Mixed Integer Linear Programming Tutorial

Pano Santos, PhD, Technical Content Manager

GUROBI OPTIMIZATION

The World's Fastest Solver

Tutorial Agenda

Video	Format	Time (min)
1- Why Mixed Integer Programming (MIP)?	Slides	10
2- Resource Assignment Problem (RAP) – Introduction	Slides	10
3- Linear Programming (LP) Formulation – RAP	Slides	10
4- Formulation With Gurobi Python API - RAP	Slides	10
5- Jupyter Notebook 001- RAP	Demo	10
6- Perfect Formulation 002 – RAP	Slides & Demo	10
7- Methods for Solving a MIP Problem	Slides	10
8- Methods for Solving a MIP Problem: Branch-and-Bound Approach	Slides	15
9- Methods for Solving a MIP Problem: Cutting Planes Approach	Slides	10
10- Why MIP Is Better Than Simple Heuristics ? RAP 003	Demo	15
11- Conclusions	Slides	5

Why Mixed Integer Programming (MIP)?

Why MIP?...1

- Mixed integer programming (MIP) can be successfully implemented to optimize the operational efficiency of a complex organization, while considering resource demand and capacity constraints, and critical business rules.

- Applications of MIP models:
  - **Supply Chain Optimization:** SAP Advanced Planning and Optimization and SAP HANA help solve complex optimization problems.
  - **Electrical Power Optimization:** NYISO managed New York's wholesale electrical power market ($7.5 billion in annual transactions) to optimize 500 power-generation units and 11,000 miles of transmission lines with consumer demand in real time.
  - **Government:** The Federal Communications Commission (FCC) used optimization to generate the first two-sided spectrum auction, generating $20 billion in revenue.

Why MIP?...2

- Mixed-integer-programming (MIP) models have been applied in a variety of business realms, often resulting in cost savings of tens or even hundreds of millions of dollars.

- MIP model formulations allow us to combine predicate logic (aka first-order-logic) with optimization.
  - This means several rules can be embedded in the mathematical problem formulation.
  - For example, a production planning model might include a rule that says if product A is produced, then product B must be produced, and product C cannot be produced.

Why MIP?...3

- Applications of MIP models relevant to data scientists:
  - **Marketing Campaign Optimization**
    - There is a set of products and a set of customers.
    - The problem is to identify which product to advertise to which customer, maximizing the advertised product's ROI and staying within a budget.
  - **Price Optimization**
    - There is a set of inventories of various retail products.
    - The problem is to determine at which price each retail product should be sold in order to maximize profits, while considering a minimum level of ROI and market share, and other business rules.
  - **Resource Management Optimization**
    - There is a set of resources with various qualifications, and there is a set of jobs with specific requirements regarding resources qualifications.
    - The problem is to determine an assignment of resources to jobs that maximizes the total matching score of the assignments.

Why MIP?...4

- Approaches to solving MIP problems:
  - Branch-and-bound approach
  - Cutting planes approach

Resource Assignment Problem

Introduction

Resource Assignment Problem…1

- Consulting company’s open positions: Tester, Java-Developer, and Architect
- Three top resources: Carlos, Joe, and Monika
- Results of competency tests: matching scores
- Assumption: Only one resource can be assigned to a job, and at most one job can be assigned to a resource.

Matching Scores	Tester	Java-Developer	Architect
Carlos 	**53%**	**27%**	**13%**
Joe 	**80%**	**47%**	**67%**
Monika 	**53%**	**73%**	**47%**

Resource Assignment Problem…2

- Simple heuristic approach
  1. Consider the highest score, match the resource with the job of the highest score, eliminate the resource and the job from the table.
  2. Choose the next highest score. If no scores are available, STOP; all jobs have been assigned to all resources. Otherwise, go to 1.

Matching Scores	Tester	Java-Developer	Architect
Carlos	53%	27%	13%
Joe	80%	47%	67%
Monika	53%	73%	47%

Resource Assignment Problem…3

- Simple heuristic approach
  1. Consider the highest score, match the resource with the job of the highest score, eliminate the resource and the job from the table.
  2. Choose the next highest score. If no scores are available, STOP; all jobs have been assigned to all resources. Otherwise, go to 1.

Matching Scores	Tester	Java-Developer	Architect
Carlos	53%	27%	13%
Joe	80%	47%	67%
Monika	53%	73%	47%

Resource Assignment Problem…4

- Simple heuristic approach
  1. Consider the highest score, match the resource with the job of the highest score, eliminate the resource and the job from the table.
  2. Choose the next highest score. If no scores are available, STOP; all jobs have been assigned to all resources. Otherwise, go to 1.

Matching Scores	Tester	Java-Developer	Architect
Carlos		**27%**	**13%**
Joe	**80%**		
Monika		**73%**	**47%**

Resource Assignment Problem…5

- Simple heuristic approach
  1. Consider the highest score, match the resource with the job of the highest score, eliminate the resource and the job from the table.
  2. Choose the next highest score. If no scores are available, STOP; all jobs have been assigned to all resources. Otherwise, go to 1.

Matching Scores	Tester	Java-Developer	Architect
Carlos			13%
Joe	80%		
Monika		73%	

Resource Assignment Problem…6

- Simple heuristic approach
    1. Consider the highest score, match the resource with the job of the highest score, eliminate the resource and the job from the table.
    2. Choose the next highest score. If no scores are available, STOP; all jobs have been assigned to all resources. Otherwise, go to 1.

Matching Scores	Tester	Java-Developer	Architect
Carlos			13%
Joe	80%		
Monika		73%	

Resource Assignment Problem…7

- Remarks
  - The total score of the assignment is **80% + 73% + 13% = 166%.**
  - Note that this assignment is far from optimal since Carlos was assigned to a job with the lowest score. We can certainly do much better than that!

Matching Scores	Tester	Java-Developer	Architect
Carlos 			**13%**
Joe 	**80%**		
Monika 		**73%**	

Simple heuristic approach

1. Consider the highest score, match the resource with the job of the highest score, eliminate the resource and the job from the table.
2. Choose the next highest score. If no scores are available, STOP; all jobs have been assigned to all resources. Otherwise go to 1.

Resource Assignment Problem…8

- Another simple approach:

  - Assign the first job to any of the three candidates.
  - Assign the second job to any of the remaining two candidates.
  - Assign the third job to the remaining candidate.
  - The number of possible assignments is 3*2*1 = 6. Note: The number (3 factorial) 3!= 3*2*1 = 6.

Resource Assignment Problem…9

Number	Assignment	Score
1	Tester - Carlos (53%) Java - Joe (47%) Archt.- Monika (47%)	**147%**
2	Tester- Joe (80%) Java - Carlos (27%) Archt.- Monika (47%)	**154%**
3	Tester - Monika (53%) Java - Joe (47%) Archt.- Carlos (13%)	**113%**
4	Tester - Carlos (53%) Java - Monika (73%) Archt.- Joe (67%)	**193%**
5	Tester - Joe (80%) Java - Monika (73%) Archt.- Carlos (13%)	**166%**
6	Tester - Monika (53%) Java - Carlos (27%) Archt.- Joe (67%)	**147%**

Another simple approach:

- Assign the first job to any of the three candidates.
- Assign the second job to any of the remaining two candidates.
- Assign the third job to the remaining candidate.
- The number of possible assignments is 3*2*1 = 6.
  Note: the number (3 factorial) 3!= 3*2*1 = 6.

Resource Assignment Problem…10

- The consulting company wins a major government contract that requires 100 jobs.

  - The problem is to assign 100 candidates to 100 jobs.

  - The enumeration algorithm proposed above enumerates 100! possible assignments.

  - $100! \approx 9.3\times10^{157}$,

  - This number is humongous, much larger than the number of atoms in the universe, which is approximately 10^{80}.

Resource Assignment Problem…11

- Even the fastest supercomputer called "Summit"
- That can make mathematical calculations at a rate of 200 petaflops/second
- Will take an astronomical number of years to enumerate all the assignments and determine the optimal one.

Linear Programming Formulation

Resource Assignment Problem (RAP)

Linear Programming Formulation…1

- Problem statement
  - Three **jobs**: Tester, Java-Developer, and Architect
  - Three **resources**: Carlos, Joe, and Monika
  - **Data**: matching scores for all job-resource combinations
  - **Assumption**: Only one resource can be assigned to a job, and at most one job can be assigned to a resource
  - **Problem**: Determine an assignment that ensures each job is fulfilled and each resource is assigned to at most one job, maximizing the total matching scores of the assignments.

Matching Scores	Tester	Java-Developer	Architect
Carlos	**53%**	**27%**	**13%**
Joe	**80%**	**47%**	**67%**
Monika	**53%**	**73%**	**47%**

Linear Programming Formulation…2

- Decision variables
  - We need to identify which resource is assigned to which job
  - Therefore, we have 9 decision variables, one per possible assignment
  - To simplify the mathematical notation of the model formulation, let's define the following indices for the resources and for the jobs
  - The decision variable $x_{r,j} = 1$ represents that resource r is assigned to job j, and 0 otherwise, for $\mathrm{r} = 1,2,3$ and $j = 1,2,3$.

Decision Variables	Tester = 1	Java-Developer = 2	Architect = 3
Carlos = 1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
Joe = 2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
Monika = 3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$

Linear Programming Formulation…3

- Job constraints
  - The job constraint of the Tester position is that either resource 1 (Carlos), resource 2 (Joe), or resource 3 (Monika) is assigned to this job.
  - Constraint (Tester = 1): $x_{1,1} + x_{2,1} + x_{3,1} = 1$
  - Similarly the constraints for the Java Developer and Architect positions can be defined as follows
  - Constraint (Java Developer = 2): $x_{1,2} + x_{2,2} + x_{3,2} = 1$
  - Constraint (Architect = 3): $x_{1,3} + x_{2,3} + x_{3,3} = 1$
  - Notice that the job constraints are defined by the columns of the following table.

Decision Variables 	Tester = 1	Java-Developer = 2	Architect = 3
Carlos = 1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
Joe = 2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$
Monika = 3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$
Requirement	1	1	1

Linear Programming Formulation…4

- Resource constraints
  - The constraint for Carlos is that either job 1 (Tester), job 2 (Java Developer), or job 3 (Architect) is assigned to this resource. Assume #resources ≥ #jobs.
  - Constraint (Carlos = 1): $x_{1,1} + x_{1,2} + x_{1,3} \leq 1$
  - Similarly the constraints for the resources Joe and Monika can be defined as follows:
  - Constraint (Joe = 2): $x_{2,1} + x_{2,2} + x_{2,3} \leq 1$
  - Constraint (Monika = 3): $x_{3,1} + x_{32} + x_{3,3} \leq 1$
  - Notice that the resource constraints are defined by the rows of the following table.

Decision Variables	Tester = 1	Java-Developer = 2	Architect = 3	Availability
Carlos = 1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	1
Joe = 2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	1
Monika = 3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	1

Linear Programming Formulation…5

- Objective function
  - The objective function is to maximize the total matching score of the assignments that satisfy the job and resource constraints. Notice that for job1 the matching score is $53x_{1,1}$ -if resource Carlos is assigned, or $80x_{2,1}$ -if resource Joe is assigned, or $53x_{3,1}$-if resource Monika is assigned. Consequently,
  - Job 1 (Tester) matching score: $(53x_{1,1}+80x_{2,1}+53x_{3,1})$
  - Job 2 (Java Developer) matching score: $(27x_{1,2}+47x_{2,2}+73x_{3,2})$
  - Job 3 (Architect): $(13x_{1,3}+67x_{2,3}+47x_{3,3})$
  - Maximize total matching score:
    $(53x_{1,1}+80x_{2,1}+53x_{3,1})+(27x_{1,2}+47x_{2,2}+73x_{3,2})+(13x_{1,3}+67x_{2,3}+47x_{3,3})$.

Decision Variables	Tester = 1	Java-Developer = 2	Architect = 3
Carlos = 1	$53x_{1,1}$	$27x_{1,2}$	$13x_{1,3}$
Joe = 2	$80x_{2,1}$	$47x_{2,2}$	$67x_{2,3}$
Monika = 3	$53x_{3,1}$	$73x_{3,2}$	$47x_{3,3}$

Linear Programming Formulation With Gurobi Python API

Resource Assignment Problem (RAP)

Linear Programming Formulation With Gurobi Python API…1

- This line of code imports the Gurobi Python callable library.

```
# import gurobi library
from gurobipy import *
```

Linear Programming Formulation With Gurobi Python API…2

- Input data

```python
# resources and jobs sets
R = ['Carlos', 'Joe', 'Monika']
J = ['Tester', 'JavaDeveloper', 'Architect']
```

- List of resources and jobs

```python
# matching score data
combinations, ms = multidict({
    ('Carlos', 'Tester'): 53,
    ('Carlos', 'JavaDeveloper'): 27,
    ('Carlos', 'Architect'): 13,
    ('Joe', 'Tester'): 80,
    ('Joe', 'JavaDeveloper'): 47,
    ('Joe', 'Architect'): 67,
    ('Monika', 'Tester'): 53,
    ('Monika', 'JavaDeveloper'): 73,
    ('Monika', 'Architect'): 47
})
```

- The “multidict” function describes the matching score associated with each possible assignment.

Linear Programming Formulation With Gurobi Python API …3

- Declare the resource assignment problem (RAP) model formulation.

```
# Declare and initialize model
m = Model('RAP')
```

Linear Programming Formulation With Gurobi Python API…4

- Decision variables
- Gurobi “tupledict” object x contains the newly created variables.

```
# Create decision variables for the RAP model
x = m.addVars(combinations, name="assign")
```

Linear Programming Formulation With Gurobi Python API…5

- Job constraints
  - The job constraint of the Tester position is that either resource 1 (Carlos), resource 2 (Joe), or resource 3 (Monika) is assigned to this job.
  - Constraint (Tester = 1): $x_{1,1} + x_{2,1} + x_{3,1} = 1$
  - Constraint (Java Developer = 2): $x_{1,2} + x_{2,2} + x_{3,2} = 1$
  - Constraint (Architect = 3): $x_{1,3} + x_{2,3} + x_{3,3} = 1$

```
# create jobs  constraints
jobs = m.addConstrs((x.sum('*',j) == 1 for j in J), 'job')
```

Linear Programming Formulation With Gurobi Python API…5

- Resource constraints
  - The constraint for Carlos is that either job 1 (Tester), job 2 (Java Developer), or job 3 (Architect) is assigned to this resource. Assume #resources ≥ #jobs.
  - Constraint (Carlos = 1): $x_{1,1} + x_{1,2} + x_{1,3} \leq 1$
  - Constraint (Joe = 2): $x_{2,1} + x_{2,2} + x_{2,3} \leq 1$
  - Constraint (Monika = 3): $x_{1,3} + x_{2,3} + x_{3,3} \leq 1$

```
# create resources constraints
resources = m.addConstrs((x.sum(r,'*') <= 1 for r in R), 'resource')
```

Linear Programming Formulation With Gurobi Python API…6

- Objective function
  - The objective function is to maximize the total matching score of the assignments that satisfy the job and resource constraints.
  - Maximize total matching score:
    $(53x_{1,1}+80x_{2,1}+53x_{3,1})+(27x_{1,2}+47x_{2,2}+73x_{3,2})+(13x_{1,3}+67x_{2,3}+47x_{3,3})$.

Decision Variables	Tester = 1	Java-Developer = 2	Architect = 3
Carlos = 1 	$53x_{1,1}$	$27x_{1,2}$	$13x_{1,3}$
Joe = 2 	$80x_{2,1}$	$47x_{2,2}$	$67x_{2,3}$
Monika = 3 	$53x_{3,1}$	$73x_{3,2}$	$47x_{3,3}$

```python
# The objective is to maximize total matching score of the assignments
m.setObjective(x.prod(ms), GRB.MAXIMIZE)
```

Linear Programming Formulation With Gurobi Python API…7

```python
# save model for inspection
m.write('RAP.lp')
```

- The "write( )" function prints the model formulation.

```
\ Model RAP
\ LP format - for model browsing. Use MPS format to capture full model detail.
Maximize
  53 assign[Carlos,Tester] + 27 assign[Carlos,JavaDeveloper]
   + 13 assign[Carlos,Architect] + 80 assign[Joe,Tester]
   + 47 assign[Joe,JavaDeveloper] + 67 assign[Joe,Architect]
   + 53 assign[Monika,Tester] + 73 assign[Monika,JavaDeveloper]
   + 47 assign[Monika,Architect]
Subject To
 job[Tester]: assign[Carlos,Tester] + assign[Joe,Tester]
   + assign[Monika,Tester] = 1
 job[JavaDeveloper]: assign[Carlos,JavaDeveloper]
   + assign[Joe,JavaDeveloper] + assign[Monika,JavaDeveloper] = 1
 job[Architect]: assign[Carlos,Architect] + assign[Joe,Architect]
   + assign[Monika,Architect] = 1
 resource[Carlos]: assign[Carlos,Tester] + assign[Carlos,JavaDeveloper]
   + assign[Carlos,Architect] <= 1
 resource[Joe]: assign[Joe,Tester] + assign[Joe,JavaDeveloper]
   + assign[Joe,Architect] <= 1
 resource[Monika]: assign[Monika,Tester] + assign[Monika,JavaDeveloper]
   + assign[Monika,Architect] <= 1
Bounds
End
```

Linear Programming Formulation With Gurobi Python API…8

- Solving the RAP problem

- The “optimize( )” function invokes the optimize method on the model object “m”.

```
# run optimization engine
m.optimize()
```

Linear Programming Formulation With Gurobi Python API…9

Gurobi Solution of the RAP Problem.

```python
# display optimal values of decision variables
for v in m.getVars():
    if (abs(v.x) > 1e-6):
        print(v.varName, v.x)

# display optimal total matching score
print('total matching scores', m.objVal)
```

```
assign[Carlos,Tester] 1.0
assign[Joe,Architect] 1.0
assign[Monika,JavaDeveloper] 1.0
total matching scores 193.0
```

Solution by Complete Enumeration.

Number	Assignment	Score
1	Tester - Carlos (53%) Java - Joe (47%) Archt.- Monika (47%)	**147%**
2	Tester- Joe (80%) Java - Carlos (27%) Archt.- Monika (47%)	**154%**
3	Tester - Monika (53%) Java - Joe (47%) Archt.- Carlos (13%)	**113%**
4	Tester - Carlos (53%) Java - Monika (73%) Archt.- Joe (67%)	**193%**
5	Tester - Joe (80%) Java - Monika (73%) Archt.- Carlos (13%)	**166%**
6	Tester - Monika (53%) Java - Carlos (27%) Archt.- Joe (67%)	**147%**

Jupyter Notebook Demo: RAP 001

Perfect Formulation

Resource Assignment Problem (RAP)

Perfect Formulation – RAP…1

- Decision variables (binary): $x_{r,j} = 1$ if resource r is assigned to job j, 0 otherwise.

```
# Create decision variables for the RAP model
x = m.addVars(combinations, name="assign")
```

- The default value of the type variables in the “addVars( )” method is: non-negative and continuous. This includes fractional values.
- The optimal solution is:

```
assign[Carlos,Tester] 1.0
assign[Joe,Architect] 1.0
assign[Monika,JavaDeveloper] 1.0
total matching scores 193.0
```

- The values of the assignment variables are binary even though we did not require them to be binary. Why?

Perfect Formulation – RAP…2

- A mathematical optimization problem with decision variables that are non-negative and continuous (fractional values), linear constraints, and a linear objective function is called a linear programming (LP) problem.

- The resource assignment problem (RAP) that we formulated is an LP problem.

- The RAP model belongs to a special class of LP problems that guarantees that if all the input data are integers (RAP: matching scores and the RHS of the job and resource constraints), then the optimal solution are integer numbers (RAP: optimal solution is defined as binary numbers).

Perfect Formulation – RAP...3

- Let's revise the RAP problem as follows:
  - There is a fixed cost associated to assigning a resource to a job: $C_{r,j}$.
  - There is a limited budget that can be used for assigning resources to jobs: B

Assignment Costs ($K)	Tester = 1	Java-Developer = 2	Architect = 3
Carlos = 1	**1**	**1**	**1**
Joe = 2	**2**	**2**	**2**
Monika = 3	**3**	**3**	**3**



- The budget is $5,000.

Perfect Formulation – RAP...4

- Jupyter Notebook demo:
  - Formulate a new mathematical optimization model considering assignment costs and budget constraint.

Methods for Solving MIP Problems

Reference book: *Integer Programming,* by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

Mixed Integer (Linear) Programming (MIP) Models

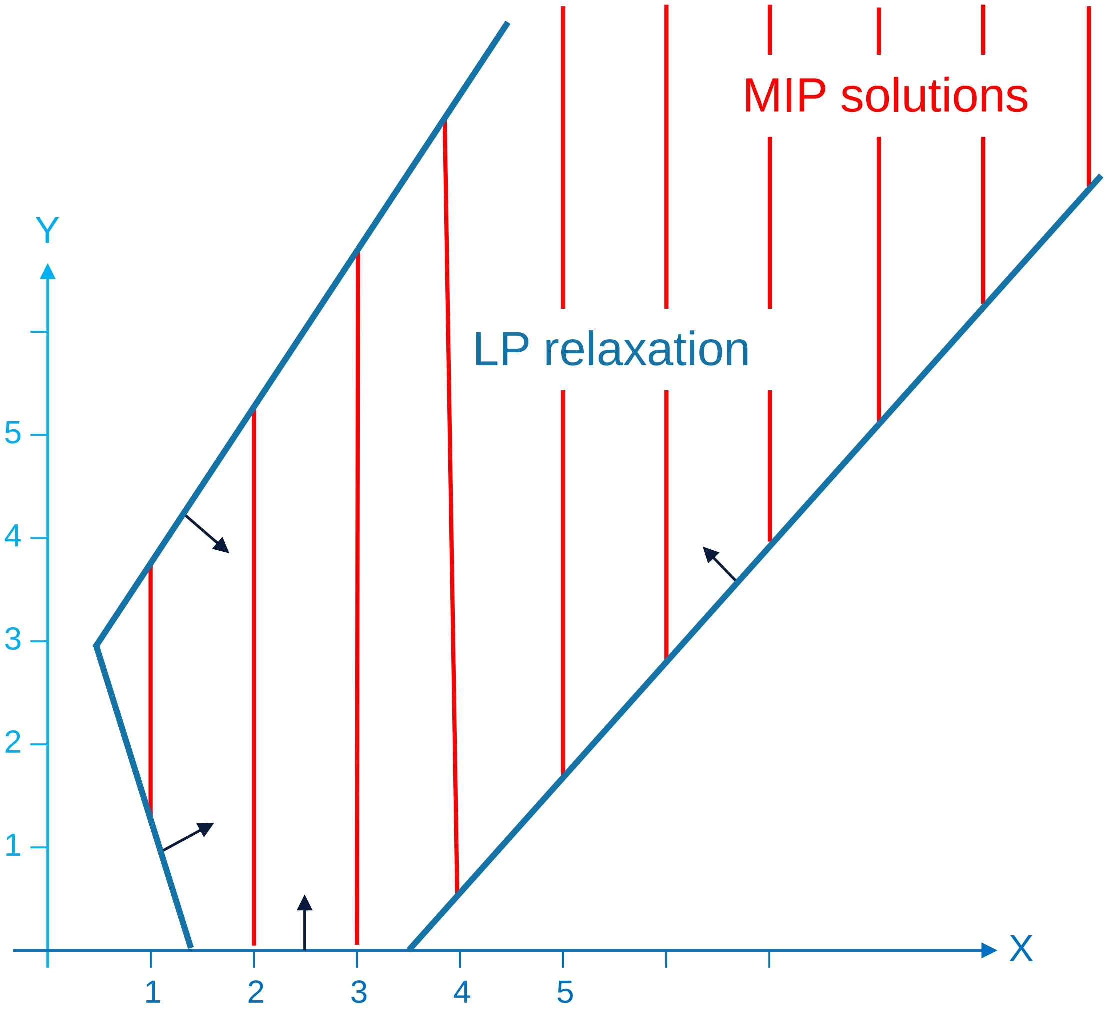
$$Max\ cx + hy$$

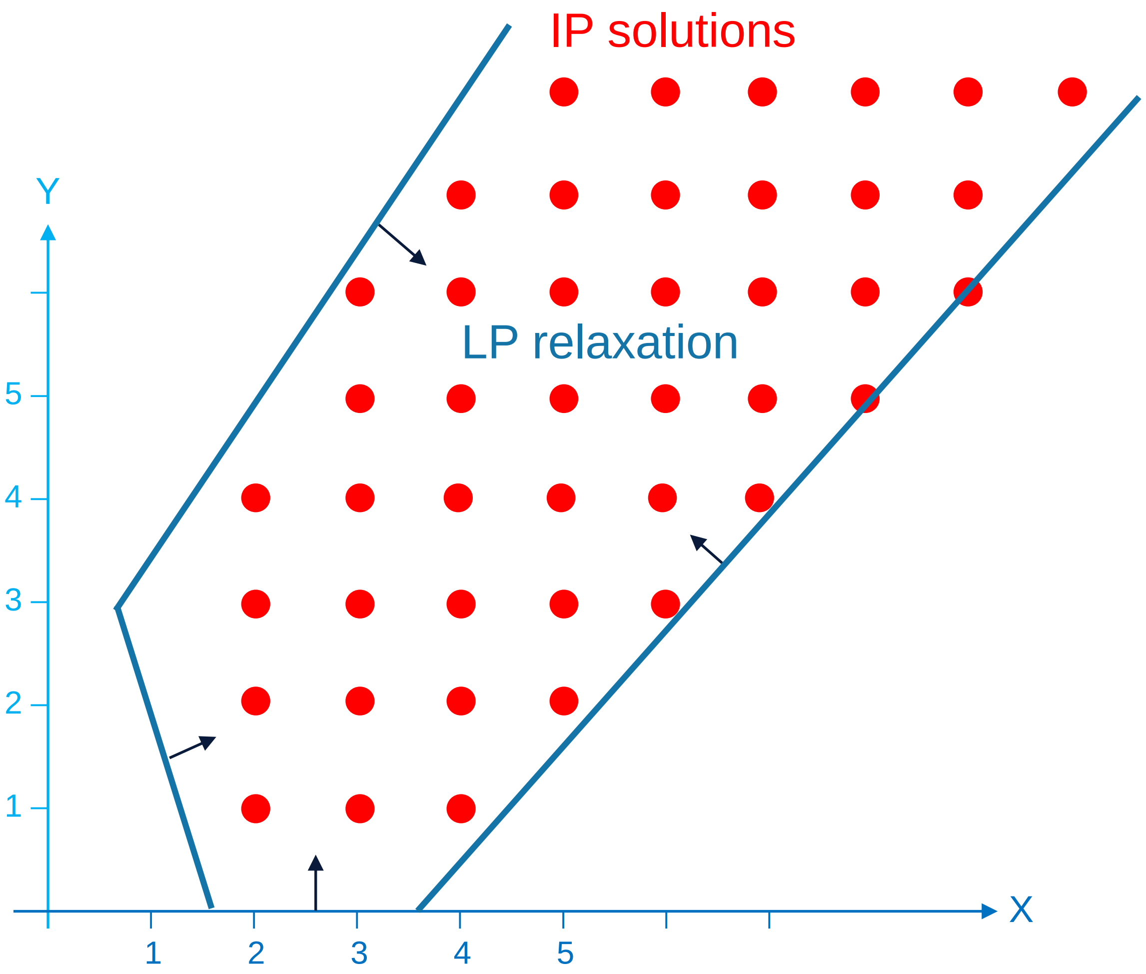
Subject to: $Ax + Gy \ \leq b$

$$x \geq 0\ integer$$

$$y \geq 0$$

Graphical Representation of a MIP and a Pure IP





Solving MIP Problems

- In mathematical programming, the key idea in solving MIP problems is to use the LP relaxation.

- As we have seen, there are LP problems, such as the assignment problem, for which the LP relaxation provides integer optimal solutions.

- In the following sections, we will present two methods to solve MIP problems. They are based on solving a series of LP problem relaxations.

  - Branch-and-bound

  - Cutting planes

Approach 1: Branch-and-Bound Methods for Solving MIP Problems

Reference book: *Integer Programming,* by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

Branch-and-Bound Method…1

- MIP problem.

$$Max\ z = 5.5x_1 + 2.1x_2$$

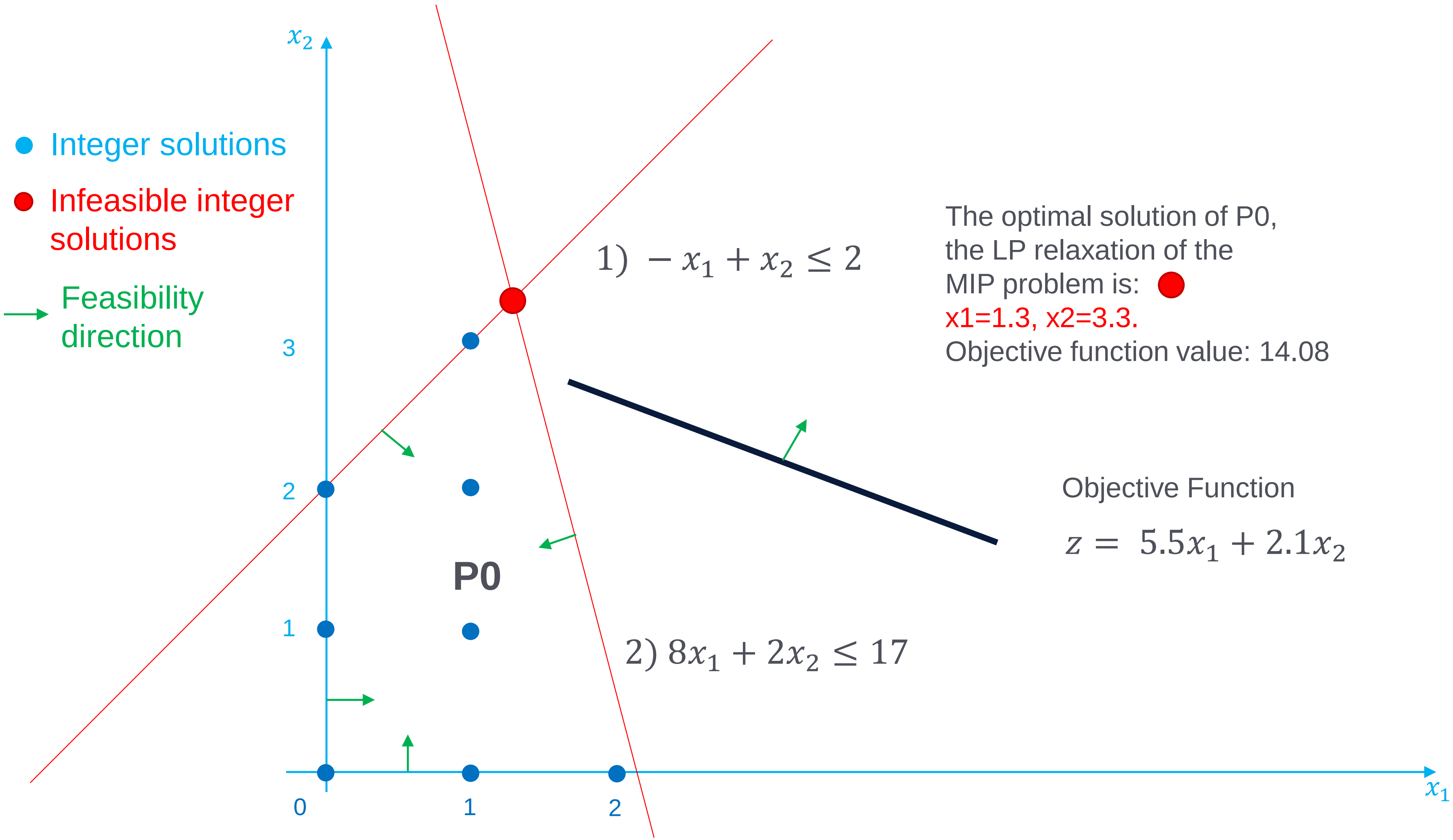
Subject to:

$$\begin{array}{ll} 1) & -x_1 + x_2 \leq 2 \\ 2) & 8x_1 + 2x_2 \leq 17 \\ & x_1, x_2 \geq 0 \text{ and integer} \end{array}$$

- P0 = LP relaxation of MIP problem (i.e., the integrality constraints are removed).

- P0 contains all the integer solutions of the original MIP problem.

Graphical Representation of P0 Problem



- Integer solutions
- Infeasible integer solutions
- Feasibility direction

The optimal solution of P0, the LP relaxation of the MIP problem is:
x1=1.3, x2=3.3.
Objective function value: 14.08

Objective Function

$$z = \ 5.5x_1 + 2.1x_2$$

Branch-and-Bound Method…2

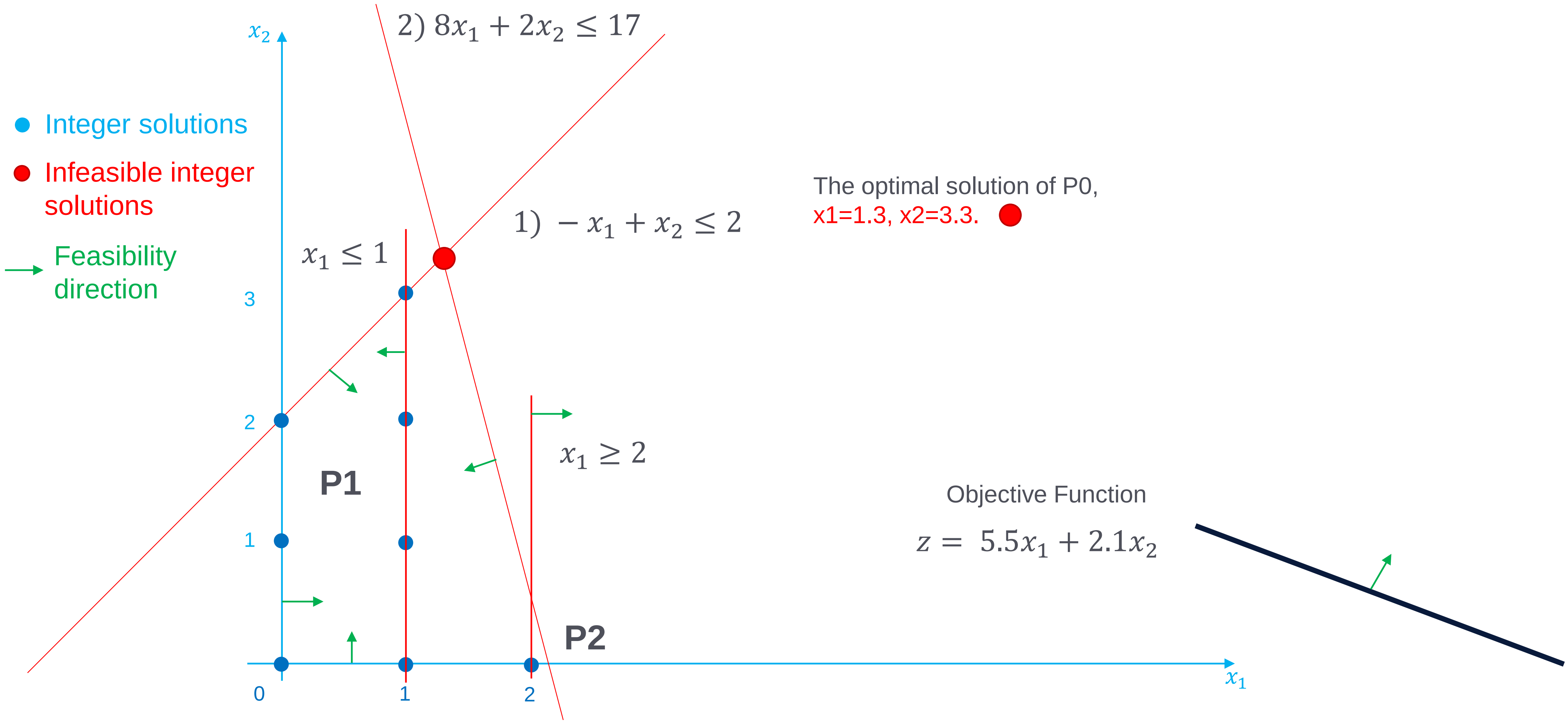
- Notice that the objective function value (14.08) of P0 is an upper bound to the MIP problem.

- Branching:
  - The optimal values of the decision variables are fractional ($x1 = 1.3$ and $x2 = 3.3$).

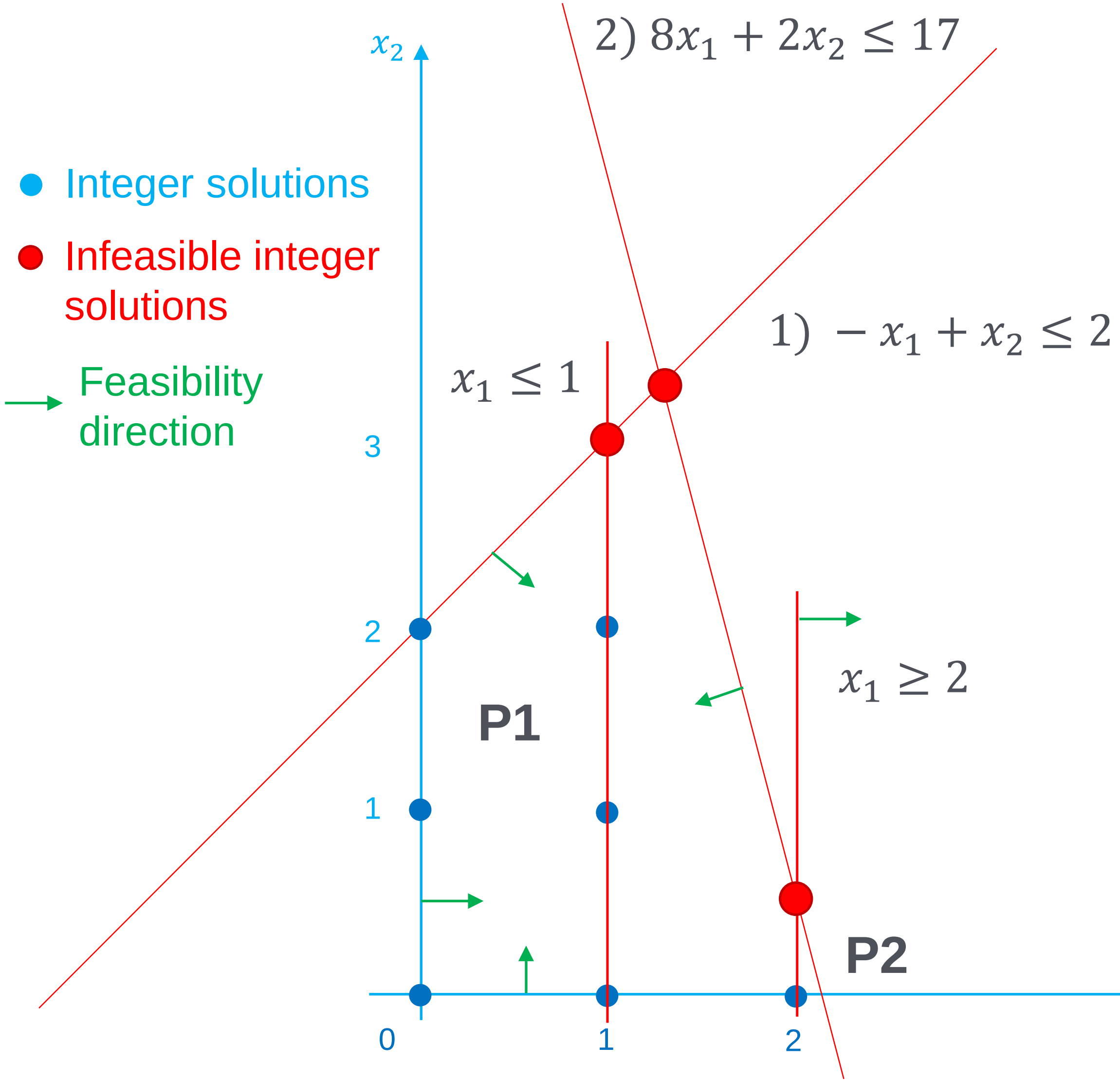
- We partition the set of solutions for the problem P0, into two subproblems.
  - P1 is the original problem P0 and the constraint $x1 \leq 1$.
  - P2 is the original problem P0 and the constraint $x1 \geq 2$.

- This process of partitioning a problem into subproblems is called branching.

Branching of P0 Problem



Solving Problems P1 and P2



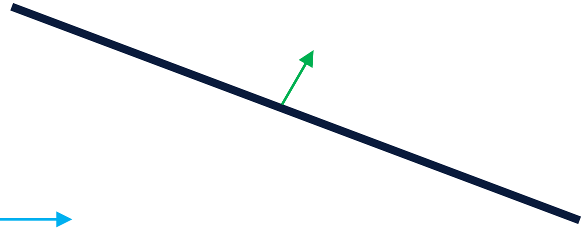
+ Notice that the optimal solution of P0: x1=1.3, x2=3.3 **is infeasible for** problems P1 and P2, these problems contain all the integer solutions of the MIP problem, and P1 ∩ P2 = ∅

+ The optimal solution of P1: x1=1, x2=3
Objective function value = 11.8

+ The optimal solution of P2: x1=2, x2=0.5
Objective function value = 12.05

Objective Function

$$z = 5.5x_1 + 2.1x_2$$



Remarks on Solving Problem P1

- The optimal solution of the LP relaxation P1 is integer ($x1 = 1$ and $x2 = 3$).

- The optimal objective function value of P1 is: 11.8. It is a lower bound of the MIP problem.

- This problem does not need to be explored further.

- We say that this problem is **pruned by integrality.**

Remarks on Solving Problem P2

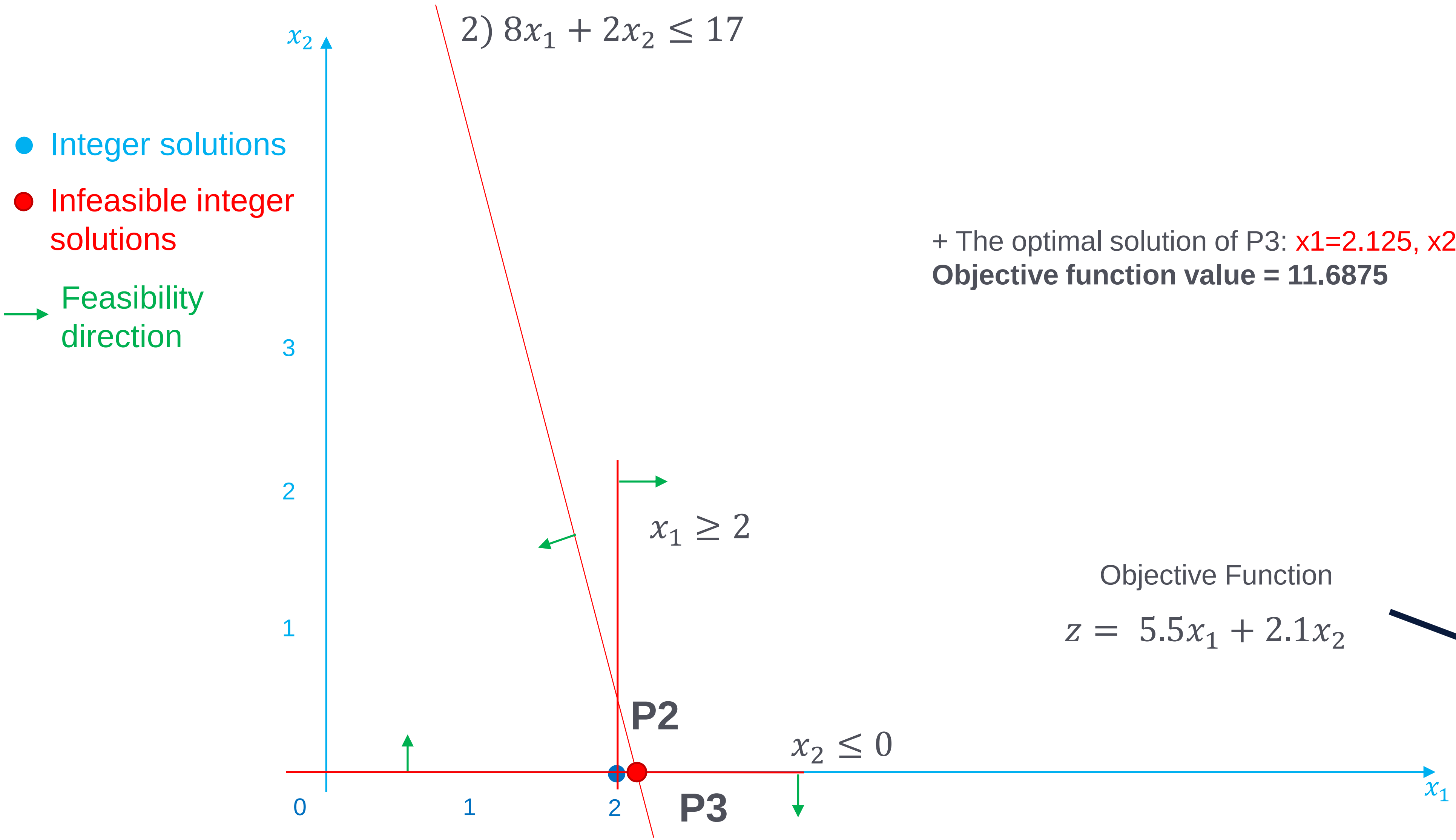
- The optimal solution of the LP relaxation P2 is fractional ($x1 = 2$ and $x2 = 0.5$).
- The optimal objective function value of P2 is: 12.05. It is a tighter upper bound of the MIP problem.
- This optimal solution is not integer; therefore, this problem needs to be explored further.

Branch-and-Bound Method…3

- Branching of P2 Problem:

  - The optimal objective function value of P2 is 12.05 > 11.8 (best integer solution found so far).
  - X2 = 0.5 ➔ branch on x2.
  - We partition the set of solutions of problem P2, into two subproblems.
    - P3 is the original problem P2 and the constraint $x2 \leq 0$,
    - P4 is the original problem P2 and the constraint $x2 \geq 1$.

Solving Problem P3



+ The optimal solution of P3: x1=2.125, x2=0
Objective function value = 11.6875

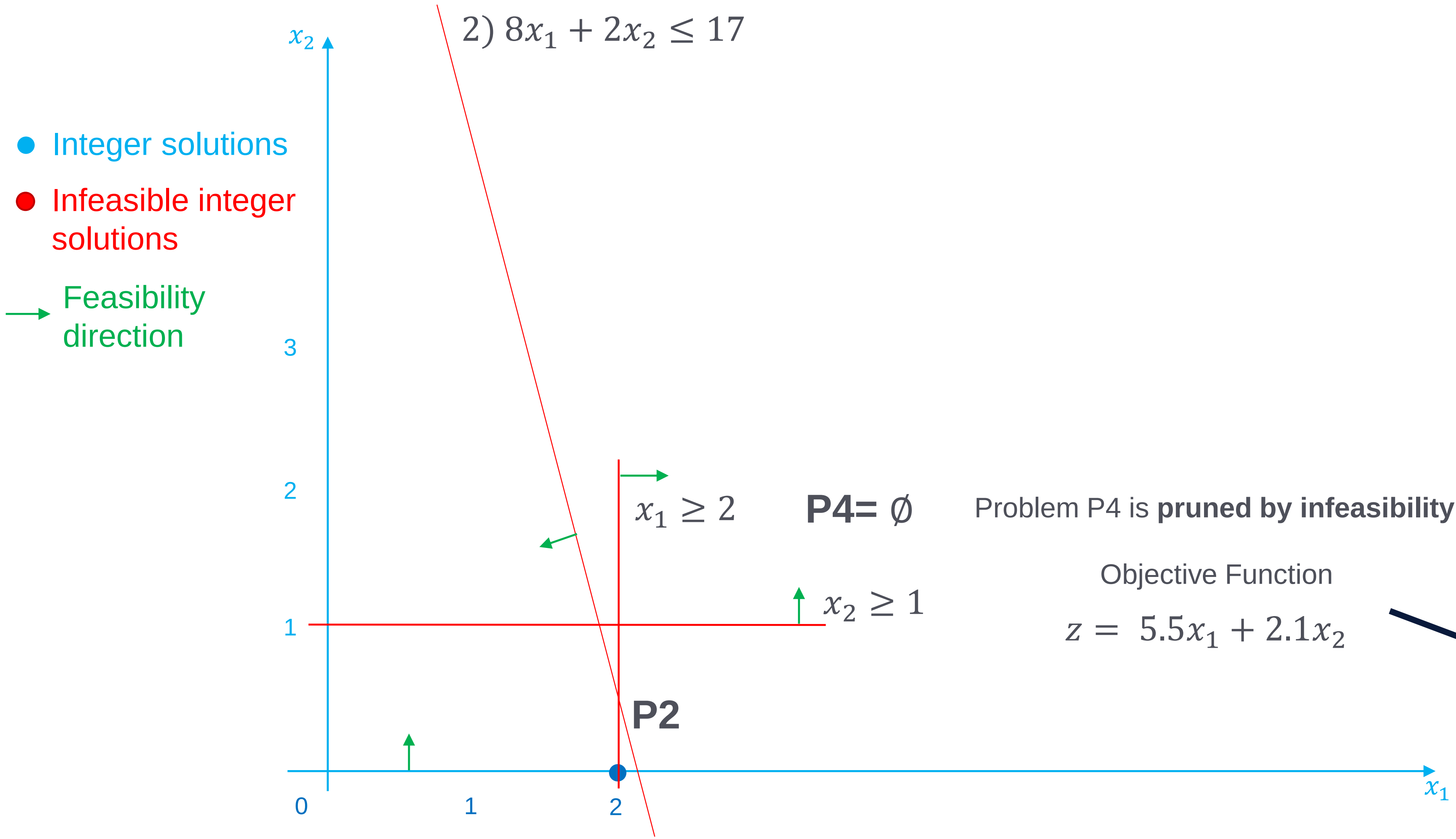
Objective Function

$$z = 5.5x_1 + 2.1x_2$$

Remarks on Solving Problem P3:

- The optimal solution of the LP relaxation P3 is fractional (x1 = 2.125 and x2 = 0).
- The optimal objective function value of P3 is: 11.6875.
- Since P3 is an LP relaxation, the value 11.6875 is an upper bound of the integer optimal solution of any subproblem derived from P3.
- Since 11.6875 < 11.8 (objective function value of the best integer solution found so far), then problem P3 does not require further exploration, and it is considered to be **pruned by bound.**

Solving Problem P4



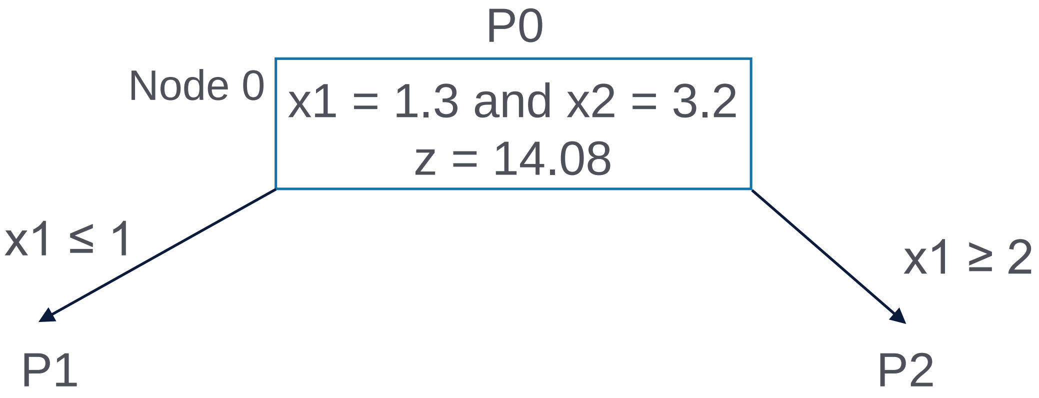
Problem P4 is **pruned by infeasibility**

Objective Function

$$z = 5.5x_1 + 2.1x_2$$

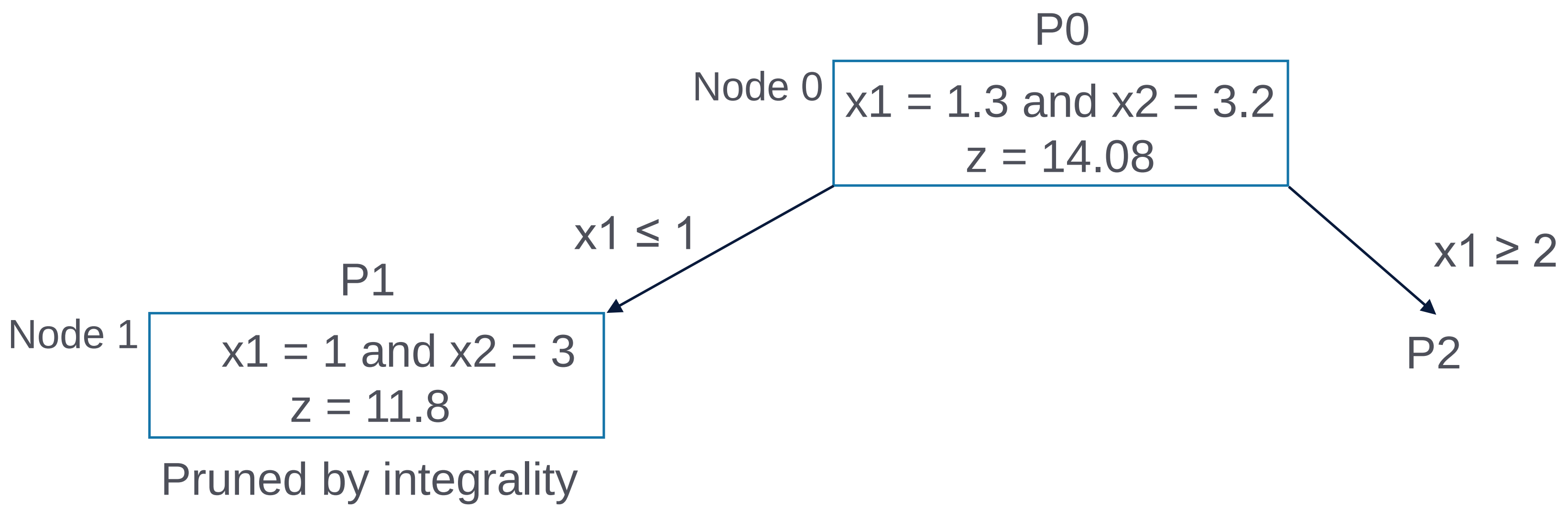
Branch-and-Bound method…4

- The branch-and-bound implicit enumeration tree is:



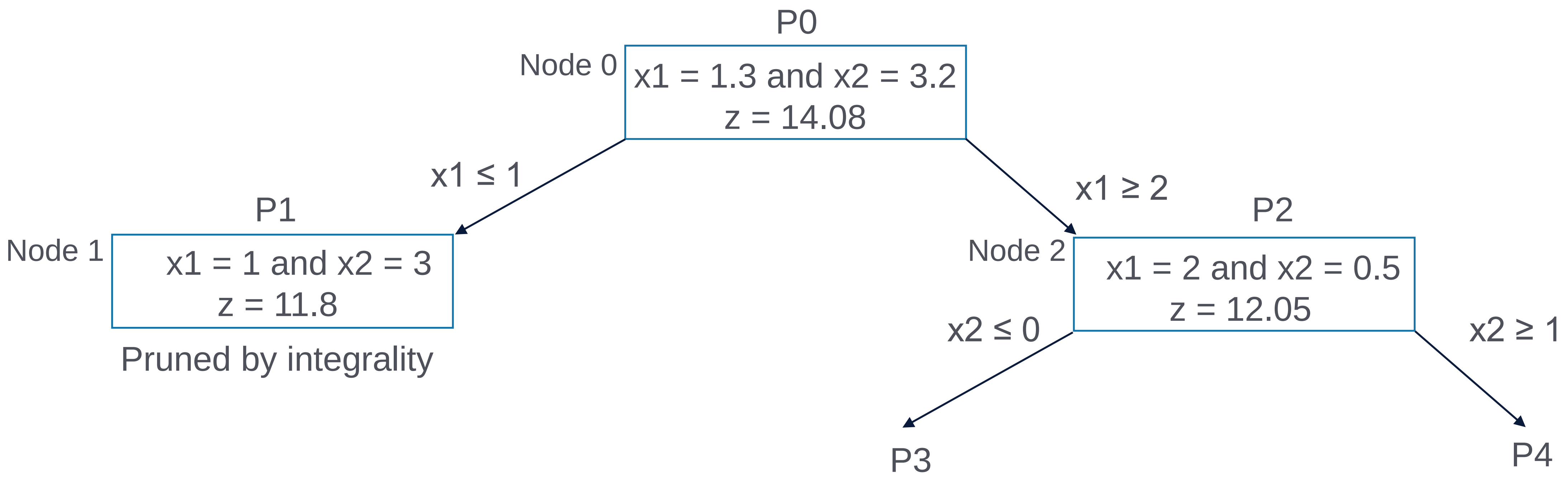
Branch-and-Bound Method…5

- The branch-and-bound implicit enumeration tree is:



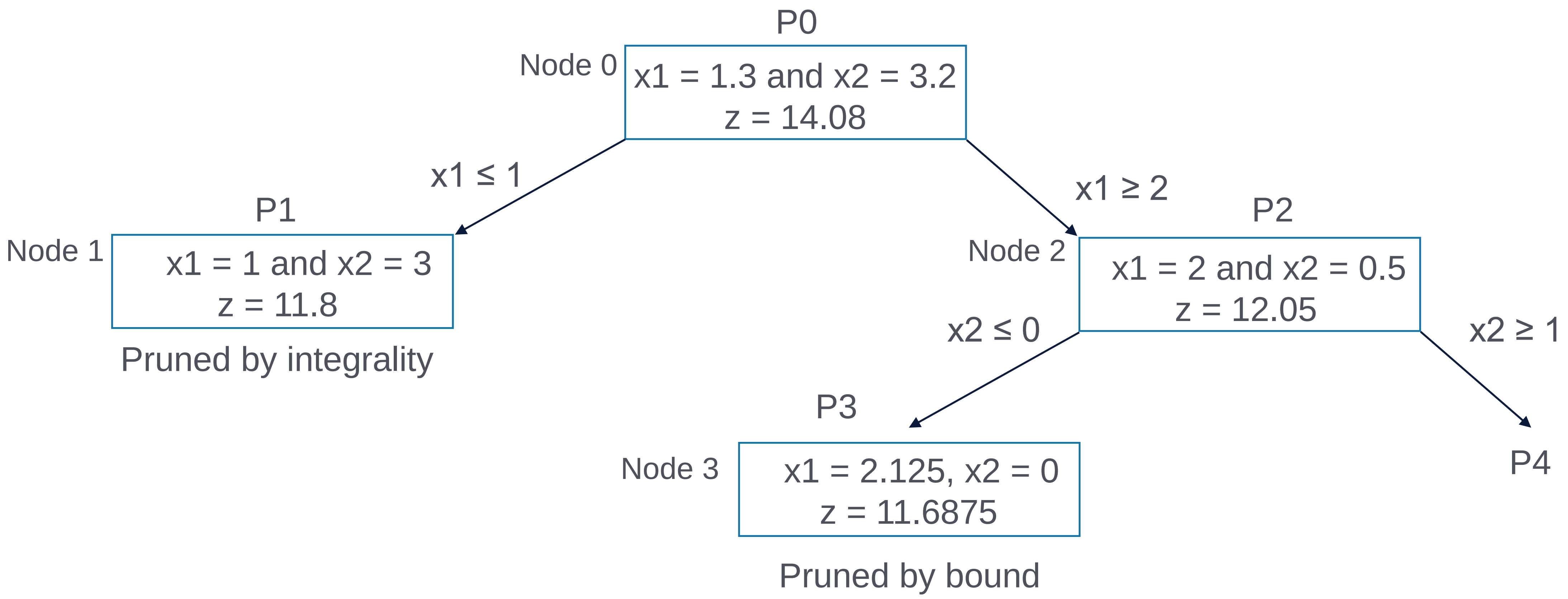
Branch-and-Bound Method…6

- The branch-and-bound implicit enumeration tree is:



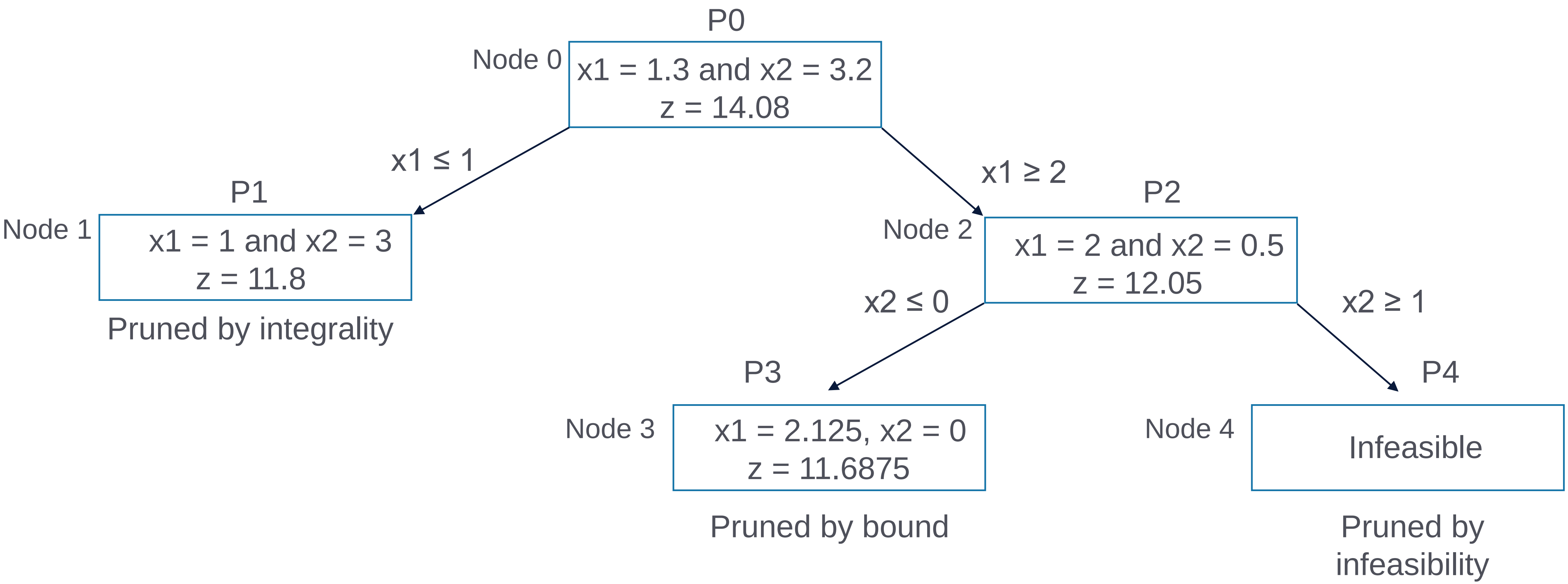
Branch-and-Bound Method…7

- The branch-and-bound implicit enumeration tree is:



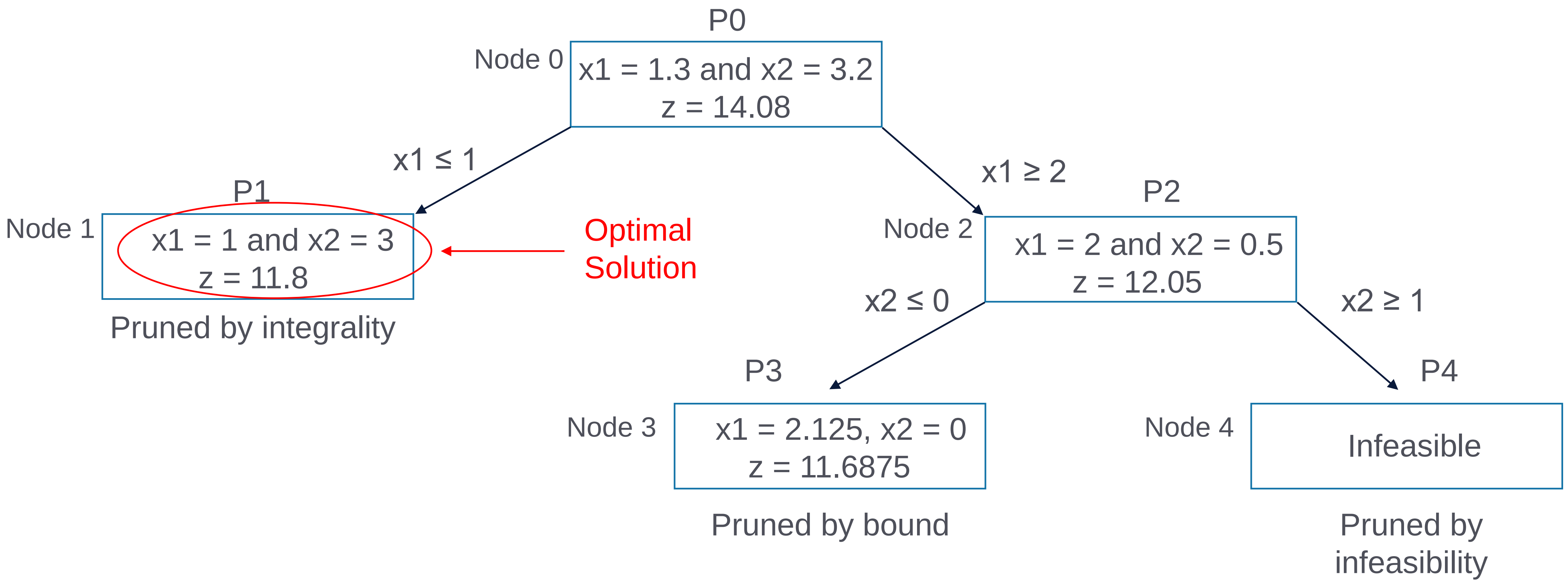
Branch-and-Bound Method...8

- The branch-and-bound implicit enumeration tree is:



Branch-and-Bound Method...9

- The branch-and-bound implicit enumeration tree is:



Branch-and-Bound Approach: Final Remarks...1

- The branch-and-bound algorithm is based on two principles.
    - **Branching**: partitions the set of solutions of the original MIP into disjoint subproblems.
    - **Bounding**: prunes the implicit enumeration tree.
- The optimal objective function value of any subproblem in the tree is checked against the current best upper bound of the original MIP and the current best lower bound of the original MIP.

Branch-and-Bound Approach: Final Remarks...2

- The process stops when there are no more subproblems to explore.
- The optimal solution of the MIP problem is the integer solution corresponding to the subproblem with the current best objective function values.
- Notice that the current best lower bound and current best upper bound of the original MIP objective function value give us information about how close we are to finding the optimal solution of the original MIP.
- This is called the optimality gap.

Approach 2: Cutting Planes Methods for Solving MIP Problems

Reference book: *Integer Programming,* by Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli

Cutting Planes Approach

- MIP problem

$$Max\ z = 5.5x_1 + 2.1x_2$$

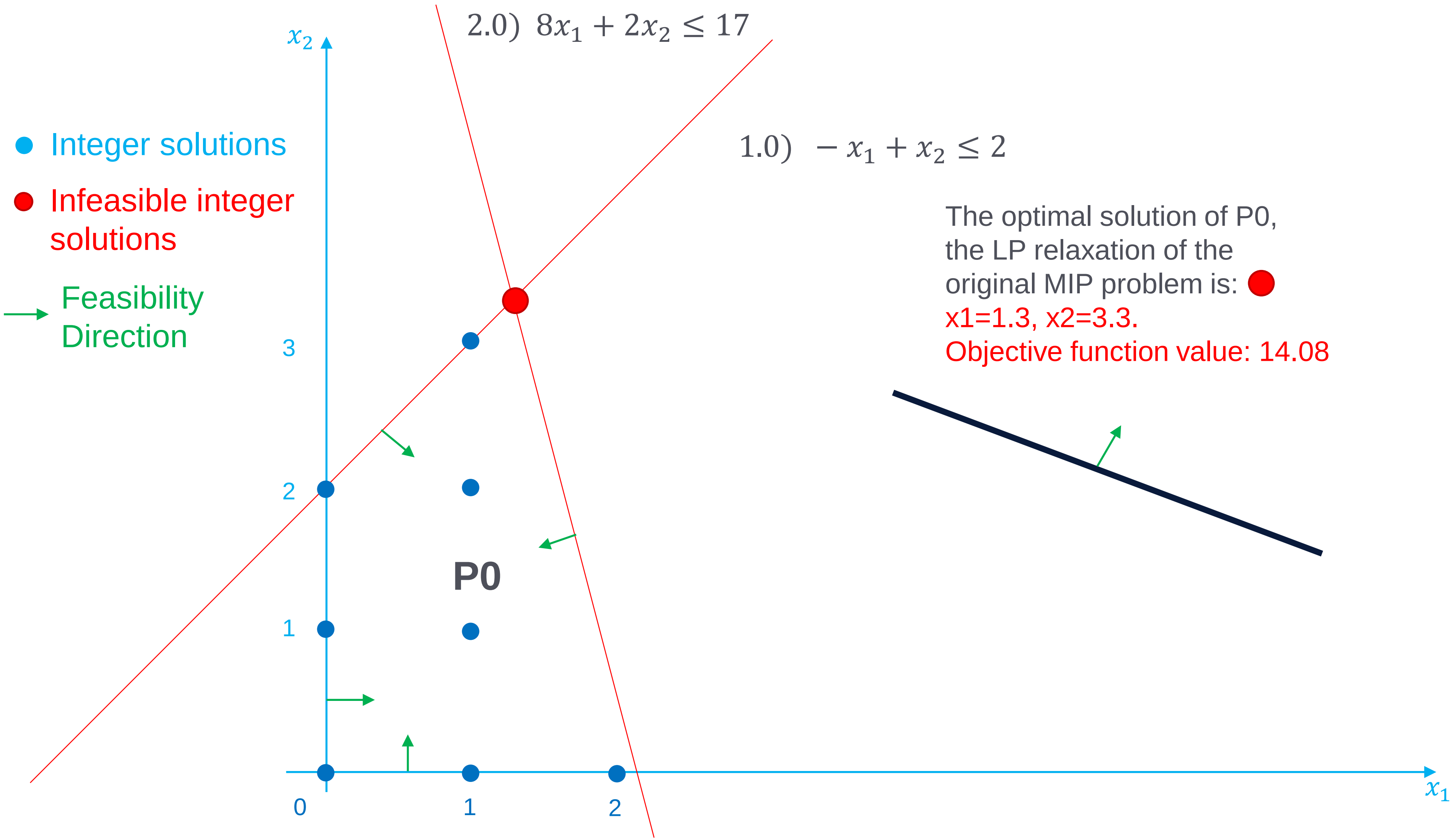
Subject to:

$$-x_1 + x_2 \leq 2$$
$$8x_1 + 2x_2 \leq 17$$
$$x_1, x_2 \geq 0 \text{ and integer}$$

- P0 = LP relaxation of MIP problem (i.e., the integrality constraints are removed).

- Similar to branch-and-bound, but rather than imposing restrictions on the fractional variables, we generate a series of constraints called Gomory cuts, add them to the LP relaxation model P0, and re-solve. Cuts do not eliminate integer solutions of the MIP problem.

Graphical Representation of P0 Problem

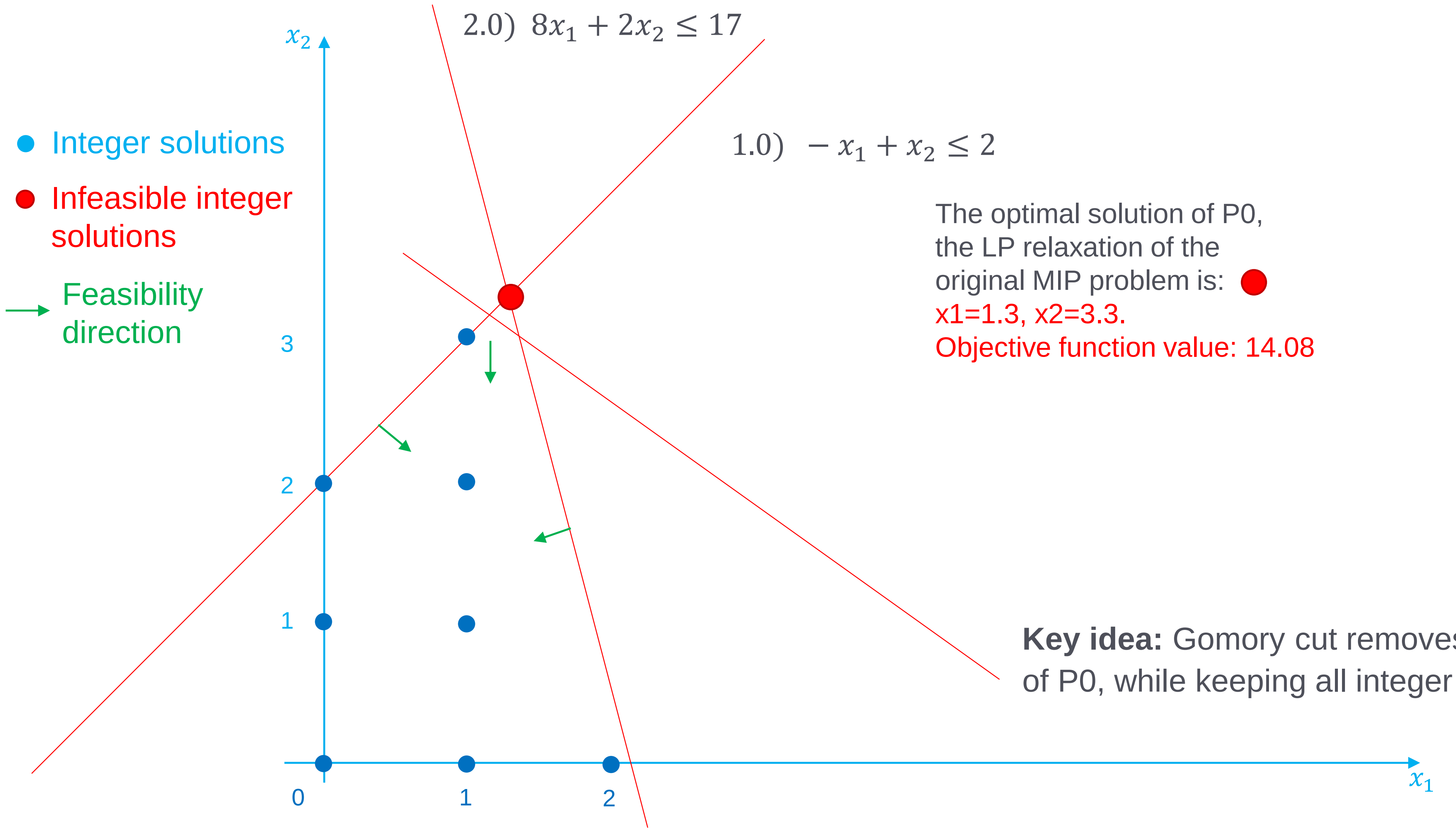


The optimal solution of P0, the LP relaxation of the original MIP problem is:

x1=1.3, x2=3.3.

Objective function value: 14.08

Graphical Representation of P0 Problem…2



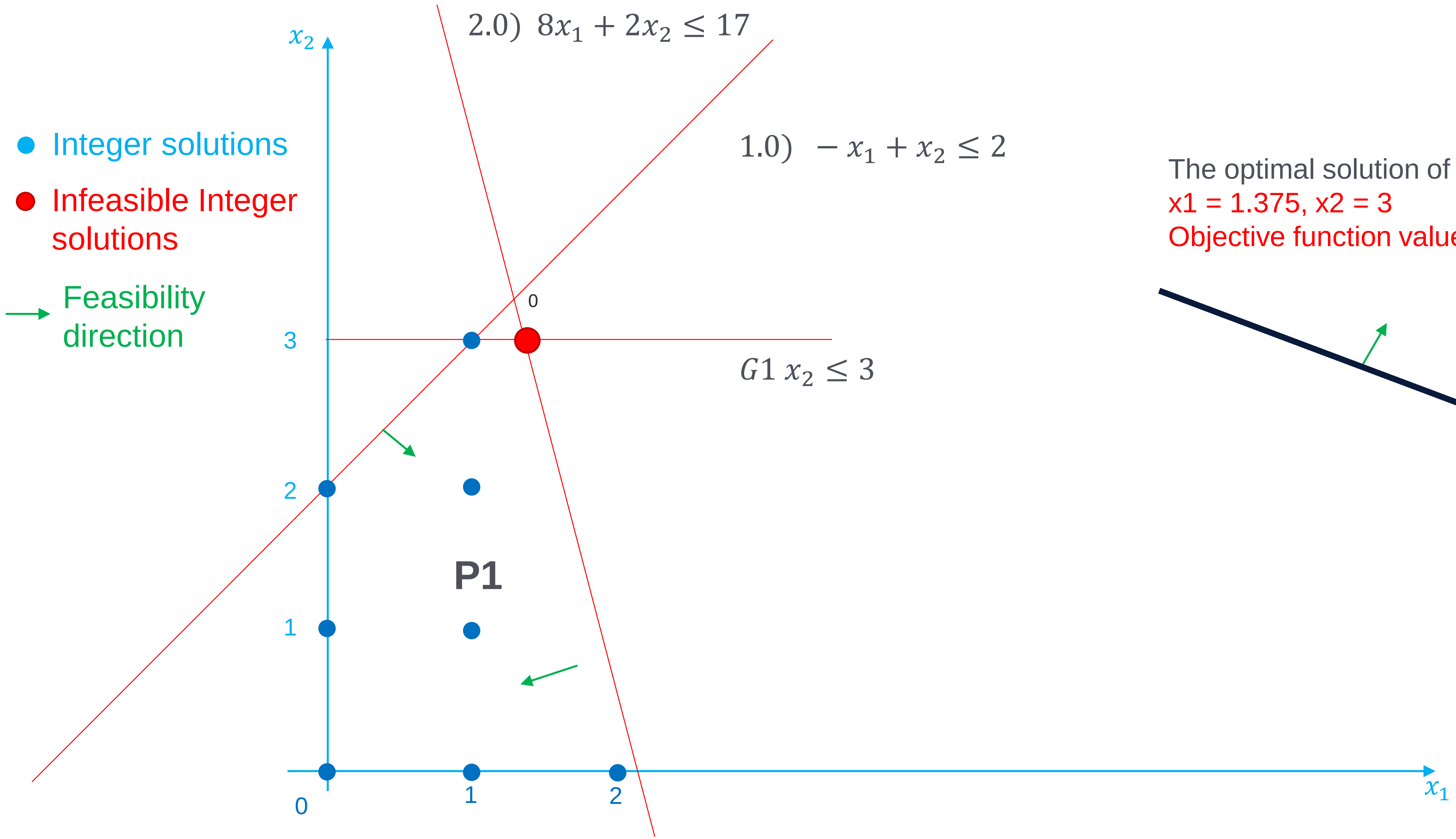
The optimal solution of P0, the LP relaxation of the original MIP problem is:

x1=1.3, x2=3.3.

Objective function value: 14.08

Key idea: Gomory cut removes fractional optimal solution of P0, while keeping all integer solutions of P0.

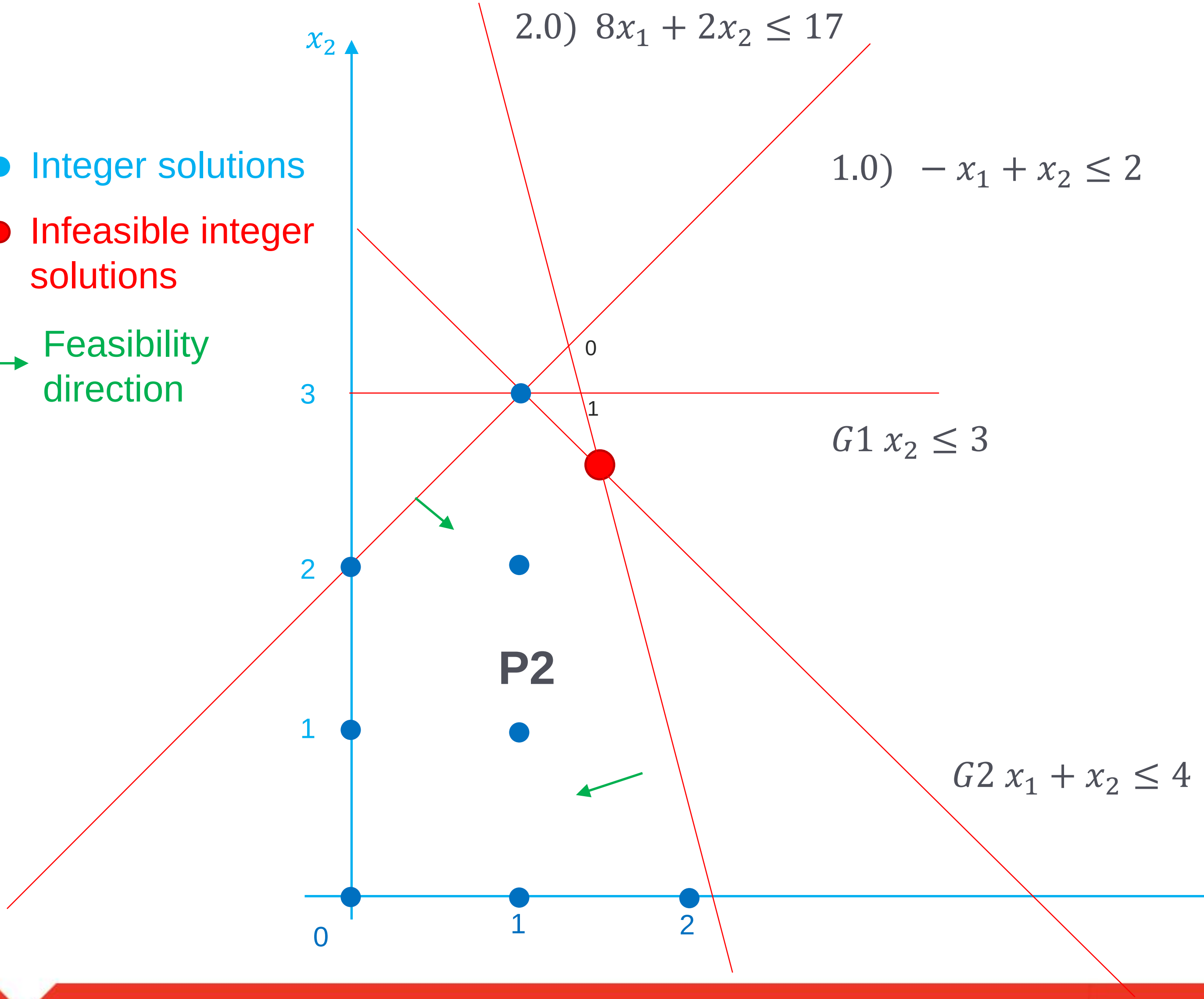
First Gomory Cut for MIP Problem



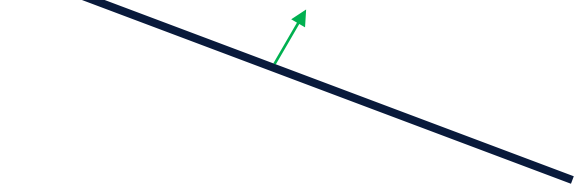
The optimal solution of P1 is:
x1 = 1.375, x2 = 3
Objective function value: 13.8625

Second Gomory Cut for MIP Problem

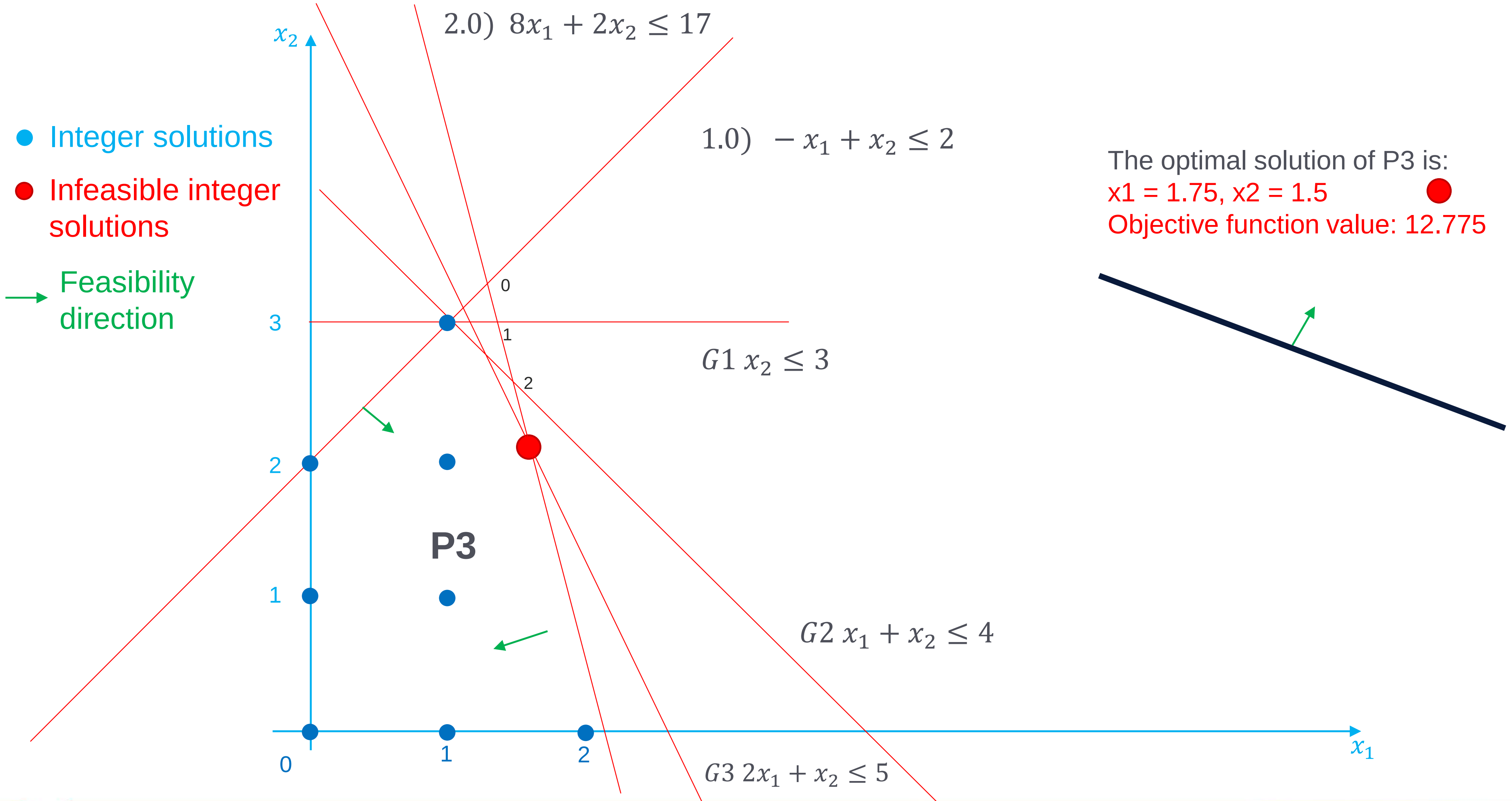
- Integer solutions
- Infeasible integer solutions
- Feasibility direction



The optimal solution of P2 is:
x1 = 1.5, x2 = 2.5
Objective function value: 13.5

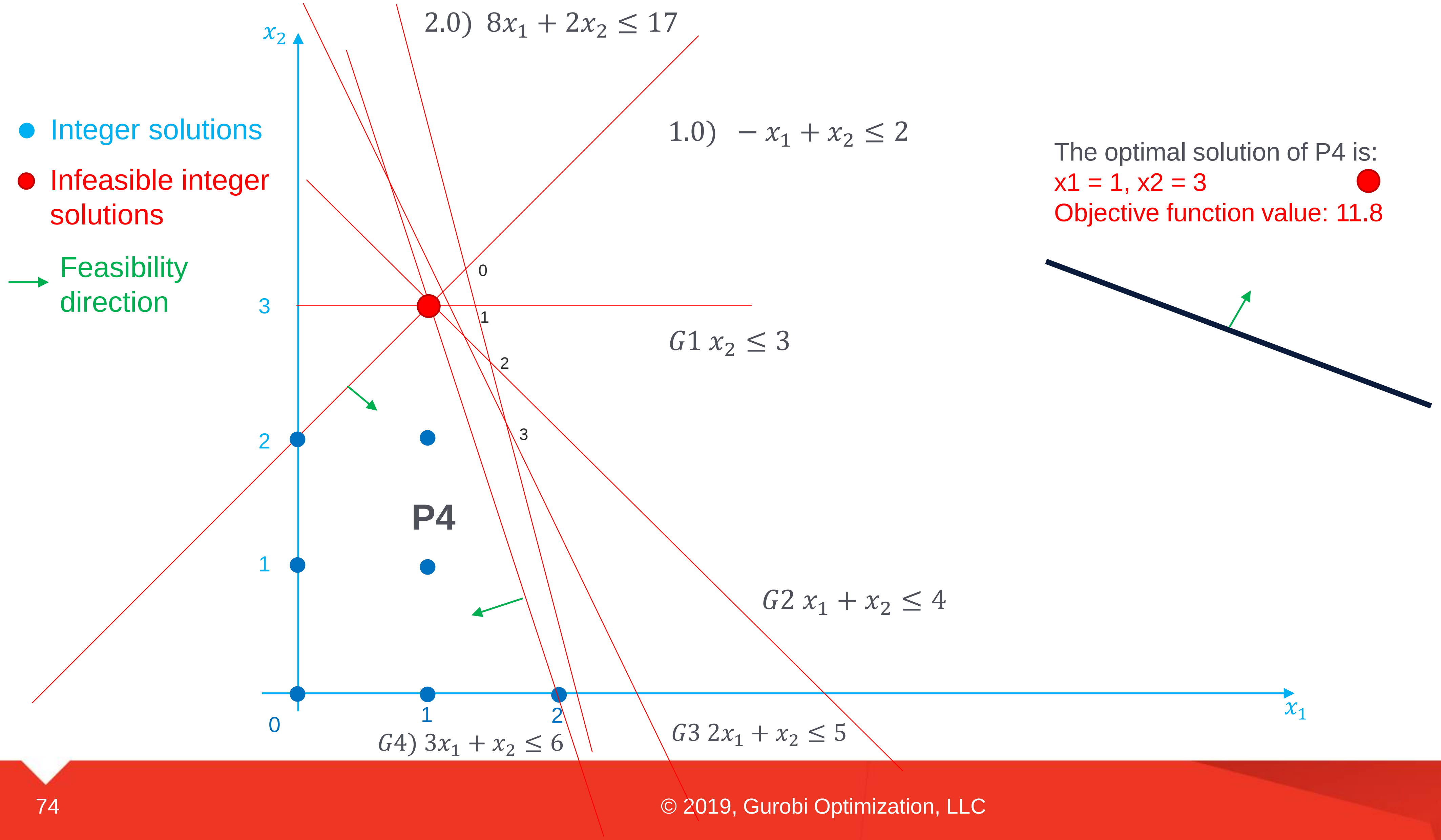


Third Gomory Cut for MIP Problem



The optimal solution of P3 is:
x1 = 1.75, x2 = 1.5
Objective function value: 12.775

Fourth Gomory Cut for MIP Problem

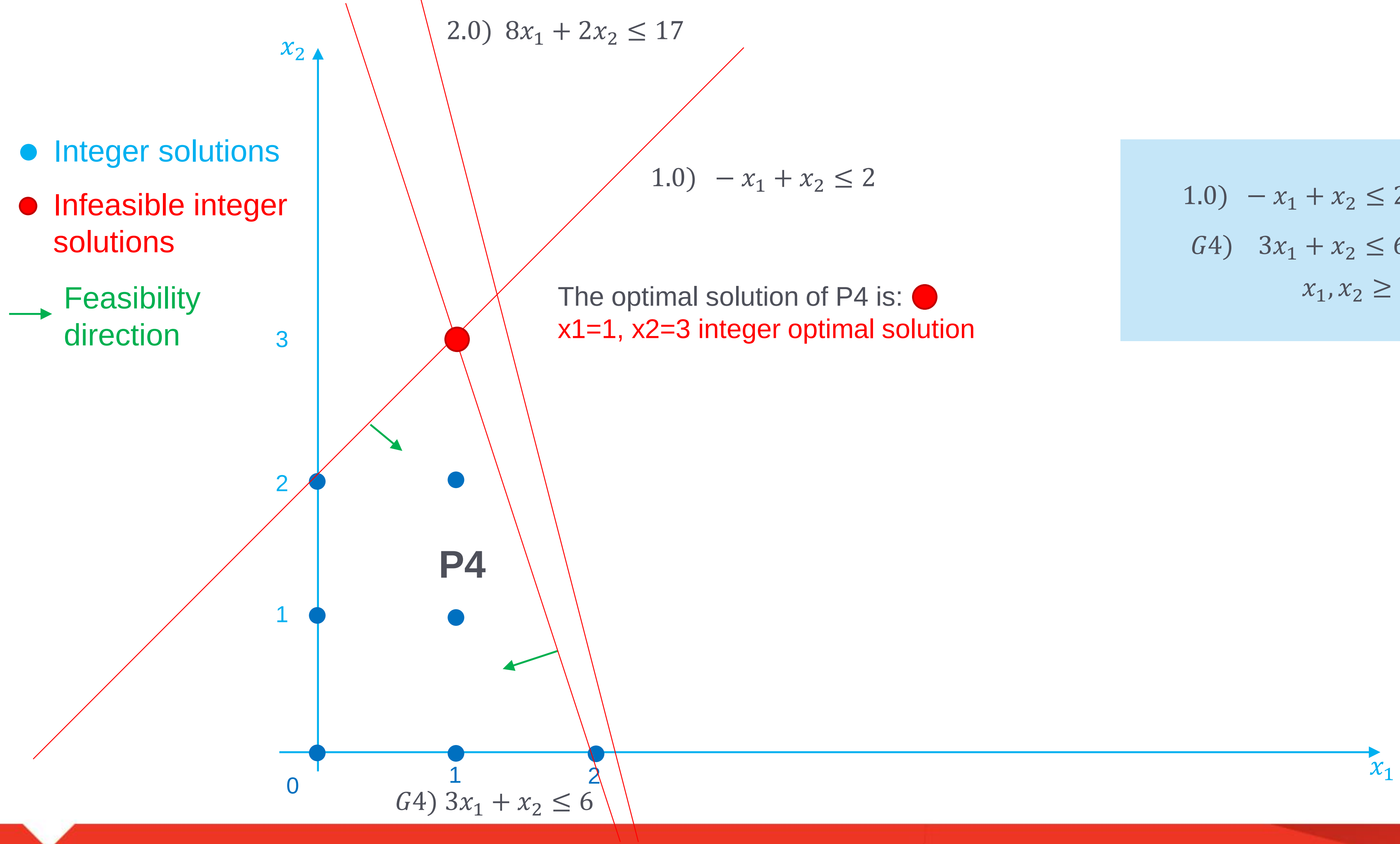


The optimal solution of P4 is:

x1 = 1, x2 = 3

Objective function value: 11.8

Solution Space of MIP Problem



The optimal solution of P4 is:

x1=1, x2=3 integer optimal solution

$$1.0)\quad -x_1 + x_2 \leq 2$$
$$G4)\quad 3x_1 + x_2 \leq 6$$
$$x_1, x_2 \geq 0$$

Branch-and-Cut Approach

- This approach combines the branch-and-bound approach with the cutting planes approach.
- The cutting planes approach is applied before the branching step in the branch-and-bound approach.
- Frequently multiple rounds of cuts are added at the root problem P0, while fewer or no cuts might be generated deeper in the branch-and-bound implicit enumeration tree.

Final Remarks

- Gurobi users formulate MIP problems that are solved by the Gurobi callable library.
- The mathematics and computer science behind Gurobi algorithms are cutting-edge.
- Gurobi has world-class experts in mathematical optimization to solve complex and high-value combinatorial optimization business problems.
- Gurobi has the best performance in the market.

Why MIP Is Better than Simple Heuristics ?

Demo

Conclusions

What Have We Covered in This MIP Tutorial?

- Why mixed integer programming (MIP) problems and solvers are important.
- Introduced the Resource Assignment Problem (RAP).
  - Demonstrated that simple heuristics can lead to poor assignments. We discussed that enumerating all possible assignments is practically impossible, since the number of possible assignments is astronomical.
- Formulated the RAP problem as a linear programming (LP) problem.
- Implemented the LP formulation of the RAP using the Gurobi Python API.
- Gave a demo in Jupyter Notebook of the RAP problem implementation using the Gurobi Python API.
- Discussed that the LP formulation of the RAP always leads to integer solutions.
  - Added a constraint to the RAP LP formulation that destroyed RAP problem's special structure that always leads to integer solutions.

What Have We Covered in This MIP Tutorial?...Continued

- The branch-and-bound approach to solving MIP problems.
- The cutting planes approach to solving MIP problems.
- The branch-and-cut approach that combines the branch-and-bound and cutting planes approaches.
- Why MIP is better than simple heuristics in the context of the RAP problem with a budget constraint.

Summary

- George Dantzig, one of the founders of mathematical programming, said:
  - Mathematical optimization is a methodology that entails three steps:
    - The formulation of a real problem as a mathematical model.
    - The development of algorithms to solve those mathematical models.
    - The use of software and hardware to run these algorithms and develop mathematical programming applications.

- The Gurobi Optimizer is the leading mathematical programming solver and incorporates state-of-the-art algorithms to tackle mathematical optimization models.

- Gurobi Optimization is developing a multitude of modeling examples of mathematical programming applications with the potential to serve as templates to solve optimization problems in many industries - including certain classes of machine learning problems.

Model Building Mathematical Programming

An Overview

GUROBI OPTIMIZATION

The World's Fastest Solver

Thank You – Questions?

GUROBI OPTIMIZATION

The World's Fastest Solver