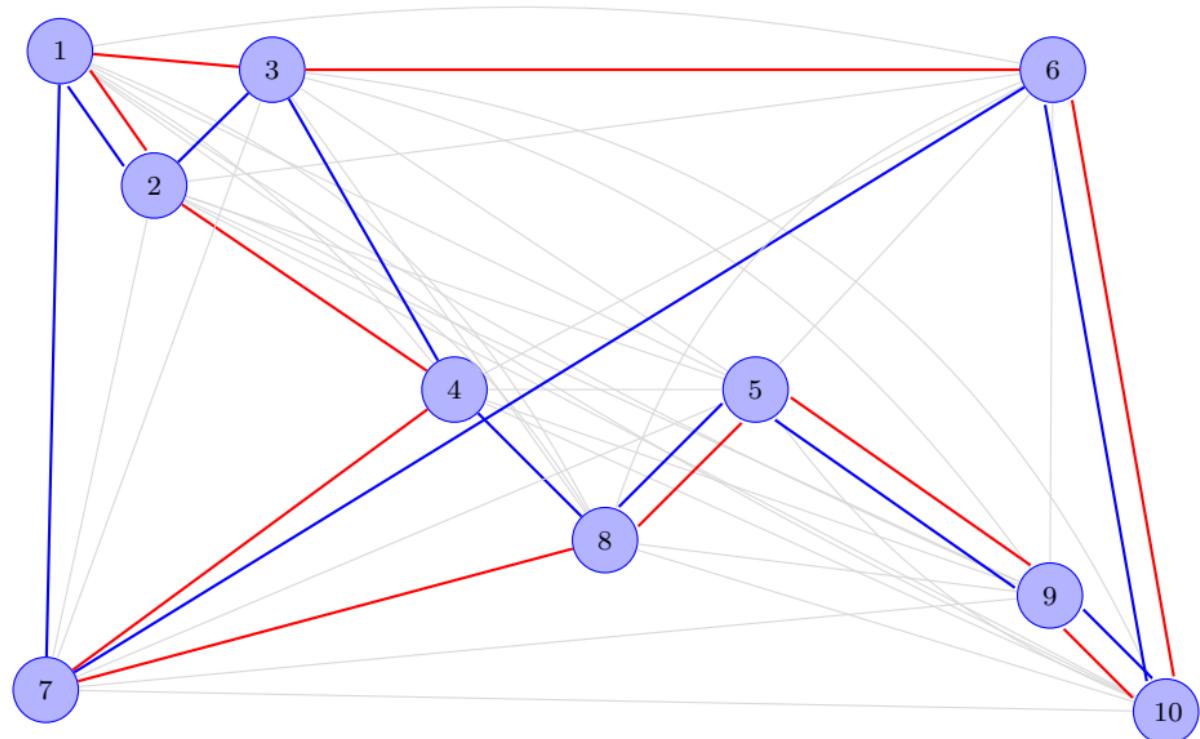
Circuit Board Design

Walk through



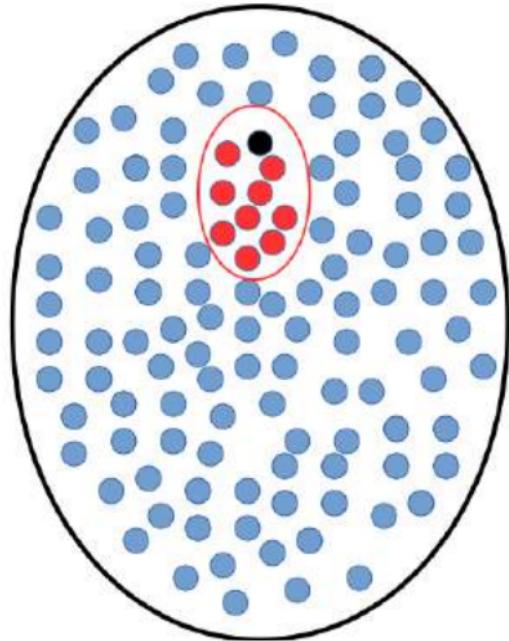
Circuit Board Design

Optimal Solution vs Nearest Neighbour



Improvement Heuristic (1)

- Starts from a candidate solution
- Iteratively considers solutions in the **neighbourhood** of the current solution
- **Neighbouring solutions typically** have a similar structure
- **Moves** to a neighbouring solution if it improves the objective value
- Known as **local search**
- Larger neighbourhoods typically give better results
- **Local optimality** attained when no neighbouring solution yields an improvement



Improvement Heuristic (2)

- Local search is a structured approach to solving optimization problems
- Strives for global optimality but only ever considers the “local” neighbourhood of a solution at any given iteration
- Will typically end up in a **local optimum**
- Assuming the set of all solutions is denoted S , how do we define the neighbourhood $N(x) \subset S$ for a given solution $x \in S$?
- We can use the following **moves**
 - ① Add elements
 - ② Remove elements
 - ③ Remove/add elements
 - ④ Changing two or more elements of a solution, i.e. swapping elements
- $y \in N(x)$ is termed a neighbour of x
- Assuming a minimization problem, x is locally optimal if

$$f(x) \leq f(y) \text{ for all } y \in N(x)$$

Generic Local Search Approach

Generate Initial Solution x_0 ;

Current Solution $x_i \leftarrow x_0$;

while $N(x_i)$ not empty **do**

 Select $x_j \in N(x_i)$;

if $f(x_j) < f(x_i)$ **then**

 Move to new solution;

$x_i = x_j$;

else

$N(x) = N(x) \setminus x_j$

end

end

Return locally optimal solution x_i



Generic Local Search Approach

Generate Initial Solution x_0 ;

Current Solution $x_i \leftarrow x_0$;

while $N(x_i)$ not empty **do**

 Select $x_j \in N(x_i)$;

if $f(x_j) < f(x_i)$ **then**

 Move to new solution;

$x_i = x_j$;

else

$N(x) = N(x) \setminus x_j$

end

end

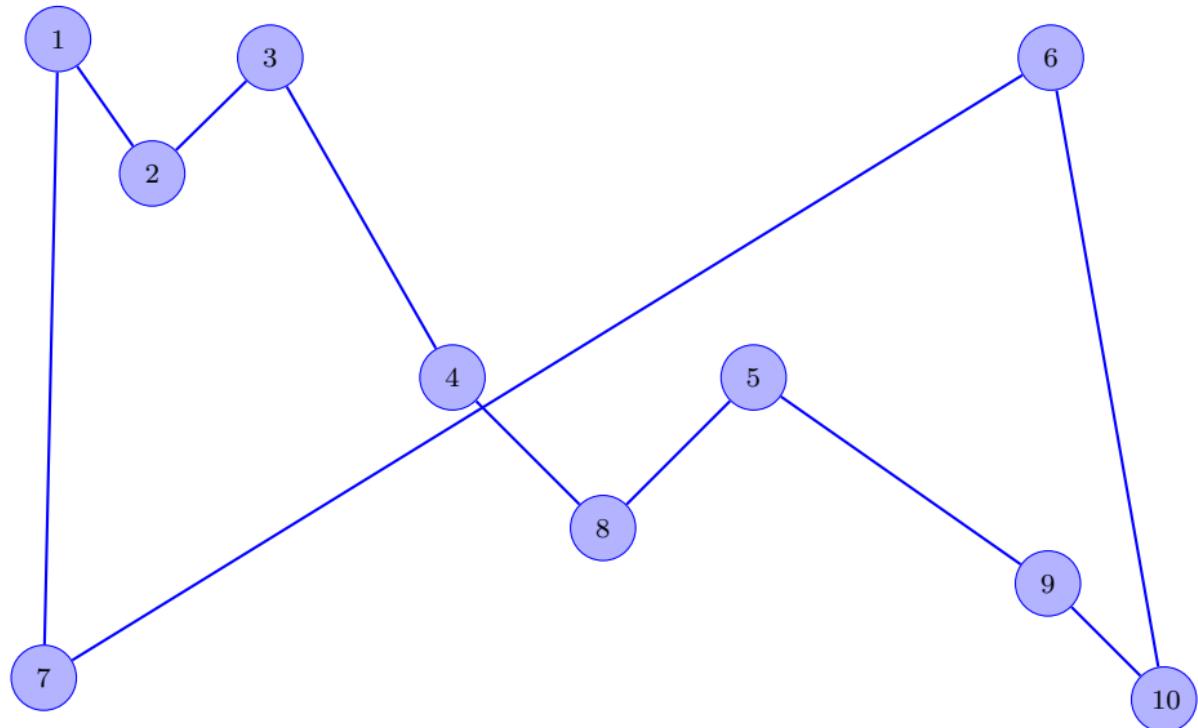
Return locally optimal solution x_i



More on optimization using metaheuristics in 42117

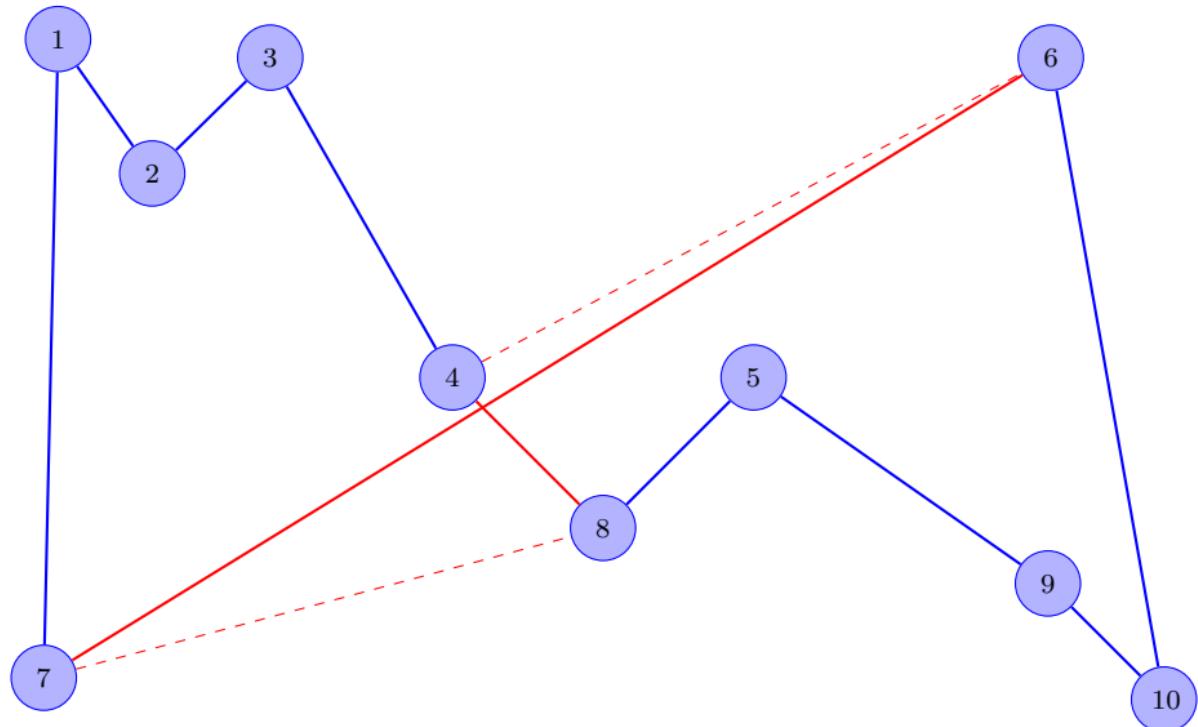
Pairwise Interchanges

2-Opt: Consider removing edges (4,8) and (6,7) and reconnecting



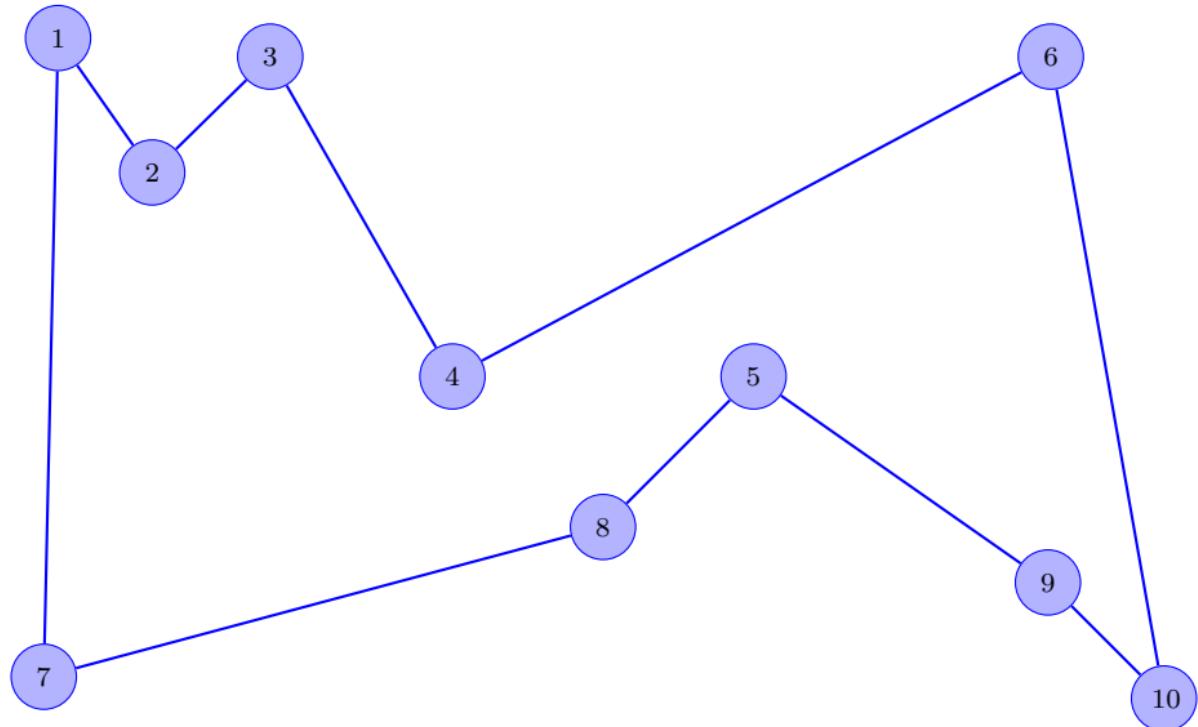
Pairwise Interchanges

2-Opt: Consider removing edges (4,8) and (6,7) and reconnecting



Pairwise Interchanges

2-Opt: Consider removing edges (4,8) and (6,7) and reconnecting



Observations

- Optimal solution

$1 - 2 - 4 - 7 - 8 - 5 - 9 - 10 - 6 - 3 - 1$, cost = 81.8mm

- Greedy solution:

$1 - 2 - 3 - 4 - 8 - 5 - 9 - 10 - 6 - 7 - 1$, cost = 90.9mm

- After the edge exchange (4,8) and (6,7):

$1 - 2 - 3 - 4 - 6 - 10 - 9 - 5 - 8 - 7 - 1$

$$\text{savings} = d_{48} + d_{67} = 4.2 + 25.0 = 29.2\text{mm}$$

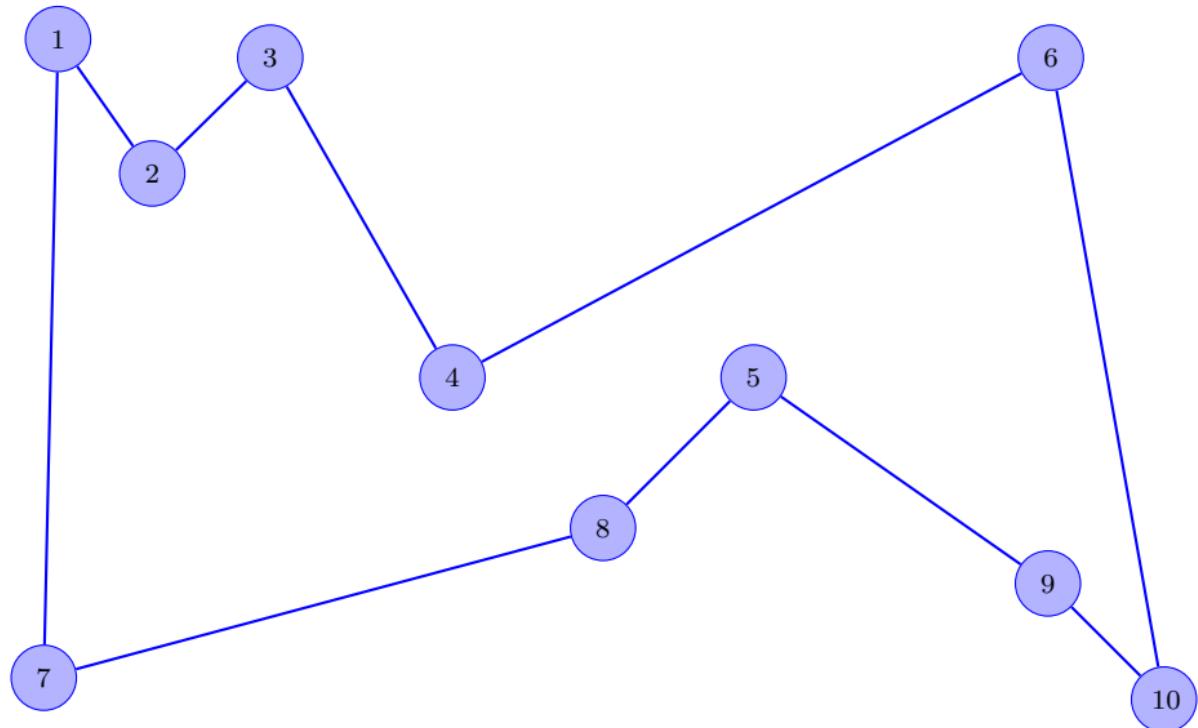
$$\text{costs} = d_{46} + d_{87} = 15.7 + 10.3 = 26.0\text{mm}$$

$$\text{solution cost} = 90.9 - 29.2 + 26.0 = 87.7\text{mm}$$

- The edge exchange has led to an improved solution!

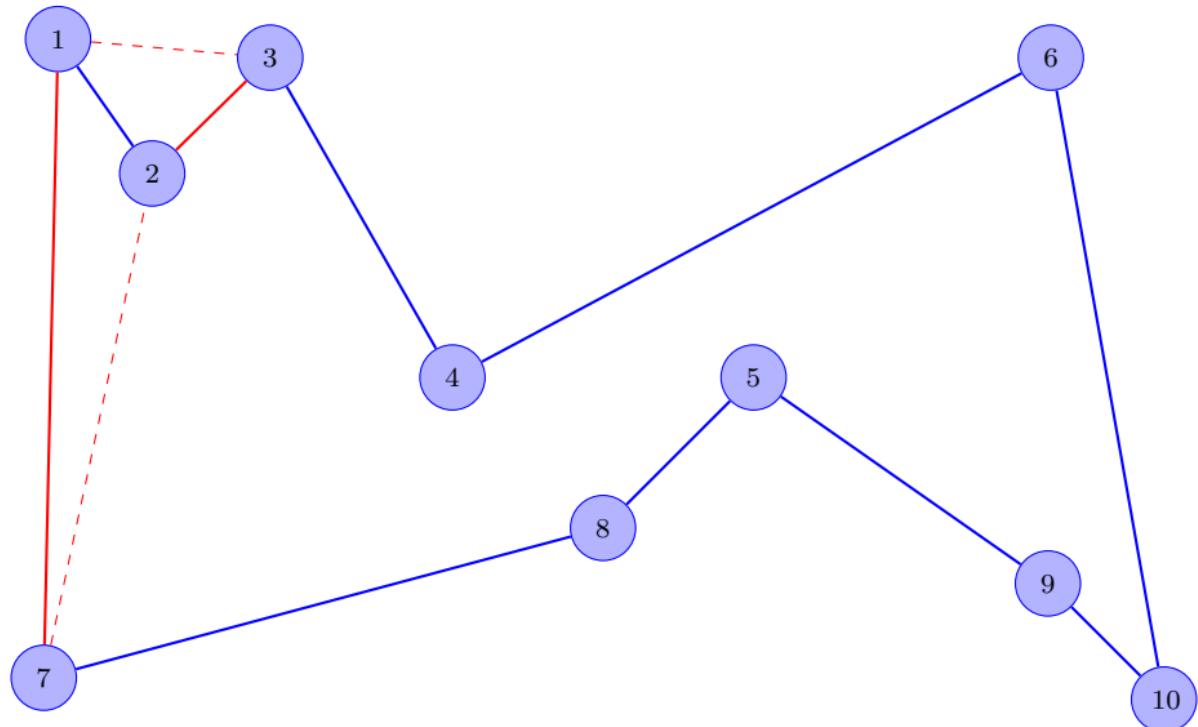
Pairwise Interchanges

2-Opt: Consider removing edges (1,7) and (2,3) and reconnecting



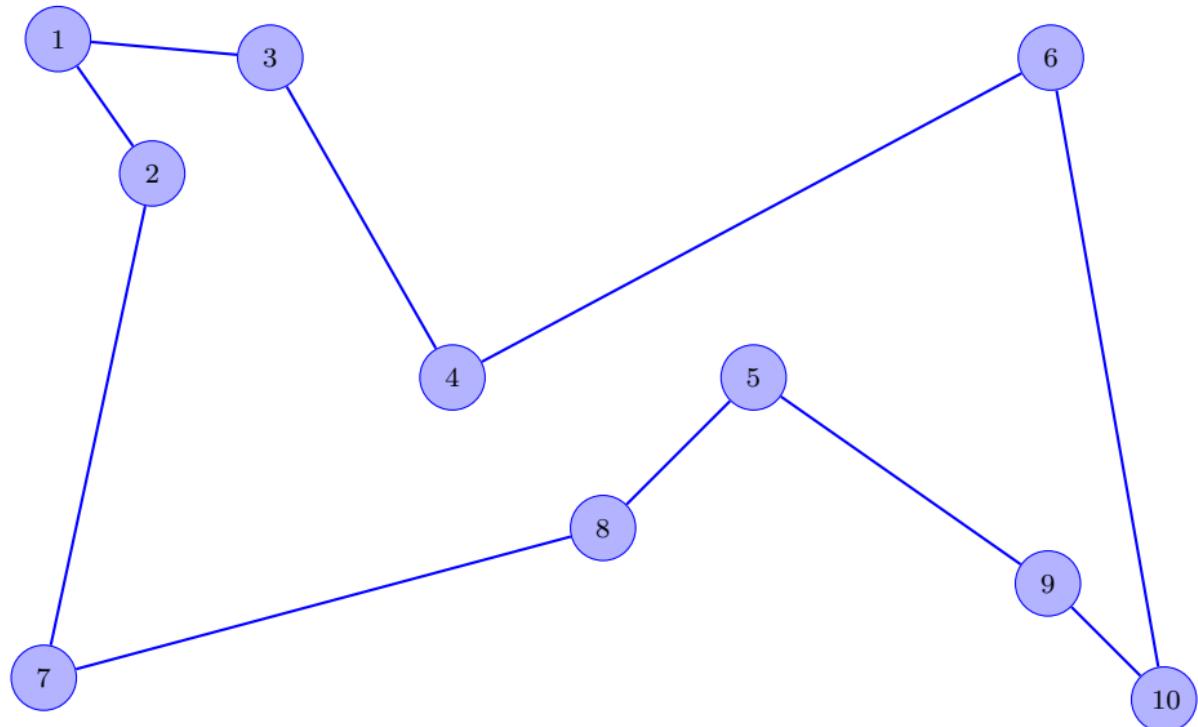
Pairwise Interchanges

2-Opt: Consider removing edges (1,7) and (2,3) and reconnecting



Pairwise Interchanges

2-Opt: Consider removing edges (1,7) and (2,3) and reconnecting



Observations

- Optimal solution

$1 - 2 - 4 - 7 - 8 - 5 - 9 - 10 - 6 - 3 - 1$, cost = 81.8mm

- Most recent solution:

$1 - 2 - 3 - 4 - 6 - 10 - 9 - 5 - 8 - 7 - 1$, cost = 87.7mm

- After the edge exchange (1,7) and (2,3):

$1 - 3 - 4 - 6 - 10 - 9 - 5 - 8 - 7 - 2 - 1$

$$\text{savings} = d_{17} + d_{23} = 16.0 + 3.6 = 19.6\text{mm}$$

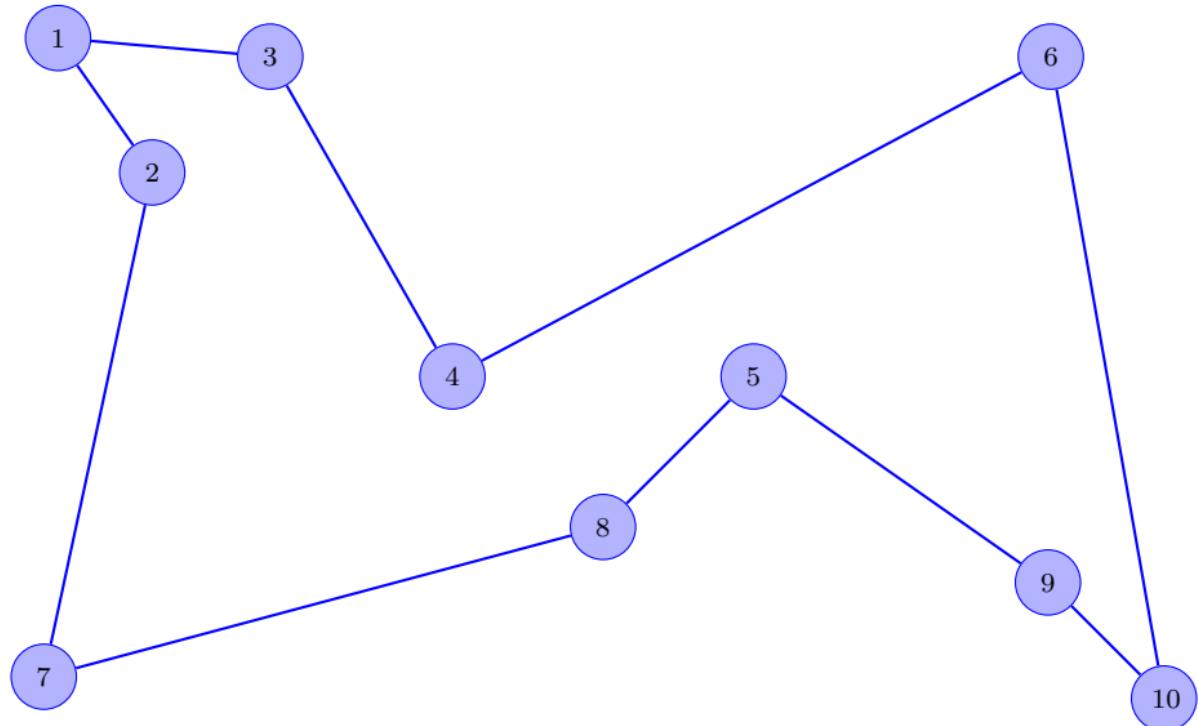
$$\text{costs} = d_{72} + d_{23} = 13.2 + 5.1 = 18.3\text{mm}$$

$$\text{solution cost} = 87.7 - 19.6 + 18.3 = 86.4\text{mm}$$

- The edge exchange has led to an improved solution!

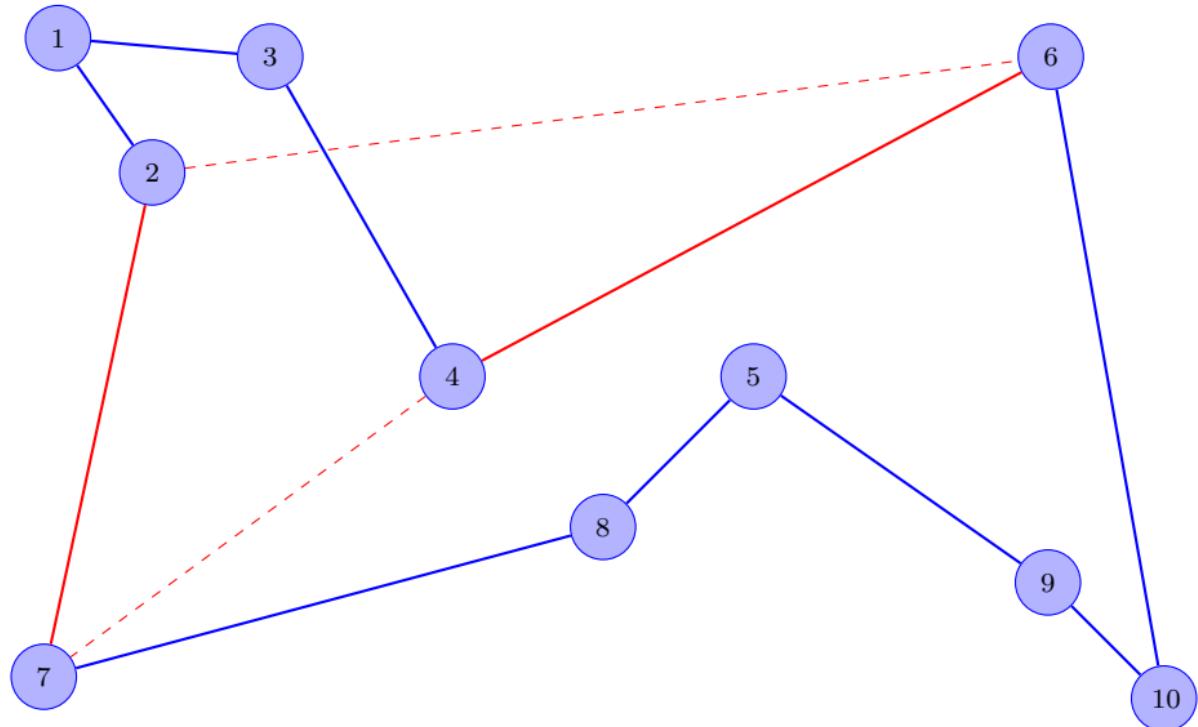
Pairwise Interchanges

2-Opt: Consider removing edges (4,6) and (2,7) and reconnecting



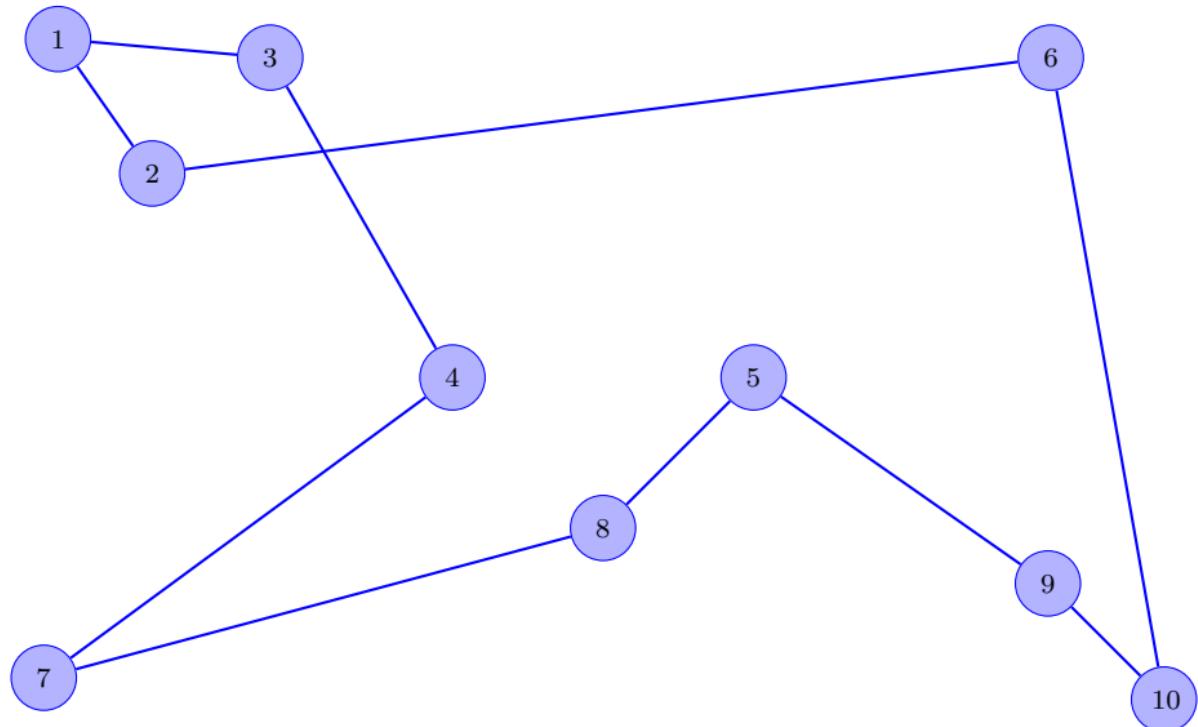
Pairwise Interchanges

2-Opt: Consider removing edges (4,6) and (2,7) and reconnecting



Pairwise Interchanges

2-Opt: Consider removing edges (4,6) and (2,7) and reconnecting



Observations

- Optimal solution

$1 - 2 - 4 - 7 - 8 - 5 - 9 - 10 - 6 - 3 - 1$, cost = 81.8mm

- Most recent solution:

$1 - 3 - 4 - 6 - 10 - 9 - 5 - 8 - 7 - 2 - 1$, cost = 86.4mm

- After the edge exchange (2,7) and (4,6):

$1 - 2 - 6 - 10 - 9 - 5 - 8 - 7 - 4 - 3 - 1$

$$\text{savings} = d_{27} + d_{46} = 13.2 + 15.7 = 28.9\text{mm}$$

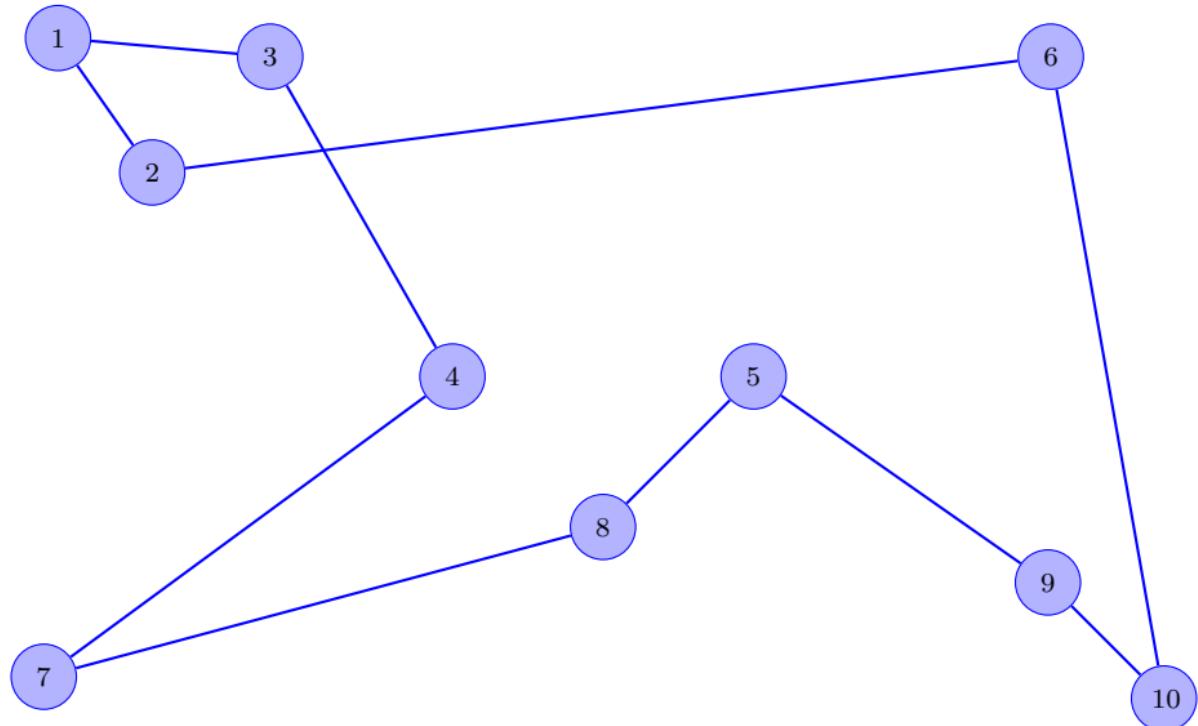
$$\text{costs} = d_{47} + d_{26} = 10 + 18.1 = 28.1\text{mm}$$

$$\text{solution cost} = 86.4 - 28.9 + 28.1 = 85.6\text{mm}$$

- The edge exchange has led to an improved solution!

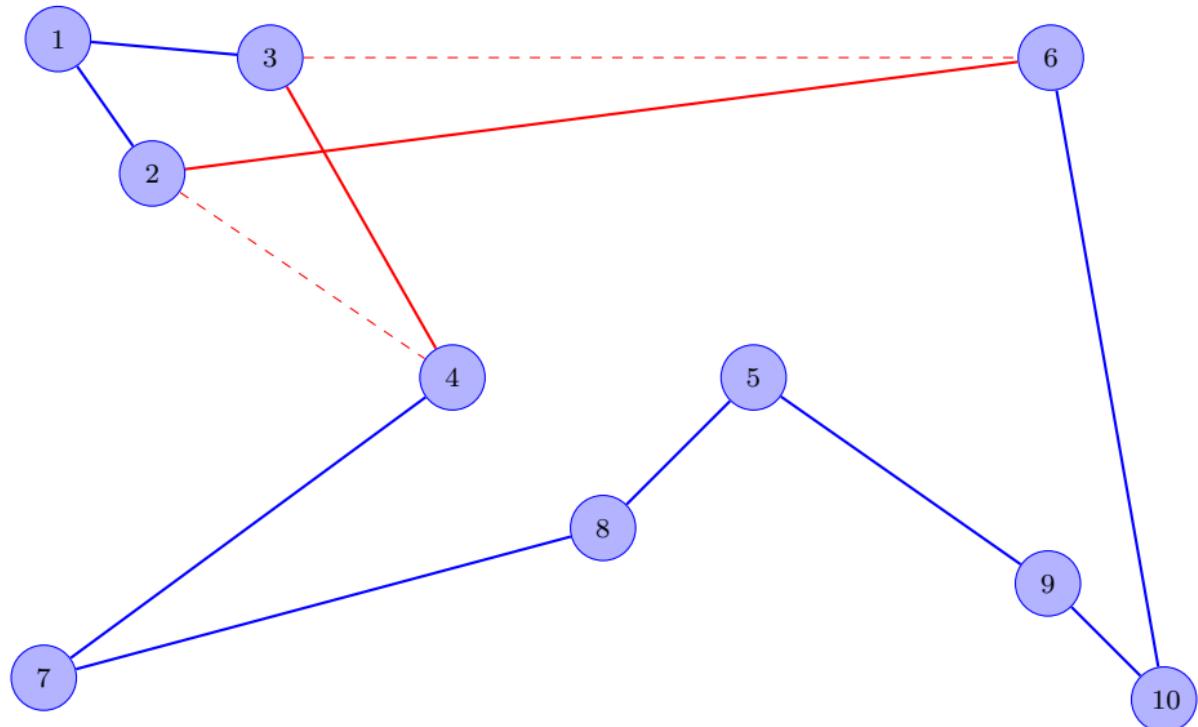
Pairwise Interchanges

2-Opt: Consider removing edges (3,4) and (2,6) and reconnecting



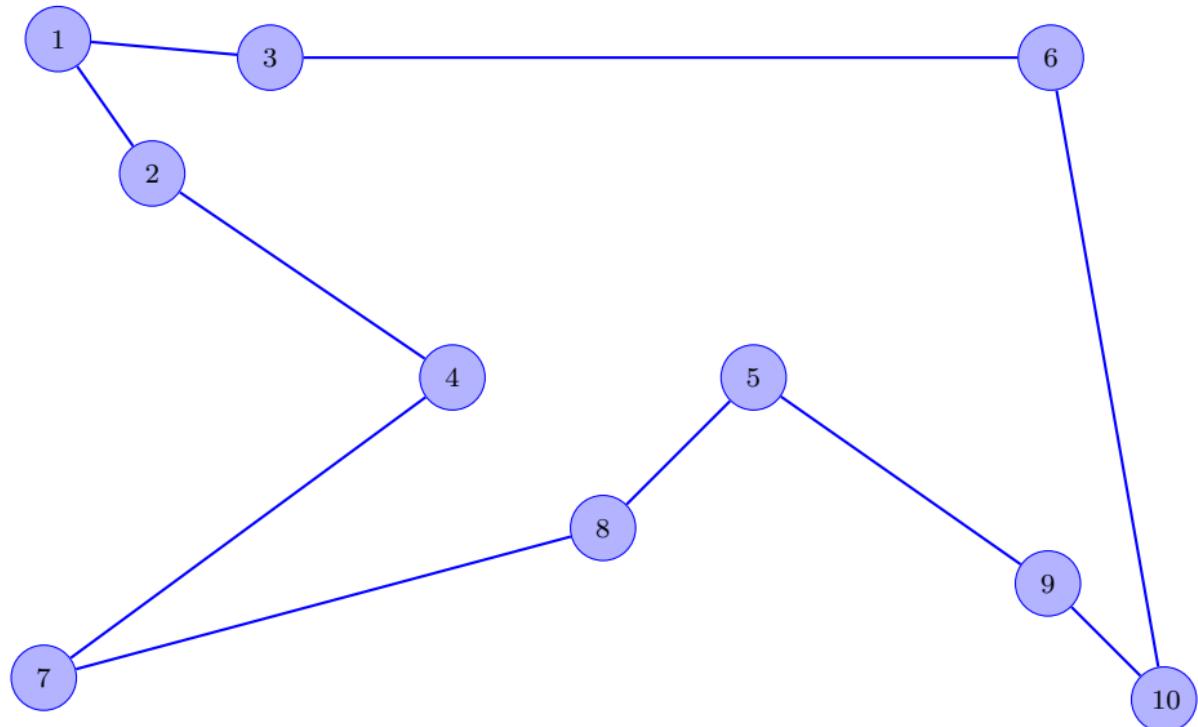
Pairwise Interchanges

2-Opt: Consider removing edges (3,4) and (2,6) and reconnecting



Pairwise Interchanges

2-Opt: Consider removing edges (3,4) and (2,6) and reconnecting



Observations

- Optimal solution

$1 - 2 - 4 - 7 - 8 - 5 - 9 - 10 - 6 - 3 - 1$, cost = 81.8mm

- Most recent solution:

$1 - 2 - 6 - 10 - 9 - 5 - 8 - 7 - 4 - 3 - 1$, cost = 85.6mm

- After the edge exchange (2,7) and (4,6):

$1 - 3 - 6 - 10 - 9 - 5 - 8 - 7 - 4 - 2 - 1 - 3$

$$\text{savings} = d_{34} + d_{26} = 7.1 + 18.1 = 25.2\text{mm}$$

$$\text{costs} = d_{36} + d_{24} = 15 + 6.4 = 21.4\text{mm}$$

$$\text{solution cost} = 85.6 - 25.2 + 21.4 = 81.8\text{mm}$$

- The edge exchange has led to the **optimal** solution!

Comments

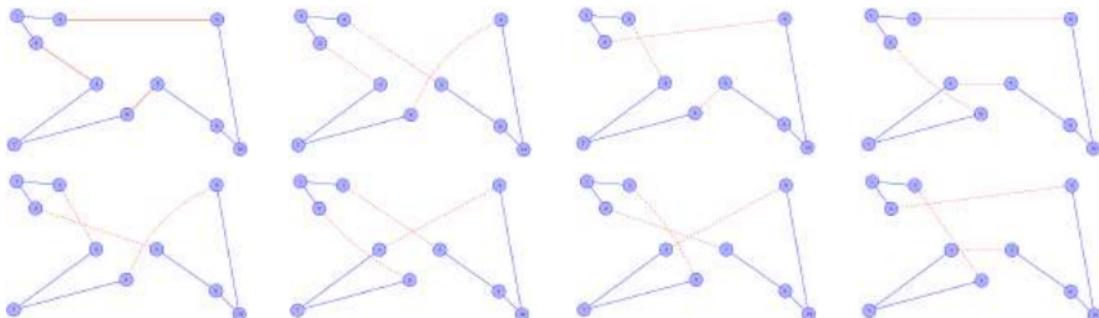
- In general the 2-opt exchange **will not** yield the optimal solution
- It will terminate with a **locally** optimal solution
- No **neighbouring** solution has a better objective value
- The final solution depends **on the order** in which the exchanges are performed
- The selection of which exchange to perform could also be greedy

3-opt Improvement

- A single 2-opt exchange results in one feasible alternative solution
- Consider a larger neighbourhood of a solution by exchanging more than two edges
- Removing three edges at random, create all feasible completions
- A larger neighbourhood generally leads to local optima of higher quality

3-opt Example - Circuit Board Problem

Consider edges (3,6), (5,8), and (2,4)



3-opt generates $2^3 - 1$ alternatives

- What about $k - opt$ exchange?
- Removing k edges results in $\mathcal{O}(n^k)$ new tours
- Neighbourhood size grows dramatically as k increases
- Terminating criteria
 - Iteration count without improvement
 - Time Limit

Learning Objectives

At the end of today's lecture you should be able to

- ① Formulate the Travelling Salesman Problem (TSP) using Integer Programming
- ② Understand the concept of solution neighbourhood
- ③ Explain what an improvement heuristic is
- ④ Implement the 2-opt and 3-opt exchange for the TSP

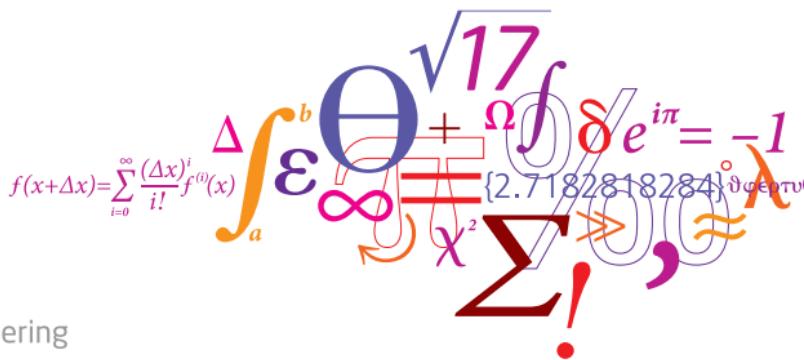
Richard Lusby
Associate Professor, DTU Management Engineering

Building 358, Room 145a rmlu@dtu.dk
2800 Kgs. Lyngby, Denmark +45 4525 3084 phone

Total Unimodularity

Richard M. Lusby

DTU Management



Today's lecture

- Julia Examples
- Introduction to Totally Unimodular matrices
- How to show that a matrix is totally unimodular
- Some examples

Learning Objectives

At the end of today's lecture you should be able to

- State what it means for a matrix to be *Totally Unimodular*
- Determine whether or not a matrix is totally unimodular
- Understand the importance from a linear programming perspective of having a totally unimodular constraint matrix
- Give examples of applications for which the constraint matrix is totally unimodular

Introduction

Consider

$$\max\{cx : Ax \leq b, x \in Z_+^n\}$$

This could also have been written as:

$$\begin{aligned} \max \quad & \sum_{i=1}^n c_i x_i \\ \text{st.} \quad & \sum_{i=1}^n a_{ij} x_i \leq b_j, \quad 1 \leq j \leq m \\ & x \in Z_+^n \end{aligned}$$

Question

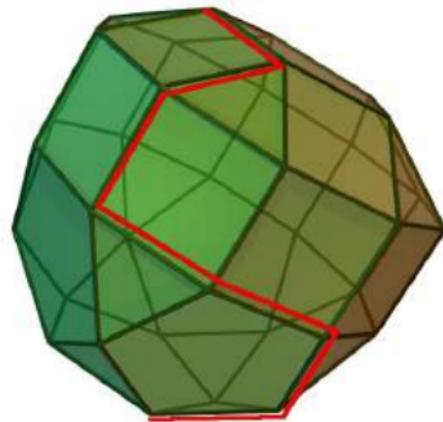
Let us assume that A and b only contains integers. When is an optimal integer solution to the LP-relaxation (LP) guaranteed?

Vertex Solution in Linear Programming

$$\sum_{i=1}^n a_{ij}x_j \leq b_j, \quad 1 \leq j \leq m$$

Vertex solution

Unique solution of n linearly independent tight inequalities.



Vertex Solution in Linear Programming

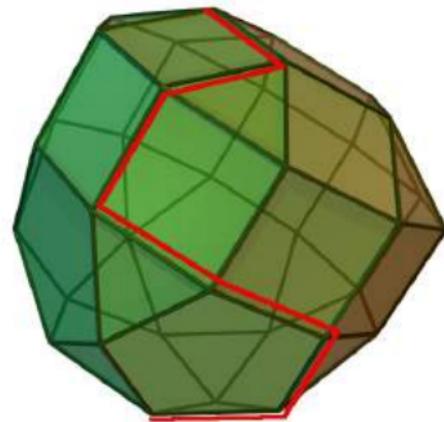
$$\sum_{i=1}^n a_{ij}x_j \leq b_j, \quad 1 \leq j \leq m$$

Vertex solution

Unique solution of n linearly independent tight inequalities.

From LP theory...

we have $(x_B, x_N) = (B^{-1}b, 0)$, where B is an $m \times m$ non-singular submatrix of (A, I) and I is a square identity matrix of size m .



Totally Unimodular Matrices

Sufficient condition

Sufficient condition: If the **optimal basis** B has $\det(B) = \pm 1$, then the LP-relaxation solves the IP.

This follows from **Cramer's Rule**: $B^{-1} = B^*/\det(B)$

Totally Unimodular Matrices

Sufficient condition

Sufficient condition: If the **optimal basis** B has $\det(B) = \pm 1$, then the LP-relaxation solves the IP.

This follows from **Cramer's Rule**: $B^{-1} = B^*/\det(B)$

Definition (TU)

A matrix A is **totally unimodular (TU)** if **every square** submatrix of A has determinant $+1, -1, 0$.

Totally Unimodular Matrices

Sufficient condition

Sufficient condition: If the **optimal basis** B has $\det(B) = \pm 1$, then the LP-relaxation solves the IP.

This follows from **Cramer's Rule**: $B^{-1} = B^*/\det(B)$

Definition (TU)

A matrix A is **totally unimodular (TU)** if **every square** submatrix of A has determinant $+1, -1, 0$.

Note: If A is TU, $a_{ij} \in \{+1, -1, 0\}$ for all i, j .

Totally Unimodular Matrices

Theorem

If A is totally unimodular then every vertex solution of $Ax \leq b$ is **integral**.

Proof

- ① A vertex solution is defined by a set of n linearly independent tight inequalities.
- ② Let A' denote the (square) submatrix of A which corresponds to those inequalities.
- ③ Then $A'x = b'$ where b' consists of the corresponding entries in b .
- ④ Since A is totally unimodular then $\det(A)$ must be 1 or -1 .
- ⑤ By Cramer's rule, x is integer.

Examples

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

This **is** totally unimodular as its determinant is equal to 1.

$$\begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

This **is not** totally unimodular as its determinant is equal to 2.

Another example

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Consider the following:

Another example

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

This **is not** totally unimodular as its determinant is equal to 2.

Consider the following:

Another example

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

This **is not** totally unimodular as its determinant is equal to 2.

Consider the following:

$$\begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$$

What about this?

$$\begin{pmatrix} 1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Is this totally unimodular or not?

Properties of TU

Proposition

$$\begin{aligned} A \text{ is TU} \\ \Updownarrow \\ A^T \text{ is TU} \\ \Updownarrow \\ (A, I) \text{ is TU.} \end{aligned}$$

More Properties of TU

Proposition (sufficient condition)

A is TU if,

- ① $a_{ij} \in \{0, 1, -1\}$ for all i, j .
- ② each column contains at most two non-zeros coefficients.
- ③ There exists a partition (M_1, M_2) of the rows such that for each column j containing two non-zeros we have $\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0$.

Examples (again)

$$\begin{pmatrix} 1 & -1 & -1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & -1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Application: Directed Graphs

Definition

Let A be the incidence matrix of a **directed** graph. Each row i represents a vertex v_i (or also just denoted i) and each column j represents an edge e_j (or also just denoted j).

- $a_{ij} = +1$ if vertex i is the tail of edge j
- $a_{ij} = -1$ if vertex i is the head of edge j
- $a_{ij} = 0$ otherwise

Application: Directed Graphs

Definition

Let A be the incidence matrix of a **directed** graph. Each row i represents a vertex v_i (or also just denoted i) and each column j represents an edge e_j (or also just denoted j).

- $a_{ij} = +1$ if vertex i is the tail of edge j
- $a_{ij} = -1$ if vertex i is the head of edge j
- $a_{ij} = 0$ otherwise

Proposition

The incidence matrix A of a directed graph is totally unimodular

Application: Directed Graphs

Definition

Let A be the incidence matrix of a **directed** graph. Each row i represents a vertex v_i (or also just denoted i) and each column j represents an edge e_j (or also just denoted j).

- $a_{ij} = +1$ if vertex i is the tail of edge j
- $a_{ij} = -1$ if vertex i is the head of edge j
- $a_{ij} = 0$ otherwise

Proposition

The incidence matrix A of a directed graph is totally unimodular

Consequence

A number of important network optimization problems like e.g. **max flow** and **min cost flow** can be solved efficiently as their matrices are totally unimodular.

Application: Bipartite Graphs

Definition

Let A be the incidence matrix of a **bipartite** graph. Now $a_{ij} = +1$ if and only if edge j is incident to vertex i .

Application: Bipartite Graphs

Definition

Let A be the incidence matrix of a **bipartite** graph. Now $a_{ij} = +1$ if and only if edge j is incident to vertex i .

Proposition

The incidence matrix A of a bipartite graph is totally unimodular

Application: Bipartite Graphs

Definition

Let A be the incidence matrix of a **bipartite** graph. Now $a_{ij} = +1$ if and only if edge j is incident to vertex i .

Proposition

The incidence matrix A of a bipartite graph is totally unimodular

Consequence

A number of important optimization problems can be modelled using bipartite graphs like eg. **maximum bipartite matching**. Due to the proposition they can be solved efficiently as their matrices are totally unimodular.

Learning Objectives

At the end of today's lecture you should be able to

- State what it means for a matrix to be *Totally Unimodular*
- Determine whether or not a matrix is totally unimodular
- Understand the importance from a linear programming perspective of having a totally unimodular constraint matrix
- Give examples of applications for which the constraint matrix is totally unimodular

Richard Lusby
Associate Professor, DTU Management Engineering

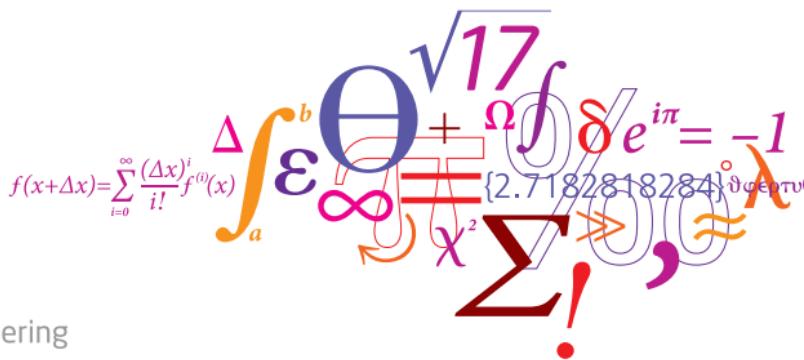
Building 358, Room 145a rmlu@dtu.dk
2800 Kgs. Lyngby, Denmark +45 4525 3084 phone

Solving Integer Linear Programs

LP Based Branch & Bound

Richard M. Lusby

DTU Management



Today's Agenda

- ① One or two exercises from last week
- ② Quick recap
- ③ Linear Programming (LP) - Based
Branch & Bound

Learning Objectives

At the end of today's lecture you should be able to

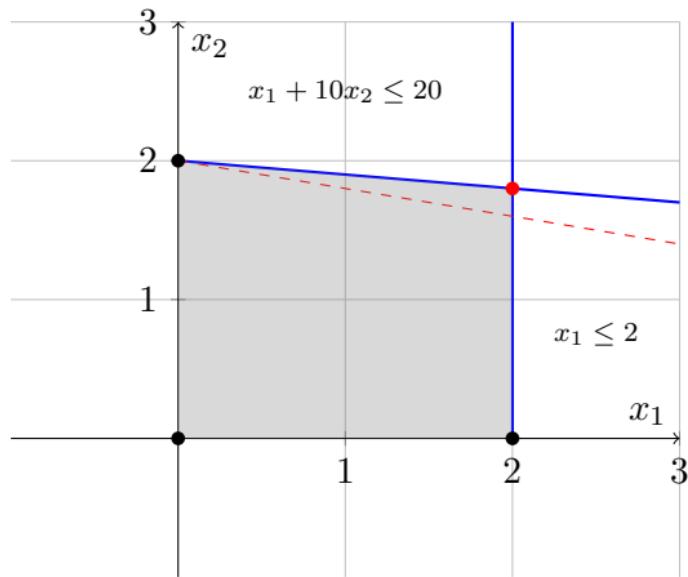
- ① Know what LP-Based Branch & Bound is
- ② Describe how LP-Based Branch & Bound works
- ③ Apply LP-Based Branch & Bound to solve integer linear programming problems

Solving an LP - What we know already

Consider the problem:

$$\begin{aligned} \text{Maximize: } & x_1 + 5x_2 \\ \text{s.t. } & x_1 + 10x_2 \leq 20 \\ & x_1 \leq 2 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

Solving an LP - What we know already



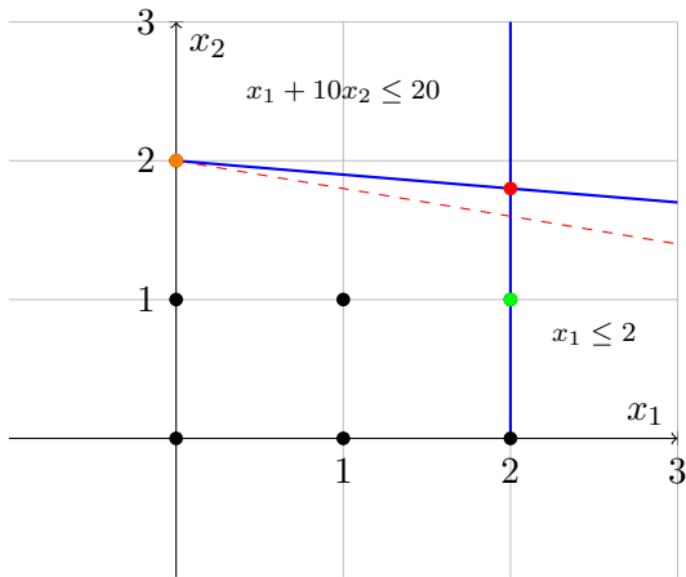
Optimal solution is $x_1 = 2, x_2 = 1.8, Z = 11$

Solving an IP - What happens?

Consider the problem:

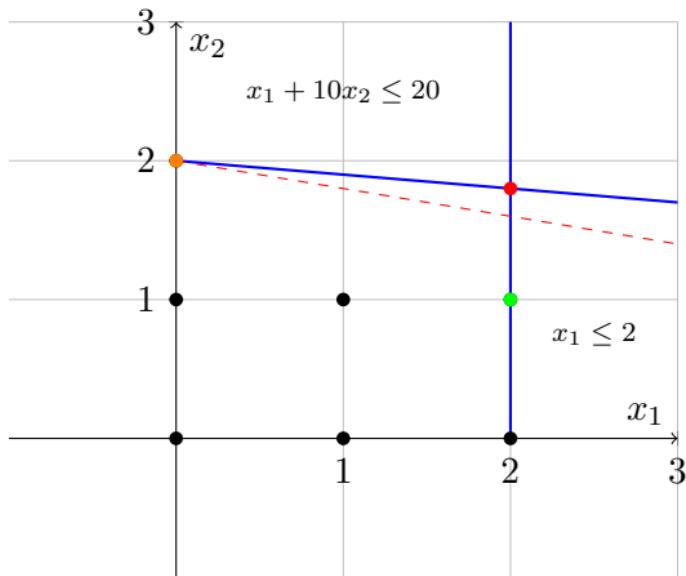
$$\begin{aligned} \text{Maximize: } & x_1 + 5x_2 \\ \text{s.t. } & x_1 + 10x_2 \leq 20 \\ & x_1 \leq 2 \\ & x_1 \in \mathbb{Z}_+ \\ & x_2 \in \mathbb{Z}_+ \end{aligned}$$

Solving an IP - What happens?



Rounded solution is $x_1 = 2, x_2 = 1, Z = 7$

Solving an IP - What happens?



Optimal solution is $x_1 = 0, x_2 = 2, Z = 10$

Comments

- Linear programs can be solved efficiently using the Simplex Algorithm
- No known integer programming algorithm possesses the same efficiency
- We could be lucky, our linear programming relaxation might be integer
- The structure of the problem could be such that all extreme points of the feasible region are naturally integer
- Integer programs have a finite number of possible solutions
 - potentially very many
- Adopt a *divide and conquer* approach
- LP-based Branch&Bound is one such approach

Solving an IP: LP-Based Branch&Bound

Consider the problem:

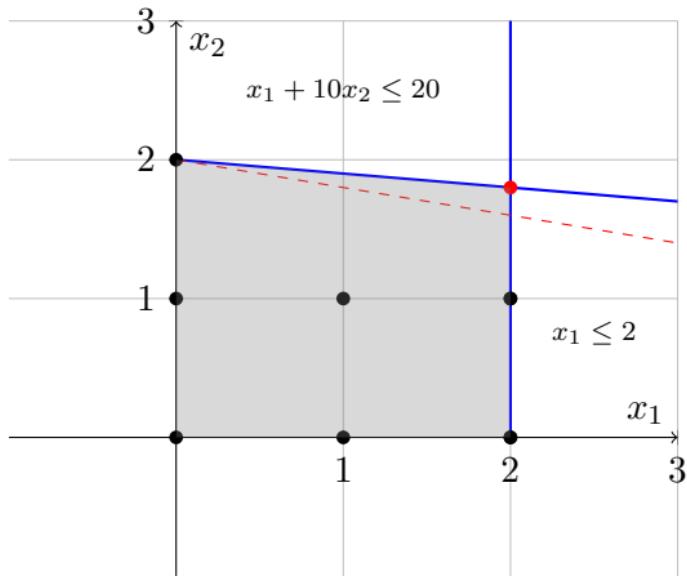
$$\begin{aligned} \text{Maximize: } & x_1 + 5x_2 \\ \text{s.t. } & x_1 + 10x_2 \leq 20 \\ & x_1 \leq 2 \\ & x_1 \in \mathbb{Z}_+ \\ & x_2 \in \mathbb{Z}_+ \end{aligned}$$

Solving an IP: LP-Based Branch&Bound

Step 1: Relax the integrality requirements

$$\begin{aligned} \text{Maximize: } & x_1 + 5x_2 \\ \text{s.t. } & x_1 + 10x_2 \leq 20 \\ & x_1 \leq 2 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{aligned}$$

The Feasible Region



Optimal solution is: $x_1 = 2, x_2 = 1.8, Z = 11$

LP-Relaxation Bound

Let us denote the optimal LP objective value by Z_{LP}^* and the optimal IP objective value by Z_{IP} . For maximization problems we have the following relationship:

$$Z_{IP}^* \leq Z_{LP}^*$$

The objective value of the integer program cannot be more than its LP-relaxation.

Fractional LP Solution

If the solution to the LP-Relaxation is not feasible for the integer program, i.e., if it contains a variable with a fractional value, then we must **branch** on that variable.

Branching on a variable removes the infeasibility by **partitioning** the solution space into smaller subproblems. Each subproblem must then be solved to determine its corresponding solution.

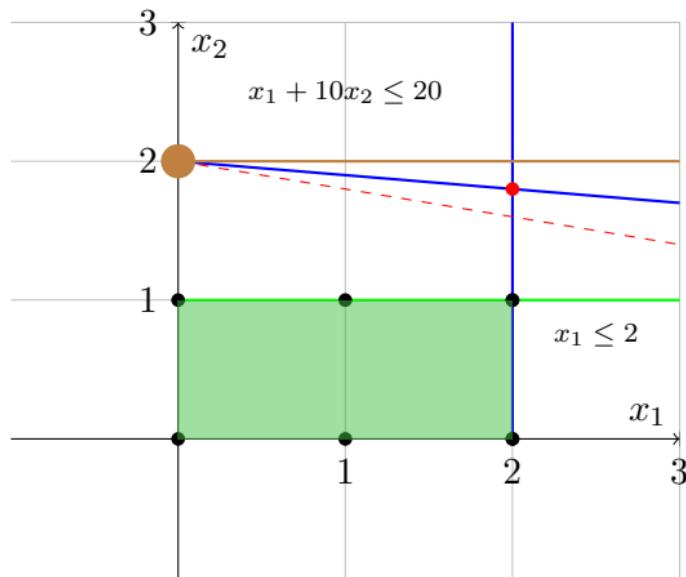
One way to do this is to define new **bounds** on the variable in each of the newly generated subproblems

In our example, $x_2 = 1.8$ is not feasible. We can create the following two new subproblems:

$$\text{LP-Relaxation} + x_2 \leq 1$$

$$\text{LP-Relaxation} + x_2 \geq 2$$

The Two New Subproblems

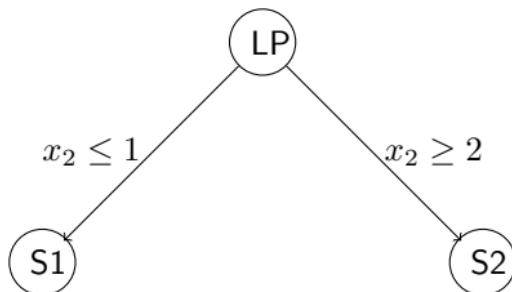


Optimal solution to Subproblem 1 (green): $x_1 = 2, x_2 = 1, Z = 7$

Optimal solution to Subproblem 2 (brown): $x_1 = 0, x_2 = 2, Z = 10$

Branch & Bound Tree

- **Branching** on a variable produces two new problems
- Can be visualized as a **Tree**



- **Branches** are depicted as arcs between nodes
- **Children** nodes inherit the parent formulation
- We continue **growing branches** (solving subproblems) until optimality is proven
- We only ever solve **linear programs**
- Not all nodes need to be considered. Nodes can be **pruned**.

Bounding Techniques (1)

Lower and upper bounds on the objective value can be used to ignore non-promising solutions

For a **Maximization Problem**, the following is true

- The LP-relaxation of any node in the tree provides an **upper bound** on the quality of any solution in the subtree emanating from the node (the subproblems that arise from a given problem are even more constrained)
- Feasible integer solutions provide **lower bounds**
- At any point in time, the **incumbent** is the feasible integer solution with the highest value found to date

Bounding Techniques (2)

A node/subproblem is removed from further consideration, **pruned** or **fathomed**, if:

- It's bound is at most the value of the incumbent's
- It's LP-relaxation has no feasible solutions
- The optimal solution to the LP-relaxation is integer. If it's value is better than that of the incumbent, the solution becomes the incumbent. More fathoming may occur as a result of this tighter bound.

Branch & Bound - Solving MIPs

For maximization problems we set $Z^* = -\infty$

While “live” subproblems exist, do the following:

- ① **Branching:** Select one of the remaining “live” subproblems and identify a fractional variable, x_j (with value x_j^*) to branch on. Create two new constraints by adding the respective constraints $x_j \leq \lfloor x_j^* \rfloor$ and $x_j \geq \lceil x_j^* \rceil$
- ② **Bounding:** For each new subproblem, obtain its bound by applying the simplex method to its relaxation and use the resulting value of Z for the resulting optimal solution.
- ③ **Fathoming/pruning:** For each new subproblem, see if it can be fathomed. Update incumbent if necessary.

Stop when there are no remaining subproblems. The incumbent is the optimal solution.

Example (1)

Solve the following Binary Integer Program with Branch & Bound

$$\begin{aligned} \text{Maximize } & 9x_1 + 5x_2 + 6x_3 + 4x_4 \\ \text{s.t. } & 6x_1 + 3x_2 + 5x_3 + 2x_4 \leq 10 \\ & x_3 + x_4 \leq 1 \\ & -x_1 + x_3 \leq 0 \\ & -x_2 + x_4 \leq 0 \\ & x_j \in \{0, 1\} \quad \text{for } j = 1, 2, 4 \end{aligned}$$

Example (2)

Solve the following Mixed Integer Program with Branch & Bound

$$\begin{aligned} \text{Maximize } & 4x_1 - 2x_2 + 7x_3 - x_4 \\ \text{s.t. } & x_1 + 5x_3 \leq 10 \\ & x_1 + x_2 - x_3 \leq 1 \\ & 6x_1 - 5x_2 \leq 0 \\ & -x_1 + 2x_3 - 2x_4 \leq 3 \\ & x_4 \geq 0 \\ & x_j \in \mathbb{Z}_+ \quad \text{for } j = 1, 2, 3 \end{aligned}$$

Learning Objectives

At the end of today's lecture you should be able to

- ① Know what LP-Based Branch & Bound is
- ② Describe how LP-Based Branch & Bound works
- ③ Apply LP-Based Branch & Bound to solve integer linear programming problems

Richard Lusby
Associate Professor, DTU Management Engineering

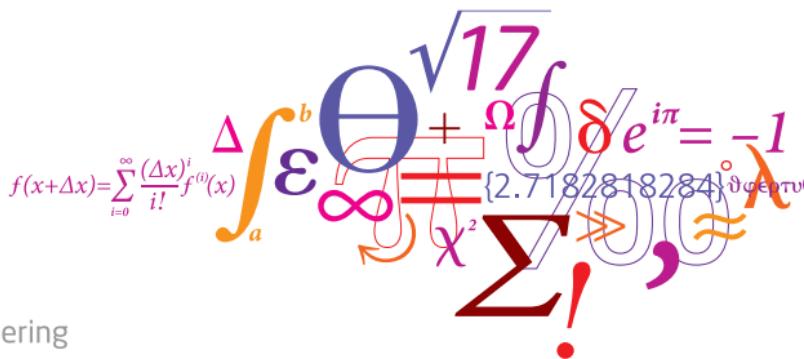
Building 424, Room 208 rmlu@dtu.dk
2800 Kgs. Lyngby, Denmark +45 4525 3084 phone

The Dual Simplex Method

The Transportation Problem and the Network Simplex Method

Richard M. Lusby

DTU Management



Today's Agenda

① Dual Simplex Method

- What is it?
- Why is it necessary?
- What can we use it for?

② Transportation Problem

- What is it?

③ Network Simplex Method

- What is it?
- A solution method for the Transportation Problem

Learning Objectives

At the end of today's lecture you should be able to

- ① Explain the connection between the Primal and Dual Simplex Methods
- ② Solve a linear program using the Dual Simplex Method
- ③ Know what Transportation problem is
- ④ Know what the Network Simplex Method is
- ⑤ Solve the Transportation problem using the Network Simplex Method

A Mathematical Model

$$\begin{aligned} \text{Maximize: } & Z = 3x_1 + 5x_2 \\ \text{s.t. } & x_1 \leq 4 \\ & 2x_2 \leq 12 \\ & 3x_1 + 2x_2 \leq 18 \\ & x_1, x_2 \geq 0 \end{aligned}$$

A Mathematical Model

$$\begin{aligned} \text{Maximize: } & Z = 3x_1 + 5x_2 \\ \text{s.t. } & x_1 \leq 4 \\ & 2x_2 \leq 12 \\ & 3x_1 + 2x_2 \leq 18 \\ & x_1, x_2 \geq 0 \end{aligned}$$

How do we solve this problem?

A Mathematical Model

$$\begin{aligned} \text{Maximize: } & Z = 3x_1 + 5x_2 \\ \text{s.t. } & x_1 + s_1 = 4 \\ & 2x_2 + s_2 = 12 \\ & 3x_1 + 2x_2 + s_3 = 18 \\ & x_1, x_2, s_1, s_2, s_3 \geq 0 \end{aligned}$$

How do we solve this problem?

(Primal) Simplex Pivoting

Initial Tableau

	Z	x_1	x_2	s_1	s_2	s_3	RHS
Z	1	-3	-5	0	0	0	0
s_1	0	1	0	1	0	0	4
s_2	0	0	2	0	1	0	12
s_3	0	3	2	0	0	1	18

Maintains **feasibility**, iterates towards **optimality**

(Primal) Simplex Pivoting

Initial Tableau

	Z	x_1	x_2	s_1	s_2	s_3	RHS
Z	1	-3	-5	0	0	0	0
s_1	0	1	0	1	0	0	4
s_2	0	0	2	0	1	0	12
s_3	0	3	2	0	0	1	18

Maintains **feasibility**, iterates towards **optimality**

(Primal) Simplex Pivoting

Updated Tableau

	Z	x_1	x_2	s_1	s_2	s_3	RHS
Z	1	-3	0	0	$\frac{5}{2}$	0	30
s_1	0	1	0	1	0	0	4
x_2	0	0	1	0	$\frac{1}{2}$	0	6
s_3	0	3	0	0	-1	1	6

Maintains **feasibility**, iterates towards **optimality**

(Primal) Simplex Pivoting

Optimal Tableau

	Z	x_1	x_2	s_1	s_2	s_3	RHS
Z	1	0	0	0	$\frac{3}{2}$	1	36
s_1	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	2
x_2	0	0	1	0	$\frac{1}{2}$	0	6
x_1	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	2

Maintains **feasibility**, iterates towards **optimality**

Another Mathematical Model

$$\text{Maximize: } Z = 3x_1 + 5x_2$$

$$\text{s.t.} \quad x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

$$x_1, x_2 \geq 0$$

$$\text{Minimize: } Z = 4y_1 + 12y_2 + 18y_3$$

$$\text{s.t.} \quad y_1 + 3y_3 \geq 3$$

$$2y_2 + 2y_3 \geq 5$$

$$y_1, y_2, y_3 \geq 0$$

What is the relationship between the two formulations?

A Simplex Tableau is optimal if it is primal and dual feasible

Consider the Dual Formulation

$$\begin{aligned} \text{Minimize: } & Z = 4y_1 + 12y_2 + 8y_3 \\ \text{s.t. } & y_1 + 3y_3 \geq 3 \\ & 2y_2 + 2y_3 \geq 5 \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

Can we solve this using the Simplex Method?

Consider the Dual Formulation

$$\begin{aligned} \text{Maximize: } & Z = -4y_1 - 12y_2 - 8y_3 \\ \text{s.t. } & -y_1 - 3y_3 + z_1 = -3 \\ & -2y_2 - 2y_3 + z_2 = -5 \\ & y_1, y_2, y_3 \geq 0 \end{aligned}$$

Can we solve this using the Simplex Method?

Simplex Pivoting

Initial Tableau

	Z	y_1	y_2	y_3	z_1	z_2	RHS
Z	1	4	12	18	0	0	0
z_1	0	-1	0	-3	1	0	-3
z_2	0	0	-2	-2	0	1	-5

Not **Dual Feasible** but is **Primal Feasible**

Can we **pivot** directly in the **Primal Space** using the **Dual Tableau**?

Dual Simplex Pivoting

Identify the Dual leaving variable/Primal entering variable

	Z	y_1	y_2	y_3	z_1	z_2	RHS	
Z	1	4	12	18	0	0	0	
z_1	0	-1	0	-3	1	0	-3	
z_2	0	0	-2	-2	0	1	-5	←

Select the most row with the most negative right hand side

Dual Simplex Pivoting

Identify the Dual entering variable/Primal leaving variable

	Z	y ₁	y ₂	y ₃	Z ₁	z ₂	RHS
Z	1	4	12	18	0	0	0
z ₁	0	-1	0	-3	1	0	-3
z ₂	0	0	-2	-2	0	1	-5 ←

How much can we increase the value of x_2 ?

Mirror Primal Ratio Test :

Find the non-basic variable i that minimizes the ratio

$$\left| \frac{\text{row}_0^i}{\text{row}_p^i} \right|$$

Consider only basic variables with negative coefficients in row p

Dual Simplex Pivoting

Identify the Dual entering variable/Primal leaving variable

	Z	y ₁	y ₂	y ₃	Z ₁	z ₂	RHS
Z	1	4	12	18	0	0	0
z ₁	0	-1	0	-3	1	0	-3
z ₂	0	0	-2	-2	0	1	-5 ←

How much can we increase the value of x_2 ?

Mirror Primal Ratio Test :

Find the non-basic variable i that minimizes the ratio

$$\left| \frac{\text{row}_0^i}{\text{row}_p^i} \right|$$

Consider only basic variables with negative coefficients in row p

Dual Simplex Pivoting

Variable y_2 is the entering variable (s_2 primal leaving variable)

	Z	y_1	y_2	y_3	z_1	z_2	RHS
Z	1	4	12	18	0	0	0
z_1	0	-1	0	-3	1	0	-3
z_2	0	0	-2	-2	0	1	-5 ←

Dual Simplex Pivoting

Variable y_2 is the entering variable (s_2 primal leaving variable)

	Z	y_1	y_2	y_3	z_1	z_2	RHS
Z	1	4	12	18	0	0	0
z_1	0	-1	0	-3	1	0	-3
z_2	0	0	-2	-2	0	1	-5 ←

Dual Simplex Pivoting - Update Tableau

Variable y_2 is the entering variable (s_2 primal leaving variable)

	Z	y_1	y_2	y_3	z_1	z_2	RHS
Z	1	4	0	6	0	6	-30
z_1	0	-1	0	-3	1	0	-3
y_2	0	0	1	1	0	$-\frac{1}{2}$	$\frac{5}{2}$

Dual Simplex Pivoting - Update Tableau

Variable y_2 is the entering variable (s_2 primal leaving variable)

	Z	y_1	y_2	y_3	z_1	z_2	RHS
Z	1	4	0	6	0	6	-30
z_1	0	-1	0	-3	1	0	-3 ←
y_2	0	0	1	1	0	$-\frac{1}{2}$	$\frac{5}{2}$

Select the row with the most negative right hand side

Dual Simplex Pivoting - Update Tableau

Variable y_2 is the entering variable (s_2 primal leaving variable)

	Z	y_1	y_2	y_3	z_1	z_2	RHS
Z	1	4	0	6	0	6	-30
z_1	0	-1	0	-3	1	0	-3
y_2	0	0	1	1	0	$-\frac{1}{2}$	$\frac{5}{2}$

Select the row with the most negative right hand side

Dual leaving variable: z_1 (x_1 entering in primal)

Dual entering variable: y_3 (s_3 leaving in primal)

Dual Simplex Pivoting - Update Tableau

Variable y_3 is the entering variable (s_3 primal leaving variable)

	Z	y_1	y_2	y_3	z_1	z_2	RHS
Z	1	2	0	0	2	6	-36
y_3	0	$\frac{1}{3}$	0	1	$-\frac{1}{3}$	0	1
y_2	0	$-\frac{1}{3}$	1	0	$\frac{1}{3}$	$-\frac{1}{2}$	$\frac{3}{2}$

Dual Simplex Pivoting - Update Tableau

Variable y_3 is the entering variable (s_3 primal leaving variable)

	Z	y_1	y_2	y_3	z_1	z_2	RHS
Z	1	2	0	0	2	6	-36
y_3	0	$\frac{1}{3}$	0	1	$-\frac{1}{3}$	0	1
y_2	0	$-\frac{1}{3}$	1	0	$\frac{1}{3}$	$-\frac{1}{2}$	$\frac{3}{2}$

Is the solution optimal?

Dual Simplex Pivoting - Update Tableau

Variable y_3 is the entering variable (s_3 primal leaving variable)

	Z	y_1	y_2	y_3	z_1	z_2	RHS
Z	1	2	0	0	2	6	-36
y_3	0	$\frac{1}{3}$	0	1	$-\frac{1}{3}$	0	1
y_2	0	$-\frac{1}{3}$	1	0	$\frac{1}{3}$	$-\frac{1}{2}$	$\frac{3}{2}$

Is the solution optimal?

Dual simplex maintains **optimality**, iterates towards **feasibility**

Dual Simplex Pivoting - Reoptimization

Consider the optimal Primal Tableau and add the constraint $x_2 \leq 5$

	Z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
Z	1	0	0	0	$\frac{3}{2}$	1	0	36
s_1	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	0	2
x_2	0	0	1	0	$\frac{1}{2}$	0	0	6
x_1	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	0	2
s_4	0	0	1	0	0	0	1	5

- Primal infeasible (but Dual feasible) - Pivot in the Dual Space!

Dual Simplex Pivoting - Reoptimization

Consider the optimal Primal Tableau and add the constraint $x_2 \leq 5$

	Z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
Z	1	0	0	0	$\frac{3}{2}$	1	0	36
s_1	0	0	0	1	$\frac{1}{3}$	$-\frac{1}{3}$	0	2
x_2	0	0	1	0	$\frac{1}{2}$	0	0	6
x_1	0	1	0	0	$-\frac{1}{3}$	$\frac{1}{3}$	0	2
s_4	0	0	0	0	$-\frac{1}{2}$	0	1	-1

- Primal infeasible (but Dual feasible) - Pivot in the Dual Space!

Dual Simplex Pivoting - Reoptimization

Consider the optimal Primal Tableau and add the constraint $x_2 \leq 5$

	Z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
Z	1	0	0	0	0	1	3	33
s_1	0	0	0	1	0	$-\frac{1}{3}$	$\frac{2}{3}$	$\frac{4}{3}$
x_2	0	0	1	0	0	0	1	5
x_1	0	1	0	0	0	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{8}{3}$
s_4	0	0	0	0	1	0	-2	2

- Primal infeasible (but Dual feasible) - Pivot in the Dual Space!

Dual Simplex Pivoting - Reoptimization

Consider the optimal Primal Tableau and add the constraint $x_2 \leq 5$

	Z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
Z	1	0	0	0	0	1	3	33
s_1	0	0	0	1	0	$-\frac{1}{3}$	$\frac{2}{3}$	$\frac{4}{3}$
x_2	0	0	1	0	0	0	1	5
x_1	0	1	0	0	0	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{8}{3}$
s_4	0	0	0	0	1	0	-2	2

- Primal infeasible (but Dual feasible) - Pivot in the Dual Space!
- **One iteration!**

Dual Simplex Pivoting - Reoptimization

Consider the optimal Primal Tableau and add the constraint $x_2 \leq 5$

	Z	x_1	x_2	s_1	s_2	s_3	s_4	RHS
Z	1	0	0	0	0	1	3	33
s_1	0	0	0	1	0	$-\frac{1}{3}$	$\frac{2}{3}$	$\frac{4}{3}$
x_2	0	0	1	0	0	0	1	5
x_1	0	1	0	0	0	$\frac{1}{3}$	$-\frac{2}{3}$	$\frac{8}{3}$
s_4	0	0	0	0	1	0	-2	2

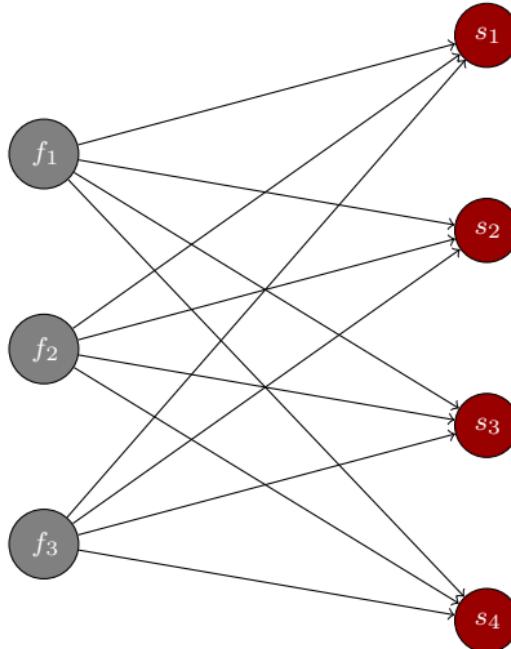
- Primal infeasible (but Dual feasible) - Pivot in the Dual Space!
- **One iteration!**
- Typically utilized in Branch & Bound

The Transportation Problem

- Well known problem in Operations Research
- Typically requires the distribution of a single commodity
- A set of sources is given and supplies the commodity
- A set of destinations is also given with known demands
- The aim is to transport the commodity from the sources to the destinations at minimum cost

Network Representation

Three factories $\{f_1, f_2, f_3\}$ supply four stores $\{s_1, s_2, s_3, s_4\}$



Problem Input

		Store				Supply
		s_1	s_2	s_3	s_4	
Factory	f_1	464	513	654	867	75
	f_2	352	416	690	791	125
	f_3	995	682	388	685	100
Demand		80	65	70	85	300

Table: Problem Data

Generic Formulation

$x_{ij} :=$ flow from supplier $i = 1, 2, \dots, n$ to destination $j = 1, 2, \dots, n$

$$\text{Minimize: } \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$$

$$\text{s.t. } \sum_{j=1}^n x_{ij} = s_i \quad \text{for } i = 1, 2, \dots, n$$

$$\sum_{i=1}^n x_{ij} = d_j \quad \text{for } j = 1, 2, \dots, n$$

$$x_{ij} \geq 0 \quad \text{for } i = 1, 2, \dots, n \text{ for } j = 1, 2, \dots, n$$

An Equivalent Formulation

x_{ij} := flow from supplier $i = 1, 2, \dots, n$ to destination $j = 1, 2, \dots, n$

$$\text{Minimize: } \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij}$$

$$\text{s.t. } -\sum_{j=1}^n x_{ij} = -s_i \quad \text{for } i = 1, 2, \dots, n$$

$$\sum_{i=1}^n x_{ij} = d_j \quad \text{for } j = 1, 2, \dots, n$$

$$x_{ij} \geq 0 \quad \text{for } i = 1, 2, \dots, n \text{ for } j = 1, 2, \dots, n$$

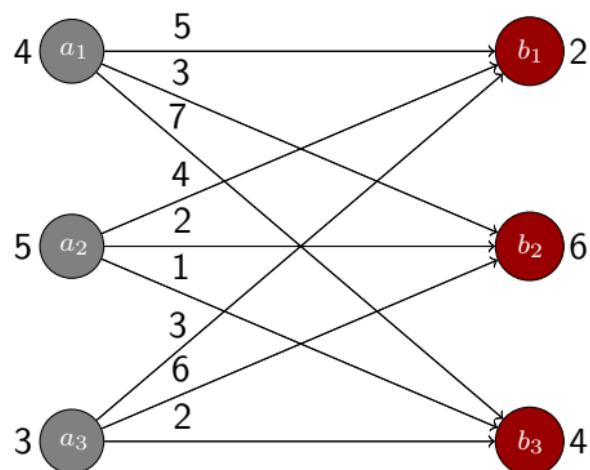
Solving the Transportation with Network Simplex

An Example

Three suppliers $\{a_1, a_2, a_3\}$ and three receivers $\{b_1, b_2, b_3\}$

	Receivers			
	b_1	b_2	b_3	Supply
a_1	5	3	7	4
a_2	4	2	1	5
a_3	3	6	2	3
Demand	2	6	4	12

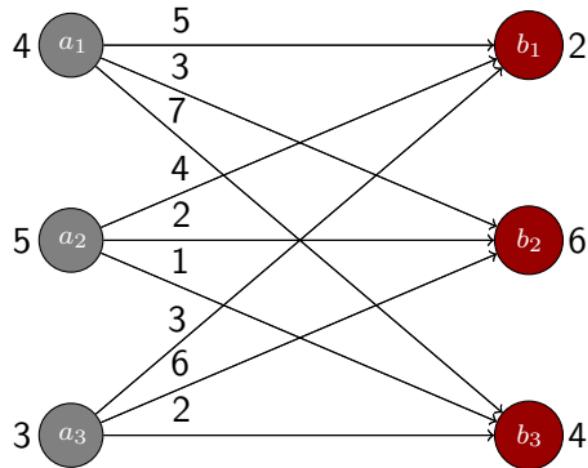
Table: Problem Data



Initial Solution: Greedy Heuristic

- Consider the suppliers in the order a_1 , a_2 , and a_3
- For each supplier, find the recipients who have not yet received all their goods.
Choose the cheapest of these and satisfy the demand (if possible)
- If the supplier has remaining goods, deliver these to the second cheapest recipient and so on, otherwise move to the next supplier

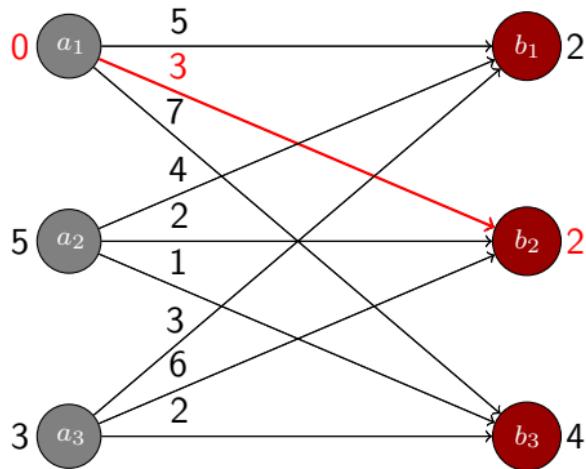
Greedy Heuristic



Solution:

Cost:

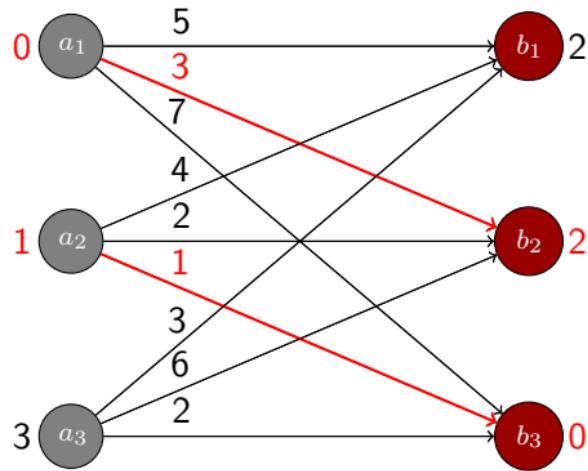
Greedy Heuristic



Solution: $(a_1, b_2) = 4$

Cost: 12

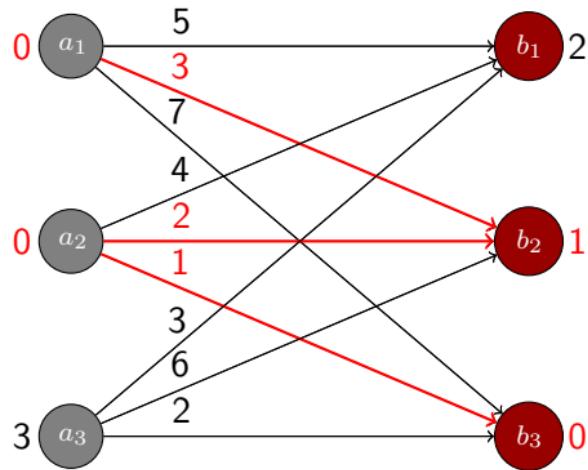
Greedy Heuristic



Solution: (a₁, b₂) = 4, (a₂, b₃) = 4

Cost: 16

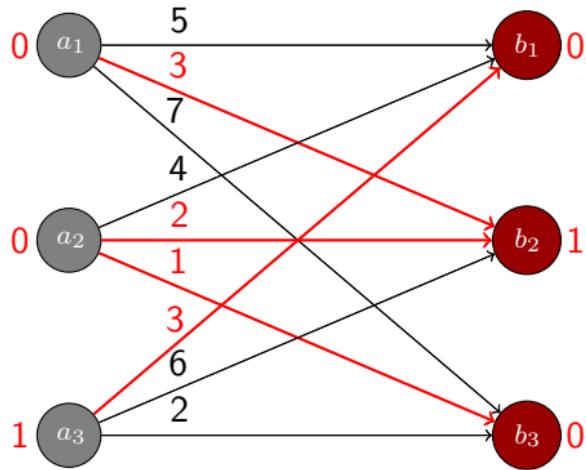
Greedy Heuristic



Solution: $(a_1, b_2) = 4, (a_2, b_3) = 4, (a_2, b_2) = 1$

Cost: 18

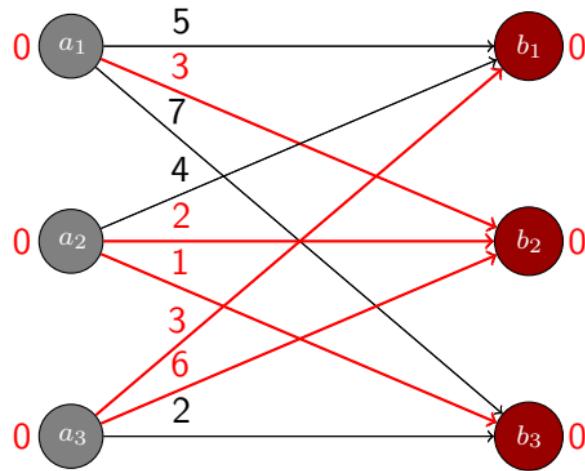
Greedy Heuristic



Solution: (a₁, b₂) = 4, (a₂, b₃) = 4, (a₂, b₂) = 1, (a₃, b₁) = 2

Cost: 24

Greedy Heuristic



Solution: (a₁, b₂) = 4, (a₂, b₃) = 4, (a₂, b₂) = 1, (a₃, b₁) = 2, (a₃, b₂) = 1

Cost: 30

Solution Characteristics

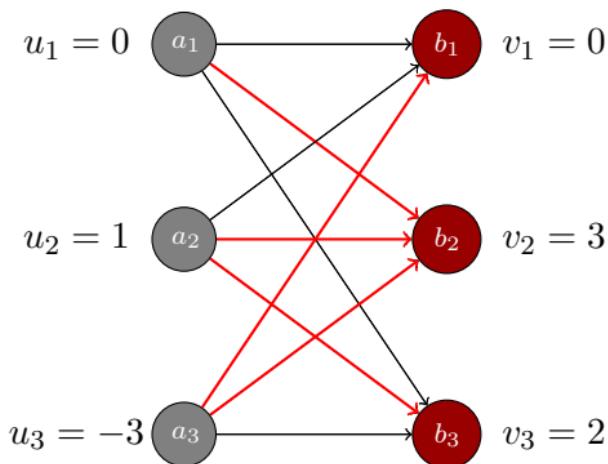
- Five arcs used (number of recipients minus one)
- Forms a **tree** in the network
- If fewer than five were used, we would select as many as needed in such a way that we get a tree in the network
- The used arcs form a “basic” solution

Iterating to a better solution

Compute the dual variables

$$c_{ij} + u_i - v_j = 0$$

Start by setting $u_1=0$



$$v_2 = u_1 + c_{12} = 0 + 3 = 3$$

$$u_2 = v_2 - c_{22} = 3 - 2 = 1$$

$$u_3 = v_2 - c_{32} = 3 - 6 = -3$$

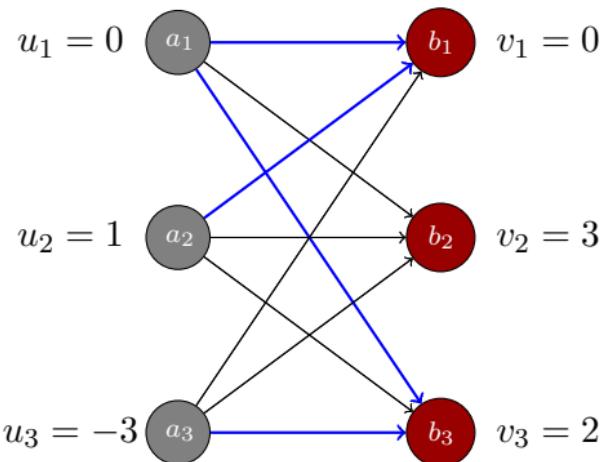
$$v_3 = u_2 + c_{23} = 1 + 1 = 2$$

$$v_1 = u_3 + c_{31} = -3 + 3 = 0$$

Iterating to a better solution

Optimality Check - Compute the reduced cost for non-basic variables

$$\bar{c}_{ij} = c_{ij} + u_i - v_j$$



$$\bar{c}_{11} = 5 + 0 - 0 = 5$$

$$\bar{c}_{13} = 7 + 0 - 2 = 5$$

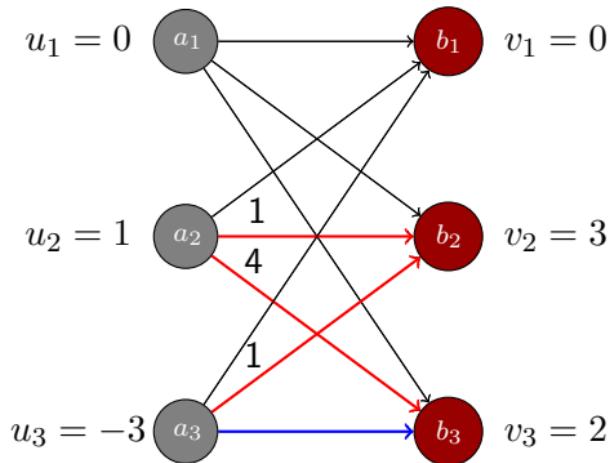
$$\bar{c}_{21} = 4 + 1 - 0 = 5$$

$$\bar{c}_{33} = 2 - 3 - 2 = -3$$

Identify Leaving Variable

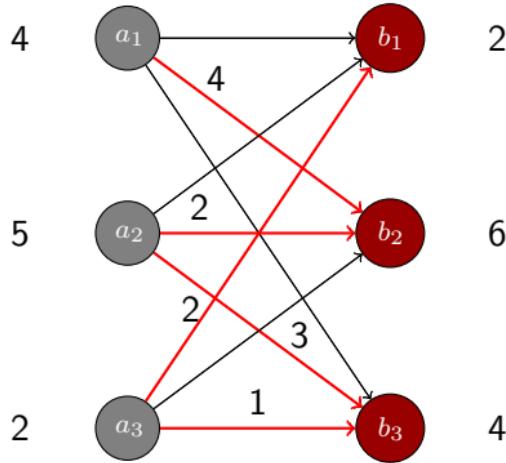
Detect the cycle $(a_3, b_3, a_2, b_2, a_3)$

How much flow can we send on arc (a_3, b_3) ?



Arcs traversed backwards limit the flow \rightarrow flow on $(a_3, b_3) = 1$

New Solution

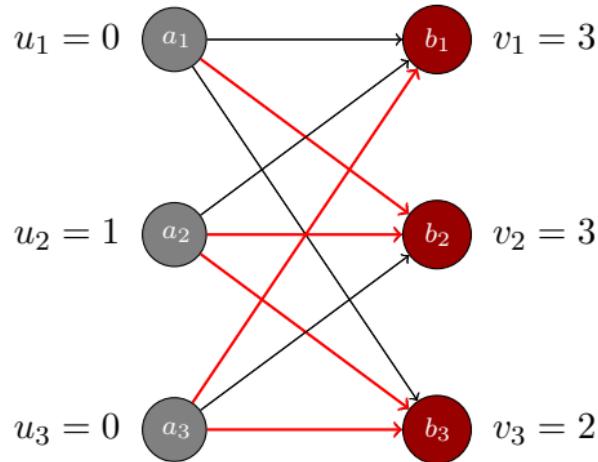


Solution: $(a_1, b_2) = 4$, $(a_2, b_3) = 3$, $(a_2, b_2) = 2$, $(a_3, b_1) = 2$, $(a_3, b_3) = 1$

Cost: 27

Compute Dual Variables

$$c_{ij} + u_i - v_j = 0$$



The solution is the optimal, why?

Learning Objectives

At the end of today's lecture you should be able to

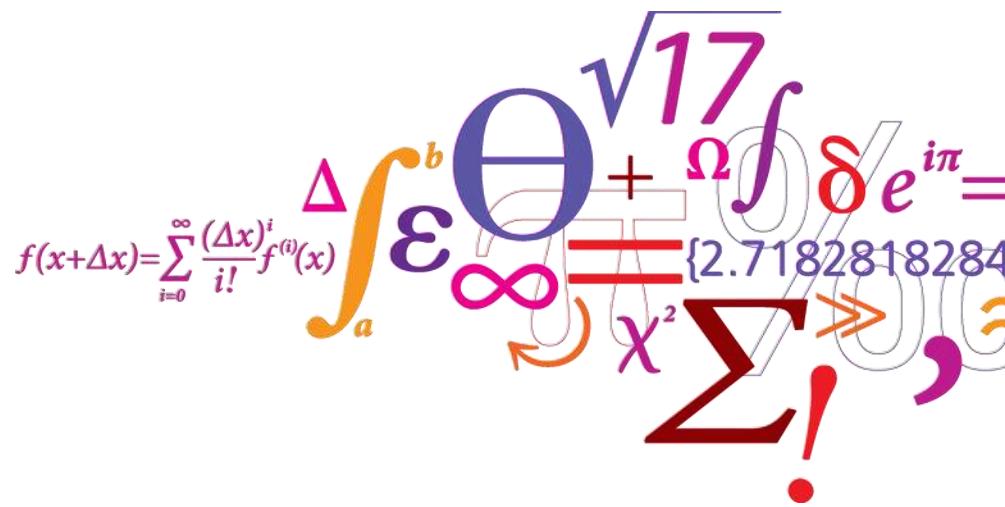
- ① Explain the connection between the Primal and Dual Simplex Methods
- ② Solve a linear program using the Dual Simplex Method
- ③ Know what Transportation problem is
- ④ Know what the Network Simplex Method is
- ⑤ Solve the Transportation problem using the Network Simplex Method

Richard Lusby
Associate Professor, DTU Management Engineering

Building 358, Room 145a rmlu@dtu.dk
2800 Kgs. Lyngby, Denmark +45 4525 3084 phone

Evaluation, exam, other courses

Stefan Røpke, Richard Lusby

$$f(x+\Delta x) = \sum_{i=0}^{\infty} \frac{(\Delta x)^i}{i!} f^{(i)}(x)$$


1 It is my opinion, that...

1.1 I learned a lot from this course.

Completely disagree (0) Disagree (2)

Neutral (3)

Agree (4) Completely agree (5)

The course 4.2

47 out of 140 has
answered (34%)

1.2 the teaching activities throughout this course has been in accordance to
the learning objectives. - [View Learning Objectives]

Completely disagree (0) Disagree (2)

Neutral (3)

Agree (4) Completely agree (5)

The course 4.3

1.3 the teaching activities has been motivating for my effort.

Completely disagree (0) Disagree (2)

Neutral (3)

Agree (4) Completely agree (5)

The course 3.9

1.4 throughout the course, I had the opportunity to get feedback on where I
stand academically in this course.

Completely disagree (0) Disagree (2)

Neutral (3)

Agree (4) Completely agree (5)

The course 3.7

1.5 the expectations to my effort in exercises, project work, etc. has been
clear.

Completely disagree (0) Disagree (2)

Neutral (3)

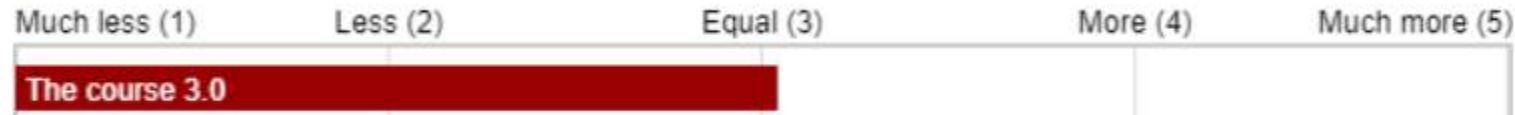
Agree (4) Completely agree (5)

The course 4.1

Evaluation:

2 Effort

2.1 5 ECTS point is set to a workload of 9 hours a week in the 13-week period (45 hours a week in the 3-week period) I think, the time I spent on this course is



Evaluation:

1 It is my opinion, that...

1.1 the teacher has given me a good grasp of the academic content of the course.

Completely disagree (Disagree (2)

Neutral (3)

Agree (4)

Completely agree (5)

Tea. 4.4

Tea. 4.3

Tea.

Richard Martin Lusby

43 out of 139 has
answered (31%)

Tea.

Stefan Røpke

43 out of 138 has
answered (31%)

1.2 the teacher has communicated the academic content in a clear way.

Completely disagree (Disagree (2)

Neutral (3)

Agree (4)

Completely agree (5)

Tea. 4.3

Tea. 4.1

1.3 the way the teacher organized the learning activities was helpful for my learning.

Completely disagree (Disagree (2)

Neutral (3)

Agree (4)

Completely agree (5)

Tea. 4.3

Tea. 4.2

What was good

- Good structure.
- Good exercises
- Exercises related to "real" problems
- Good teaching assistants
- Good teachers

What was not so good

- Sometimes the lectures are too light
- Sometimes too much repetition
- Some examples are only on the blackboard
- Some did not learn anything from the lectures but only from the teaching assistants
- Some lectures are rushed (too many subjects)

Suggestions for improvement

- More examples on Julia code.
- More reading material for some of the topics.
- Consider teaching in Danish.
- More teaching assistants during project work.
- Projects should count in grading.

- **3 Qualitative comments**

- **3.1 General constructive feedback about this course**

The course was structured very good with a nice introduction to the world of OR in the beginning, followed by going more into depth with integer programming and BIP. The exercises were good in variance and in learning objective. Both teachers were good at handing out of their knowledge, and so was the TAs.

- Pensum i anden halvdel af kurset har ikke altid stemt overens med hvad der gennemgås til forelæsningen. Bl.a. blev der arbejdet med heltalsprogrammering i 4 uger, hvor man skulle læse det samme i bogen
- Det er godt at mange af øvelserne har baggrund i virkeligheden, da det ellers ville blive meget abstrakt. Flere eksempler på opgaver løbende ville være godt
- The last part of the Course is a bit too Light. A little too much repetition.
- Mange opgaver efter hver forelæsning, men efter efterårsferien når vi har fået ny underviser, skriver han ikke sine slides grundigt nok til at man kan forstå stoffet, fx dual og det vi har lært efterfølge med slides på kun 25 slides uden eksempler, alle eksemplerne hertil er kun på tavlen og ikke slidsne, dette er virkelig dårligt
- Ærgerligt at rapporterne ikke betyder noget for den samlede karakter
- Kurset er generelt godt Dog er niveauet på forelæsninger ofte lavt, ved gennemgang af gamle opgaver og eksempler som bliver gennemgået lidt for grundigt og langsomt

- **3 Qualitative comments**

- **3.1 General constructive feedback about this course**

Hjælpelæreren er den eneste der har lært mig noget. Jeg fik ikke noget ud af forelæsningerne, det var meget svært at følge med og stoffet virkede meget anderledes end de reelle opgaver man undervejs blev stillet.

- Gode øvelsesopgaver. Gode hjælpelærer. Dejligt, det er muligt at skrive til underviserens og få svar. Det er dejligt, at vi får meget hjælp til Juliakoder i forelæsningen, så det kun er lidt vi skal programmere selv.
- More examples on Julia code. More examples or readings on the different topics, so we can search for help ourselves during exercises and projects. Super friendly Professors.
- Jeg kan godt lide kurset, men jeg vil forstå det meget bedre hvis det var på dansk. Dejligt med alle disse slides, som hjælper en til at kunne lave projekterne og opgaverne. Jeg kan godt lide, at der er både forelæsning og studiehjælp til selv at lave opgaver. Jeg forstår det bedst når vi laver opgaverne og får hjælp.
- Very good course in general
- Jeg synes, det kunne være rart også at få samme form for feedback på fremlæggelser som vi får på opgaverne
- Hvis det kan lade sig gøre så have ekstra hjælpelærer, når der arbejdes med projekterne. De hjælpelærer der var til stede var ret pressede, og vi ventede nogengange i 20 min for at få hjælp.

Exam

- 4 hour written exam
- All aids allowed (you can bring your computer, calculator, tablet, etc.)
- No internet
- Around 20 questions.
 - Most are multiple choice
 - Some could be "text questions"
- For multiple choice questions we only look at "where you put the cross"
- Each multiple choice question counts for 5 points
 - Correct answer: 5 points
 - Wrong answer: 0 points
 - More than one choice in a multiple choice question: 0 points
- Text questions give more points (read first page of the exam)
- Questions have varying difficulty. Start with the easiest ones!

Content of exam

- All material we have covered (book, exercises, projects and slides)
- Exam text is normally long. Don't let that frighten you!
 - We need a lot of space to list choices
 - We try to avoid long "chains" of exercises
- Questions are a mix of
 - "Standard" exercises (you will have seen something similar before, especially if you looked at the old exams)
 - Some more "creative" exercises that will have you use the material in a new way.

Tips & tricks

- Tasks are not sorted by difficulty
 - Browse through all the tasks and start with the easy ones.
- If you get stuck in a task, put it aside and go on with the next one
- Remember that in the multiple choice question you may as well select an answer even if you are not sure – it's stupid to leave a multiple choice question unanswered!
- You are allowed to use common sense and eliminate some answers in multiple choice
- If you can see a “short-cut” for solving a multiple-choice question it's fine to do so.

Grading



As a rule of thumb we use the table below to decide the grade, but we (Stefan+Richard+external examinor) can choose to change the numbers in the table.

Exam score	Grade
]90;100]	12
]80;90]	10
]65;80]	7
]55;65]	4
[50;55]	2
[0;50[0 or -3

Question hour?

Questions?

What comes after intro to OR?

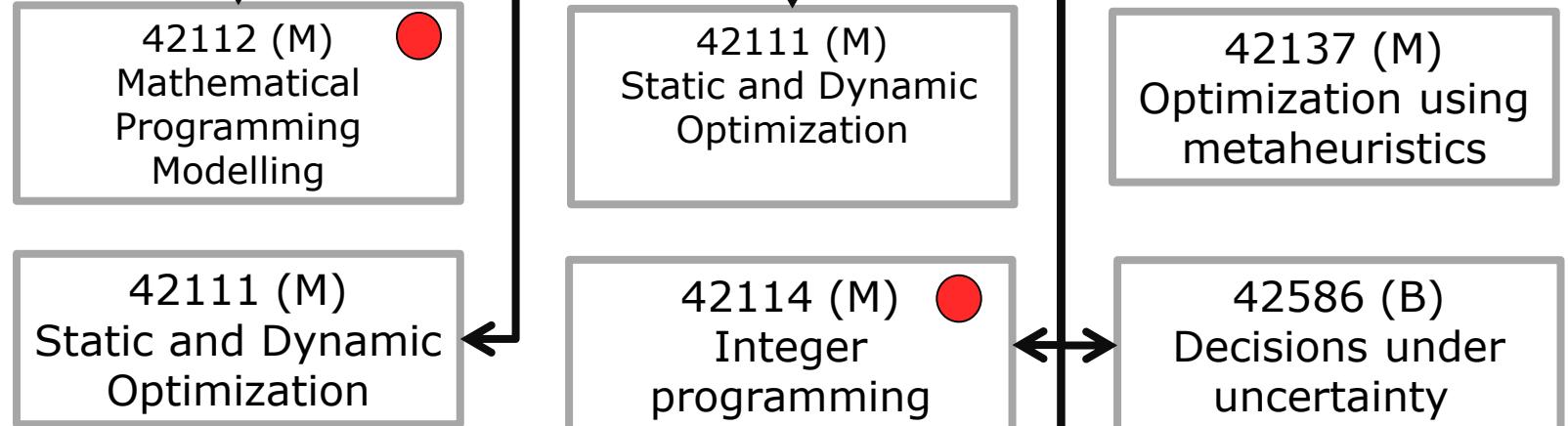
Intro

Solid lines: Mandatory prerequisites
Dashed lines: recommended prerequisites
(M)=master, (B)=bachelor

OR courses

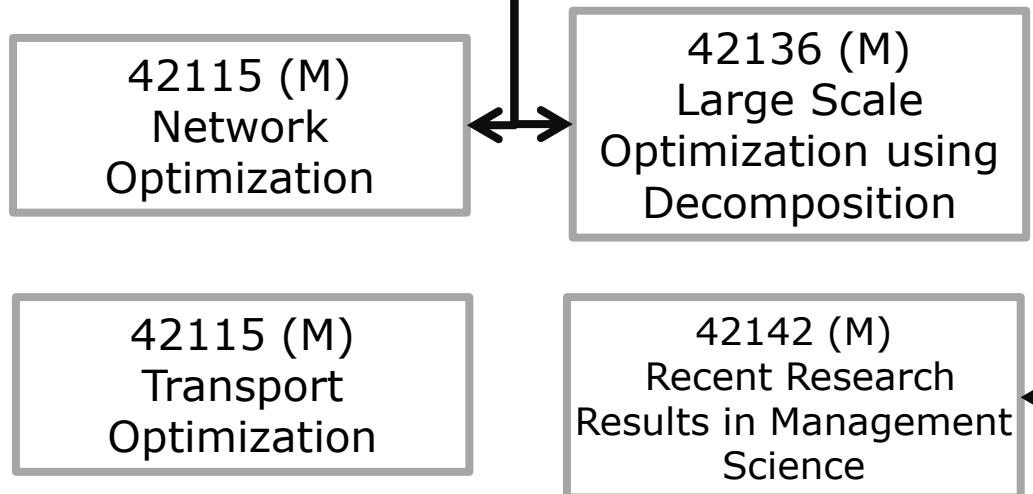


Follow-up



Advanced:

● = minimum requirement to do a OR MSc Thesis (just my rule of thumb, not official)



Questions?

Plan for the rest of today

- Let's go to exercise rooms and work on exam from fall 2018 (on DTU inside)
- If exercise room are occupied we will have to head back to this room
- Richard and I will be in two of the exercise rooms. We can mainly answer questions in our own material.
- From 15:00 the teaching assistants will join us.

Julia

Stefan Røpke

A vibrant collage of mathematical symbols and numbers. It includes a blue integral from a to b, a red summation symbol with an exclamation mark, a purple infinity symbol, a green theta symbol, a yellow epsilon symbol, a pink sigma symbol with a double arrow, and a red delta symbol with a plus sign. The numbers 17 and 2.7182818284 are also present.

What is Julia?

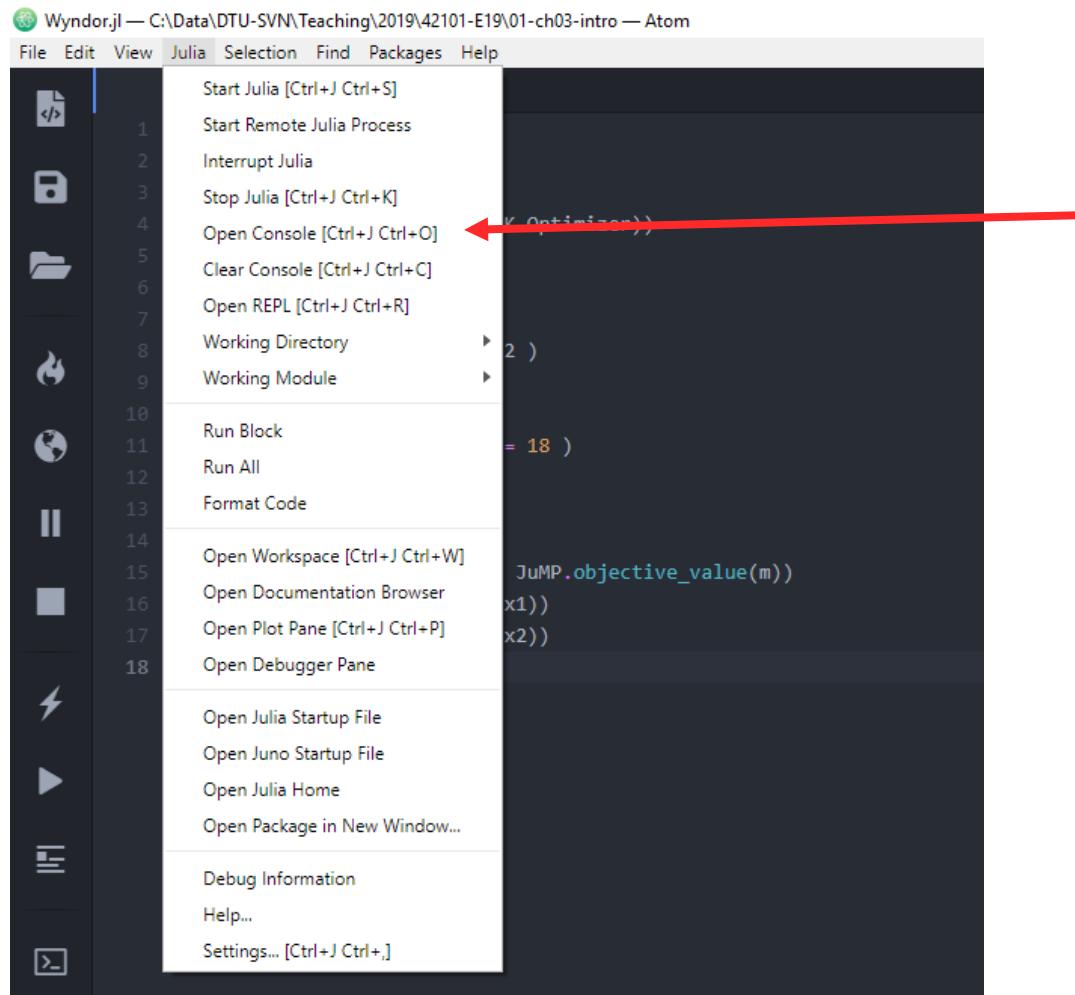
- A programming language like Python, C++, Java, etc.
- Julia is convenient for working with operation research.
 - It is, for example, easy to solve linear programming problems using Julia
- We use it in several other OR courses
- Free and open source so you can easily use it in a company.

How to use Julia?

- Two options. Either
 - Install Julia on your computer.
 - Install Julia and Juno following the instructions on
 - <http://docs.junolab.org/latest/man/installation/>
- or
 - Jupyter (All you need is a browser)
 - Enter: <https://class42101.man.dtu.dk> in your browser

Julia and Juno (Julia on your own PC)

- Inside Juno select “Open Console” in the Julia menu in order to type in some Julia commands



Julia and Juno

- In console window type

```
julia> using Pkg

julia> Pkg.add(["GLPK", "JuMP"])
Resolving package versions...
  Updating `C:\Users\Stefan\.julia\environments\v1.1\Project.toml`
[no changes]
  Updating `C:\Users\Stefan\.julia\environments\v1.1\Manifest.toml`
[no changes]

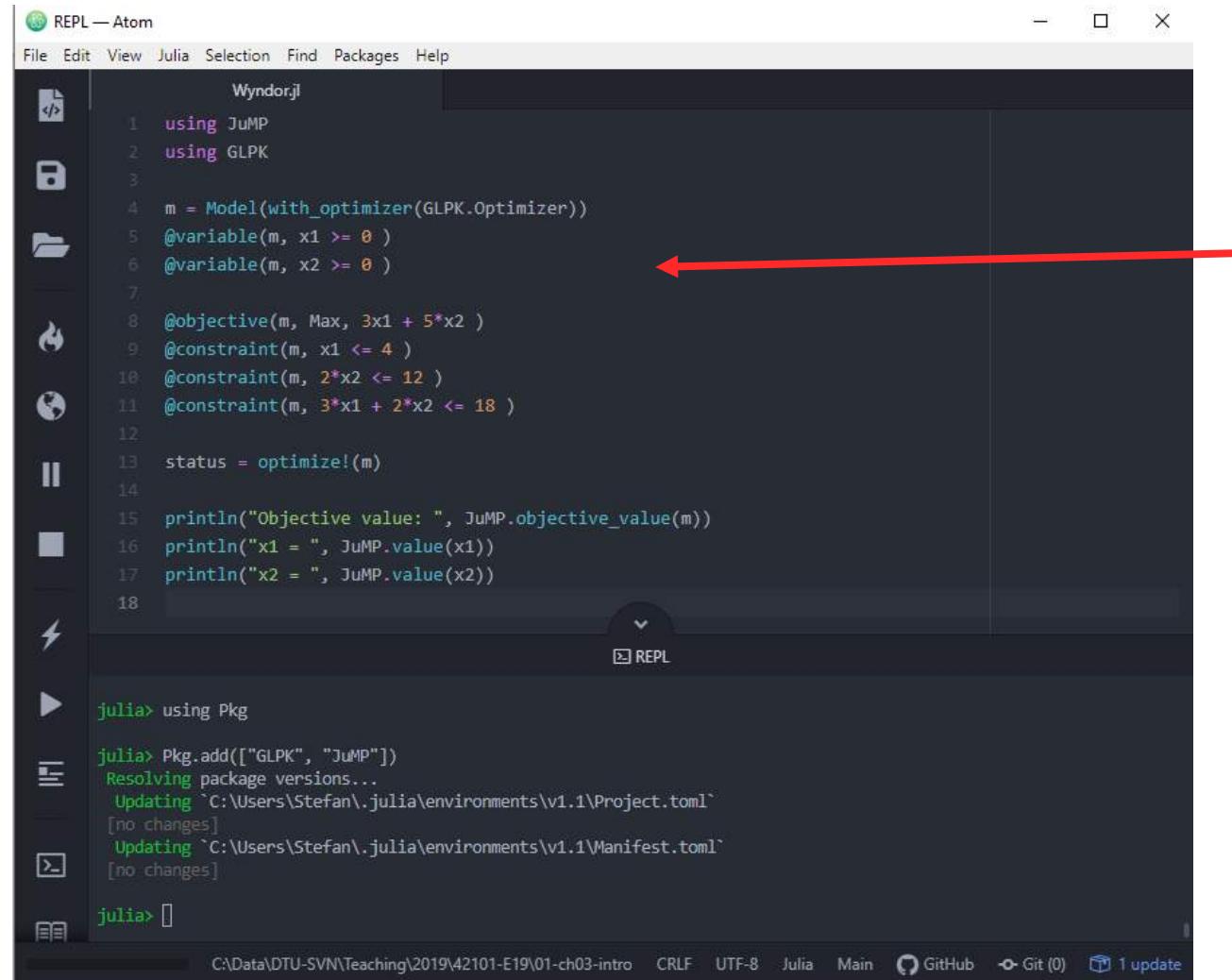
julia> █
```

(output on your computer is going to look different, and it make take some time for the commands to finish)

- This installs the packages that allows Julia to solve linear programs

Julia and Juno

- Now we can type in the Wyndor model and solve it



```
REPL — Atom
File Edit View Julia Selection Find Packages Help
Wyndor.jl
1 using JuMP
2 using GLPK
3
4 m = Model(with_optimizer(GLPK.Optimizer))
5 @variable(m, x1 >= 0 )
6 @variable(m, x2 >= 0 )
7
8 @objective(m, Max, 3*x1 + 5*x2 )
9 @constraint(m, x1 <= 4 )
10 @constraint(m, 2*x2 <= 12 )
11 @constraint(m, 3*x1 + 2*x2 <= 18 )
12
13 status = optimize!(m)
14
15 println("Objective value: ", JuMP.objective_value(m))
16 println("x1 = ", JuMP.value(x1))
17 println("x2 = ", JuMP.value(x2))
18
julia> using Pkg
julia> Pkg.add(["GLPK", "JuMP"])
Resolving package versions...
Updating `C:\Users\Stefan\.julia\environments\v1.1\Project.toml`
[no changes]
Updating `C:\Users\Stefan\.julia\environments\v1.1\Manifest.toml`
[no changes]
julia> 
```

C:\Data\DTU-SVN\Teaching\2019\42101-E19\01-ch03-intro CRLF UTF-8 Julia Main GitHub Git (0) 1 update

Open a new file, type in the wyndor model. Save to a file that ends with ".jl"

Julia and Juno

- Now we can type in the Wyndor model and solve it

The screenshot shows the Atom code editor with a JuMP optimization script named `Wyndor.jl`. The script defines a linear programming model with two variables (x_1 and x_2), constraints, and an objective function. It then solves the model and prints the results.

```
Wyndor.jl
1 using JuMP
2 using GLPK
3
4 m = Model(with_optimizer(GLPK.Optimizer))
5 @variable(m, x1 >= 0 )
6 @variable(m, x2 >= 0 )
7
8 @objective(m, Max, 3*x1 + 5*x2 )
9 @constraint(m, x1 <= 4 )
10 @constraint(m, 2*x2 <= 12 )
11 @constraint(m, 3*x1 + 2*x2 <= 18 )
12
13 status = optimize!(m)
14
15 println("Objective value: ", JuMP.objective_value(m))
16 println("x1 = ", JuMP.value(x1))
17 println("x2 = ", JuMP.value(x2))
```

The output in the REPL shows the following:

- Updating `C:\Users\Stefan\.julia\environments\v1.1\Project.toml` [no changes]
- Updating `C:\Users\Stefan\.julia\environments\v1.1\Manifest.toml` [no changes]
- [Info: Recompiling stale cache file C:\Users\Stefan\.julia\compiled\v1.1\JuMP\DMXqY.ji for JuMP [4076af6c-e467-56a e-b986-b466b2749572]
- Objective value: 36.0
- x1 = 2.0
- x2 = 6.0
- julia>

Make sure that
the cursor is in
the window with
the model.
Press the play
button.

Output will appear in the console window

Wyndor in Julia

```
using JuMP
using GLPK

m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 )

@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 )

status = optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("x1 = ", JuMP.value(x1))
println("x2 = ", JuMP.value(x2))
```

Tells Julia to include definitions and functions from the two packages we downloaded and creates a new model.

Just copy-paste these three lines when you make a new model

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

```
using JuMP
using GLPK

m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 ) }
```

Here we define the variables for the model

```
@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 )

status = optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("x1 = ", JuMP.value(x1))
println("x2 = ", JuMP.value(x2))
```

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

```
using JuMP
using GLPK

m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 )

@objective(m, Max, 3*x1 + 5*x2 ) }
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 )

status = optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("x1 = ", JuMP.value(x1))
println("x2 = ", JuMP.value(x2))
```

Here we define the objective function.

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

```
using JuMP
using GLPK

m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 )

@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 ) }

status = optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("x1 = ", JuMP.value(x1))
println("x2 = ", JuMP.value(x2))
```

Here we define constraints
Equality constraints are written ==
Greater-than-equal are written >=

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

```
using JuMP
using GLPK

m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 )

@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 )

status = optimize!(m)
```

We ask Julia / GLPK to solve the model

```
println("Objective value: ", JuMP.objective_value(m))
println("x1 = ", JuMP.value(x1))
println("x2 = ", JuMP.value(x2))
```

Wyndor in Julia

```
using JuMP
using GLPK

m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 )

@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 )

status = optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("x1 = ", JuMP.value(x1))
println("x2 = ", JuMP.value(x2))
```

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

We print the optimal objective value and optimal solution.

Jupyter

- All you need is a browser
 - Can be a little sluggish
 - DTU's solution at: <https://class42101.man.dtu.dk> or

DTU's Jupyter server at: <https://class42101.man.dtu.dk>

A screenshot of a web browser window. The address bar shows a secure connection to "https://class42101.man.dtu.dk/hub/login". The title bar has tabs for "JupyterHub" and "Julia Computing". Below the address bar, the browser menu bar includes "Sikker", "Stefan", and other icons. The main content area displays a "Sign in" form with fields for "Username" and "Password", and a "Sign In" button. A large green arrow points from a callout box on the right towards the "Sign in" form. The callout box contains the text: "Log in with your DTU username and password".

Sign in

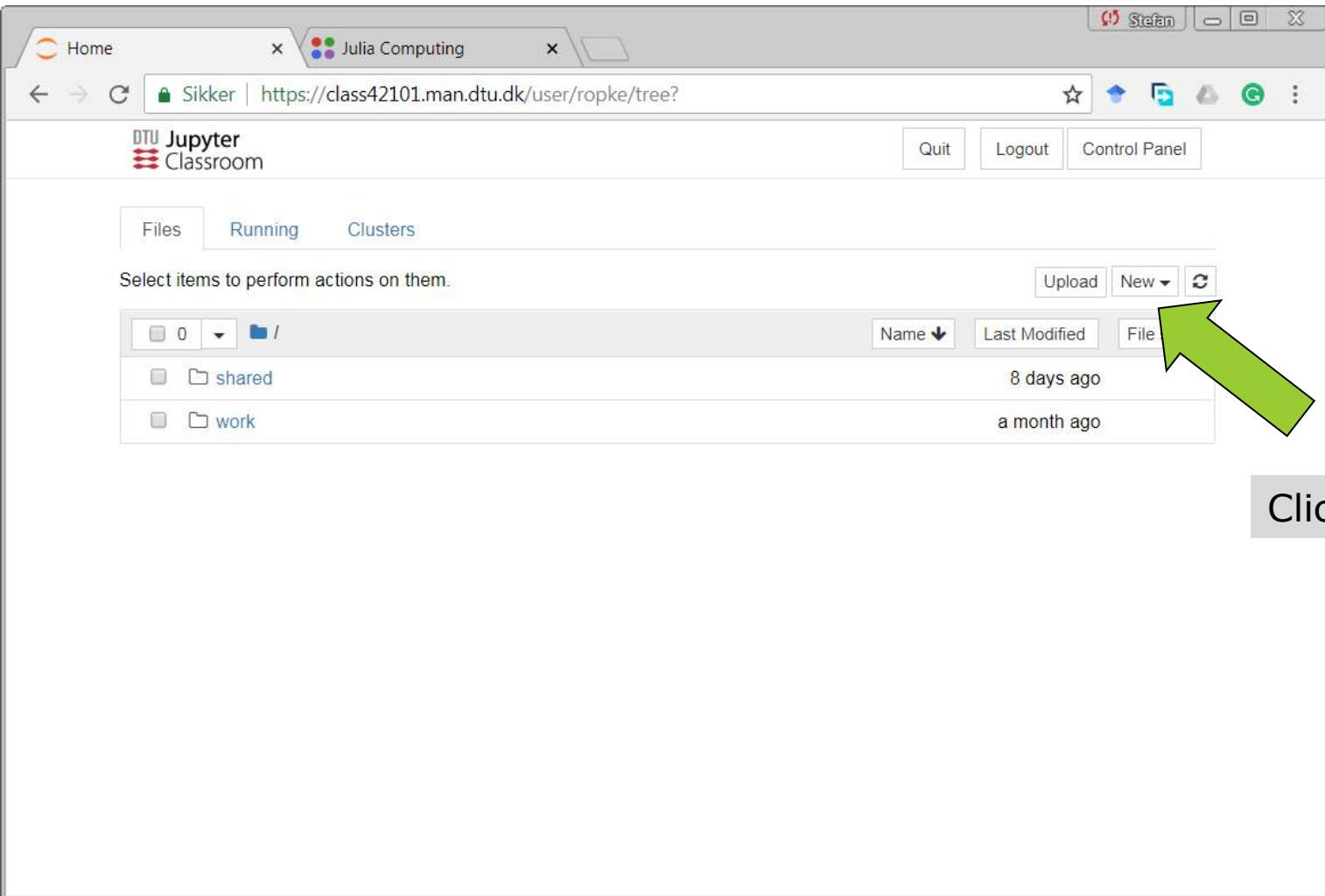
Username:

Password:

Sign In

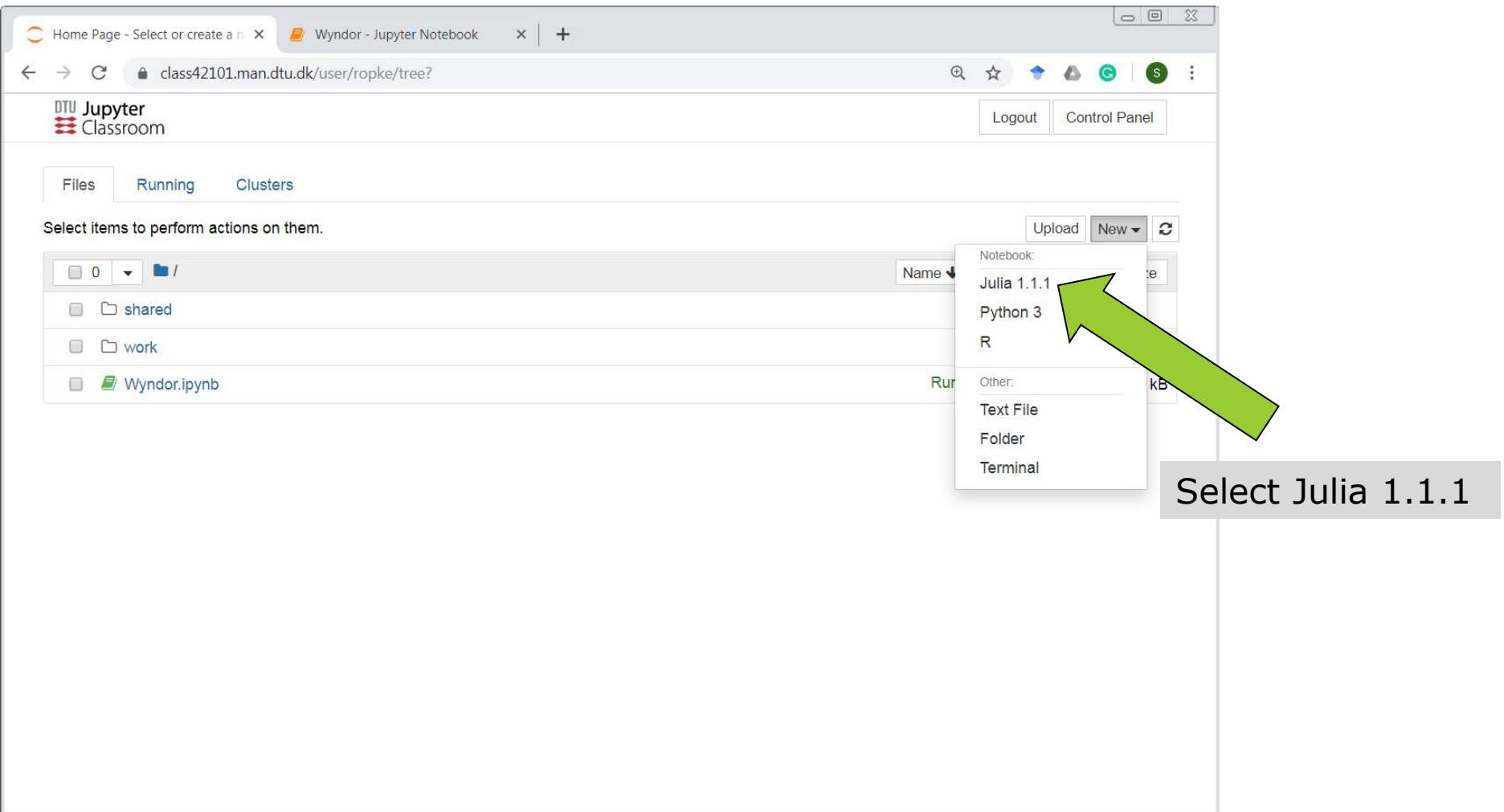
Log in with your DTU username and password

Jupyter



The screenshot shows the DTU Jupyter Classroom interface. At the top, there is a browser header with tabs for "Home" and "Julia Computing". The URL is https://class42101.man.dtu.dk/user/ropke/tree?. The title bar displays "Sikker" and the user "Stefan". Below the header, the interface includes a "DTU Jupyter Classroom" logo, "Quit", "Logout", and "Control Panel" buttons. A navigation bar at the top has tabs for "Files", "Running", and "Clusters", with "Files" being the active tab. A message "Select items to perform actions on them." is displayed above the file list. The file list shows two entries: "shared" (modified 8 days ago) and "work" (modified a month ago). To the right of the file list are buttons for "Upload", "New", and "File". A large green arrow points from a grey box containing the text "Click new" towards the "New" button. The bottom of the interface has a light grey footer.

Jupyter



Home Page - Select or create a new notebook Wyndor - Jupyter Notebook

class42101.man.dtu.dk/user/ropke/tree?

DTU Jupyter Classroom

Logout Control Panel

Files Running Clusters

Select items to perform actions on them.

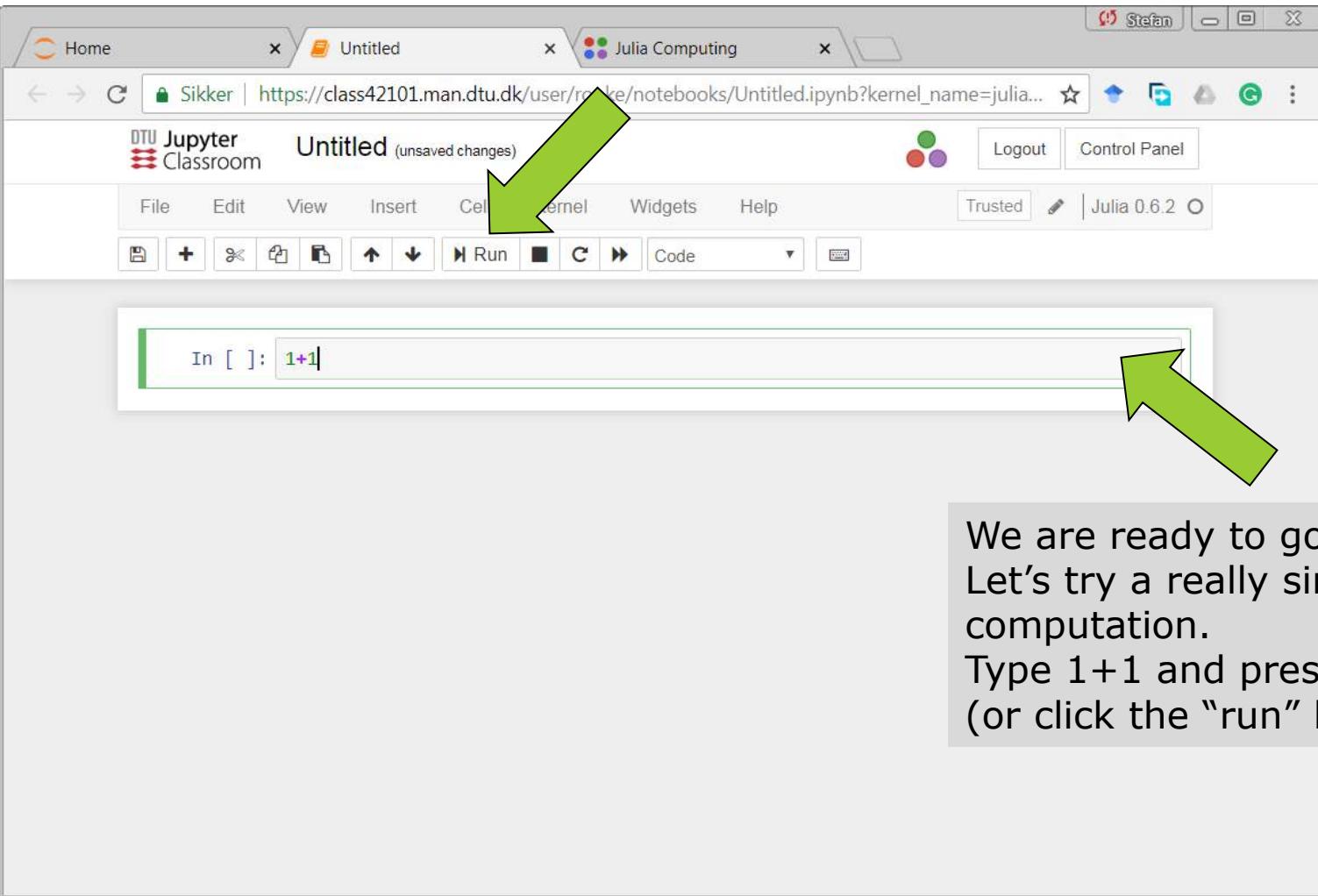
0 / shared work Wyndor.ipynb

Name: Upload New

Notebook: Julia 1.1.1 Python 3 R Other: Text File Folder Terminal

Select Julia 1.1.1

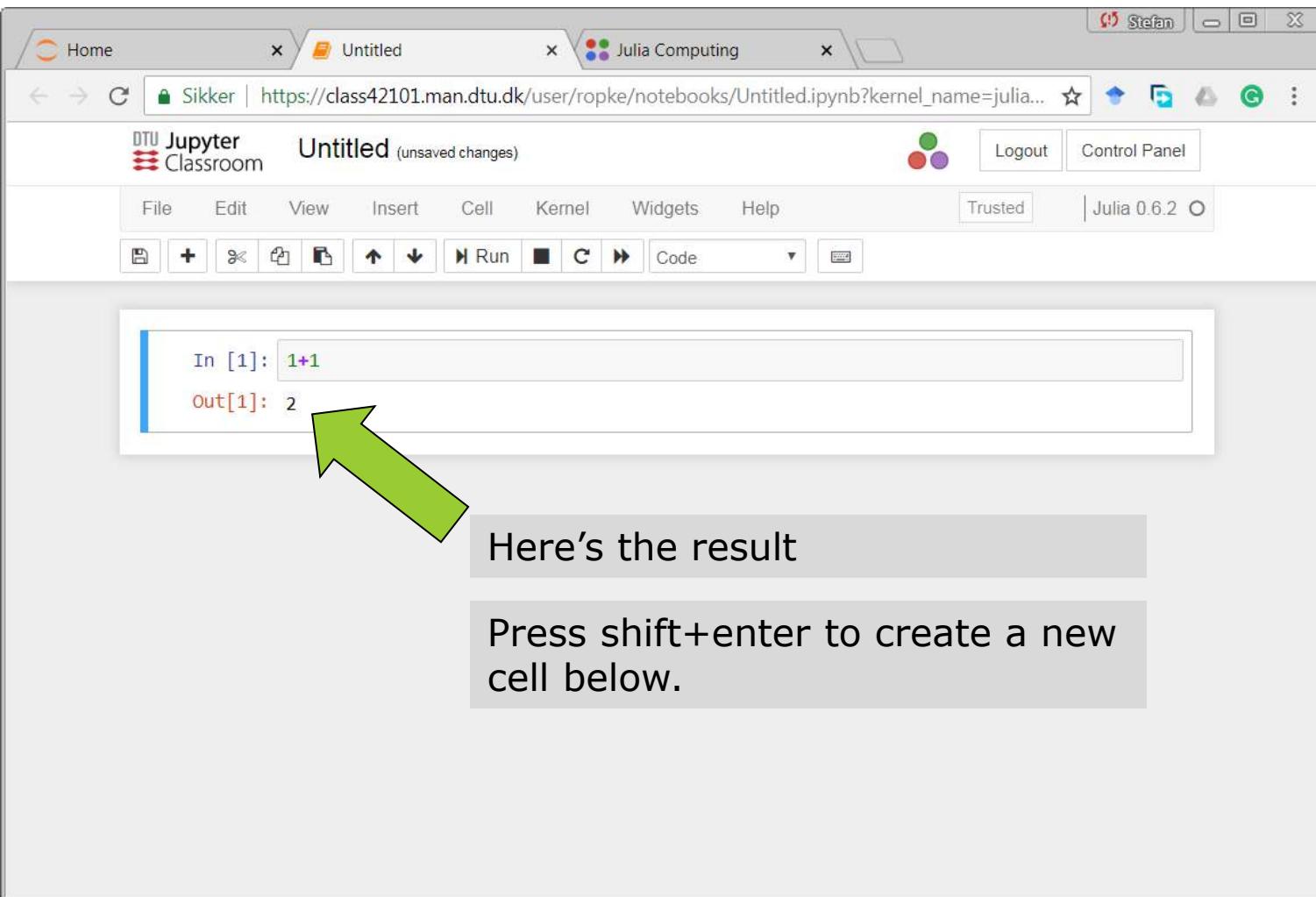
Jupyter



A screenshot of a Jupyter Notebook interface. The top bar shows tabs for "Home", "Untitled", and "Julia Computing". The main area shows a notebook cell with the input "In []: 1+1". A large green arrow points from the right side of the slide towards this cell. To the right of the cell, a text box contains the following text:

We are ready to go.
Let's try a really simple computation.
Type $1+1$ and press CTRL+enter
(or click the "run" button)

Jupyter



The screenshot shows a Jupyter Notebook interface running in a browser window. The title bar indicates the user is Stefan. The main content area displays a code cell with the following content:

```
In [1]: 1+1
Out[1]: 2
```

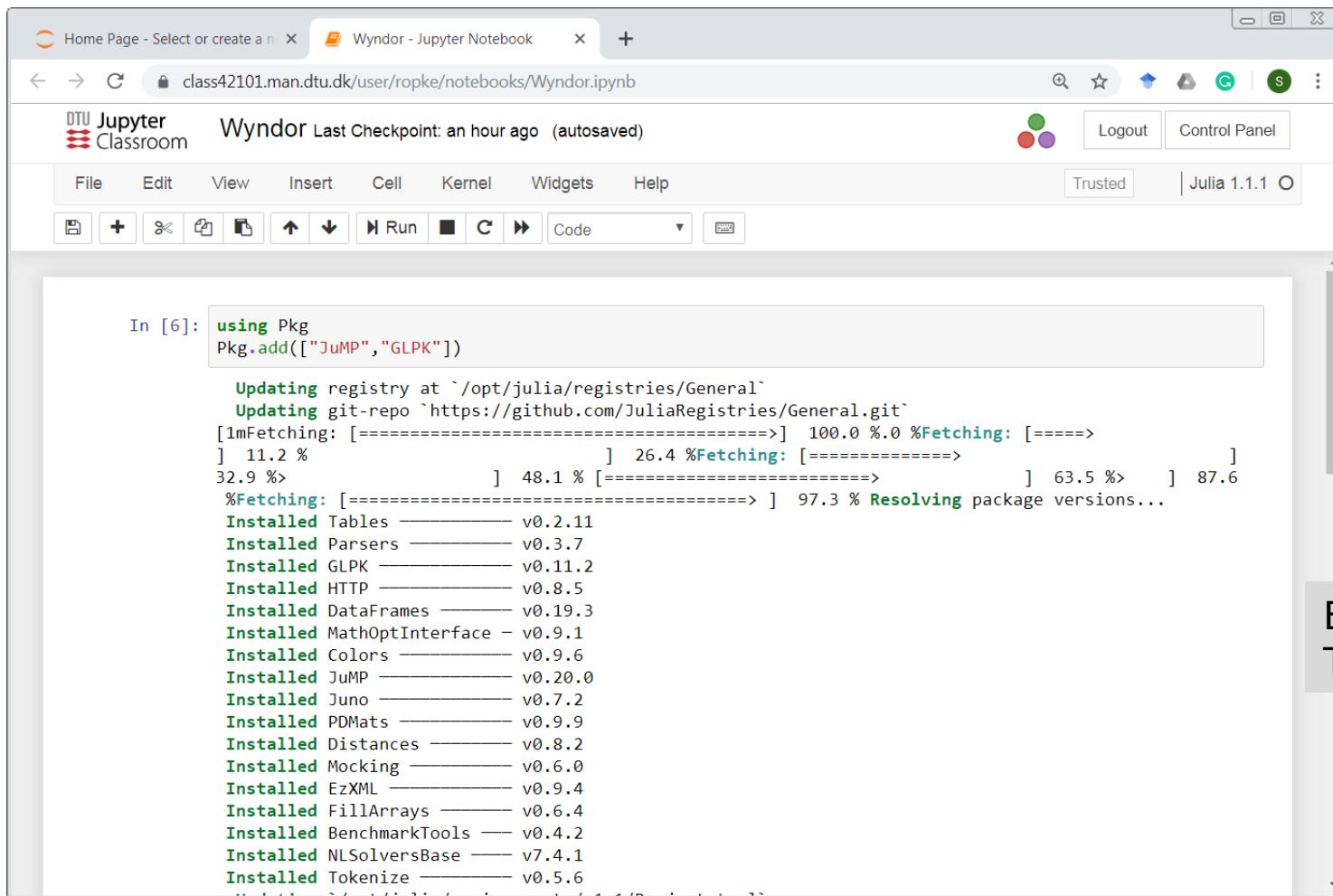
A large green arrow points from the text "Here's the result" below the code cell towards the output "2".

Below the code cell, there is a text box containing the following instructions:

Here's the result

Press shift+enter to create a new cell below.

Jupyter: install JuMP and GLPK packages



The screenshot shows a Jupyter Notebook interface running on a DTU Classroom server. The notebook title is "Wyndor - Jupyter Notebook". The code cell In [6] contains the command `using Pkg` followed by `Pkg.add(["JuMP", "GLPK"])`. The output shows the package registry being updated from GitHub and the installation of various packages, including JuMP and GLPK. A progress bar indicates the completion of the process.

```
In [6]: using Pkg
Pkg.add(["JuMP", "GLPK"])

Updating registry at `/opt/julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
[1mFetching: [=====>] 100.0 %.0 %Fetching: [=====>
] 11.2 % ] 26.4 %Fetching: [=====>] 63.5 %> ] 87.6
32.9 %> ] 48.1 % [=====> ] 97.3 % Resolving package versions...
Installed Tables v0.2.11
Installed Parsers v0.3.7
Installed GLPK v0.11.2
Installed HTTP v0.8.5
Installed DataFrames v0.19.3
Installed MathOptInterface v0.9.1
Installed Colors v0.9.6
Installed JuMP v0.20.0
Installed Juno v0.7.2
Installed PDMats v0.9.9
Installed Distances v0.8.2
Installed Mocking v0.6.0
Installed EzXML v0.9.4
Installed FillArrays v0.6.4
Installed BenchmarkTools v0.4.2
Installed NLSolversBase v7.4.1
Installed Tokenize v0.5.6
... . . . . .
```

Be patient!
This takes a while!

Jupyter

Home Page - Select or create a new notebook Wyndor - Jupyter Notebook + New

Back Forward Search Star Upload Download Cell Kernel Widgets Help

DTU Jupyter Classroom Wyndor Last Checkpoint: an hour ago (autosaved) Logout Control Panel

In [8]:

```
using JuMP
using GLPK

m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 )

@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 )

status = optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("x1 = ", JuMP.value(x1))
println("x2 = ", JuMP.value(x2))
```

Objective value: 36.0
x1 = 2.0
x2 = 6.0

In []:

Let's try to solve the wyndor model using Julia. Compare to:

$$\max Z = 3x_1 + 5x_2$$

subject to

$$x_1 \leq 4$$

$$2x_2 \leq 12$$

$$3x_1 + 2x_2 \leq 18$$

and

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

Home Page - Select or create a new notebook Wyndor - Jupyter Notebook + New

[←](#) [→](#) [C](#) [Logout](#) [Control Panel](#)

File Edit View Insert Cell Kernel Widgets Help

In [8]:

```
using JuMP
using GLPK

m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 )

@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 )

status = optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("x1 = ", JuMP.value(x1))
println("x2 = ", JuMP.value(x2))
```

Objective value: 36.0
x1 = 2.0
x2 = 6.0

In []:

1) We solve the model (run the program) by pressing CTRL+enter or clicking “run”

2) Output appears here.

Very slow the first time we solve an LP model.

Be patient!

Home Page - Select or create a new notebook Wyndor - Jupyter Notebook + New

[←](#) [→](#) [C](#) [Logout](#) [Control Panel](#)

File Edit View Insert Cell Kernel Widgets Help

In [8]:

```
using JuMP
using GLPK

m = Model(with_optimizer(GLPK.Optimizer))
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 )

@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 )

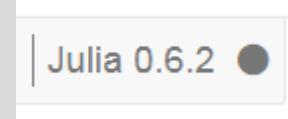
status = optimize!(m)

println("Objective value: ", JuMP.objective_value(m))
println("x1 = ", JuMP.value(x1))
println("x2 = ", JuMP.value(x2))
```

Objective value: 36.0
x1 = 2.0
x2 = 6.0

In []:

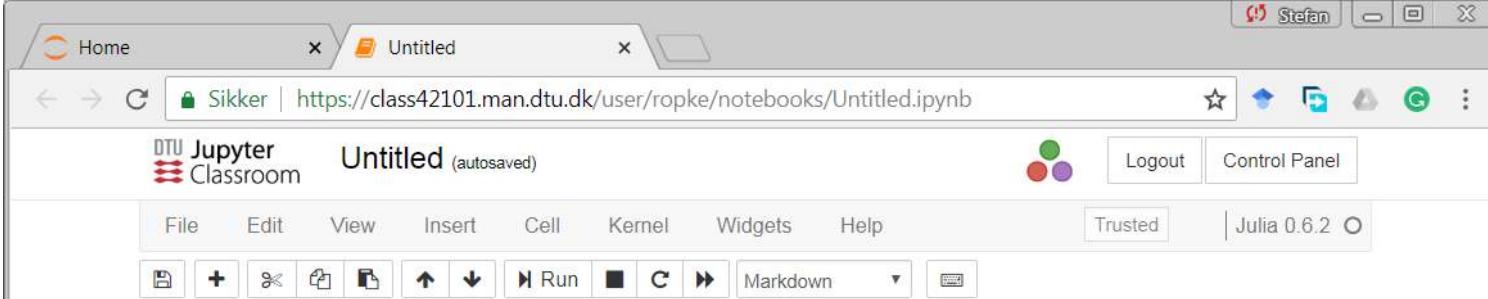
The first time you run a model it can take a while (up to 30 seconds?) At the top right you can check if Julia is busy:

Working: 

Ready: 

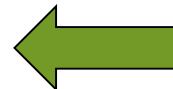
**In the lecture we skip the following slides
You can look at them yourself**





A screenshot of a Jupyter Notebook interface. At the top, there are tabs for "Home" and "Untitled". The address bar shows "Sikker | https://class42101.man.dtu.dk/user/ropke/notebooks/Untitled.ipynb". On the right, there are icons for "Logout" and "Control Panel". Below the address bar is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are buttons for "Trusted" and "Julia 0.6.2". The main area contains two code cells. The first cell, "In [2]:", contains the expression $1+1$ and "Out[2]: 2". The second cell, which has a blue border, contains text about solving a linear programming model:

Below we are solving the following model:
max $Z = 3x_1 + 5x_2$
subject to
 $x_1 \leq 4$
 $2x_2 \leq 12$
 $3x_1 + 2x_2 \leq 18$
and
 $x_1 \geq 0, x_2 \geq 0$



You can enter text in between computation cells (could be used to document what you are doing). Math formulas can be entered using Latex.

You do so by switching from "code" to "Markdown" in the toolbar

In [2]: `1+1`

Out[2]: 2

Below we are solving the following model:
\$\$\max Z=3x_1+5x_2\$\$
subject to
\$\begin{aligned} & x_1 + 2x_2 \leq 4 \\ & 3x_1 + 2x_2 \leq 12 \\ & x_1, x_2 \geq 0 \end{aligned}\$

This is what we enter in order to get the output from the previous slide.

See <https://tinyurl.com/y8oaz7go> for more information about “markdown” cells.

Sikker | https://class42101.man.dtu.dk/user/ropke/notebooks/Untitled.ipynb

DTU Jupyter Classroom Untitled (unsaved changes) Logout Control Panel

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 0.6.2

New Notebook ▾ Open... Make a Copy... Save as... Rename... Save and Checkpoint Revert to Checkpoint ▾ Print Preview Download as ▾ Trusted Notebook Close and Halt

Give your notebook a nice name using “rename” from the file menu.

I named my notebook “First notebook”

using GLPKMathProgInterface

```
m = Model(solver = GLPKSolverLP())
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 )

@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 )

status = solve(m)

println("Objective value: ", getobjectivevalue(m))
println("x1 = ",.getvalue(x1))
println("x2 = ",.getvalue(x2))
```

Objective value: 36.0
x1 = 2.0
x2 = 6.0

Sikker | https://class42101.man.dtu.dk/user/ropke/notebooks/First%20notebook.ipynb

DTU Jupyter Classroom First notebook (autosaved) Logout Control Panel

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 0.6.2

New Notebook Open... Make a Copy... Save as... Rename... Save and Checkpoint Revert to Checkpoint Print Preview Download as Trusted Notebook Close and Halt

You can ensure that your notebook is saved using "Save and Checkpoint" (Jupyter also autosaves regularly).

$x_1 \geq 0, x_2 \geq 0$

```
using GLPKMathProgInterface

m = Model(solver = GLPKSolverLP())
@variable(m, x1 >= 0 )
@variable(m, x2 >= 0 )

@objective(m, Max, 3*x1 + 5*x2 )
@constraint(m, x1 <= 4 )
@constraint(m, 2*x2 <= 12 )
@constraint(m, 3*x1 + 2*x2 <= 18 )

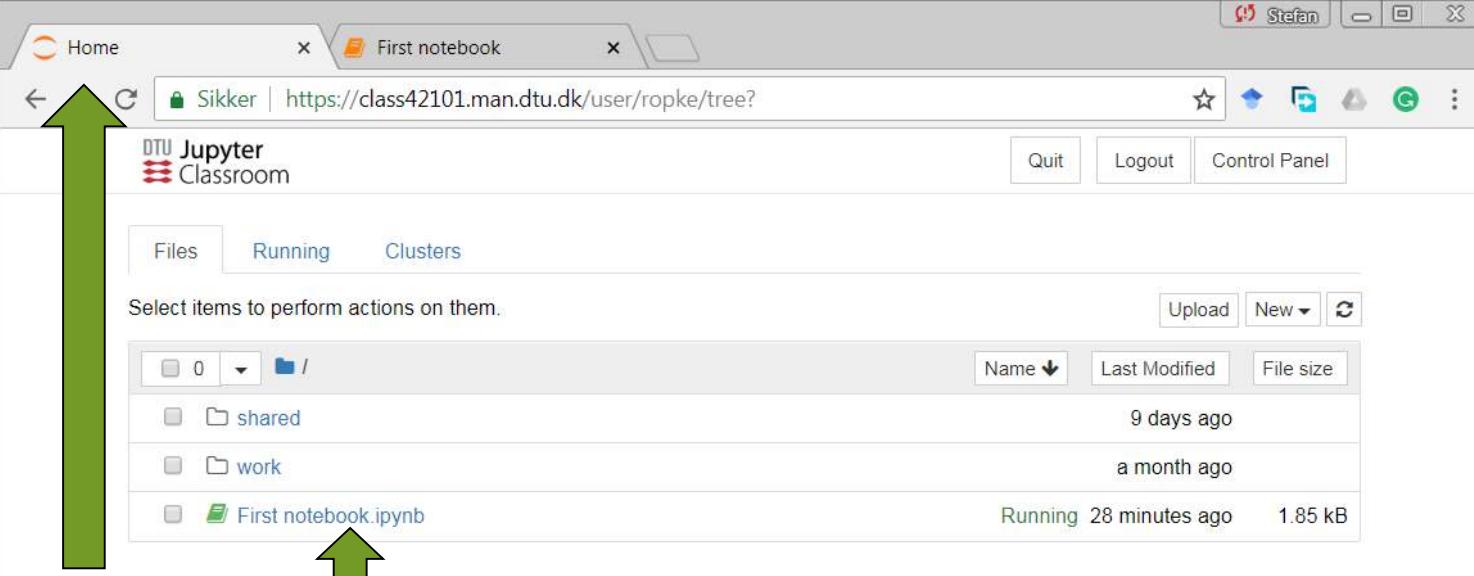
status = solve(m)

println("Objective value: ", getobjectivevalue(m))
println("x1 = ", getvalue(x1))
println("x2 = ", getvalue(x2))

Objective value: 36.0
x1 = 2.0
x2 = 6.0
```

28 Introduction to Operations Research

https://class42101.man.dtu.dk/user/ropke/notebooks/First notebook.ipynb#



Sikker | https://class42101.man.dtu.dk/user/ropke/tree?

DTU Jupyter Classroom

Files Running Clusters

Select items to perform actions on them.

Upload New

	Name	Last Modified	File size
0	/		
<input type="checkbox"/>	shared	9 days ago	
<input type="checkbox"/>	work	a month ago	
<input type="checkbox"/>	First notebook.ipynb	Running 28 minutes ago	1.85 kB

If you go back to your “home tab” you can see your notebook in the overview.

You can create as many notebooks as you like.

The notebooks are stored in “the cloud” so you can log in from another computer and your notebooks would still be there.

If you are doing important work you should make a backup of the notebook to your own computer.

Sikker | https://class42101.man.dtu.dk/user/ropke/notebooks/First%20notebook.ipynb

DTU Jupyter Classroom First notebook (autosaved) Logout Control Panel

File Edit View Insert Cell Kernel Widgets Help Trusted Julia 0.6.2

New Notebook ▾ Open...
Make a Copy... Save as... Rename... Save and Checkpoint Revert to Checkpoint ▾ Print Preview Download as ▾ Trusted Notebook Close and Halt

Notebook (.ipynb) Julia (.jl) HTML (.html) Reveal.js slides (.html) Markdown (.md) reST (.rst) LaTeX (.tex) PDF via LaTeX (.pdf)

using GLPK
m = Model(GLPK())
@variables x1 x2
@constraints m begin
 @constraint(m, 3*x1 + 5*x2)
 @constraint(m, x1 <= 4)
 @constraint(m, 2*x2 <= 12)
 @constraint(m, 3*x1 + 2*x2 <= 18)

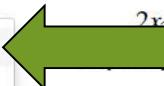
 status = solve(m)

 println("Objective value: ", getobjectivevalue(m))
 println("x1 = ", getvalue(x1))
 println("x2 = ", getvalue(x2))

 Objective value: 36.0
 x1 = 2.0
 x2 = 6.0

We are solving the following model:
$$\max Z = 3x_1 + 5x_2$$
$$x_1 \geq 0, x_2 \leq 12$$

You can save a copy of your notebook on your own computer by selecting "Download as" and then selecting "Notebook (.ipynb)"



Sikker | https://class42101.man.dtu.dk/user/ropke/tree?

DTU Jupyter Classroom

Files Running Clusters

Select items to perform actions on them.

Upload New

0 /

shared

work

First notebook.ipynb

Name Last Modified File size

9 days ago 0

a month ago 0

Running 36 minutes ago 1.85 kB

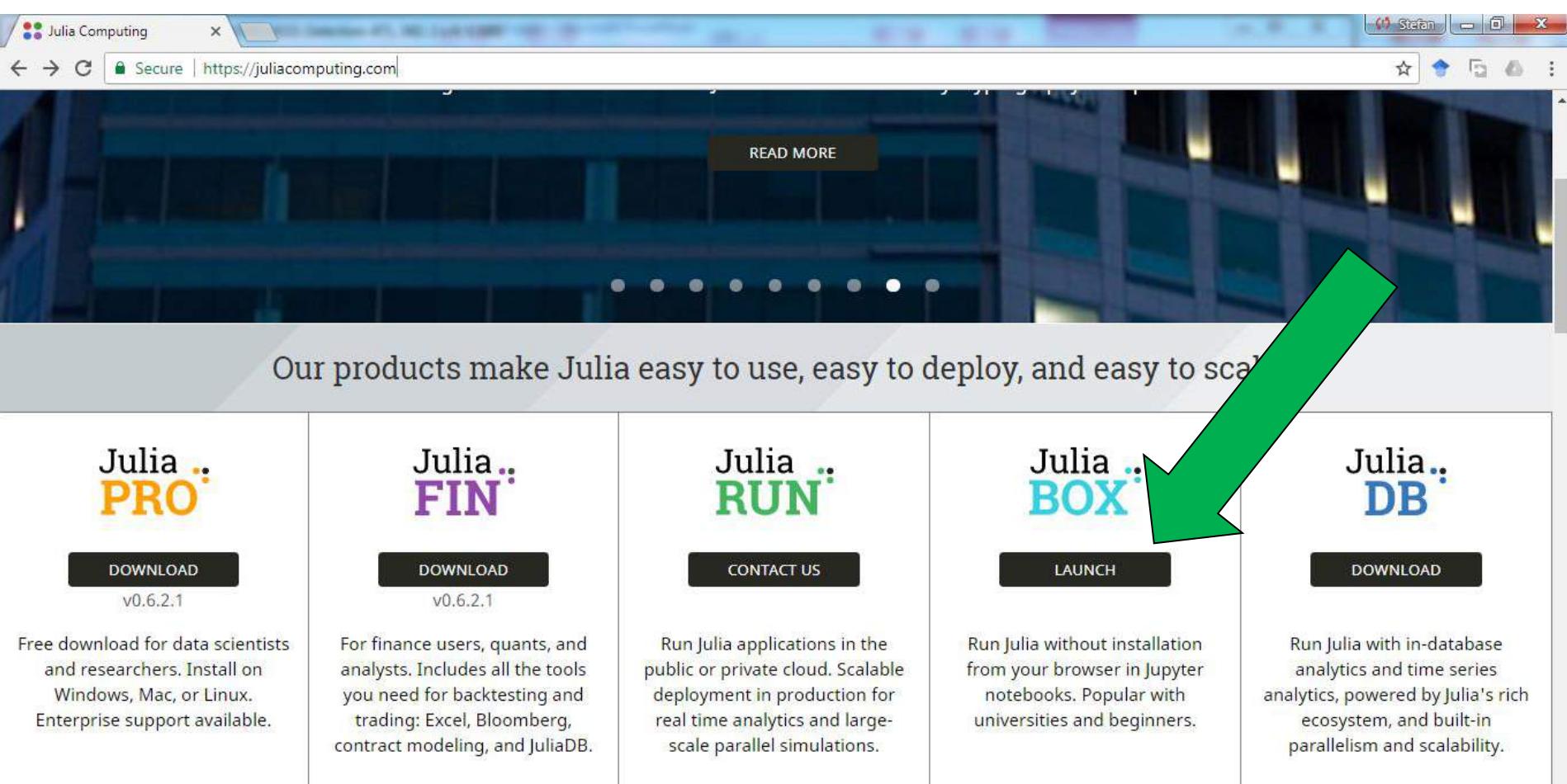


You can read in a notebook from your computer using "upload".

You can also use the "download" / "upload" functionality to share notebooks with your friends.

JuliaBox - <https://juliacomputing.com/>

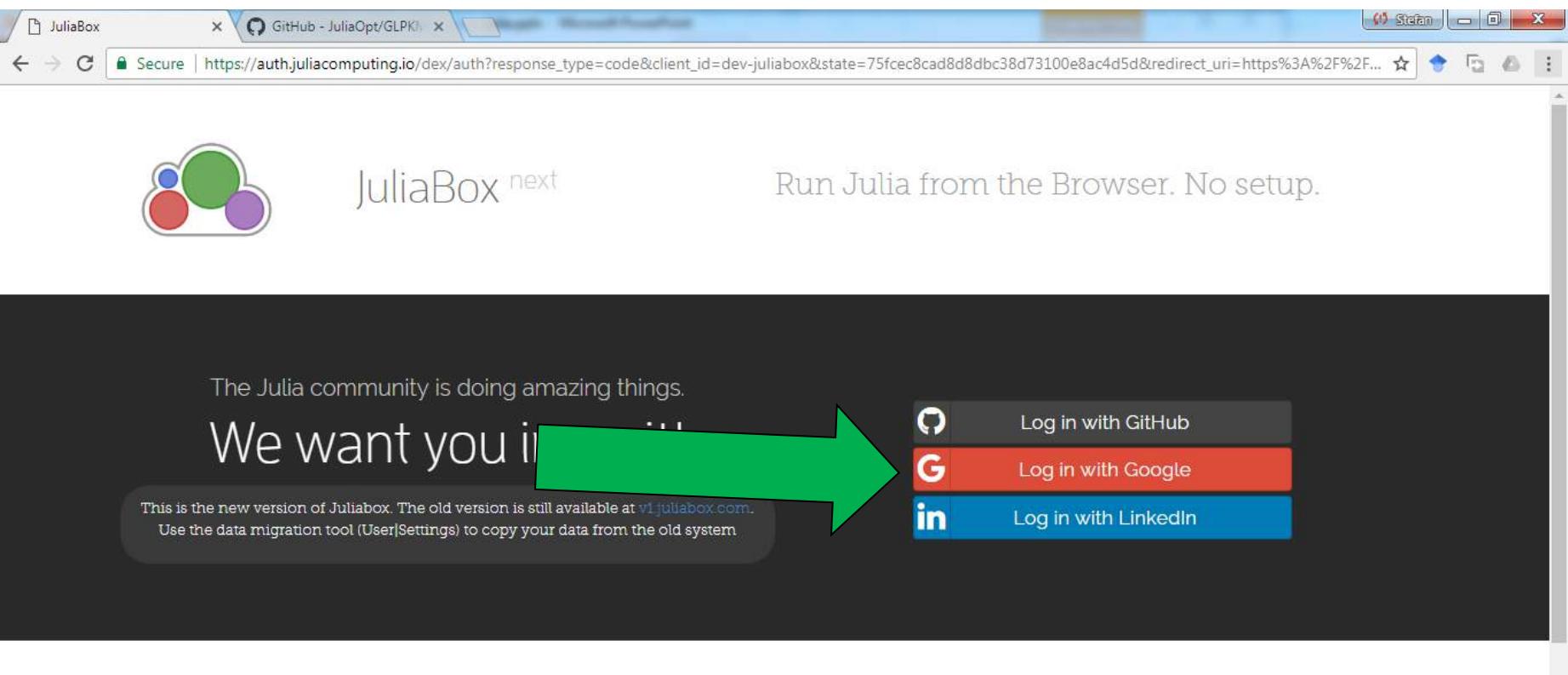
- It's basically the same as Jupyter on the DTU servers
- However they seem to use an earlier version of JuMP, so our examples may not work directly there.



The screenshot shows the Julia Computing website with a banner image of a modern building at night. Below the banner, a headline reads "Our products make Julia easy to use, easy to deploy, and easy to scale".

Product	Description	Action Buttons
Julia PRO	Free download for data scientists and researchers. Install on Windows, Mac, or Linux. Enterprise support available.	DOWNLOAD v0.6.2.1
Julia FIN	For finance users, quants, and analysts. Includes all the tools you need for backtesting and trading: Excel, Bloomberg, contract modeling, and JuliaDB.	DOWNLOAD v0.6.2.1
Julia RUN	Run Julia applications in the public or private cloud. Scalable deployment in production for real time analytics and large-scale parallel simulations.	CONTACT US
Julia BOX	Run Julia without installation from your browser in Jupyter notebooks. Popular with universities and beginners.	LAUNCH
Julia DB	Run Julia with in-database analytics and time series analytics, powered by Julia's rich ecosystem, and built-in parallelism and scalability.	DOWNLOAD

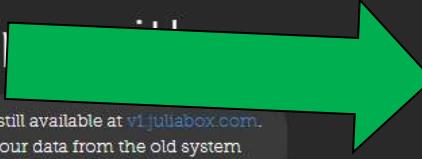
JuliaBox



The screenshot shows a web browser window with the URL https://auth.juliacomputing.io/dex/auth?response_type=code&client_id=dev-juliabox&state=75fcec8cad8d8dbc38d73100e8ac4d5d&redirect_uri=https%3A%2F%2F.... The page features a large green arrow pointing from the text "We want you in" towards the social login buttons.

 JuliaBox next

Run Julia from the Browser. No setup.

The Julia community is doing amazing things.
We want you in 

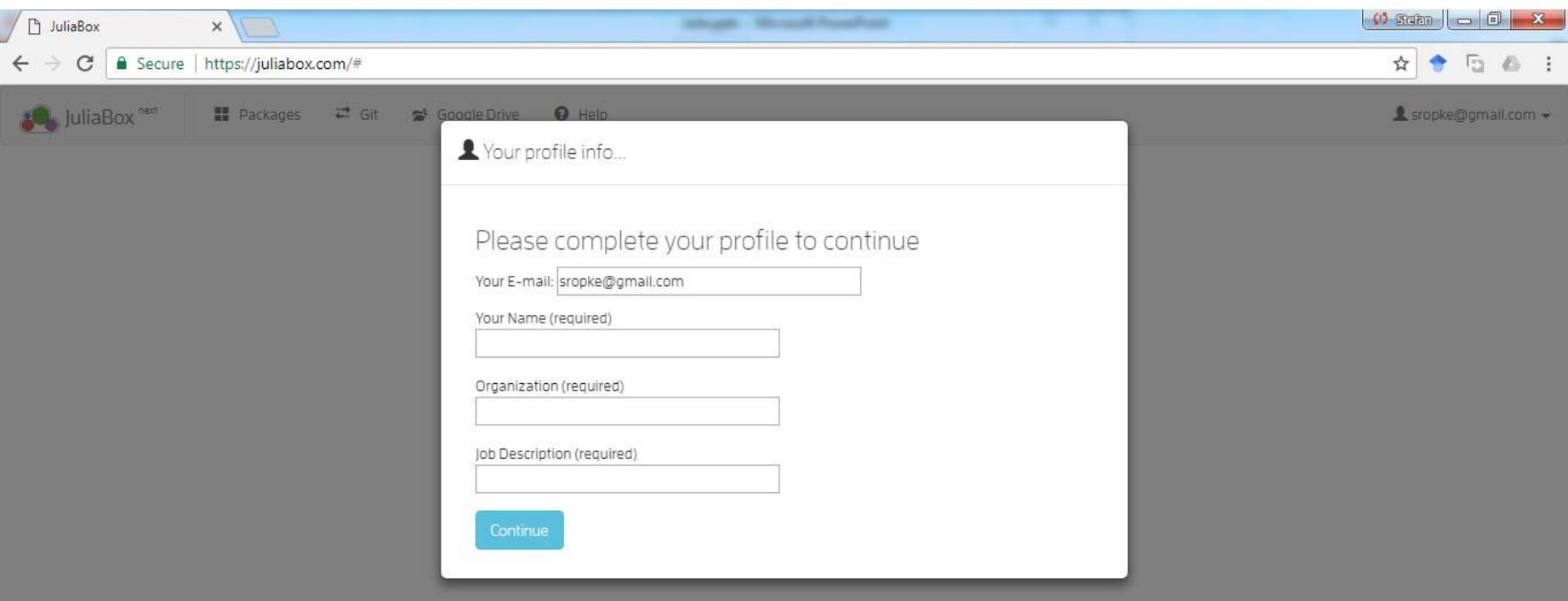
This is the new version of Juliabox. The old version is still available at v1.juliabox.com.
Use the data migration tool (User|Settings) to copy your data from the old system.

 Log in with GitHub
 Log in with Google
 Log in with LinkedIn

This is a beta service. Please backup your data.

JuliaBox

- Fill out information



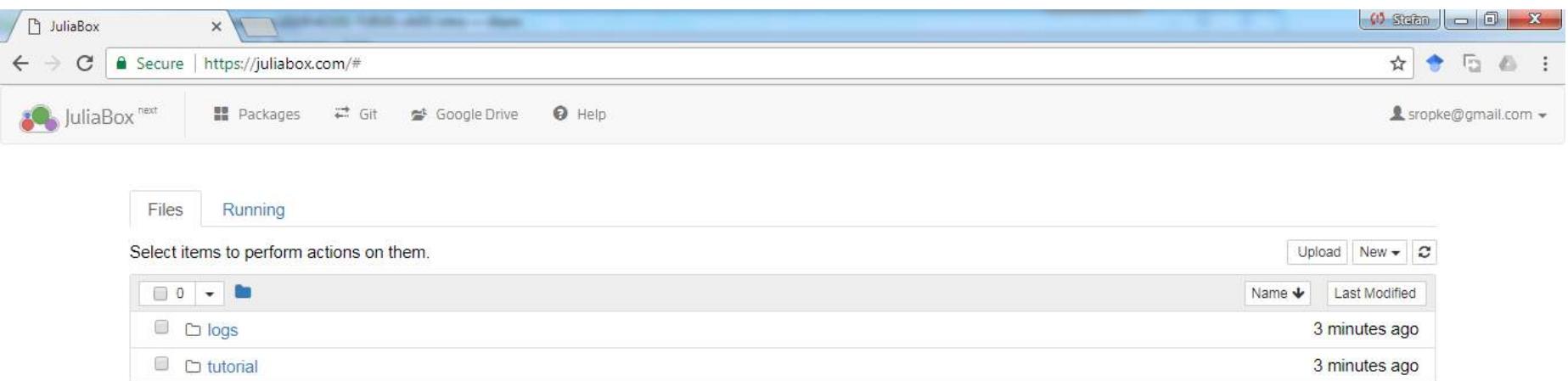
The screenshot shows a web browser window for JuliaBox. The address bar indicates a secure connection to <https://juliabox.com/#>. The main content area displays a "Your profile info..." form with the following fields:

- Your E-mail:
- Your Name (required):
- Organization (required):
- Job Description (required):

A blue "Continue" button is at the bottom of the form.

JuliaBox

From here it is like the DTU solution!



The screenshot shows a web browser window for JuliaBox. The address bar indicates a secure connection to <https://juliabox.com/#>. The top navigation bar includes links for Packages, Git, Google Drive, and Help, along with a user profile for Stefan and an email link to sropke@gmail.com.

The main content area features two tabs: "Files" (selected) and "Running". Below the tabs, there is a message: "Select items to perform actions on them." To the right of this message are buttons for Upload, New, and Refresh.

The file list displays the following entries:

	Name	Last Modified
<input type="checkbox"/>	logs	3 minutes ago
<input type="checkbox"/>	tutorial	3 minutes ago