COMP9334: Capacity Planning of Computer Systems and Networks

Week 8A: Optimisation (2): Integer programming

Optimisation

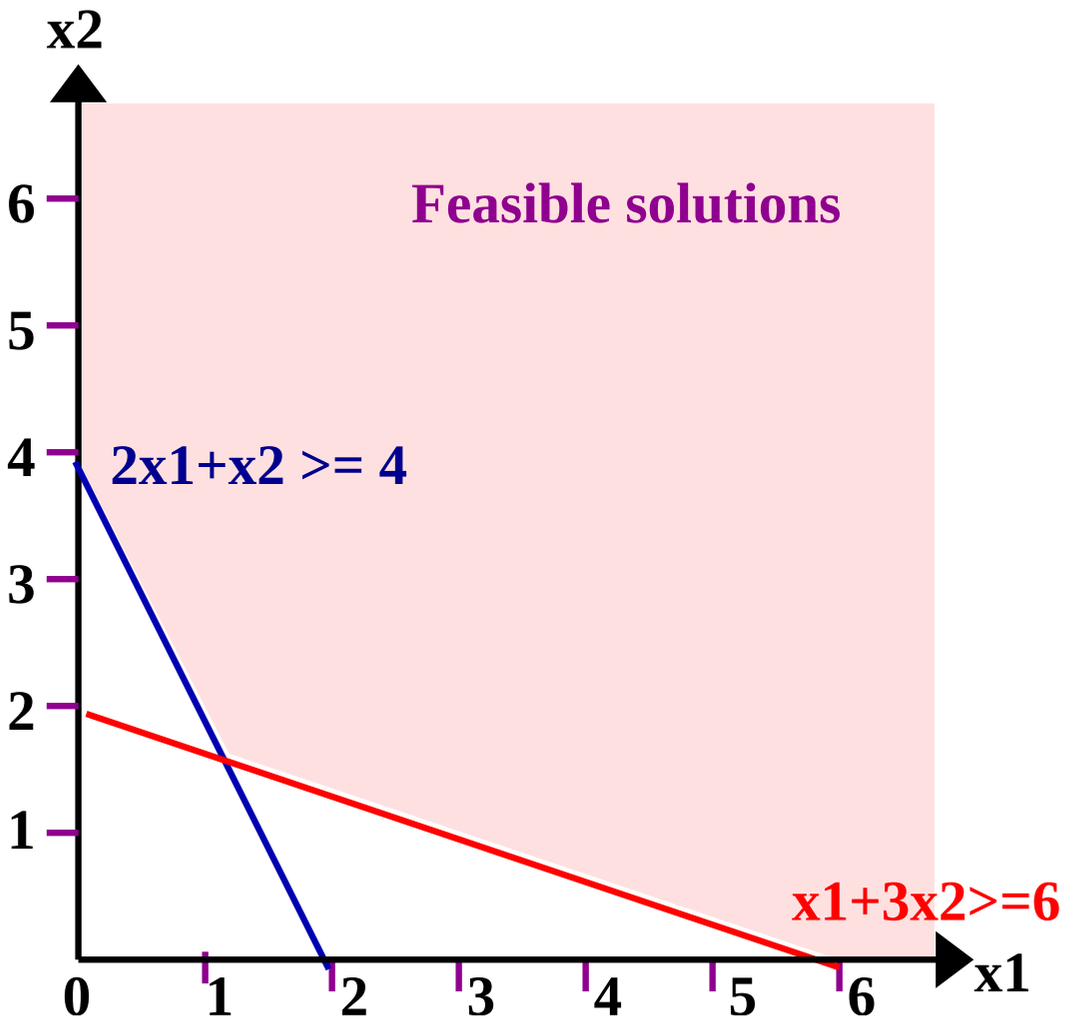
- What have we covered so far?
  - Formulation of linear programming (LP) problems
  - Using AMPL to solve LP problems
- This lecture
  - Different types of LP problems
  - Formulation of integer programming (IP) problems
  - Using AMPL to solve IP problems

Linear programming

- Linear programming (LP) is a tool for solving a type of optimization problems where
  - Decision variables are real numbers
  - Objective function is linear in the decision variables
  - All the constraints are linear in the decision variables
- LP problems with 2 variables have a nice graphical solution, e.g.

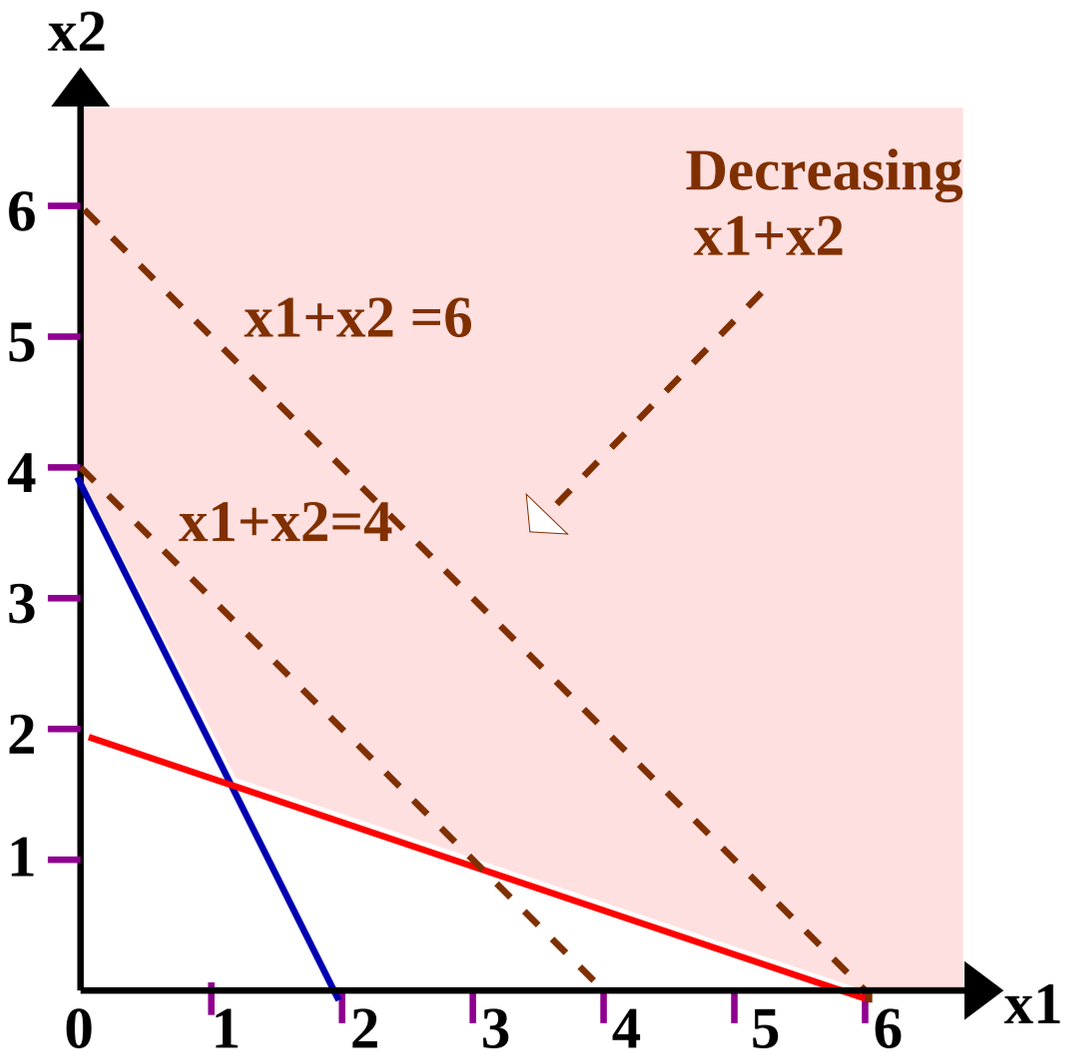
$$
\begin{array}{rl}
\text{min} & x_1 + x_2 \\
\text{subject to} & 2x_1 + x_2 \geq 4 \\
& x_1 + 3x_2 \geq 6 \\
& x_1, x_2 \geq 0
\end{array}
$$

Feasible solutions



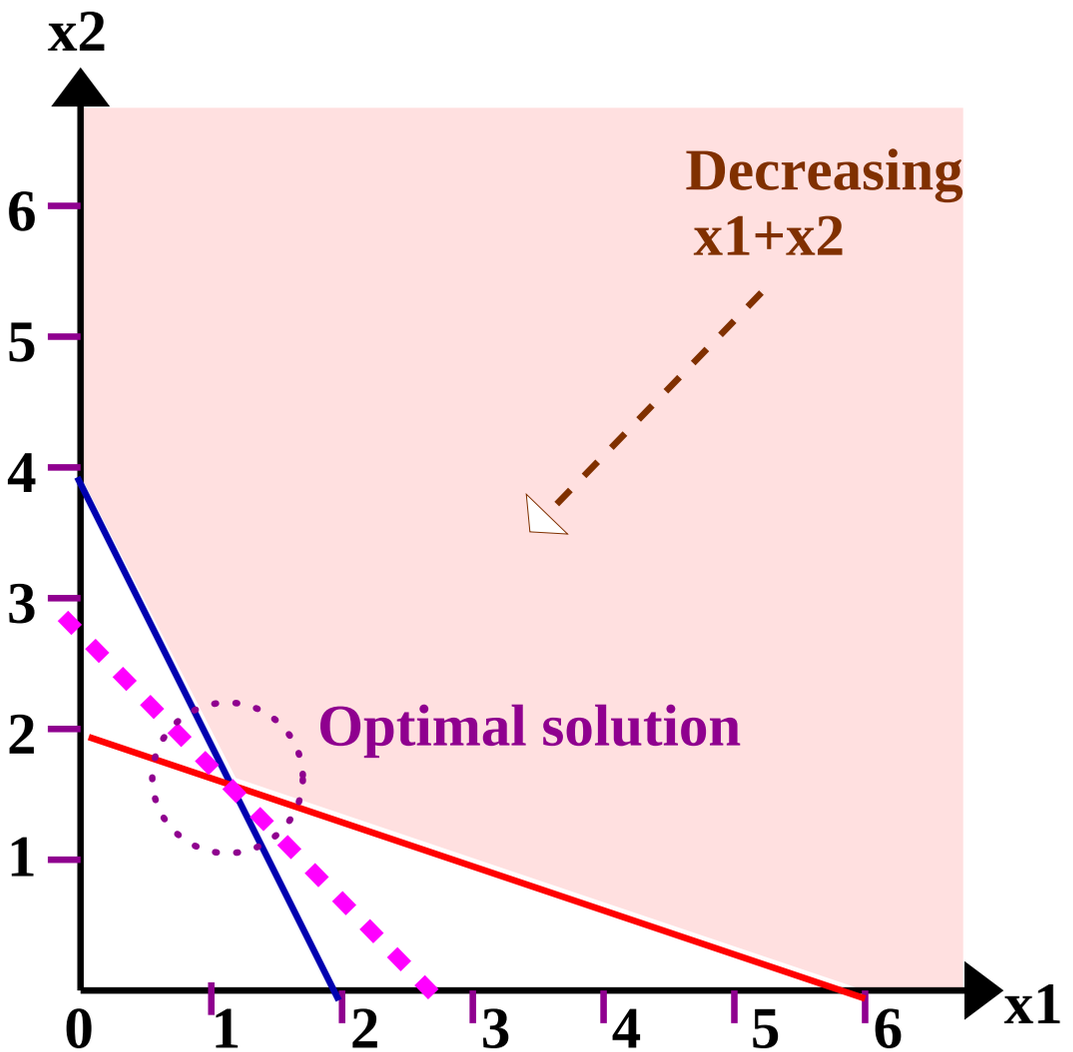
- Any (x_1, x_2) that satisfies all the constraints is a feasible solution. Otherwise, it is an infeasible solution
- For LP problems with 2 variables, the feasible region is a polygon
- In general, it is a polyhedron

Objective function



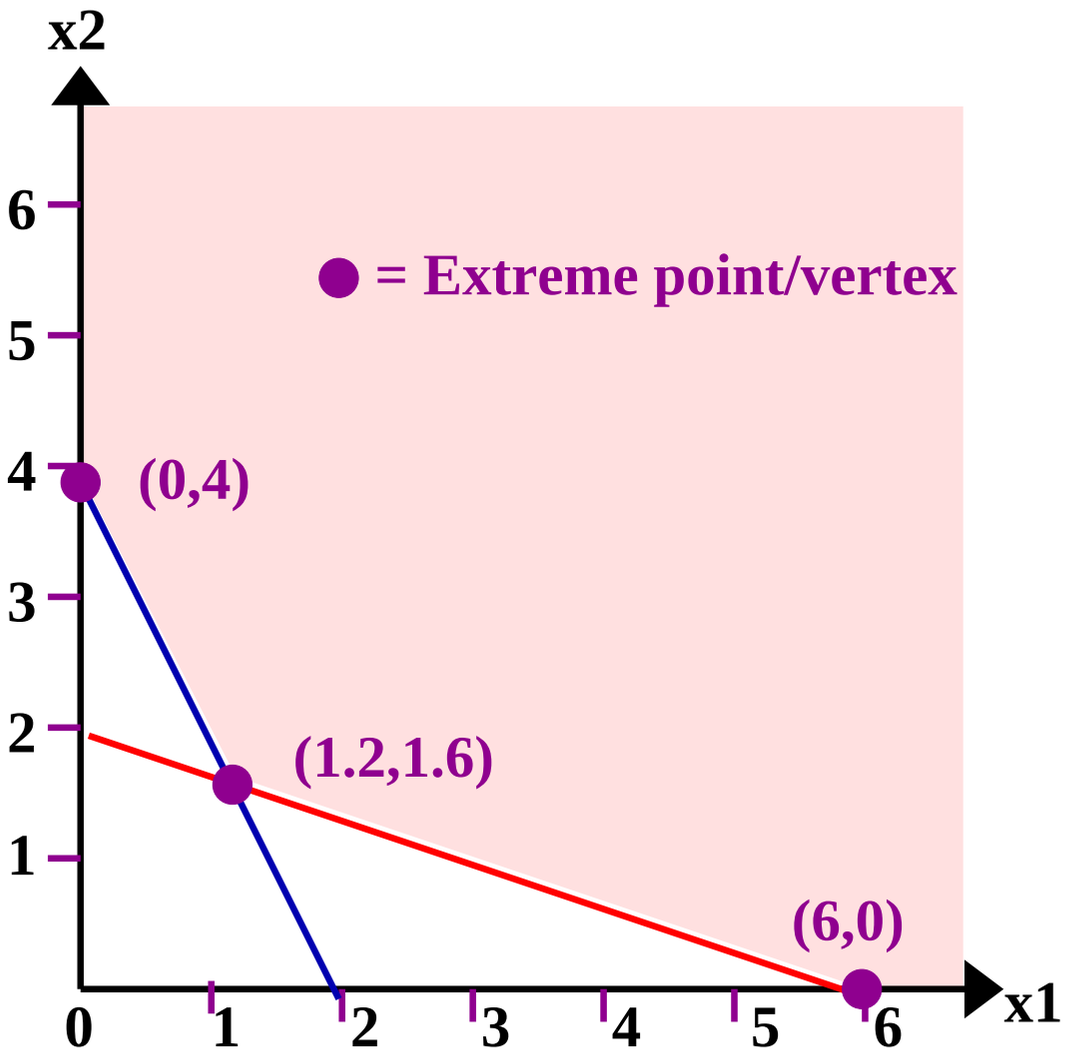
- Since the aim is to minimize $x_1 + x_2$, we look for contour of $x_1 + x_2$

Optimal solution



- Optimal solution of this LP problem is $x_1 = 1.2$, $x_2 = 1.6$, with objective function value = 2.8

Extreme point



- Regardless of the objective function, the optimal solution of an LP problem (if it exists) must be at one of the extreme points or vertices of the polyhedron formed by the constraints

Special cases: Multiple optimal solutions

- What is the optimal solution to the following LP problem?

$$
\begin{aligned}
\text{min} \quad & x_1 + x_2 \\
\text{subject to} \quad & x_1 + x_2 \geq 4 \\
& x_1 + 3x_2 \geq 6 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

Special cases: Infeasible LP

- What is the optimal solution to the following LP problem?

$$
\begin{aligned}
\text{min} \quad & x_1 + x_2 \\
\text{subject to} \quad & x_1 + 3x_2 \leq 4 \\
& x_1 + 3x_2 \geq 6 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

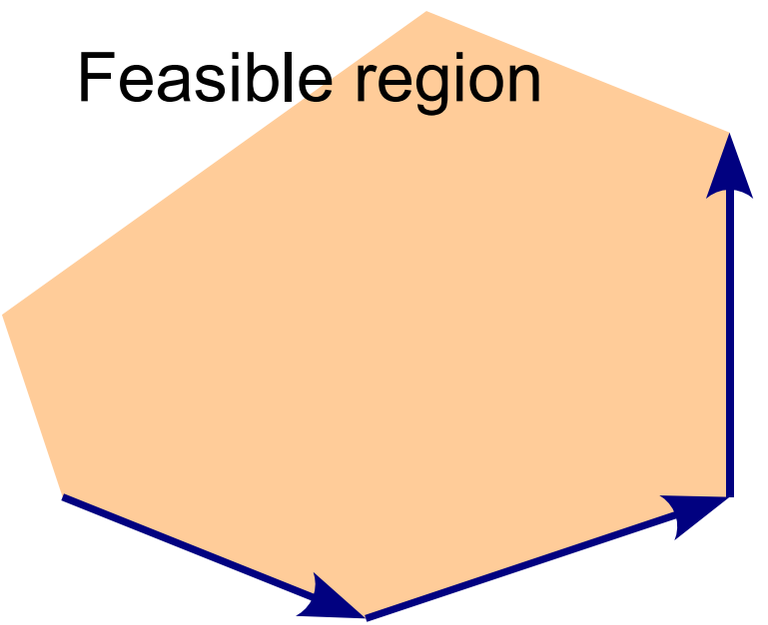
Special cases: Unbounded LP

- What is the optimal solution to the following LP problem?

$$
\begin{array}{rl}
\text{max} & x_1 + x_2 \\
\text{subject to} & 2x_1 + x_2 \geq 4 \\
& x_1 + 3x_2 \geq 6 \\
& x_1, x_2 \geq 0
\end{array}
$$

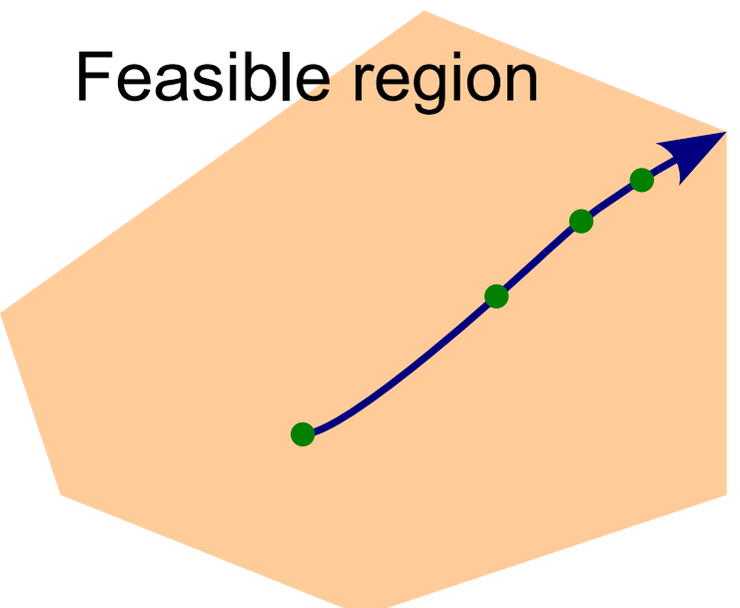
Algorithms for LP

- Simplex method (Dantzig 1947)
  - Found in many LP software
  - Principle: Move from an extreme point to another
  - Worst-case complexity: Exponential
  - Average performance on practical LP problems: Very good



Algorithms for LP (cont.)

- Interior-point method (Karmarkar 1984)
  - Very competitive compared with the simplex method
  - Principle: Move from an interior point to another
  - Worst-case complexity: Polynomial
  - The algorithm for large LP problems



Motivating example 2: Cloud computing

- Now, each resource further charges a set up cost of fixed amount
  - Resource 1: Set up cost = 200 dollars
  - Resource 2: Set up cost = 100 dollars
  - Resource 3: Set up cost = 50 dollars
- Again, the computation job has the following requirements:
  - Requires 10^7 million cycles
  - Completion time $\leq$ 4,800 sec
  - Cost $\leq$ 1,500 dollars
  - Minimize the completion time
- Problem faced by the job:
  - From which resource should we buy the computing power?
  - How many cycles to buy from each chosen resource?

Yes-or-no decision

- What is the cost of buying cycles from a chosen resource?
- Yes-or-no questions: Is Resource i chosen?
  - E.g. is Resource 3 chosen?
    - Yes $\Rightarrow x_3 > 0 \Rightarrow$ Cost = $2000 \times x_3 + 200 \times 1$
    - No $\Rightarrow x_3 = 0 \Rightarrow$ Cost = $2000 \times x_3 + 200 \times 0$
- This type of optimization problems involves 0-or-1 logical decisions among other decisions
- You will learn in the rest of this lecture
  - How to formulate this type of optimization problems
  - How to solve this type of optimization problems

Formulating optimization problem

- Given:
  - n resources
  - Resource i offers computing power at a speed of p_i million cycles/sec with cost c_i dollars/sec
  - Set up cost for using Resource i is s_i dollars
  - Customer requires N million cycles
  - Completion time $\leq T_{\max}$
  - Cost $\leq C_{\max}$
- Decision variables:

$$
\begin{aligned}
y_i &= \begin{cases} 1 & \text{if Resource } i \text{ is chosen} \\ 0 & \text{otherwise} \end{cases} \\
x_i &= \text{fraction of the job allocated to Resource } i \\
T &= \text{completion time, which is } \max_i \frac{N x_i}{p_i}
\end{aligned}
$$

- Exercise: Express the cost of using Resource i in terms of c_i, N, x_i, p_i, s_i, y_i

Formulating optimization problem (cont.)

- Cost:

$$C = \sum_{i=1}^{n} \left(\frac{c_i N x_i}{p_i} + s_i y_i \right)$$

- Constraint: Cannot have cycles from Resource i if it is not chosen

$$x_i \leq y_i, \quad i = 1, 2, \ldots, n$$

  - Note that we require $x_i \geq 0$, thus
    - Exercise: If $y_i = 0$, what value(s) can x_i take?
    - Exercise: If $y_i = 1$, what value(s) can x_i take?

Formulating optimization problem (cont.)

- The problem formulation is

$$\min \quad T$$

subject to

$$\begin{aligned}
T &\geq \frac{Nx_i}{p_i}, \quad i = 1, 2, \ldots, n \\
T &\leq T_{\max} \\
\sum_{i=1}^{n} \left(\frac{c_i N x_i}{p_i} + s_i y_i \right) &\leq C_{\max} \\
x_i &\leq y_i, \quad i = 1, 2, \ldots, n \\
\sum_{i=1}^{n} x_i &= 1 \\
x_i &\geq 0, \quad i = 1, 2, \ldots, n \\
y_i &\in \{0, 1\}, \quad i = 1, 2, \ldots, n
\end{aligned}$$

Integer programming

- LP in which some or all decision variables can only take non-negative integer values
- Specific type of integer programming (IP):
  - Mixed integer programming (MIP): Some decision variables are integers, others are real
  - Pure integer programming: All decision variables are integers
  - Binary integer programming: All decision variables must be either 0 or 1

LP relaxation

- Relaxing the integer constraints may not give you the right solution

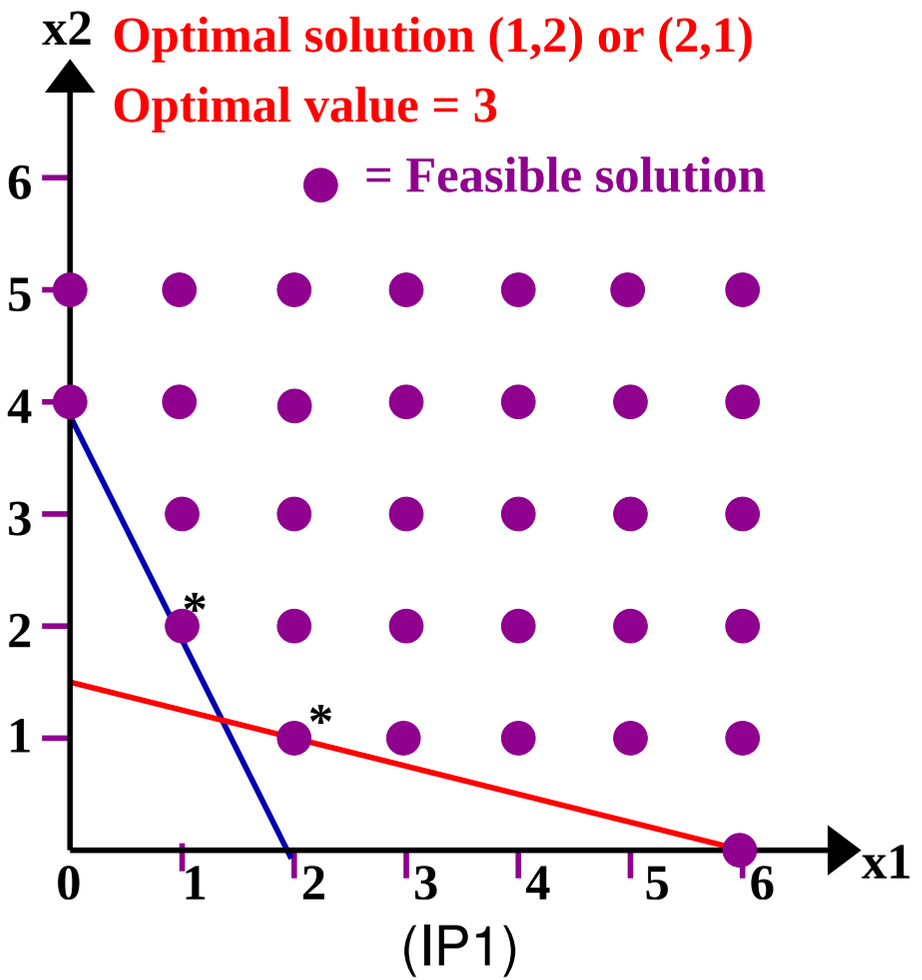
IP problem (IP1)

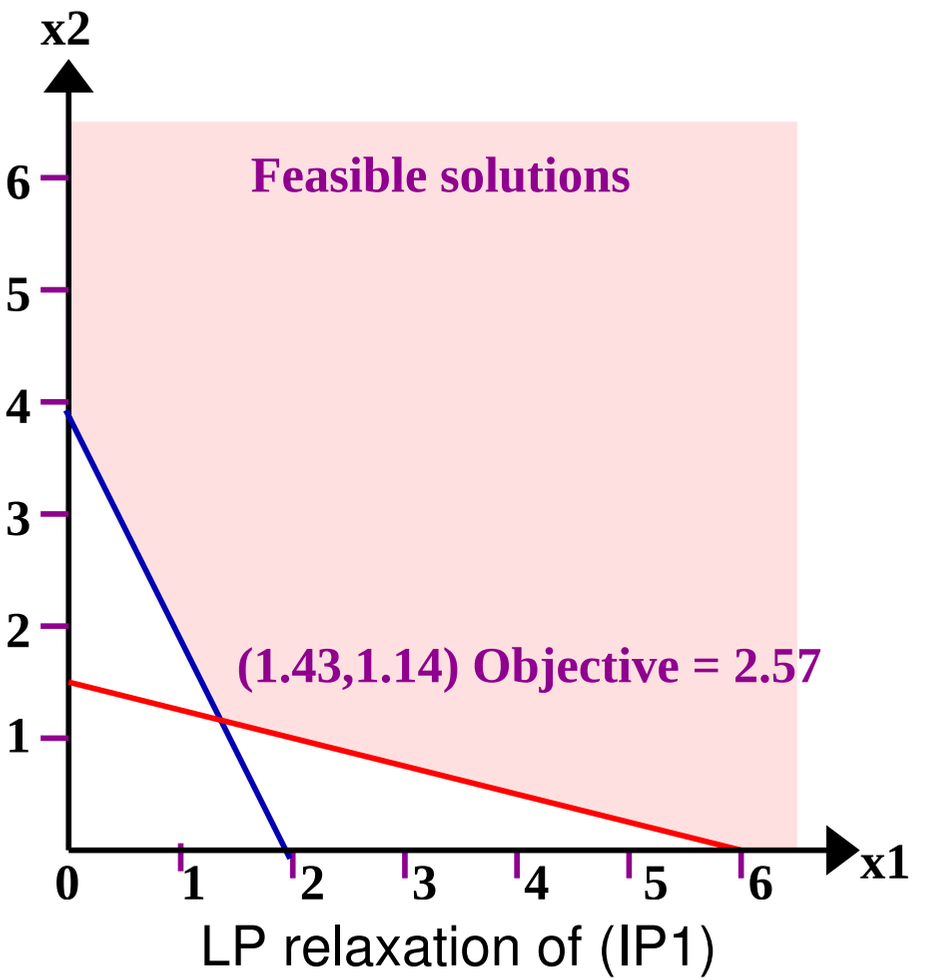
$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \geq 4 \\ & x_1 + 4x_2 \geq 6 \\ & x_1, x_2 \geq 0, \text{ integer} \end{aligned}$$

LP relaxation of (IP1)

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{subject to} \quad & 2x_1 + x_2 \geq 4 \\ & x_1 + 4x_2 \geq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Feasible solutions





- Question: Can we round the LP relaxation solution to obtain a solution to (IP1)?

Results and observations

- Rounding the LP relaxation solution may produce an infeasible solution for the original IP problem
- However, for minimization problems, it is always true that:
  Optimal value of an IP problem $\geq$ Optimal value of its LP relaxation
- If the LP relaxation gives an integer solution
  - The optimal solution to the LP relaxation is also an optimal solution to the original problem
  - This is true for some special classes of IP problems e.g. network flow problems

Solving IP exactly

- Complete enumeration will require a lot of computation
- A smarter way is to use the branch-and-bound method (you are encouraged to read Winston sections 9.3, 9.4 in your own time)
- In principle, the branch-and-bound method can find the optimal solution but practically it may take a very long time
  - Some IP problems with 60 variables may take hours to run

AMPL/CPLEX for solving example 2

- This is saved in the file `grid_mip.mod`

```
set COMP;   #Set of resources
param c {COMP}; #Cost
param p {COMP}; #Speed
param s {COMP}; #Set up cost
param Tmax;
param Cmax;
param N;
var x {i in COMP} >= 0;
var y {i in COMP} binary;
var T >= 0;
minimize time: T;
subject to T_constraint {i in COMP}: T >= N * x[i] / p[i];
subject to Tmax_constraint: T <= Tmax;
subject to Cmax_constraint: sum {i in COMP}
    (N * c[i] * x[i] / p[i] + s[i] * y[i]) <= Cmax;
subject to x_sum: sum {i in COMP} x[i] = 1;
subject to restriction {i in COMP}: x[i] <= y[i];
```

AMPL/CPLEX for solving example 2 (cont.)

- Results from the solver

```
CPLEX 12.6.0.0: optimal integer solution; objective 3333.33333
5 MIP simplex iterations
0 branch-and-bound nodes
x [*] :=
COMP_1  0.333333
COMP_2  0.666667
COMP_3  0
;
y [*] :=
COMP_1  1
COMP_2  1
COMP_3  0
;
T = 3333.33

sum{i in COMP} (N*c[i]*x[i]/p[i] + s[i]*y[i]) = 1466.67
```

Acknowledgment

- Grid computing example based on Menascé and Casalicchio, “QoS in computing”, IEEE Internet Computing, pp. 85–87, Jul./Aug. 2004.