JUNIA Grande école d'ingénieurs
HEI·ISEN·ISA

#Class 4

Mixed Integer Linear Programming

Amina El Yaagoubi
Samuel Deleplanque

October 2024

Before we start…

Some keywords

- Integer programming,
- Decision variables,
- (0-1) decision variables,
- Mixed Integer Programming,
- Graph representation,
- Assignment problem,
- Knapsack problem,
- Bin-Packing problem,
- LP relaxation,
- Location problem.

Let’s go back to “Transportation problem 3” of #Class2

Transportation problem 3

- Consider a market with I suppliers, J buyers, and K products that are bought and sold. Supplier i has a quantity of S_{ik} of product k and sells it for a price of A_{ik} euros per unit. Buyer j demands at most D_{jk} units of product k and is willing to purchase it for a price of B_{jk} euros per unit.
- In this market, you are responsible for matching the suppliers with the buyers: you gather all the price and offer information, and then you allocate the suppliers to the buyers. Your profit is the difference between the purchase price and the selling price for each unit of product k that supplier i sells to buyer j.
- Formulate the problem of maximizing profit as a mathematical program (generic model).

… More constraints?

- Now, suppose that the suppliers and buyers are not in the same location. The transport capacity between supplier i and buyer j is at most U_{ij} units. Additionally, you are obligated to pay for the transportation of all products sold by supplier i to buyer j, at a cost of C_{ij} euros per unit.
- Also, assume that $|I| \geq 3$ and $|J| \geq 2$, and that buyer $j = 2$ does not wish to purchase any products from supplier $i = 3$.
- Modify your generic model to account for these new constraints.

… What if ?

- But what if we can only produce an integer number of products?
  - Indivisible (discrete) quantities: We assume that the products cannot be fractional.
  - Let's consider the products as laptops.

- Write the new generic model.

Beyond Linear Programming :

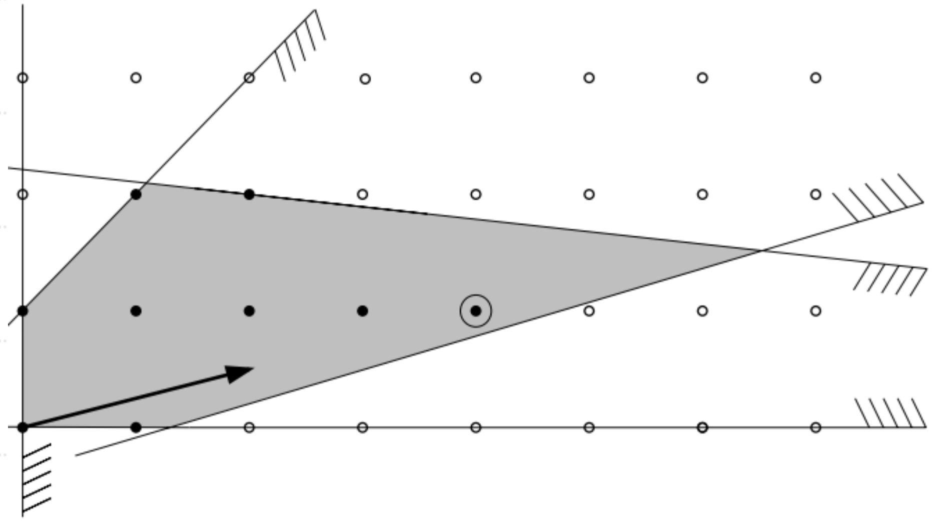
Integer (Linear) Programming (IP/ILP) and Mixed Integer (Linear) Programming (MIP/MILP)

Applicability of IP

- The real-world is often abrupt, unexpected, discontinuous, no-smooth, …
- Clearly, we can think of situations where it is only meaningful to make integral quantities of certain goods, for example, containers, laptops, electric vehicles, or use integral quantities of some resource, for example, employees.
- This is the point where (Mixed-)Integer (Linear) Programming ((M)I(L)P) comes to play!
- We might use an IP (sometimes known as discrete programming) model instead of an LP model.
  - The wide applicability of IP as a method of modeling is not obvious.
  - Integrality condition may arise from indivisibility (people, containers, chairs, …).
  - But it also can be used as a “trigger” or “switch”.
  - Logical conditions such as disjunctions, implications, precedence can be modeled using this tool.

What does it mean graphically?

- The set of all feasible solutions of an integer program is no longer a convex polyhedron, as was the case for linear programming, but it consists of separate integer points.
- Feasible solutions are shown as solid dots, and the optimal solution is marked by a circle.
- IP can be understood as the universal tool for modeling non-convexities and discontinuities.



More formally... IP (or ILP), MIP (or MILP)

- In (M)IP models, constraints and objective function are similar to LP models.

IP: All variables have to take on integer values in the final solution.

$$\begin{aligned} &Maximize\ \ c^T x \\ &s.t.\ \ Ax \le b \\ &x \in \mathbb{Z}^n \end{aligned}$$

Difference from LP models

MIP: Some variables have to take on integer values in the final solution.

$$\begin{aligned} &Maximize\ \ c^T x \\ &s.t.\ \ Ax \le b \\ &x \ge 0 \\ &x_i \in \mathbb{Z}, \forall i \in I \subset \{1, \dots, n\} \end{aligned}$$

- The constraint that enforces variables to take integer values is called the **integrality** (or integrity) constraint.
- When this constraint is removed, it's referred to as a relaxed problem or continuous relaxation, and then you have a linear optimization problem (LP).
- **LP has no integrality constraints**.

What changes in .lp files for IP models?

The names of all variables which are to be restricted to general integer values, separated by at least one space.

```
Minimize/Maximize
 obj: ....
Subject To
  c1: ....
  c2: ....
   ....
Bounds (optional)
   .......
General
   ........
End
```

Objective function section

Constraints section

Bounds section (optional)

General section

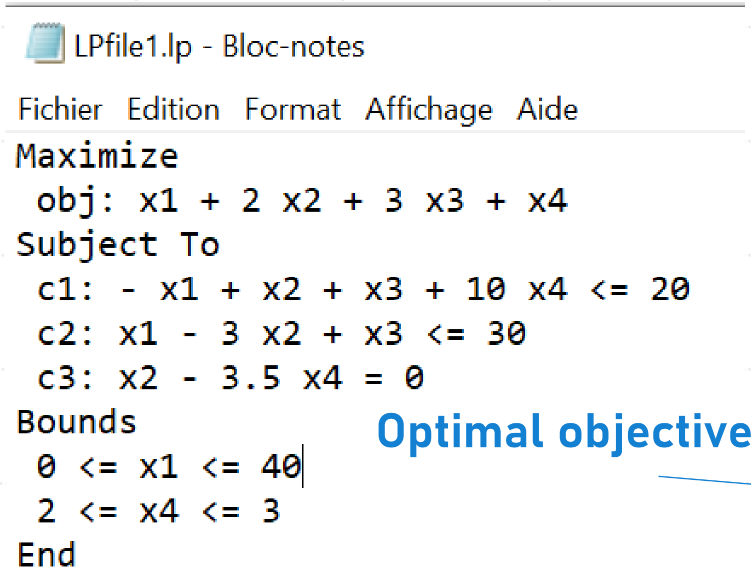
Example (1/2)

$$\begin{array}{ll} Max & x_1 + 2x_2 + 3x_3 + x_4 \\ s.t. & \\ & -x_1 + x_2 + x_3 + 10x_4 \leq 20 \\ & x_1 - 3x_2 + x_3 \leq 30 \\ & x_2 - 3.5x_4 = 0 \\ & x_1 \leq 40 \\ & x_4 \leq 3 \\ & x_4 \geq 2 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{array}$$

LP

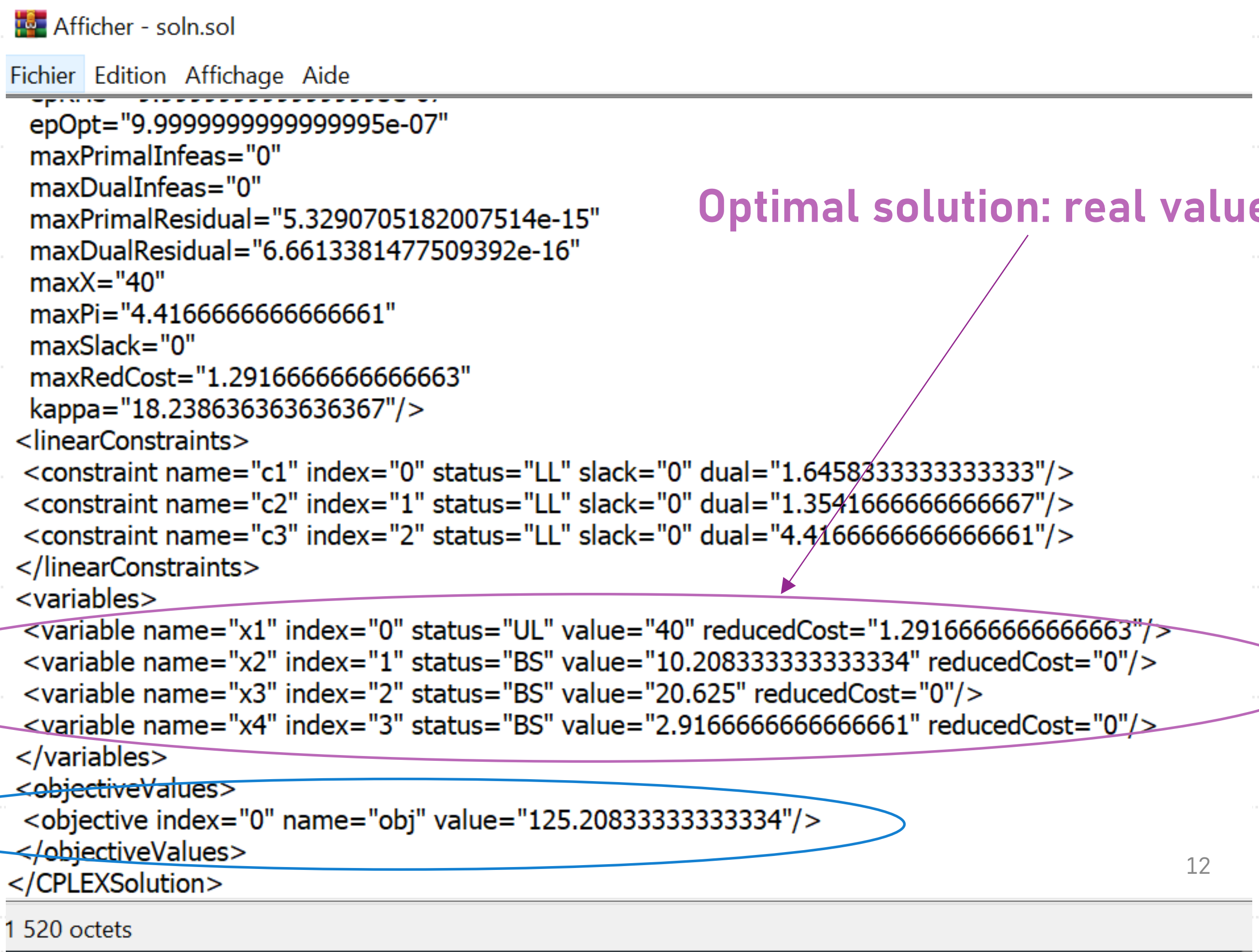
NEOS SERVER SOLVERS

CPLEX



```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
End
```

Optimal objective value



```
 epOpt="9.9999999999999995e-07"
 maxPrimalInfeas="0"
 maxDualInfeas="0"
 maxPrimalResidual="5.3290705182007514e-15"
 maxDualResidual="6.6613381477509392e-16"
 maxX="40"
 maxPi="4.4166666666666661"
 maxSlack="0"
 maxRedCost="1.2916666666666663"
 kappa="18.238636363636367"/>
<linearConstraints>
 <constraint name="c1" index="0" status="LL" slack="0" dual="1.6458333333333333"/>
 <constraint name="c2" index="1" status="LL" slack="0" dual="1.3541666666666667"/>
 <constraint name="c3" index="2" status="LL" slack="0" dual="4.4166666666666661"/>
</linearConstraints>
<variables>
 <variable name="x1" index="0" status="UL" value="40" reducedCost="1.2916666666666663"/>
 <variable name="x2" index="1" status="BS" value="10.208333333333334" reducedCost="0"/>
 <variable name="x3" index="2" status="BS" value="20.625" reducedCost="0"/>
 <variable name="x4" index="3" status="BS" value="2.9166666666666661" reducedCost="0"/>
</variables>
<objectiveValues>
 <objective index="0" name="obj" value="125.20833333333334"/>
</objectiveValues>
</CPLEXSolution>
```

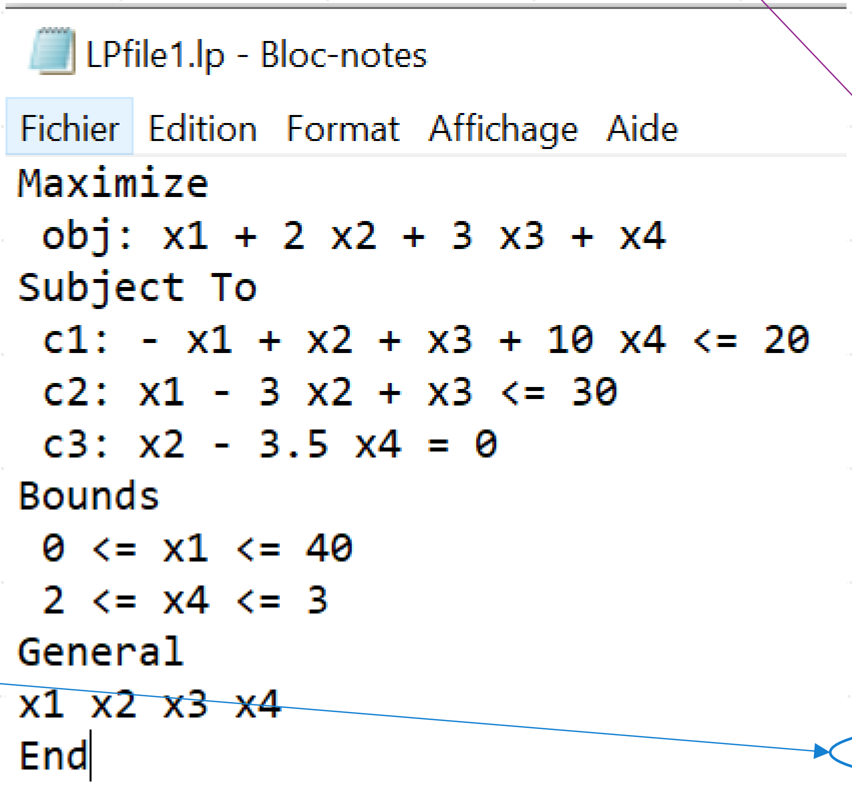
Optimal solution: real values

Example (2/2)

$$\begin{aligned} Max \quad & x_1 + 2x_2 + 3x_3 + x_4 \\ s.t. \quad & \\ & -x_1 + x_2 + x_3 + 10x_4 \leq 20 \\ & x_1 - 3x_2 + x_3 \leq 30 \\ & x_2 - 3.5x_4 = 0 \\ & x_1 \leq 40 \\ & x_4 \leq 3 \\ & x_4 \geq 2 \\ & x_1, x_2, x_3, x_4 \in \mathbb{N} \end{aligned}$$

IP

CPLEX

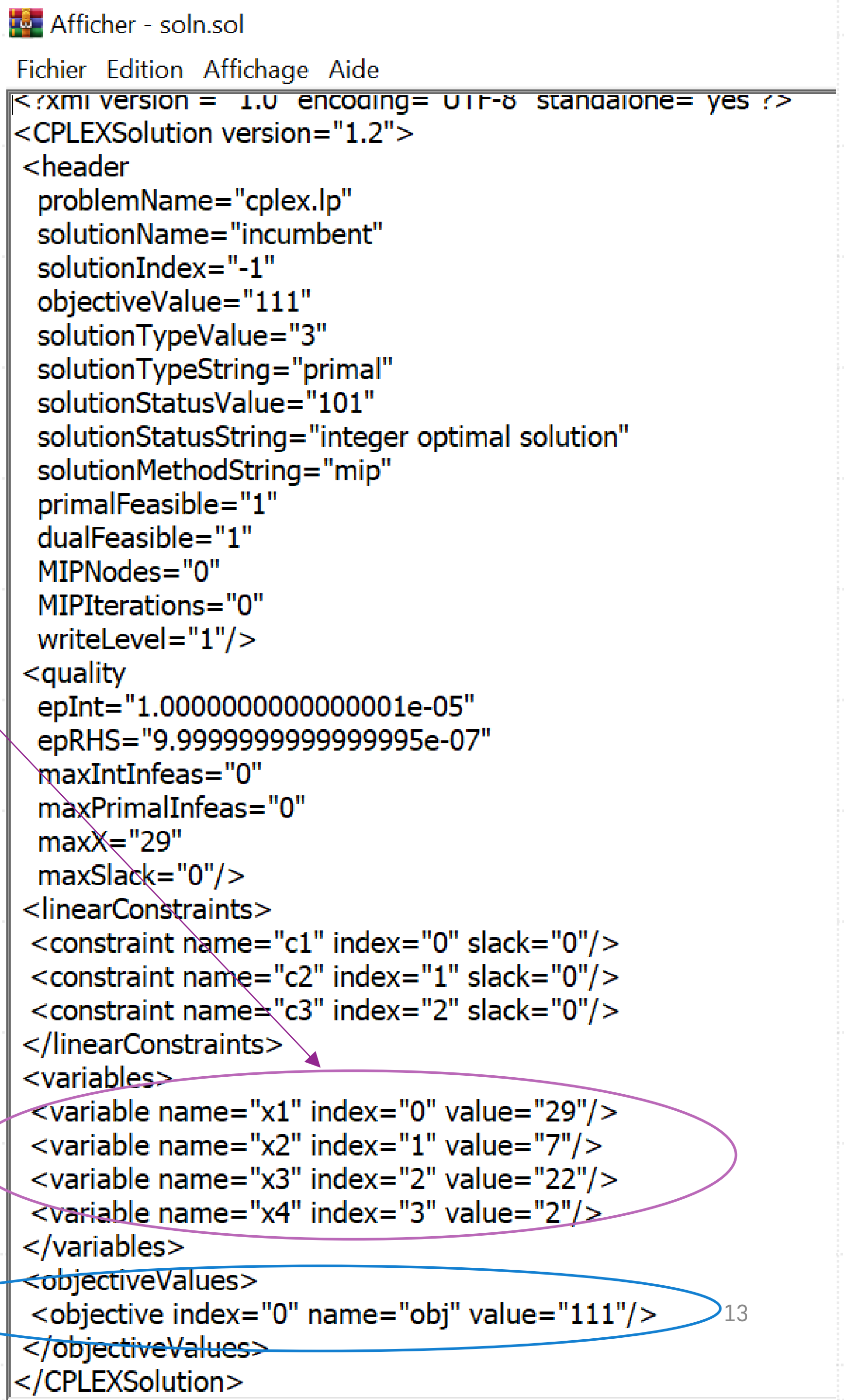

LPfile1.lp - Bloc-notes

Fichier Edition Format Affichage Aide

```
Maximize
 obj: x1 + 2 x2 + 3 x3 + x4
Subject To
 c1: - x1 + x2 + x3 + 10 x4 <= 20
 c2: x1 - 3 x2 + x3 <= 30
 c3: x2 - 3.5 x4 = 0
Bounds
 0 <= x1 <= 40
 2 <= x4 <= 3
General
x1 x2 x3 x4
End
```

Optimal solution: integer values

Optimal objective value

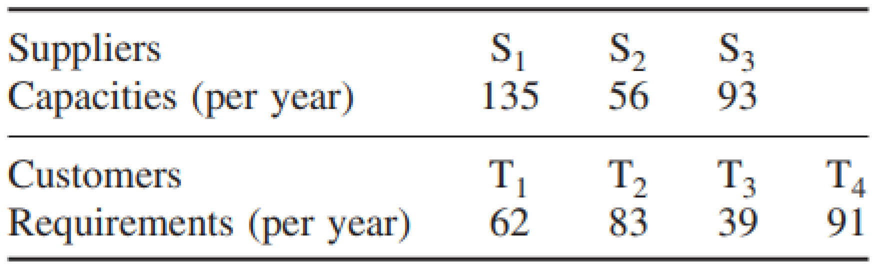

Afficher - soln.sol

Fichier Edition Affichage Aide

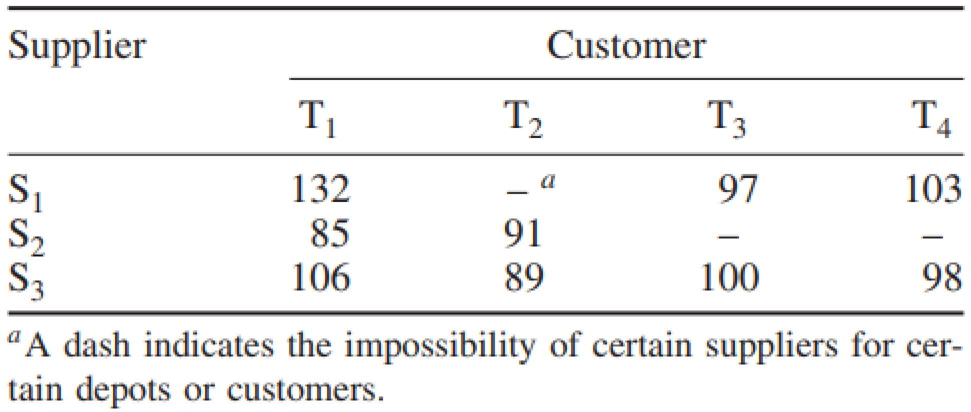
```
<?xml version = "1.0" encoding="UTF-8" standalone="yes"?>
<CPLEXSolution version="1.2">
 <header
   problemName="cplex.lp"
   solutionName="incumbent"
   solutionIndex="-1"
   objectiveValue="111"
   solutionTypeValue="3"
   solutionTypeString="primal"
   solutionStatusValue="101"
   solutionStatusString="integer optimal solution"
   solutionMethodString="mip"
   primalFeasible="1"
   dualFeasible="1"
   MIPNodes="0"
   MIPIterations="0"
   writeLevel="1"/>
 <quality
   epInt="1.0000000000000001e-05"
   epRHS="9.9999999999999995e-07"
   maxIntInfeas="0"
   maxPrimalInfeas="0"
   maxX="29"
   maxSlack="0"/>
 <linearConstraints>
  <constraint name="c1" index="0" slack="0"/>
  <constraint name="c2" index="1" slack="0"/>
  <constraint name="c3" index="2" slack="0"/>
 </linearConstraints>
 <variables>
  <variable name="x1" index="0" value="29"/>
  <variable name="x2" index="1" value="7"/>
  <variable name="x3" index="2" value="22"/>
  <variable name="x4" index="3" value="2"/>
 </variables>
 <objectiveValues>
  <objective index="0" name="obj" value="111"/>
 </objectiveValues>
</CPLEXSolution>
```

Transportation problem 4

- Three suppliers (S1, S2, S3) are used to provide four customers (T1, T2, T3, T4) with their requirements for a particular commodity over a year. The yearly capacities of the suppliers and the minimum requirements of the customers are given in the first table (in suitable units).
- The unit costs for supplying each customer from each supplier are given in the second table (in pounds per unit). Suppose that these commodities are containers.
  1) Propose a graph to represent the problem
  2) Give the IP model to minimize the total cost.
  3) Write the .lp file and solve it using NEOS Server.
  4) Illustrate the optimal solution using the graph.



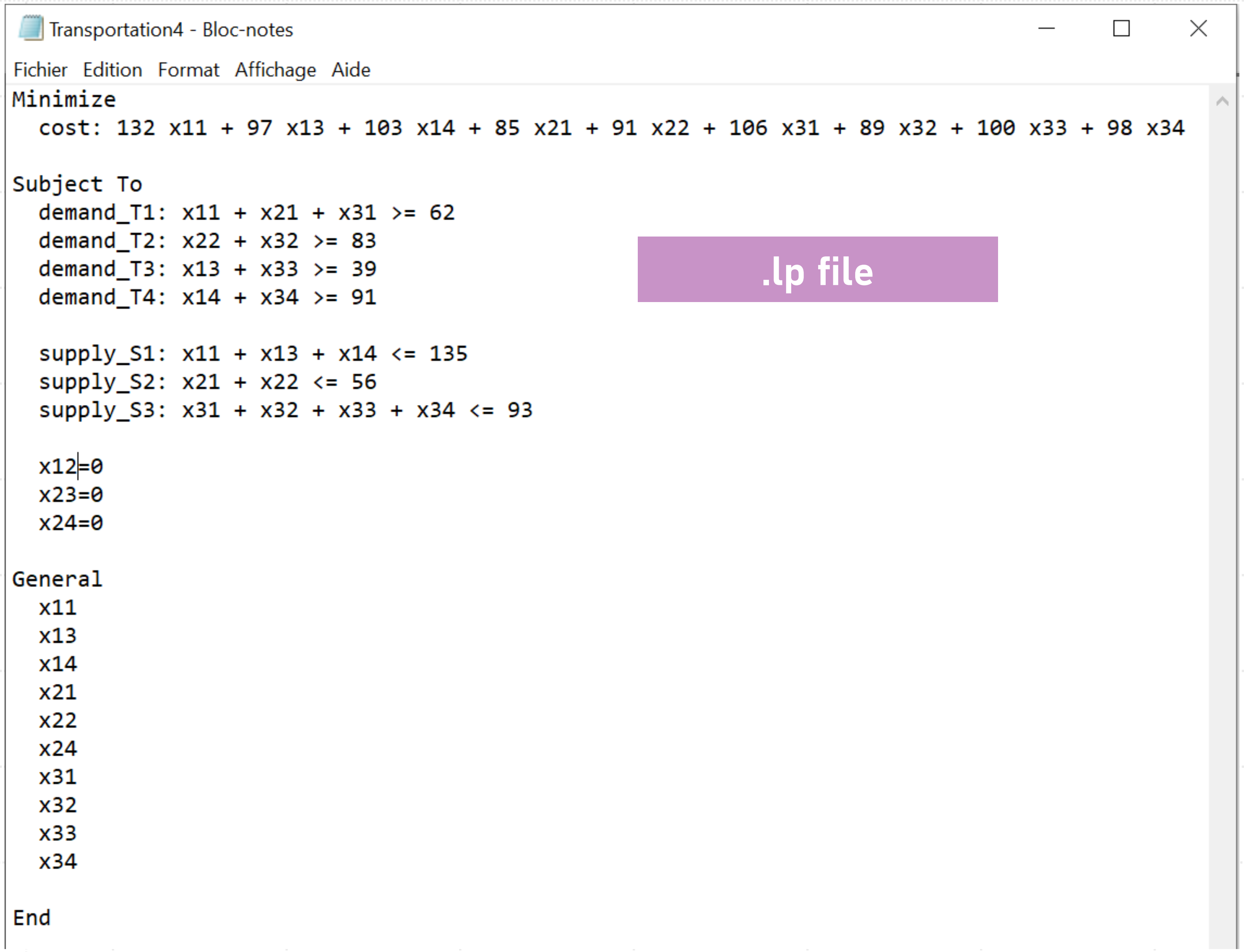
Suppliers	S_1	S_2	S_3	
Capacities (per year)	135	56	93	
Customers	T_1	T_2	T_3	T_4
Requirements (per year)	62	83	39	91



Supplier	Customer			
	T_1	T_2	T_3	T_4
S_1	132	–[a]	97	103
S_2	85	91	–	–
S_3	106	89	100	98

[a] A dash indicates the impossibility of certain suppliers for certain depots or customers.

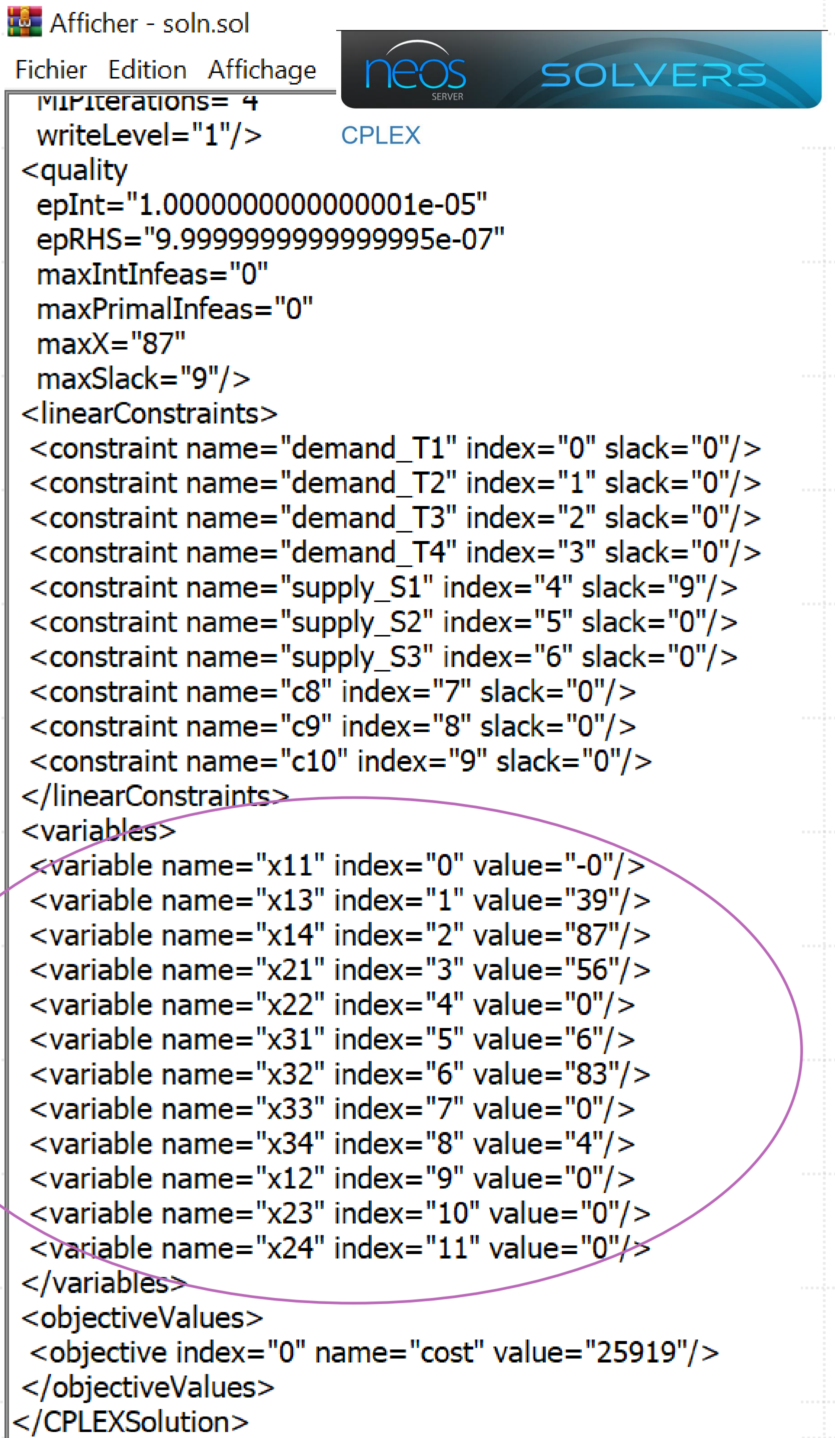
Transportation problem 4



Transportation4 - Bloc-notes

```
Minimize
  cost: 132 x11 + 97 x13 + 103 x14 + 85 x21 + 91 x22 + 106 x31 + 89 x32 + 100 x33 + 98 x34

Subject To
  demand_T1: x11 + x21 + x31 >= 62
  demand_T2: x22 + x32 >= 83
  demand_T3: x13 + x33 >= 39
  demand_T4: x14 + x34 >= 91

  supply_S1: x11 + x13 + x14 <= 135
  supply_S2: x21 + x22 <= 56
  supply_S3: x31 + x32 + x33 + x34 <= 93

  x12=0
  x23=0
  x24=0

General
  x11
  x13
  x14
  x21
  x22
  x24
  x31
  x32
  x33
  x34

End
```

.lp file



Afficher - soln.sol

CPLEX

```
  MIPIterations="4"
  writeLevel="1"/>
 <quality
  epInt="1.0000000000000001e-05"
  epRHS="9.9999999999999995e-07"
  maxIntInfeas="0"
  maxPrimalInfeas="0"
  maxX="87"
  maxSlack="9"/>
 <linearConstraints>
  <constraint name="demand_T1" index="0" slack="0"/>
  <constraint name="demand_T2" index="1" slack="0"/>
  <constraint name="demand_T3" index="2" slack="0"/>
  <constraint name="demand_T4" index="3" slack="0"/>
  <constraint name="supply_S1" index="4" slack="9"/>
  <constraint name="supply_S2" index="5" slack="0"/>
  <constraint name="supply_S3" index="6" slack="0"/>
  <constraint name="c8" index="7" slack="0"/>
  <constraint name="c9" index="8" slack="0"/>
  <constraint name="c10" index="9" slack="0"/>
 </linearConstraints>
 <variables>
  <variable name="x11" index="0" value="-0"/>
  <variable name="x13" index="1" value="39"/>
  <variable name="x14" index="2" value="87"/>
  <variable name="x21" index="3" value="56"/>
  <variable name="x22" index="4" value="0"/>
  <variable name="x31" index="5" value="6"/>
  <variable name="x32" index="6" value="83"/>
  <variable name="x33" index="7" value="0"/>
  <variable name="x34" index="8" value="4"/>
  <variable name="x12" index="9" value="0"/>
  <variable name="x23" index="10" value="0"/>
  <variable name="x24" index="11" value="0"/>
 </variables>
 <objectiveValues>
  <objective index="0" name="cost" value="25919"/>
 </objectiveValues>
</CPLEXSolution>
```

Types of Integer Programming Problems

- Pure Integer (Linear) Programming **(IP)**
  - All variables must have integer solutions
- Mixed Integer (Linear) Programming **(MIP)**
  - Some, but not all variables have integer solutions
- Binary (0–1) Integer (Linear) Programming **(BIP)**
  - All variables have values of 0 or 1: **decision variables**
- Mixed Binary Integer (Linear) Programming **(MBIP)**
  - Some decision variables are binary, and other decision variables are either general integer or continuous valued.

Decision variables … Yes, or No?

- Variables that are frequently used in IP to indicate which of a number of possible decisions should be made.
- Usually, these variables can only take the two values, zero or one.
- Such variables are known as zero–one (0–1) (or binary) variables.
- More formally, (0–1)IP is :

$$\begin{aligned} Maximize \quad & c^T x \\ s.t. \quad & Ax \leq b \\ & x_i \in \{0,1\}\ \forall i \end{aligned}$$

What changes in .lp files for (0-1) models?

The names of all variables which are to be restricted to binary integer values, separated by at least one space.

```
Minimize/Maximize
 obj: ....
Subject To
  c1: ....
  c2: ....
   ....
Bounds (optional)
   .......
Binary
   ........
End
```

Objective function section

Constraints section

Bounds section (optional)

Binary section

In general

.lp files sections:

Minimize/Maximize obj: ….	**Objective function section**
Subject To c1: …. c2: …. ….	**Constraints section**
Bounds (optional) …….	**Bounds section (optional)**
General ……..	**General section**
Binary ……..	**Binary section**
End	

The names of all variables which are to be restricted to general integer values, separated by at least one space.

The names of all variables which are to be restricted to binary integer values, separated by at least one space.

→ Note that continuous variables don't need declaration.

Useful tips

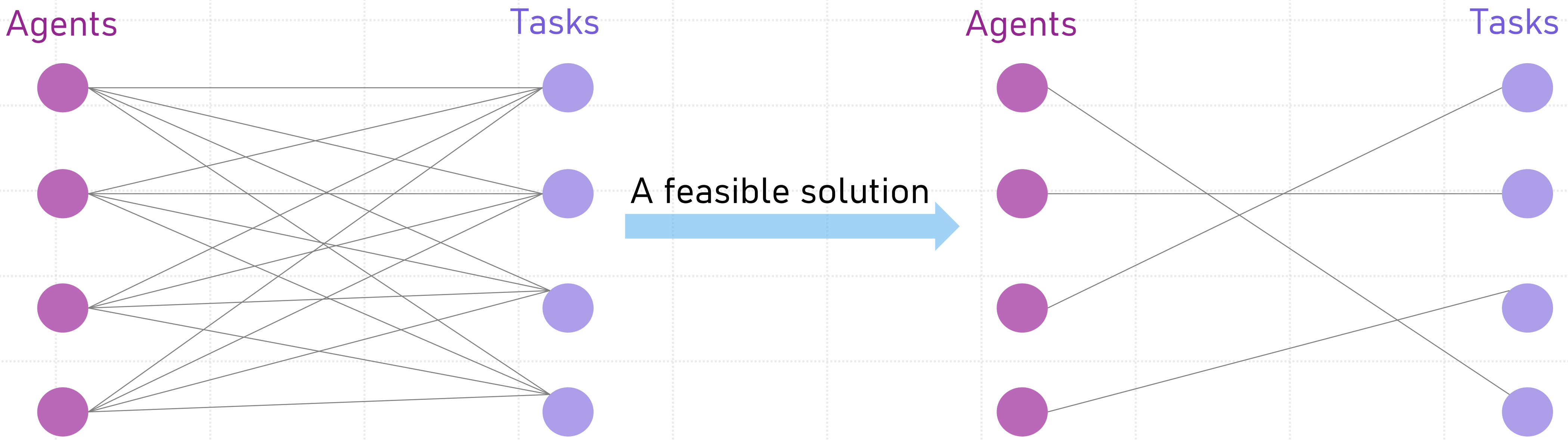
Binary variables can be used to represent logical constraints. Here are a few examples, where p_i represents a logical proposition and x_i the corresponding logical binary variable.

Logical constraints	Algebraic Forms
$p_1 \oplus p_2 = true$: This constraint represents an exclusive OR (XOR). It means either p_1 is true or p_2 is true, but not both at the same time.	$x_1 + x_2 = 1$: The sum of x_1 and x_2 must equal 1, meaning that one of the two variables is 1 (true), and the other is 0 (false), but not both.
$p_1 \vee p_2 \vee \cdots \vee p_n = true$: This is a logical disjunction, meaning that at least one of the propositions $p_1, p_2, \ldots, p_n$ must be true.	$x_1 + x_2 + \cdots + x_n \geq 1$: The sum of all variables must be at least 1, indicating that at least one of these variables is 1 (true).
$p_1 \wedge p_2 \wedge \cdots \wedge p_n = true$: This represents a logical conjunction, meaning that all propositions $p_1, p_2, \ldots, p_n$ must be true simultaneously.	$x_1 + x_2 + \cdots + x_n \geq n$ (or $= n$): If the sum of the all variables is greater than or equal to n, it means that all the variables are 1 (all true).
$p_1 \Rightarrow p_2$: This is a logical implication. It means that if p_1 is true, then p_2 must also be true.	$x_2 \geq x_1$: If $x_1 = 1$, then x_2 must also be at least 1 (true). If $x_1 = 0$, x_2 can take any value (0 or 1).
$p_1 \Leftrightarrow p_2$: This represents a logical equivalence, meaning p_1 and p_2 are either both true or both false.	$x_1 = x_2$: Variables x_1 and x_2 must be equal, meaning if one is 1, the other is also 1, and if one is 0, the other is 0 as well.

Let’s model some well-known problems

Assignment problem 1

- The assignment problem is one of the fundamental combinatorial optimization problems. In its most general form, the problem is as follows:
- There are n agents and n tasks. Any agent can be assigned to perform any task, but some agents are more experienced than others, and the time it takes for agent i to complete task j has a duration of t_{ij}. It is required to perform all tasks by assigning exactly one agent to each task in such a way that the average task completion time is minimized.
- Propose a graph to represent the problem, and then suggest a feasible solution.



Assignment problem 1

- The assignment problem is one of the fundamental combinatorial optimization problems. In its most general form, the problem is as follows:
- There are n agents and n tasks. Any agent can be assigned to perform any task, but some agents are more experienced than others, and the time it takes for agent i to complete task j has a duration of t_{ij}. It is required to perform all tasks by assigning exactly one agent to each task in such a way that the average task completion time is minimized.

$$Min \quad \sum_{i=1}^{n} \sum_{j=1}^{n} t_{ij} x_{ij}$$

$$s.t. \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in \{1, \dots, n\} \longrightarrow \boxed{\text{Each agent must be assigned to exactly one task}}$$

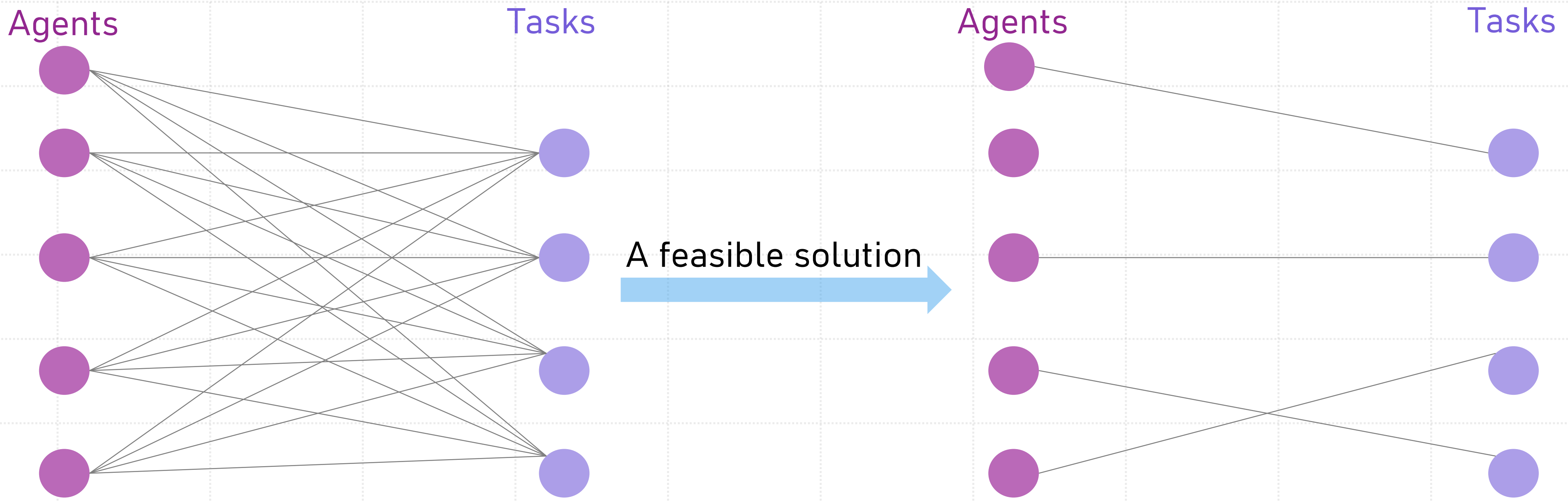
$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in \{1, \dots, n\} \longrightarrow \boxed{\text{Each task is to be performed by exactly one agent}}$$

$$x_{ij} \in \{0,1\}, \qquad \forall i, j \in \{1, \dots, n\}$$

Where x_{ij} is a binary (decision) variable: $x_{ij} = \begin{cases} 1 \; if \; agent \; i \; performs \; task \; j \\ 0 \; otherwise \end{cases}$

Assignment problem 2

- There are n agents and m tasks (with $n \geq m$). Any agent can be assigned to perform any task, but some agents are more experienced than others, and the time it takes for agent i to complete task j has a duration of t_{ij}. The objective is to assign agents to tasks in such a way that the total completion time is minimized. Although **not every agent has to be assigned**, we require that no agent is assigned to more than one task, and **each task is assigned to exactly one agent**. Propose a graph to represent the problem, and then suggest a feasible solution



Assignment problem 2

- There are n agents and m tasks (with $n \geq m$). Any agent can be assigned to perform any task, but some agents are more experienced than others, and the time it takes for agent i to complete task j has a duration of t_{ij}. The objective is to assign agents to tasks in such a way that the total completion time is minimized. Although **not every agent has to be assigned**, we require that no agent is assigned to more than one task, and **each task is assigned to exactly one agent**.

$$Min \quad \sum_{i=1}^{n} \sum_{j=1}^{m} t_{ij} x_{ij}$$

$$s.t. \quad \sum_{j=1}^{m} x_{ij} \leq 1 \quad \forall i \in \{1, \dots, n\}$$

Each agent is assigned to at most one task

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in \{1, \dots, m\}$$

Each task is to be performed by exactly one agent

$$x_{ij} \in \{0,1\}, \ \forall i \in \{1, \dots, n\}, j \in \{1, \dots, m\}$$

Where x_{ij} is a binary (decision) variable: $x_{ij} = \begin{cases} 1 \; if \; agent \; i \; performs \; task \; j \\ 0 \; otherwise \end{cases}$

The knapsack problem

- The name ‘knapsack’ arises from the rather contrived application of a hiker trying to fill his knapsack to maximum total value. Each item he considers taking with him has a certain value and a certain weight. An overall weight limitation gives the single constraint.
- More formally:
- Given a set $I = \{1, \dots, n\}$ of items $i \in I$, each with a weight w_i and a value v_i, the goal is to determine the maximum value that can be obtained by selecting a subset of the items to fit into a knapsack of a limited weight capacity W.

The knapsack problem

- The name ‘knapsack’ arises from the rather contrived application of a hiker trying to fill his knapsack to maximum total value. Each item he considers taking with him has a certain value and a certain weight. An overall weight limitation gives the single constraint.
- More formally,
- Given a set $I = \{1, \dots, n\}$ of items $i \in I$, each with a weight w_i and a value v_i, the goal is to determine the maximum value that can be obtained by selecting a subset of the items to fit into a knapsack of a limited weight capacity W. The IP (P1) is given as:

$$Max \sum_{i=1}^{n} v_i x_i$$

$$s.t. \sum_{i=1}^{n} w_i x_i \leq W \longrightarrow$$

The only constraint: the knapsack weight capacity is not exceeded.

$$x_i \in \{0,1\}, \qquad \forall i \in I$$

Where x_i is a binary (decision) variable: $x_i = \begin{cases} 1 \; if \; item \; i \; is \; put \; in \; the \; knapsack \\ 0 \; otherwise \end{cases}$

What type of IP program is this?

Knapsack applications

- The applications in logistics and supply chain management are evident:
  - Packing containers and optimizing cargo shipments.
  - Inventory management to maximize profit while adhering to storage constraints.

However, this problem can be found in many other domains as well. For example, in:

- Finance: **you have a finite budget W, you have financial products i, each costing w_i, and yielding a return of v_i in the coming year; maximize the profit.**
- Manufacturing:
  - Cutting stock problem: Cutting raw materials into smaller pieces to fulfill orders efficiently.
  - Job scheduling and assignment to machines in manufacturing processes.
- Telecommunications:
  - Channel assignment in wireless communication to maximize bandwidth while minimizing interference.
- Computer Science and Algorithms:
  - Memory allocation in compilers and operating systems.
  - Load balancing in distributed computing.

Continuous relaxation

- If we leave out the integrality constraints, i.e., if we allow each x_i to attain all values in the interval $[0,1]$, we obtain the following linear program:

$$
\begin{aligned}
&Max \sum_{i=1}^{n} v_i x_i \\
&\text{s.}t. \ \sum_{i=1}^{n} w_i x_i \le W \\
&\qquad 0 \le x_i \le 1, \qquad \forall i \in I
\end{aligned}
$$

- It is called a continuous relaxation (or LP relaxation) of the integer program P1 (slide 27).
  - we have relaxed the constraints $x_i \in \{0,1\}$ to the weaker constraints $0 \le x_i \le 1$.
- We can solve the LP relaxation, say by the simplex algorithm, and either we obtain an optimal solution x^*, or we learn that the LP relaxation is infeasible. In the latter case, the original integer program must be infeasible as well.
- Let us now assume that the LP relaxation has an optimal solution x^*. Certainly, x^* provides an upper bound on the best possible solution of the original integer program P1. More precisely, the optimum of the objective function in the integer program P1 is bounded above by the value of the objective function at x^*. This is because every feasible solution of the integer program is also a feasible solution of the LP relaxation, and so we are maximizing over a larger set of vectors in the LP relaxation.

The Bin-Packing problem (BPP)

- The BPP in its simplest version can be described as follows: we have n objects $i \in \{1, \ldots, n\}$ of varying sizes a_i and K bins $j \in \{1, \ldots, K\}$ of the same size W; the objective is to find the minimum number of bins required to store all the objects.
- This problem has direct applications in the field of logistics, such as storing products, loading trucks, and more.
- In the most general forms of this problem, we can also take into account the shapes of objects (this is referred to as 2D BPP or 3D BPP), incompatibilities, and more.
- This problem is sometimes also referred to as cutting-stock. In fact, cutting problems (for cutting pieces of fabric, metal, etc.) where the goal is to minimize waste are modeled in a similar way. Here, we are focusing on the simplest case, 1D BPP. This case is already very practical because it can provide bounds, be used in subroutines, or be directly applied (**such as determining the minimum number of cloud storage containers required to store data from a server, optimizing the placement of files in data blocks in solid-state drives (SSDs),** cutting boards of varying lengths from fixed-length sheets, or cutting fabric rolls, etc.).

The 1D BPP

The (0-1) ILP:

$$Min \sum_{j=1}^{K} z_j$$

$$s.t. \sum_{j=1}^{K} x_{ij} = 1, \qquad \forall i \in \{1, \dots, n\}$$

$$\sum_{i=1}^{n} a_i x_{ij} \leq W z_j, \qquad \forall j \in \{1, \dots, K\}$$

$$x_{ij}, z_j \in \{0,1\}, \ \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, K\}$$

Where decision variables are:

$$z_j = \begin{cases} 1 \ if \ bin \ j \ is \ used \\ 0 \ otherwise \end{cases}$$

$$x_{ij} = \begin{cases} 1 \ if \ object \ i \ is \ stored \ in \ bin \ j \\ 0 \ otherwise \end{cases}$$

Minimize the total number of bins used

All the objects must be stored, and each one is stored in exactly one bin.

The total size of items placed in a bin (if it is used) does not exceed its size W:

- If bin j is used then $z_j = 1$, and so the constraint can be written as:

$$\sum_{i=1}^{n} a_i x_{ij} \leq W$$

- If bin j is not used then $z_j = 0$, and so the constraint can be written as:

$$\sum_{i=1}^{n} a_i x_{ij} \leq 0$$

Which implies that $x_{ij} = 0, \forall i \in \{1, \dots, n\}$ for this bin j.

The integrality constraints

The 1D BPP

The (0-1) ILP:

$$Min \ \sum_{j=1}^{K} z_j$$

$$s.t. \ \sum_{j=1}^{K} x_{ij} = 1, \qquad \forall i \in \{1,\dots,n\}$$

$$\sum_{i=1}^{n} a_i x_{ij} \le W z_j, \qquad \forall j \in \{1,\dots,K\}$$

$$x_{ij}, z_j \in \{0,1\}, \ \forall i \in \{1,\dots,n\}, \forall j \in \{1,\dots,K\}$$

Relaxation

$$Min \ \sum_{j=1}^{K} z_j$$

$$s.t. \ \sum_{j=1}^{K} x_{ij} = 1, \qquad \forall i \in \{1,\dots,n\}$$

$$\sum_{i=1}^{n} a_i x_{ij} \le W z_j, \qquad \forall j \in \{1,\dots,K\}$$

$$0 \le x_{ij} \le 1, \forall i \in \{1,\dots,n\}, \forall j \in \{1,\dots,K\}$$

$$0 \le z_j \le 1, \ \forall j \in \{1,\dots,K\}$$

Where decision variables are:

$$z_j = \begin{cases} 1 \ if \ bin \ j \ is \ used \\ 0 \ otherwise \end{cases}$$

$$x_{ij} = \begin{cases} 1 \ if \ object \ i \ is \ stored \ in \ bin \ j \\ 0 \ otherwise \end{cases}$$

The most natural lower bound of IP is obtained through **LP relaxation**.

Capacitated Facility Location problem

- This problem involves determining the optimal locations for facilities within a finite set of sites, while considering the minimum requirements of the clients that need to be served and optimizing the total profit. Usually, setting up (opening) a facility involves significant costs that do not depend on the production level of the facility.
- More formally, let :

Data
  - I : the set of clients
  - J: the set of sites where facilities can be opened
  - f_j : the fixed cost of opening the facility placed at j for $j \in J$
  - a_{ij} : the profit per unit of sale of the goods originated at facility j to client i, $j \in J$ and $i \in I$
  - u_j : the maximum capacity of the facility located at $j \in J$
  - b_i : the (minimum/at-least) demand of client $i \in I$

Capacitated Facility Location problem

- **Decision variables** :
  - $x_j = \begin{cases} 1\ if\ facility\ j\ is\ opened\ j \in J \\ 0\ otherwise \end{cases}$
  - $y_{ij} = the\ quantity\ of\ goods\ sold\ by\ facility\ j\ to\ client\ i, (\in \mathbb{R}^+), j \in J\ and\ i \in I$
- **Objective function:** maximize the total profit: $maximize \sum_{i \in I} \sum_{j \in J} a_{ij}\, y_{ij} - \sum_{j \in J} f_j x_j$
- **Constraints:**
  - Each client's demand must be satisfied: $\sum_{j \in J} y_{ij} \geq b_i, \quad \forall i \in I$
  - Since a client i cannot be served from j unless a facility is placed at j, we have the following constraints that ensure that the client i can be served from j only if a facility is opened at site j, since $x_j = 0$ implies that $y_{ij} = 0$, and $x_j = 1$ yields the constraint $\sum_{i \in I} y_{ij} \leq u_j$, which means that the production level of the facility j cannot exceed its capacity : $\sum_{i \in I} y_{ij} \leq u_j x_j, \quad \forall j \in J$
  - The integrality and non-negativity constraints:

$$x_j \in \{0,1\}, \quad \forall j \in J$$
$$y_{ij} \geq 0, \quad \forall i \in I, \forall j \in J$$

Capacitated Facility Location problem

- The final generic model is:

$$
\begin{aligned}
& Maximize \sum_{i \in I} \sum_{j \in J} a_{ij}\, y_{ij} - \sum_{j \in J} f_j x_j \\
& s.t. \quad \sum_{j \in J} y_{ij} \geq b_i, \qquad \forall i \in I \\
& \qquad \sum_{i \in I} y_{ij} \leq u_j x_j, \qquad \forall j \in J \\
& \qquad x_j \in \{0,1\}, \qquad \forall j \in J \\
& \qquad y_{ij} \geq 0, \qquad \forall i \in I, \forall j \in J
\end{aligned}
$$

This is a Mixed Integer Linear Program

Translating models into words ☺

- Here is an overview of a MIP model for a container stacking problem (a previous paper written by your professor et al. ☺). The goal is to assign containers to slots while respecting a set of constraints. Below are three constraints written mathematically. Your task is to explain each constraint using the mathematical notation and logic provided. Think about the meaning of each variable and how the constraint ensures proper storage and management of containers.

- Notations:
  - C_p: Set of containers to be stored in the buffer
  - Lev: Number of levels in a stack
  - PP: Number of stacks in the buffer
  - P_P : The set of all stacks in the buffer
  - TT: Total number of time slots during the entire handling period of containers.

- Decisions variables

$$X_{ph}^{ct} = \begin{cases} 1 & \text{if container } c \in Cp \text{ is stowed in slot } (p,h),\ p \in Pp,\ h \in \{1,\dots,Lev\} \text{ at instant } t \\ 0 & \text{otherwise} \end{cases}$$

- Constraints

(1) $$1 - X_{ph}^{ct} \geqslant \sum_{p'=1}^{PP} \sum_{h'=1}^{Lev} X_{p'h'}^{ct} \qquad \forall c \in Cp,\ \forall p \in \{1,\dots,PP\}, \forall h \in \{1,\dots,Lev\}, \forall t \in \{1,\dots,TT\}$$

(2) $$1 - X_{ph}^{ct} \geqslant \sum_{\substack{c' \in Cp \\ c' \neq c}} X_{ph}^{c't} \qquad \forall c \in Cp,\ \forall p \in \{1,\dots,PP\}, \forall h \in \{1,\dots,Lev\}, \forall t \in \{1,\dots,TT\}$$

(3) $$(h-1)\sum_{c \in CP} X_{ph}^{ct} \leqslant \sum_{h'=1}^{h-1} \sum_{c' \in CP} X_{ph'}^{c't} \qquad \forall p \in \{1,\dots,PP\}, \forall h \in \{2,\dots,Lev\}, \forall t \in \{1,\dots,TT\}$$

Translating models into words ☺

- Constraints (1) ensure that each container occupies one and only one slot at any instant t.
- Constraints (2) indicate that if the slot (p, h) is occupied at any instant t, it must be occupied by one and only one container.
- Constraints (3) ensure that containers are stacked on top of each other, that is, if one slot (p, h) is occupied at any instant t, all the other slots below must be already occupied.

If you're curious, you can check the complete version here: El Yaagoubi, Amina, et al. "A logistic model for a french intermodal rail/road freight transportation system." *Transportation Research Part E: Logistics and Transportation Review* 164 (2022): 102819.

- Constraints

(1) $$1 - X^{ct}_{ph} \geqslant \sum_{p'=1}^{PP} \sum_{h'=1}^{Lev} X^{ct}_{p'h'} \qquad \forall c \in Cp,\ \forall p \in \{1, \dots, PP\}, \forall h \in \{1, \dots, Lev\}, \forall t \in \{1, \dots, TT\}$$

(2) $$1 - X^{ct}_{ph} \geqslant \sum_{\substack{c' \in Cp \\ c' \neq c}} X^{c't}_{ph} \qquad \forall c \in Cp,\ \forall p \in \{1, \dots, PP\}, \forall h \in \{1, \dots, Lev\}, \forall t \in \{1, \dots, TT\}$$

(3) $$(h-1) \sum_{c \in CP} X^{ct}_{ph} \leqslant \sum_{h'=1}^{h-1} \sum_{c' \in CP} X^{c't}_{ph'} \qquad \forall p \in \{1, \dots, PP\}, \forall h \in \{2, \dots, Lev\}, \forall t \in \{1, \dots, TT\}$$

Complexity : LP vs. IP

- Introducing integer variables significantly enhances the modeling capabilities but also increases complexity.
- While LP problems can be solved in polynomial time, IP is classified as an NP-hard problem. This means:
  - There is no known polynomial-time algorithm to solve it.
  - It is unlikely that such an algorithm will ever be discovered.
  - Even small instances of Integer Programming problems can be difficult to solve.

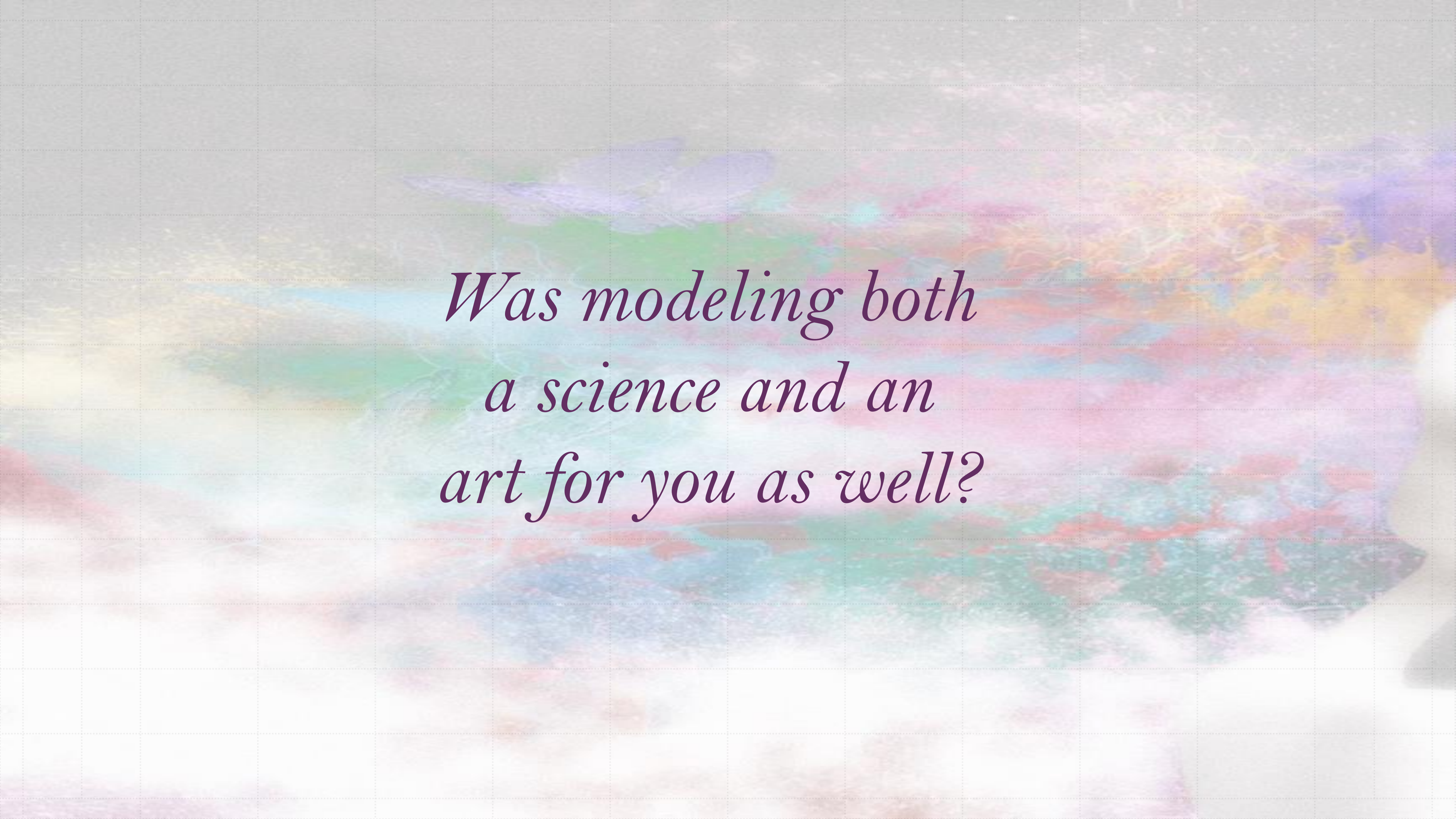
A very brief overview of a resolution method

- An integer linear program (e.g., a model with only 0-1 variables) has a finite number of solutions. We could consider enumerating all of them, but the number of solutions rapidly explodes.
- For 20 binary variables, there are more than a million possible solutions.
- For 30, it's more than a billion.
- As it becomes quickly unreasonable to perform a complete enumeration of solutions, we will try to leverage linear programming relaxation to eliminate some of these solutions.
- This technique of partial enumeration is known as branch-and-bound (B&B). It is a divide-and-conquer approach:
  - Decomposition of the problem into simpler sub-problems;
  - Combination of solving these sub-problems to obtain the solution to the original problem.
- In the branch-and-bound algorithm, each sub-problem corresponds to a node in the tree of solutions. We solve the linear relaxation of each sub-problem. The information derived from the linear relaxation may (possibly) allow us to eliminate all solutions that can be obtained from that node.

Return to keywords

- Integer programming,
- Decision variables,
- (0-1) decision variables,
- Mixed Integer Programming,
- Graph representation,
- Assignment problem,
- Knapsack problem,
- Bin-Packing problem,
- LP relaxation,
- Location problem.

- Provide a brief explanation of the mentioned keywords.



Was modeling both a science and an art for you as well?