MATH1326
Advanced Optimisation with Python

Week 1

- Course introduction
  - Canvas
  - Syllabus
  - Assessments
  - Software
- Optimisation
  - Quick refresher
  - Solving MIPs

Course Introduction

- Formulate and solve real-life problems using solvers
  - Mixed Integer Programming (MIP)
- Develop computational optimisation skills to tackle complex problems
  - Column generation
  - Decomposition approaches
  - Dynamic Programming
  - Meta heuristic approaches
- Analyse and interpret solutions

Learning

- Video lectures covering fundamentals
- Self-directed learning using real life problems
- Hands on practical problem solving using PuLP, and Python implementation of advanced optimisation algorithms
- Face to face or online interactive sessions to support learning

Textbook

Guéret, C., Prins, C., & Sevaux, M. (2000). Applications of optimization with Xpress-MP.

Out of print book but available to download as PDF (264 pages) from various online sources.

Mainly be used for the diverse set of applications described in the book

No need to learn the Mosel programming language or install Express MP.

Online Assessments through Canvas

- Open for 5 days with a time limit (>3hr) once you start the assessment.
- Multiple (2-3) randomly assigned problems in each assessment
- Marking based on code and noncode responses – detailed marking rubrics for each question will be provided before the release of each assessment

Software of choice

- Python – PuLP (Open source)
- IBM ILOG CPLEX Optimization Studio (Free to use through IBM Academic Initiative using RMIT Student account)
- Gurobi Optimizer (Free to use as a student through RMIT student account)

Software alternatives

Classical alternatives

- C, C++, C#, Java, Matlab, etc.

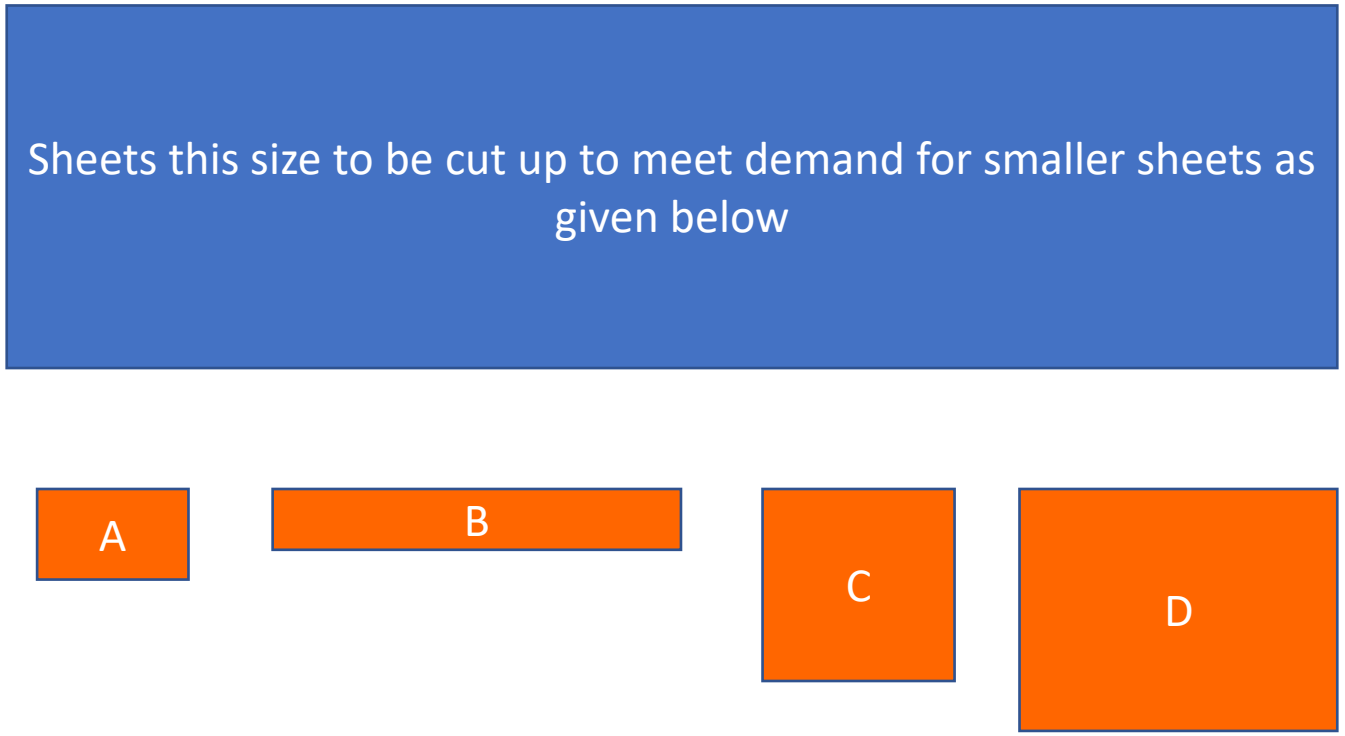
Next generation alternative

- Julia – JuMP (Open source)

A Quick Optimisation Refresher

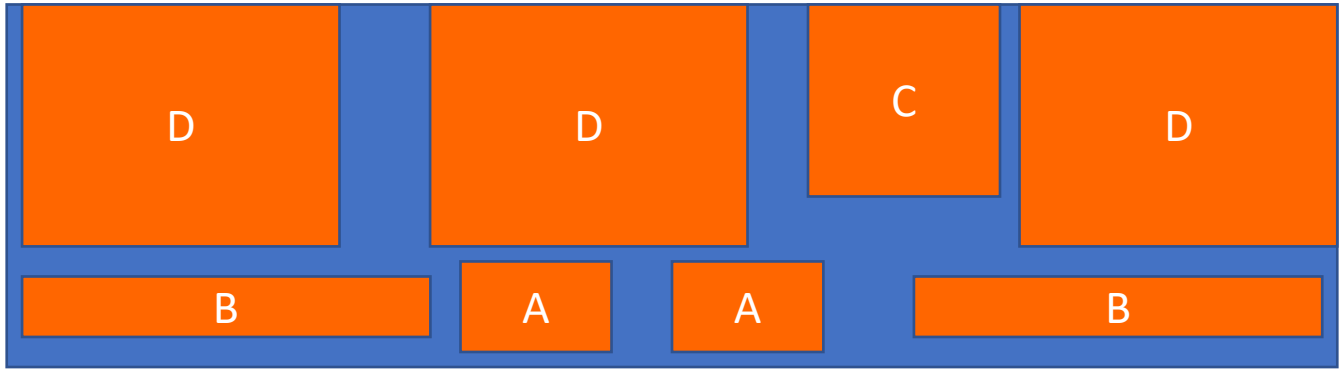
- Optimisation problems
  - Objective function
  - Decision variables
  - Constraints
- Linear Programming (LP)
  - Linear constraints and objective function
- (Mixed) Integer Programming (MIP)
  - Binary variables for (yes/no) decisions

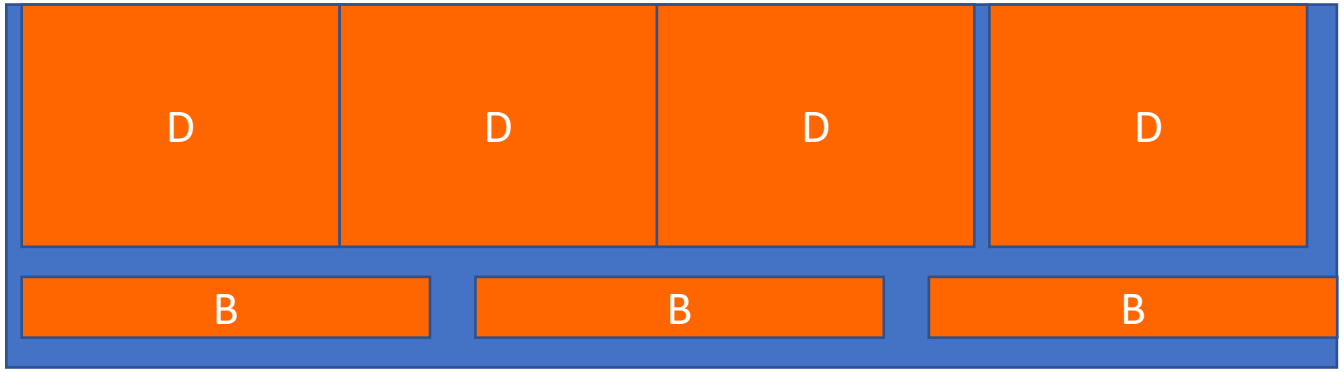
Meet demand with least number of large sheets



Required	22	30	10	15

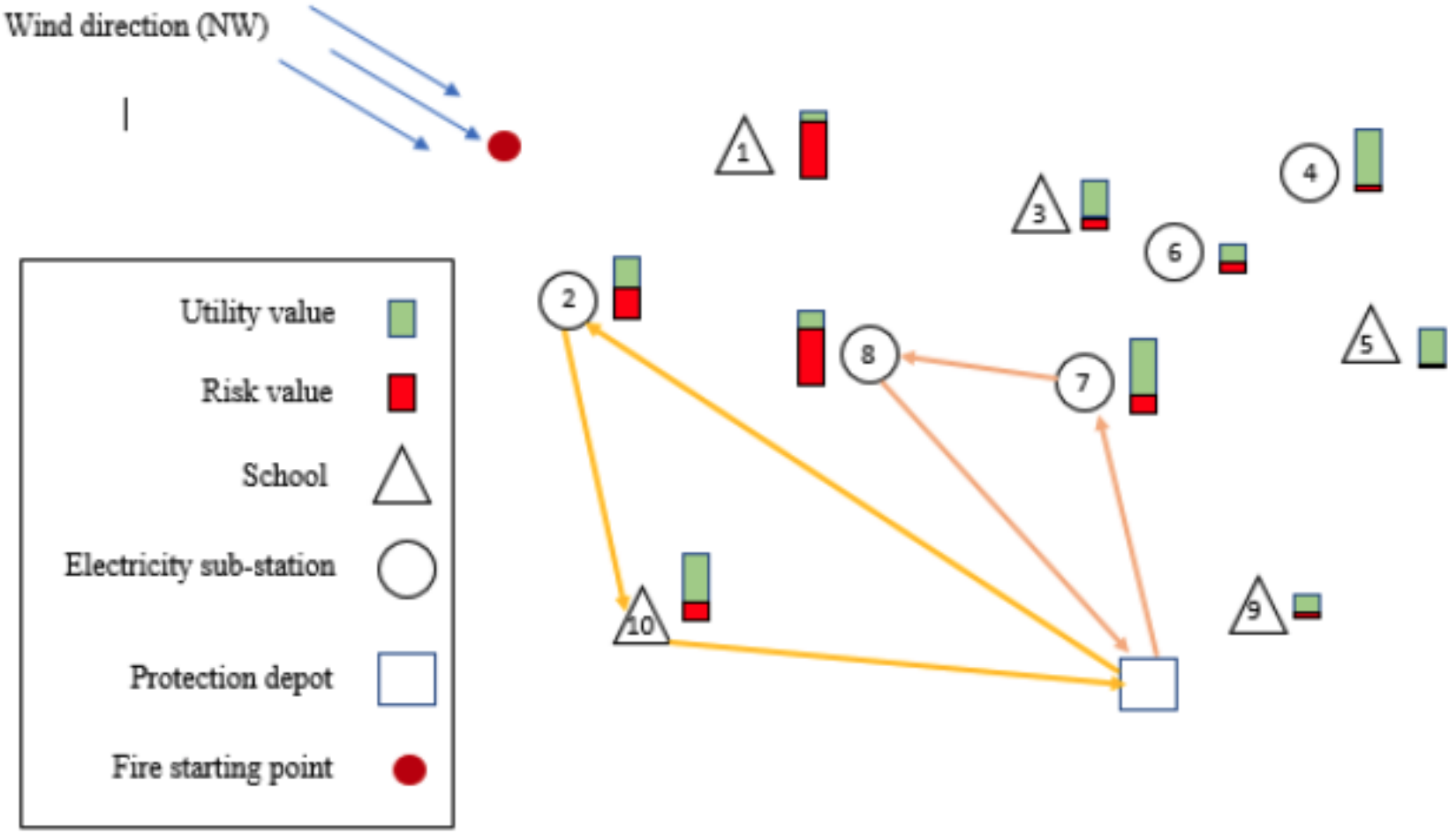
Many ways to cut up the large sheet





Location and optimal routes

Locate depots so that response vehicles can maximise protection of assets



Examples - Mining

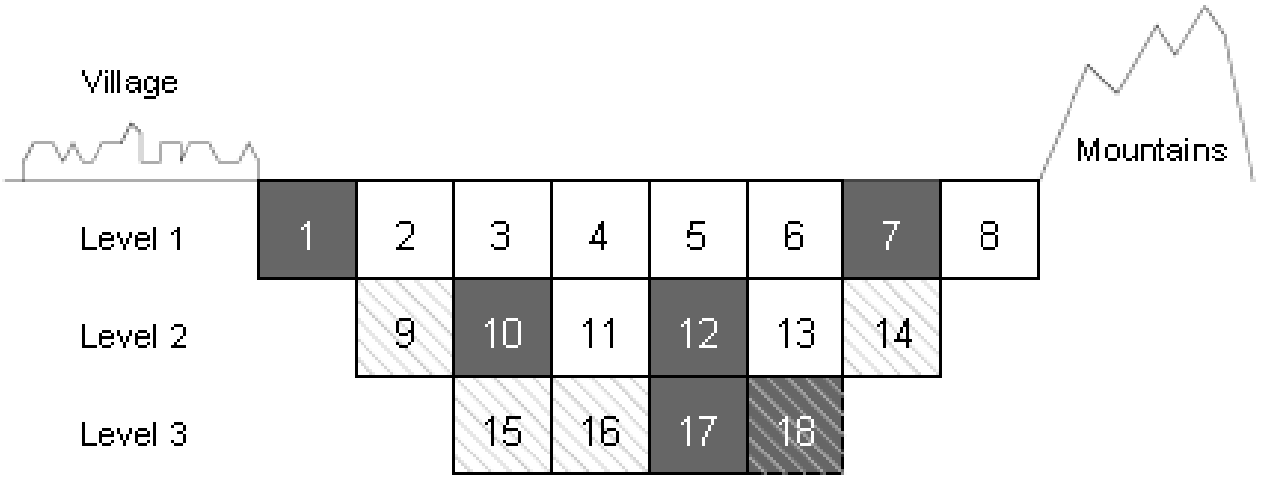
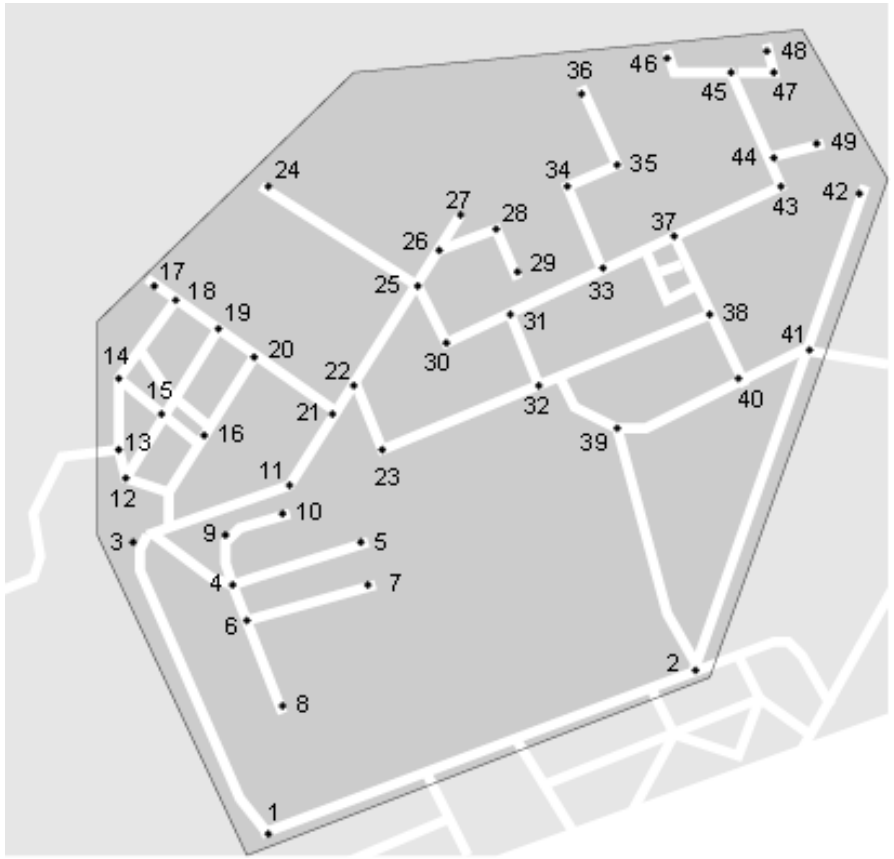


Figure 6.3: Length cut through the opencast mine

To extract a block, three blocks of the level above it need to be extracted: the block immediately on top of it, and also, due to the constraints on the slope, the blocks to the right and to the left.

Examples - Location of Surveillance Cameras

Where to install surveillance cameras to aid the security? What is the minimum number of cameras that have to be installed to survey all the streets and where should they be placed?



Special Kinds of MIP Problems

- Covering problems
- Packing problems
- Partitioning problems
- Knapsack problem
- Vehicle Routing problems / Travelling salesperson problem
- Timetabling problems
- Scheduling problems

Application Areas

- Mining & process industries
- Scheduling
- Loading and cutting
- Transport
- Telecommunication
- Timetabling & personnel planning
- Local authorities
- Many more

Solving MIPs

- Branch and bound
- Cutting planes

Relaxation

- Original IP

  MAX: $2X_1 + 3X_2$

  S.T.: $X_1 + 3X_2 <= 8.25$

  $2.5X_1 + X_2 <= 8.75$

  $X_1, X_2 >= 0$

  X_1, X_2 must be integers

- LP Relaxation

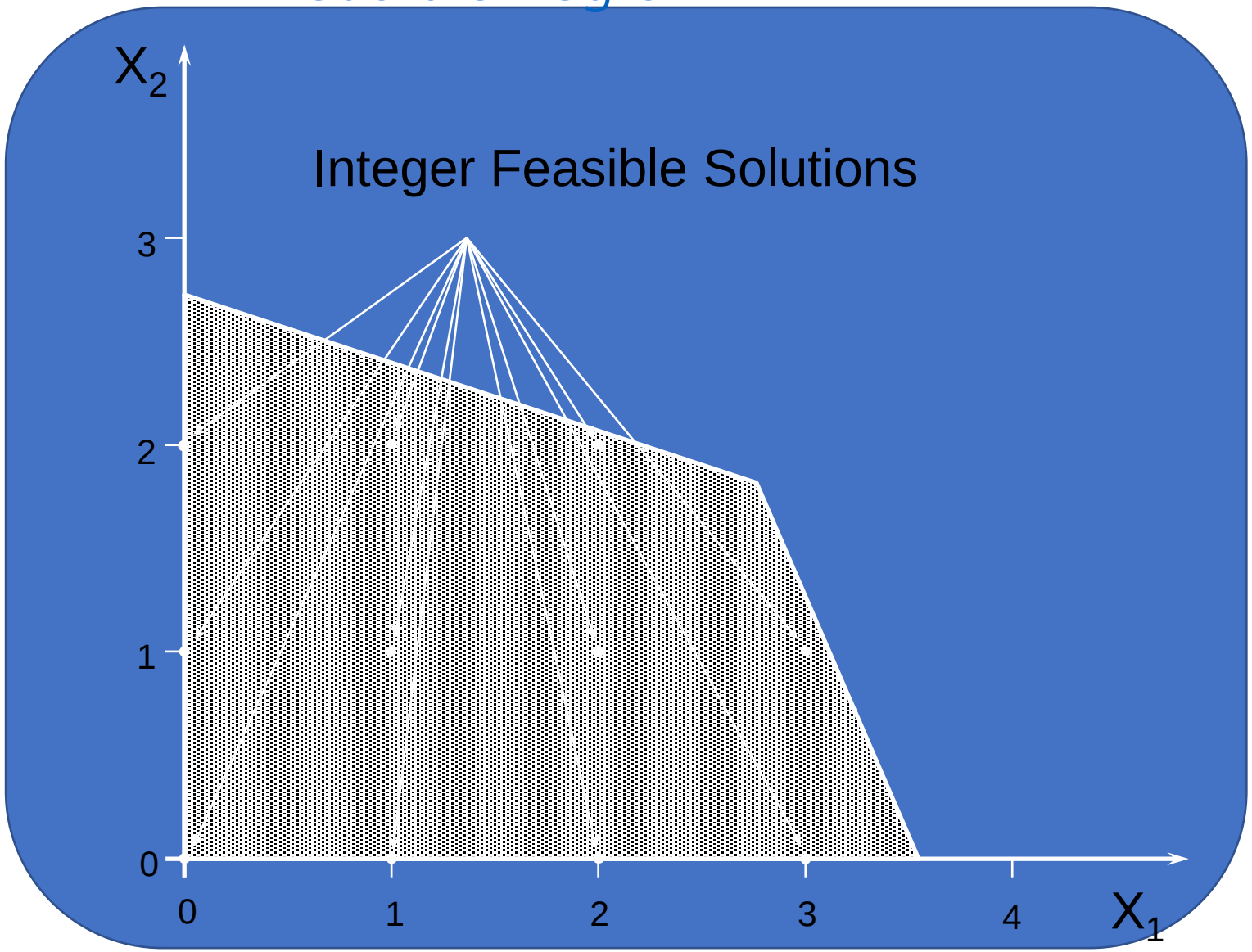
  MAX: $2X_1 + 3X_2$

  S.T.: $X_1 + 3X_2 <= 8.25$

  $2.5X_1 + X_2 <= 8.75$

  $X_1, X_2 >= 0$

Integer Feasible vs. LP Feasible Region



Bounds

- The optimal solution to an LP relaxation of an IP problem gives us a *bound* on the optimal objective function value.
- For maximisation problems, the optimal relaxed objective function values is an *upper bound* on the optimal integer value.
- For minimisation problems, the optimal relaxed objective function values is a *lower bound* on the optimal integer value.

Rounding

- It is tempting to simply round a fractional solution to the closest integer solution.
- In general, this does not work reliably:
  - The rounded solution may be infeasible.
  - The rounded solution may be suboptimal.

Integer Programming Solutions

Maximise $x+y$

Subject to $-2x+2y \geq 1$

$-8x+10y \leq 13$

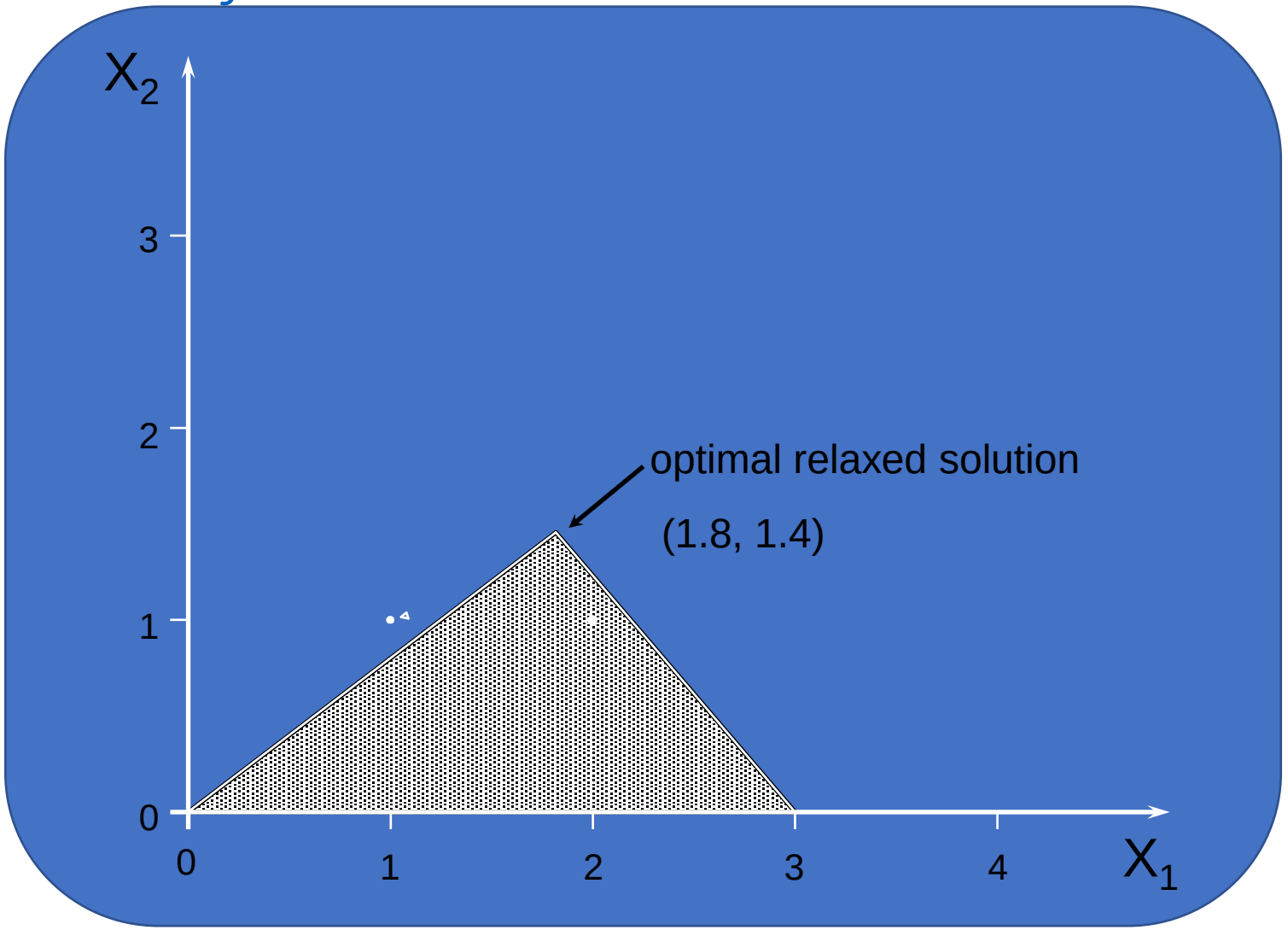
$x,y \geq 0$

Solution $x=4, y=4.5$

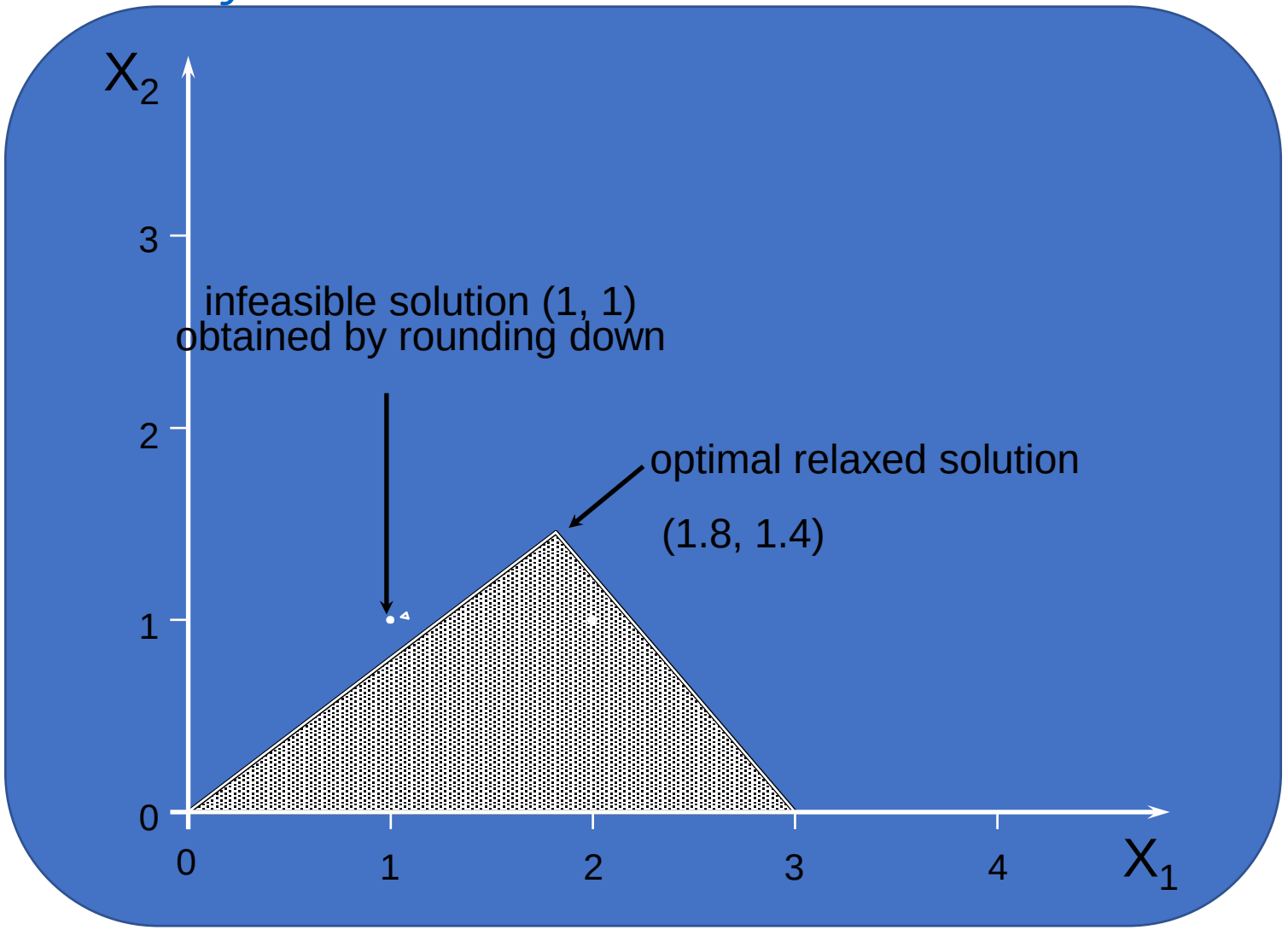
Restricting x and y to integer values

Solution: $x=1, y=2$

How Rounding Down Can Result in an Infeasible Solution



How Rounding Down Can Result in an Infeasible Solution



Branch-and-Bound

- The Branch-and-Bound (B&B) algorithm can be used to solve IP problems.
- Requires the solution of a series of LP problems termed "candidate problems".
- *Theoretically*, this can solve any IP.
- *Practically*, it often takes *LOTS* of computational effort (and time).

The Branch-And-Bound Algorithm

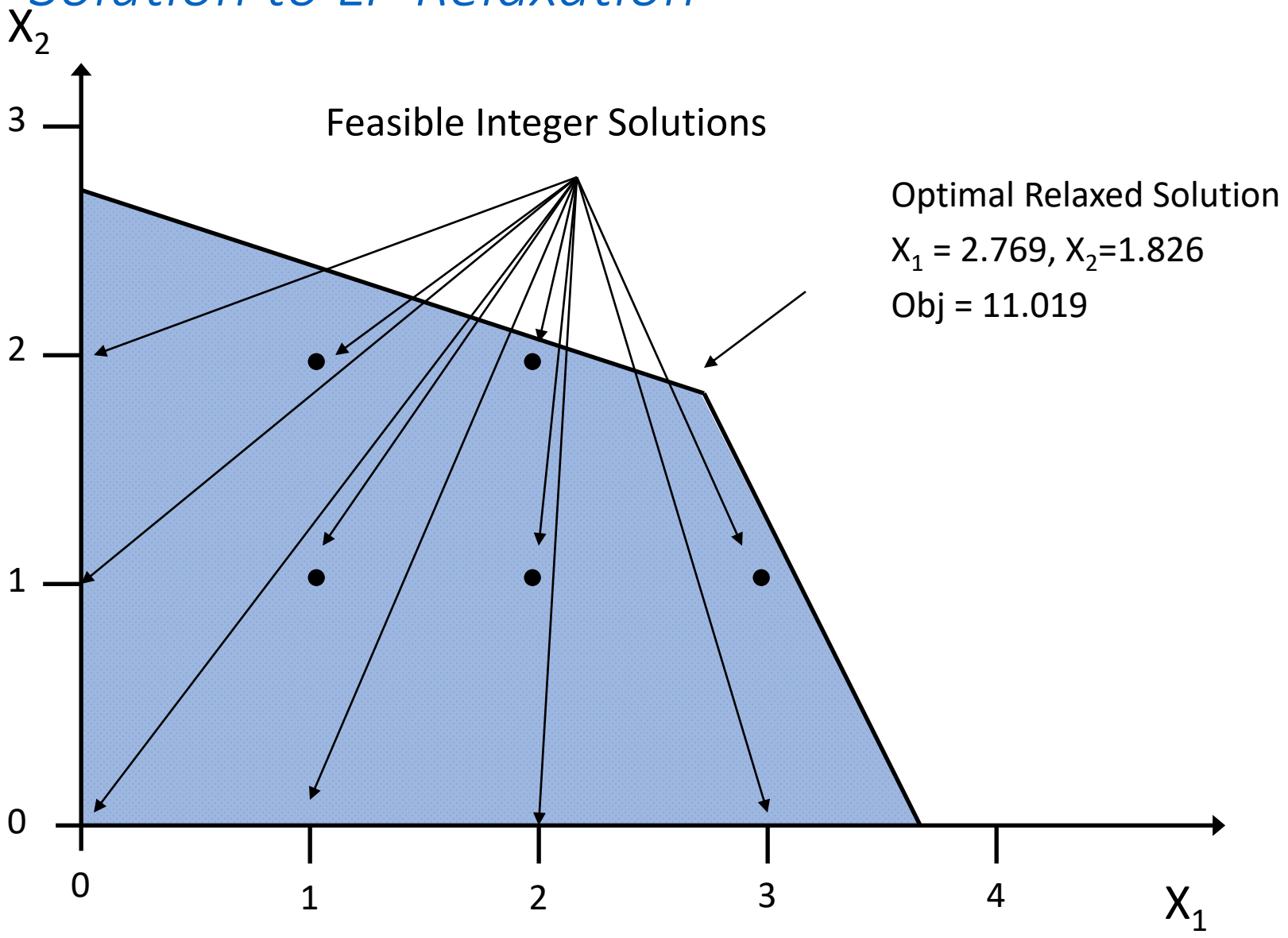
MAX: $2X_1 + 3X_2$

S.T. $X_1 + 3X_2 <= 8.25$

$2.5X_1 + X_2 <= 8.75$

$X_1, X_2 >= 0$ and integer

Solution to LP Relaxation



The Branch-And-Bound Algorithm

Problem I

MAX: $2X_1 + 3X_2$

S.T. $X_1 + 3X_2 <= 8.25$

$2.5X_1 + X_2 <= 8.75$

$X_1 <= 2$

$X_1, X_2 >= 0$ and integer

Problem II

MAX: $2X_1 + 3X_2$

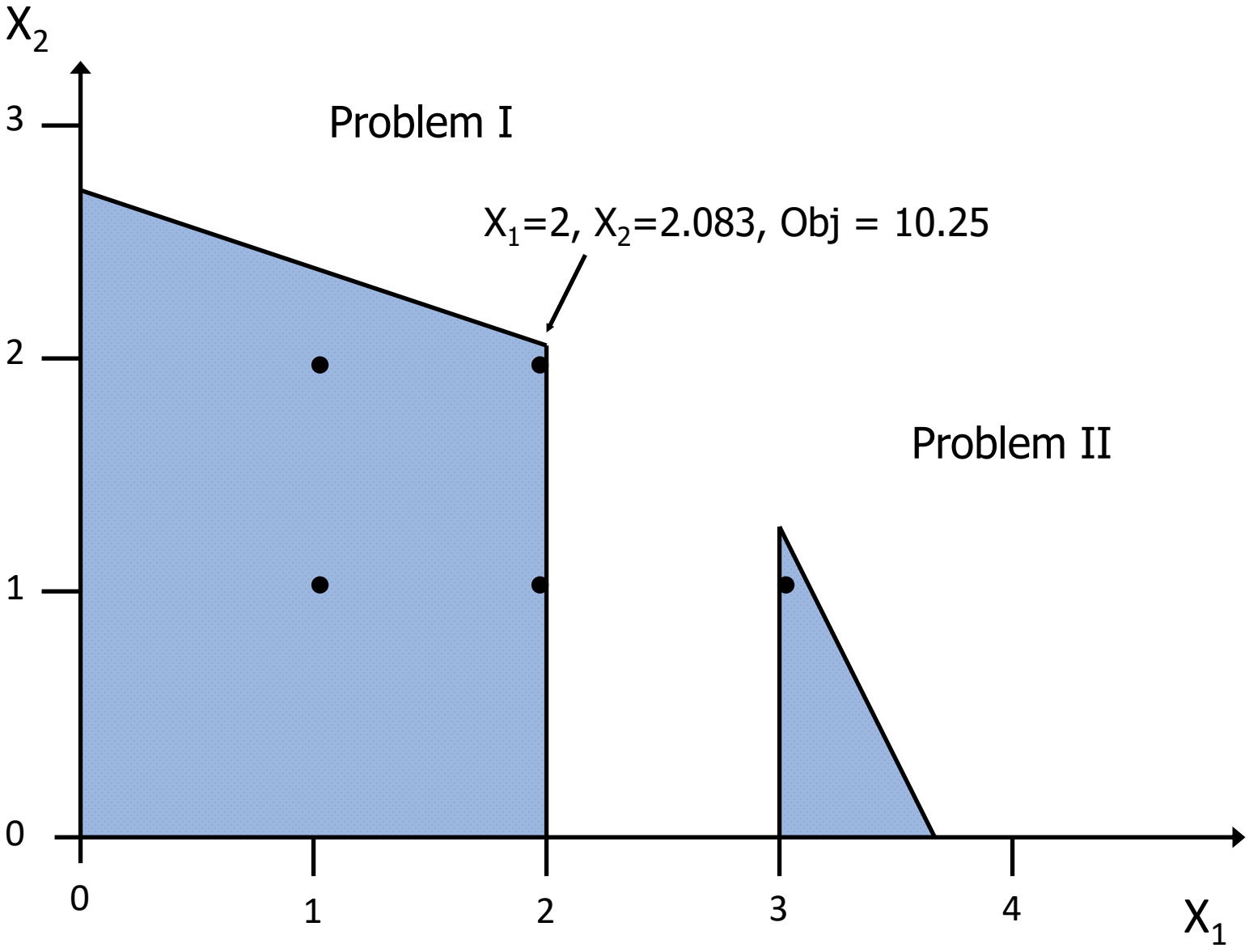
S.T. $X_1 + 3X_2 <= 8.25$

$2.5X_1 + X_2 <= 8.75$

$X_1 >= 3$

$X_1, X_2 >= 0$ and integer

Solution to LP Relaxation



The Branch-And-Bound Algorithm

Problem III

MAX: $2X_1 + 3X_2$

S.T. $X_1 + 3X_2 <= 8.25$

$2.5X_1 + X_2 <= 8.75$

$X_1 <= 2$

$X_2 <= 2$

$X_1, X_2 >= 0$ and integer

Problem IV

MAX: $2X_1 + 3X_2$

S.T. $X_1 + 3X_2 <= 8.25$

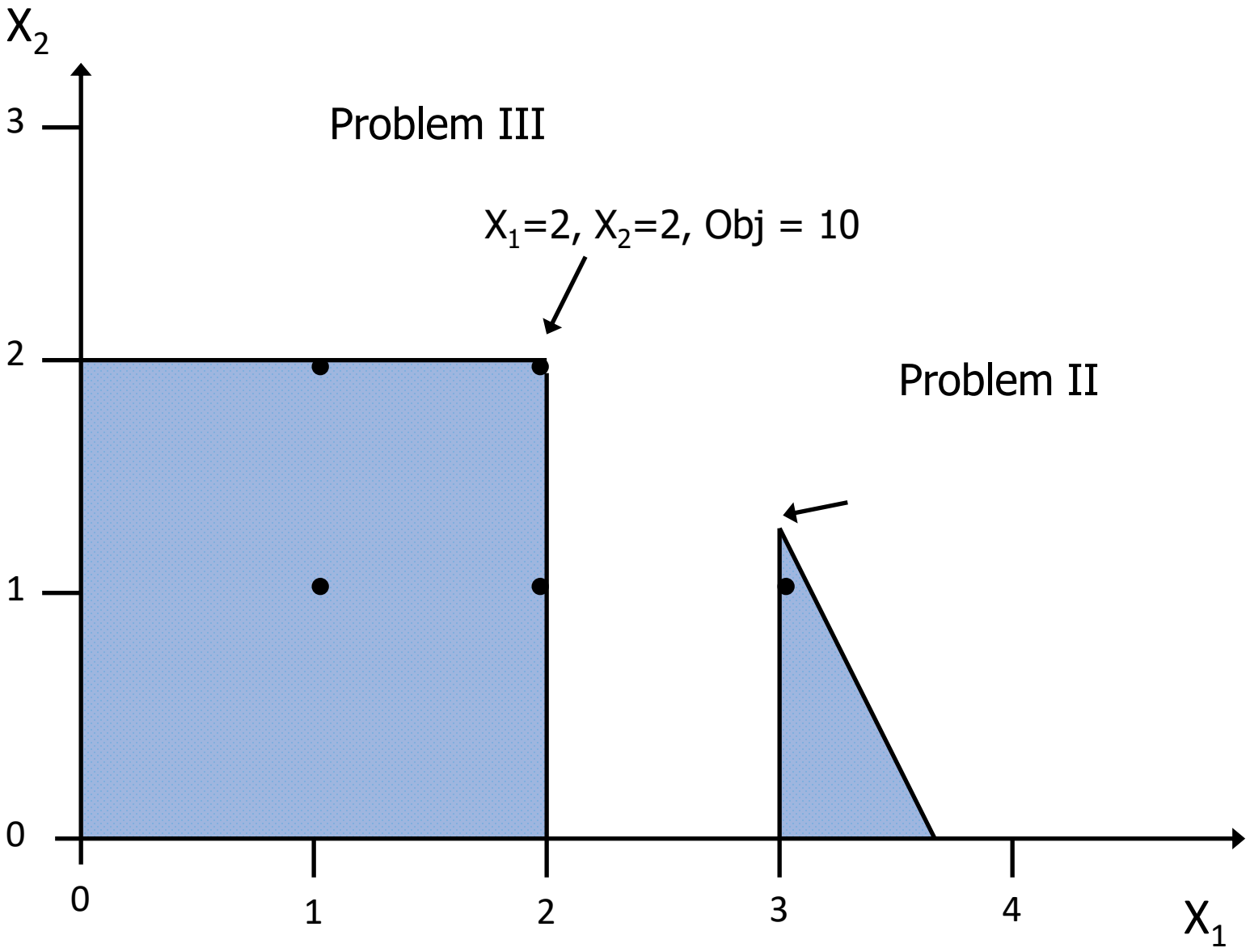
$2.5X_1 + X_2 <= 8.75$

$X_1 <= 2$

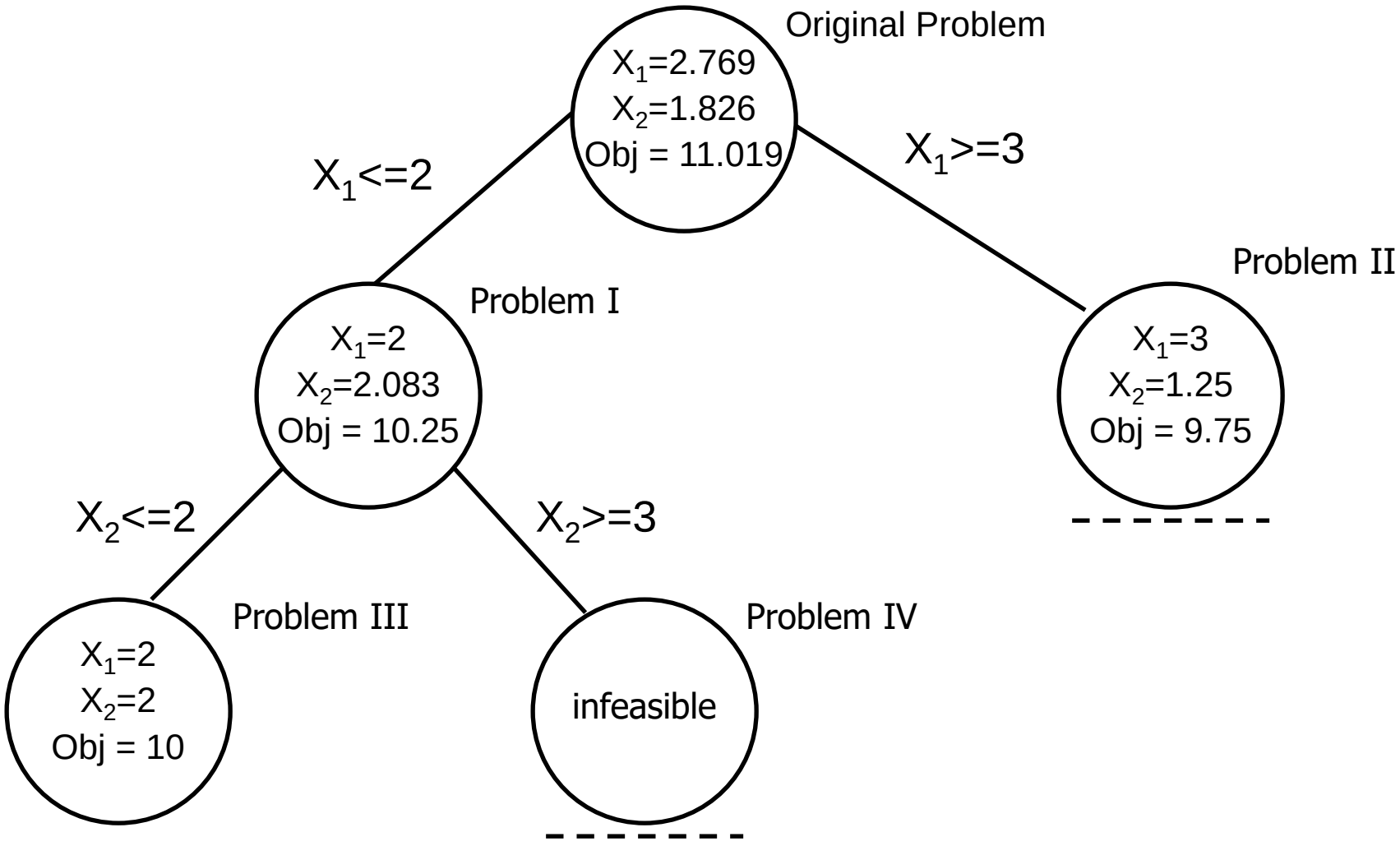
$X_2 >= 3$

$X_1, X_2 >= 0$ and integer

Solution to LP Relaxation



B&B Summary


Original Problem
X1=2.769
X2=1.826
Obj = 11.019
X1<=2
X1>=3
Problem I
X1=2
X2=2.083
Obj = 10.25
Problem II
X1=3
X2=1.25
Obj = 9.75
X2<=2
X2>=3
Problem III
X1=2
X2=2
Obj = 10
Problem IV
infeasible

Best bound and Gap

- Once we have an incumbent in a maximisation problem, the objective value for this incumbent is a valid lower bound on the optimal solution of the given MIP.
- We will never have to accept an integer solution of value lower than this value.
- There is also have a valid upper bound (aka best bound). This bound is obtained by taking the maximum of the optimal objective values of all the current leaf nodes.
- The difference between the current upper and lower bounds is known as the gap. When the gap is zero we have demonstrated optimality.
- For large problems that take a long time to solve we might be happy to accept a solution where the gap is non-zero (e.g. gap=0.1%)

Cutting planes (cuts)

- Cuts tighten the formulation by removing undesirable fractional solutions
- They do this during the solution process and without the undesirable side-effect of creating additional sub-problems (unlike branching).
- It is not usually possible to add them upfront.
- Single most important contributor to the computational advances that have been made in integer programming over the last several years.

Presolve

- Before starting the branch and bound process, some problem reductions are possible.

Consider

$$x_1 + x_2 + x_3 \geq 12$$
$$x_1 \leq 5$$
$$x_2 \leq 4$$
$$x_3 \leq 3$$

There is only one set of values that satisfy all the constraints.

Presolve

Consider

$$2x_1 + 2x_2 \leq 1$$

$$x_1 + x_2 \leq \frac{1}{2}$$

$$x_1, x_2 \in \{0,1\}$$

They must both be zero.

Initial Heuristics

- The better the objective value of the incumbent, the more likely it is that the value of an LP relaxation at another node will be worse and hence lead to a node being fathomed.

- This can save a lot of computational time and effort.

- There are numerous other aspects to a good MIP solver such as branching selection strategies and symmetry detection.

Minimum Cost Network Flow Problems (MCNFP)

- The Branch & Bound method can be computationally expensive and hence slow
- Minimum Cost Network Flow Problems (MCNFP) produce integer solutions without requiring integer constraints
- Transportation, assignment, transhipment, shortest path, maximum flow, and Critical Path Method can all be formulated as MCNFP

Software for solving MIPs

- Why not a spreadsheet?
- Python - a programming language
- PuLP – Python library for linear optimisation (it covers MIPs) works with most open source and commercial solvers.
- IBM ILOG CPLEX and Gurobi – Commercial solvers freely available for academic use
- COIN-OR – Open source tools for optimisation including solvers
- Install Python and PuLP optimisation library
- Install CPLEX and/or Gurobi
- Start experimenting with PuLP, CPLEX and/or Gurobi. Try some of the sample problems.
- Google “Applications of Optimization with Xpress” and download the free pdf book.