Optimisation

Lecture 6 - Integer Programming

Fall semester - 2024

Dr. Eng. Valentin Leplat
Innopolis University
October 8, 2024

Outline

Introduction

Integer programming

- Many optimization problems can be formulated as linear optimization problems, with the *additional* constraint that some of the variables are **integer** or **binary**.
- In some cases, these additional constraints are easy to take into account.
- In other cases, the problem becomes considerably harder to solve. We present a method for solving integer optimization problems: the branch and bound method.

Integer programming

Integer programming.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{such that} \quad & Ax \geqslant b, \\ & x_i \text{ integer for all } i. \end{aligned}$$

Integer programming

Integer programming.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{such that} \quad & Ax \geqslant b, \\ & x_i \text{ integer for all } i. \end{aligned}$$

Mixed optimization.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{such that} \quad & Ax \geqslant b, \\ & x_i \text{ integer for } i \in \mathcal{I} \subset \{1, 2, \ldots, n\}. \end{aligned}$$

Integer programming

Integer programming.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{such that} \quad & Ax \geqslant b, \\ & x_i \text{ integer for all } i. \end{aligned}$$

Mixed optimization.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{such that} \quad & Ax \geqslant b, \\ & x_i \text{ integer for } i \in \mathcal{I} \subset \{1, 2, \ldots, n\}. \end{aligned}$$

Optimization in binary variables.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^T x \\ \text{such that} \quad & Ax \geqslant b, \\ & x \in \{0, 1\}^n. \end{aligned}$$

With integers, things get complicated ...

$$\max_x \quad 17x_1 + 12x_2 \quad \text{such that} \quad \begin{array}{c} 10x_1 + 7x_2 \leqslant 40, \\ x_1 + x_2 \leqslant 5, \\ x_1, x_2 \geqslant 0 \text{ and integer.} \end{array}$$

With integers, things get complicated ...

$$\max_x \quad 17x_1 + 12x_2 \quad \text{such that} \quad \begin{array}{c} 10x_1 + 7x_2 \leqslant 40, \\ x_1 + x_2 \leqslant 5, \\ x_1, x_2 \geqslant 0 \text{ and integer.} \end{array}$$

- The linear problem obtained without the constraint 'x_1, x_2 integers' is an relaxation of the initial problem. The optimal solution of the *relaxed* problem is given by $(5/3; 10/3)$. The corresponding value for the objective function is $205/3 = 68.33$.

With integers, things get complicated ...

$$\max_x \quad 17x_1 + 12x_2 \quad \text{such that} \quad \begin{array}{c} 10x_1 + 7x_2 \leqslant 40, \\ x_1 + x_2 \leqslant 5, \\ x_1, x_2 \geqslant 0 \text{ and integer.} \end{array}$$

- The linear problem obtained without the constraint 'x_1, x_2 integers' is an relaxation of the initial problem. The optimal solution of the *relaxed* problem is given by $(5/3; 10/3)$. The corresponding value for the objective function is $205/3 = 68.33$.
- By rounding the components of the optimal solution of the relaxed problem to the nearest integers, we find the solution (2, 3) which is not feasible.

With integers, things get complicated ...

$$\max_x \quad 17x_1 + 12x_2 \quad \text{such that} \quad \begin{array}{c} 10x_1 + 7x_2 \leqslant 40, \\ x_1 + x_2 \leqslant 5, \\ x_1, x_2 \geqslant 0 \text{ and integer.} \end{array}$$

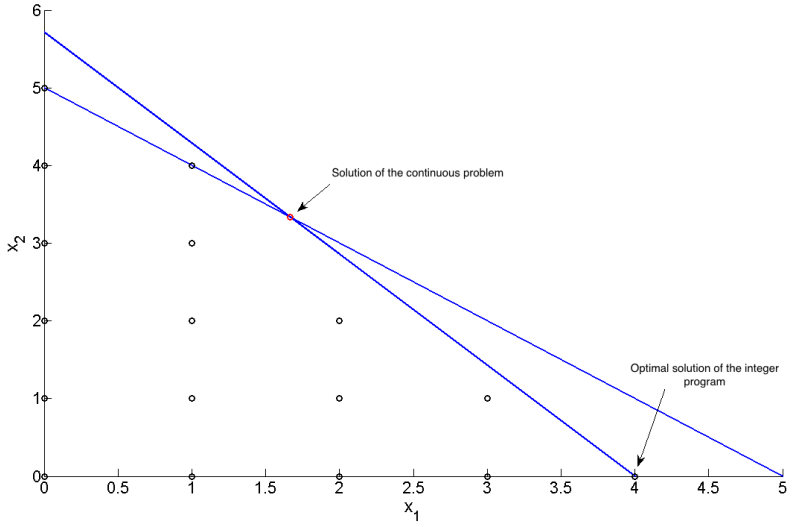
- The linear problem obtained without the constraint 'x_1, x_2 integers' is an relaxation of the initial problem. The optimal solution of the *relaxed* problem is given by $(5/3; 10/3)$. The corresponding value for the objective function is $205/3 = 68.33$.
- By rounding the components of the optimal solution of the relaxed problem to the nearest integers, we find the solution (2, 3) which is not feasible.
- The feasible solution closest to the optimal solution of the relaxed problem is (1, 3). This solution is not optimal.

With integers, things get complicated ...

$$\max_x \quad 17x_1 + 12x_2 \quad \text{such that} \quad \begin{array}{c} 10x_1 + 7x_2 \leqslant 40, \\ x_1 + x_2 \leqslant 5, \\ x_1, x_2 \geqslant 0 \text{ and integer.} \end{array}$$

- The linear problem obtained without the constraint 'x_1, x_2 integers' is an relaxation of the initial problem. The optimal solution of the *relaxed* problem is given by $(5/3; 10/3)$. The corresponding value for the objective function is $205/3 = 68.33$.
- By rounding the components of the optimal solution of the relaxed problem to the nearest integers, we find the solution (2, 3) which is not feasible.
- The feasible solution closest to the optimal solution of the relaxed problem is (1, 3). This solution is not optimal.
- The optimal solution is (4, 0).
  This solution is far from the optimal solution of the relaxed problem.

With integers, things get complicated ...



Integer programming

- The situation is *even worse in some cases.*
  For example, the case where the variables are binary and the optimal solution of the relaxed problem is (0.5, 0.5, . . . , 0.5); this does not give much information about the optimal solution in integers. . . .
- In fact, it is often difficult to determine whether a problem in binary variables has a feasible solution.

Here's an example. Decision variables

- **Context**: Four decisions $x_1, x_2, x_3, x_4 \in \{0, 1\}$ have to be made. The profit resulting from positive decisions is given by 9, 5, 6 and 4 respectively.

Here's an example. Decision variables

- **Context**: Four decisions $x_1, x_2, x_3, x_4 \in \{0, 1\}$ have to be made. The profit resulting from positive decisions is given by 9, 5, 6 and 4 respectively.
- Exclusive conditions: decisions 3 and 4 are mutually exclusive; it's 3 or 4, but not both:

Here's an example. Decision variables

- **Context**: Four decisions $x_1, x_2, x_3, x_4 \in \{0, 1\}$ have to be made. The profit resulting from positive decisions is given by 9, 5, 6 and 4 respectively.
- Exclusive conditions: decisions 3 and 4 are mutually exclusive; it's 3 or 4, but not both: $x_3 + x_4 \leqslant 1$.

Here's an example. Decision variables

- **Context**: Four decisions $x_1, x_2, x_3, x_4 \in \{0, 1\}$ have to be made. The profit resulting from positive decisions is given by 9, 5, 6 and 4 respectively.
- Exclusive conditions: decisions 3 and 4 are mutually exclusive; it's 3 or 4, but not both: $x_3 + x_4 \leqslant 1$.
- Alternativity constraints. At least one of the 1 or 3 decisions must be positive; it's either 1 or 3, or possibly both:

Here's an example. Decision variables

- **Context**: Four decisions $x_1, x_2, x_3, x_4 \in \{0, 1\}$ have to be made. The profit resulting from positive decisions is given by 9, 5, 6 and 4 respectively.
- Exclusive conditions: decisions 3 and 4 are mutually exclusive; it's 3 or 4, but not both: $x_3 + x_4 \leqslant 1$.
- Alternativity constraints. At least one of the 1 or 3 decisions must be positive; it's either 1 or 3, or possibly both: $x_1 + x_3 \geqslant 1$.
- Contingency constraints. Decision 3 can only be positive if decision 2 is positive:

Here's an example. Decision variables

- **Context**: Four decisions $x_1, x_2, x_3, x_4 \in \{0, 1\}$ have to be made. The profit resulting from positive decisions is given by 9, 5, 6 and 4 respectively.
- Exclusive conditions: decisions 3 and 4 are mutually exclusive; it's 3 or 4, but not both: $x_3 + x_4 \leqslant 1$.
- Alternativity constraints. At least one of the 1 or 3 decisions must be positive; it's either 1 or 3, or possibly both: $x_1 + x_3 \geqslant 1$.
- Contingency constraints. Decision 3 can only be positive if decision 2 is positive:

Here's an example. Decision variables

- **Context**: Four decisions $x_1, x_2, x_3, x_4 \in \{0, 1\}$ have to be made. The profit resulting from positive decisions is given by 9, 5, 6 and 4 respectively.
- Exclusive conditions: decisions 3 and 4 are mutually exclusive; it's 3 or 4, but not both: $x_3 + x_4 \leqslant 1$.
- Alternativity constraints. At least one of the 1 or 3 decisions must be positive; it's either 1 or 3, or possibly both: $x_1 + x_3 \geqslant 1$.
- Contingency constraints. Decision 3 can only be positive if decision 2 is positive: $x_3 \leqslant x_2$.

$$
\begin{aligned}
\min_{x} \quad & 9x_1 + 5x_2 + 6x_3 + 4x_4 \\
\text{such that} \quad & x_3 + x_4 \leqslant 1, \\
& x_1 + x_3 \geqslant 1, \\
& x_2 - x_3 \geqslant 0, \\
& x_1, x_2, x_3, x_4 \in \{0, 1\}.
\end{aligned}
$$

Here's an example. Project selection

- 5 projects are to be evaluated over 3 years.
- Given the cost of each of each project for each year, and the profit to be made by executing a of a project, decide which projects to execute within the available budget for each year.

	Cost	per	year	Profit
Project	1	2	3	
1	5	1	8	20
2	4	7	10	40
3	3	9	2	20
4	7	4	1	15
5	8	6	10	30
Budget	25	25	25	

Here's an example. Project selection

Choice of variables:

Here's an example. Project selection

Choice of variables:

$$x_i = \begin{cases} 1 & \text{if the project } i \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

Here's an example. Project selection

Choice of variables:

$$x_i = \begin{cases} 1 & \text{if the project } i \text{ is selected,} \\ 0 & \text{otherwise.} \end{cases}$$

We then have to solve

$$\begin{aligned} \max_{x} \quad & 20x_1 + 40x_2 + 20x_3 + 15x_4 + 30x_5 \\ \text{such that} \quad & 5x_1 + 4x_2 + 3x_3 + 7x_4 + 8x_5 \leqslant 25, \\ & x_1 + 7x_2 + 9x_3 + 4x_4 + 6x_5 \leqslant 25, \\ & 8x_1 + 10x_2 + 2x_3 + x_4 + 10x_5 \leqslant 25, \\ & x_1, x_2, x_3, x_4, x_5 \in \{0, 1\}. \end{aligned}$$

Example. Alternativity constraints

- Recall: The set of vectors satisfying the constraints $0 \leqslant x \leqslant 1$, $a_1^T x \leqslant b_1$ **and** $a_2^T x \leqslant b_2$ is a polyhedron.

Example. Alternativity constraints

- Recall: The set of vectors satisfying the constraints $0 \leqslant x \leqslant 1$, $a_1^T x \leqslant b_1$ **and** $a_2^T x \leqslant b_2$ is a polyhedron.
- The set of vectors satisfying the constraints $0 \leqslant x \leqslant 1$, $a_1^T x \leqslant b_1$ **or** $a_2^T x \leqslant b_2$ is not a polyhedron.
- To model these constraints, we introduce binary variables y_1 and y_2 and

$$a_1^T x - b_1 \leqslant M(1 - y_1),$$

$$a_2^T x - b_2 \leqslant M(1 - y_2),$$

and $y_1 + y_2 = 1$. M is a sufficiently large constant so that the constraints $a_i^T x - b_i \leqslant M$ are always satisfied for $0 \leqslant x \leqslant 1$. (For example, $M \geqslant \max(||a_1||_1 + |b_1|, ||a_2||_1 + |b_2|)$).

Example. Alternativity constraints

- Recall: The set of vectors satisfying the constraints $0 \leqslant x \leqslant 1$, $a_1^T x \leqslant b_1$ **and** $a_2^T x \leqslant b_2$ is a polyhedron.
- The set of vectors satisfying the constraints $0 \leqslant x \leqslant 1$, $a_1^T x \leqslant b_1$ **or** $a_2^T x \leqslant b_2$ is not a polyhedron.
- To model these constraints, we introduce binary variables y_1 and y_2 and

$$a_1^T x - b_1 \leqslant M(1 - y_1),$$

$$a_2^T x - b_2 \leqslant M(1 - y_2),$$

and $y_1 + y_2 = 1$. M is a sufficiently large constant so that the constraints $a_i^T x - b_i \leqslant M$ are always satisfied for $0 \leqslant x \leqslant 1$. (For example, $M \geqslant \max(||a_1||_1 + |b_1|, ||a_2||_1 + |b_2|)$).

Example. Alternativity constraints

- Recall: The set of vectors satisfying the constraints $0 \leqslant x \leqslant 1$, $a_1^T x \leqslant b_1$ **and** $a_2^T x \leqslant b_2$ is a polyhedron.
- The set of vectors satisfying the constraints $0 \leqslant x \leqslant 1$, $a_1^T x \leqslant b_1$ **or** $a_2^T x \leqslant b_2$ is not a polyhedron.
- To model these constraints, we introduce binary variables y_1 and y_2 and

$$a_1^T x - b_1 \leqslant M(1 - y_1),$$

$$a_2^T x - b_2 \leqslant M(1 - y_2),$$

  and $y_1 + y_2 = 1$. M is a sufficiently large constant so that the constraints $a_i^T x - b_i \leqslant M$ are always satisfied for $0 \leqslant x \leqslant 1$. (For example, $M \geqslant \max(||a_1||_1 + |b_1|, ||a_2||_1 + |b_2|)$).
- These constraints are equivalent to the condition that at least one of the constraints $a_i^T x \leqslant b_i$ is satisfied.

Famous Problems

Assignment problem

- Optimal task allocation: There are n people available and n tasks to be completed. Each person is assigned exactly one task. We want all tasks to be assigned. The cost of performing task j by person i is c_{ij}.
- We're looking for a task assignment that minimizes the total cost.
- **Equivalent problem**: In a square matrix, choose exactly one element per row and one element per column. The sum of the elements chosen is minimum.

Assignment problem - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ does task } j, \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

Assignment problem - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ does task } j, \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

Assignment problem - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ does task } j, \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

Assignment problem - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ does task } j, \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

$$\sum_{j=1}^{n} x_{ij} = 1 \; \forall i, \qquad \text{(each person has a task)}$$

Assignment problem - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ does task } j, \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

$$\sum_{j=1}^{n} x_{ij} = 1 \ \forall i, \qquad \text{(each person has a task)}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \ \forall j, \qquad \text{(every task is completed)}$$

Assignment problem - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ does task } j, \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

$$\sum_{j=1}^{n} x_{ij} = 1 \ \forall i, \qquad \text{(each person has a task)}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \ \forall j, \qquad \text{(every task is completed)}$$

$$x_{ij} \in \{0, 1\} \text{ for all } i, j.$$

Assignment problem - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ does task } j, \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

$$\sum_{j=1}^{n} x_{ij} = 1 \ \forall i, \qquad \text{(each person has a task)}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \ \forall j, \qquad \text{(every task is completed)}$$

$$x_{ij} \in \{0, 1\} \text{ for all } i, j.$$

- **Objective function**: total cost is minimum

$$\min_{x} \sum_{i,j} c_{ij} x_{ij}.$$

Knapsack problem

- We fill a backpack with objects.
- There are n objects that can fit in the bag.
- Object i is of weight a_i and provides satisfaction c_i (for example, its price).
- Among the sets of objects whose sum of weights does not exceed b, find those for which the sum of satisfactions is maximum.

Knapsack problem

- **Choice of variables**:

Knapsack problem

- **Choice of variables**:

$$x_i = \begin{cases} 1 & \text{if product } i \text{ is included,} \\ 0 & \text{otherwise.} \end{cases}$$

Knapsack problem

- **Choice of variables**:

$$x_i = \begin{cases} 1 & \text{if product } i \text{ is included,} \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

Knapsack problem

- **Choice of variables**:

$$x_i = \begin{cases} 1 & \text{if product } i \text{ is included,} \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

$$\sum_{i=1}^{n} a_i x_i \leqslant b \; \forall i, \qquad \text{(backpack capacity)}$$

Knapsack problem

- **Choice of variables**:

$$x_i = \begin{cases} 1 & \text{if product } i \text{ is included,} \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

$$\sum_{i=1}^{n} a_i x_i \leqslant b \ \forall i, \qquad \text{(backpack capacity)}$$

$$x \in \{0,1\}^n.$$

Knapsack problem

- **Choice of variables**:

$$x_i = \begin{cases} 1 & \text{if product } i \text{ is included,} \\ 0 & \text{otherwise.} \end{cases}$$

- **Constraints**:

$$\sum_{i=1}^{n} a_i x_i \leqslant b \ \forall i, \qquad \text{(backpack capacity)}$$

$$x \in \{0,1\}^n.$$

- **Objective function**: maximum overall satisfaction

$$\max_x \sum_{i=1}^{n} c_i x_i.$$

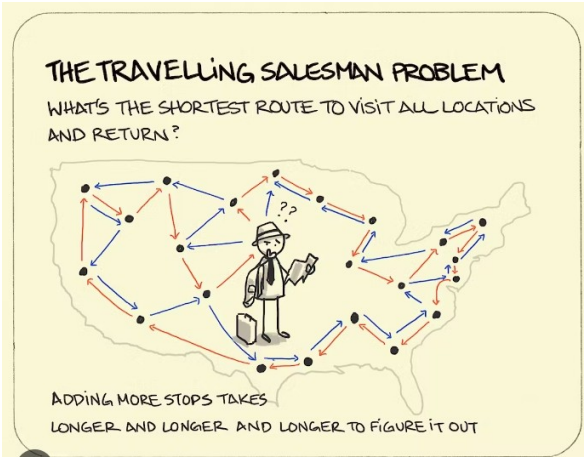
Traveling salesman problem

- A travelling salesman visits n cities.
- It passes through each town exactly once and finally returns to its starting town.
- The cost (time, distance, etc.) of travelling from city i to city j is c_{ij}. Find the route that minimizes the sum of these costs.

Traveling salesman problem

- A travelling salesman visits n cities.
- It passes through each town exactly once and finally returns to its starting town.
- The cost (time, distance, etc.) of travelling from city i to city j is c_{ij}. Find the route that minimizes the sum of these costs.
- This is probably the best-known optimization problem. The problem is simple to explain and it calls for a solution.
- The problem manifests itself in many ways. It could be the choice of route for a deliveryman, letter carrier or home nurse, or the programming of a machine that places components on a printed circuit board, and so on.

Traveling salesman problem



The history of TSP

- 1920's: The mathematician and economist Karl Menger publicizes it among his colleagues in Vienna.
- 1930's: The problem reappears in the mathematical circles of Princeton.
- 1940's: The problem is studied by statisticians in connection with an agricultural application and the mathematician Merill Flood popularizes it among his colleagues at the RAND Corporation. Eventually, the TSP gained notoriety as the prototype of a hard problem in combinatorial optimization: examining the tours one by one is out of the question because of their large number, and no other idea was on the horizon for a long time.
- 1954: George Dantzig and co-authors publish a description of a method for solving the TSP and illustrate the power of this method by solving an instance with 49 cities, an impressive size at that time. They created this instance by picking one city from each of the 48 states in the U.S.A. (Alaska and Hawaii became states only in 1959) and adding Washington, D.C.; the costs of travel between these cities were defined by road distances.

The history of TSP

- 1962: Proctor and Gamble runs a contest requiring solving a TSP on a specified 33 cities. There was a tie between many people who found the optimum.
- 1977: Groetschel finds the optimal tour of 120 cities from what was then West Germany.
- 1987: Padberg and Rinaldi find the optimal tour of 532 AT&T switch locations in the USA.
- 1998: Applegate, Bixby, Chvatal, and Cook find the optimal tour of the 13,509 cities in the USA with populations > 500.
- From: `http://www.keck.caam.rice.edu/tsp/`. Applegaten, Bixby, Chvatal, Vasek and Cook, On the solution of traveling salesman problems. Proceedings of the International Congress of Mathematicians, Vol. III (Berlin, 1998).

TSP - formulation

- **Choice of variables**:

TSP - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if the traveller goes directly from } i \text{ to } j. \\ 0 & \text{otherwise.} \end{cases}$$

(No variable x_{ii}.)

TSP - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if the traveller goes directly from } i \text{ to } j. \\ 0 & \text{otherwise.} \end{cases}$$

  (No variable x_{ii}.)

- **Constraints**:

TSP - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if the traveller goes directly from } i \text{ to } j. \\ 0 & \text{otherwise.} \end{cases}$$

(No variable x_{ii}.)

- **Constraints**:

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \ \forall i, \qquad \text{(leaves the city } i \text{ once)}$$

TSP - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if the traveller goes directly from } i \text{ to } j. \\ 0 & \text{otherwise.} \end{cases}$$

(No variable x_{ii}.)

- **Constraints**:

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \ \forall i, \qquad \text{(leaves the city } i \text{ once)}$$

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \ \forall i, \qquad \text{(arrives in city } j \text{ once)}$$

TSP - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if the traveller goes directly from } i \text{ to } j. \\ 0 & \text{otherwise.} \end{cases}$$

(No variable x_{ii}.)

- **Constraints**:

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \ \forall i, \qquad \text{(leaves the city } i \text{ once)}$$

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \ \forall i, \qquad \text{(arrives in city } j \text{ once)}$$

- **Objective function**: total cost is minimized

$$\min_{x} \sum_{i,j=1}^{n} c_{ij} x_{ij}.$$

TSP - formulation

- **Choice of variables**:

$$x_{ij} = \begin{cases} 1 & \text{if the traveller goes directly from } i \text{ to } j. \\ 0 & \text{otherwise.} \end{cases}$$

(No variable x_{ii}.)

- **Constraints**:

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1 \ \forall i, \qquad \text{(leaves the city } i \text{ once)}$$

$$\sum_{i=1, i\neq j}^{n} x_{ij} = 1 \ \forall i, \qquad \text{(arrives in city } j \text{ once)}$$

- **Objective function**: total cost is minimized

$$\min_{x} \sum_{i,j=1}^{n} c_{ij} x_{ij}.$$

But that's not enough...

TSP - formulation

- **Cutting constraints**. Whatever the partition $S_1 \cup S_2 = \{1, 2, \ldots, n\}$ with $S_1 \cap S_2 = \varnothing$ and $S_1, S_2 \neq \varnothing$, there must be an arc between S_1 and S_2

$$\sum_{i \in S_1} \sum_{j \in S_2} x_{ij} \geqslant 1.$$

[1] AKA the Dantzig–Fulkerson–Johnson formulation

TSP - formulation

- **Cutting constraints**. Whatever the partition $S_1 \cup S_2 = \{1, 2, \ldots, n\}$ with $S_1 \cap S_2 = \varnothing$ and $S_1, S_2 \neq \varnothing$, there must be an arc between S_1 and S_2

$$\sum_{i \in S_1} \sum_{j \in S_2} x_{ij} \geqslant 1.$$

- **Subtour elimination constraint**[1]: There are no circuits smaller than n. Whatever the subset S with $2 \leqslant |S| \leqslant n - 1$ is

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leqslant |S| - 1.$$

[1] AKA the Dantzig–Fulkerson–Johnson formulation

TSP - formulation

- **Cutting constraints**. Whatever the partition $S_1 \cup S_2 = \{1, 2, \ldots, n\}$ with $S_1 \cap S_2 = \varnothing$ and $S_1, S_2 \neq \varnothing$, there must be an arc between S_1 and S_2

$$\sum_{i \in S_1} \sum_{j \in S_2} x_{ij} \geqslant 1.$$

- **Subtour elimination constraint**[1]: There are no circuits smaller than n. Whatever the subset S with $2 \leqslant |S| \leqslant n - 1$ is

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leqslant |S| - 1.$$

  - ensures that no proper subset S can form a sub-tour, so the solution returned is a single tour and not the union of smaller tours.
  - leads to an exponential number of possible constraints (why?).

[1] AKA the Dantzig–Fulkerson–Johnson formulation

TSP - formulation

- **Cutting constraints**. Whatever the partition $S_1 \cup S_2 = \{1, 2, \ldots, n\}$ with $S_1 \cap S_2 = \varnothing$ and $S_1, S_2 \neq \varnothing$, there must be an arc between S_1 and S_2

$$\sum_{i \in S_1} \sum_{j \in S_2} x_{ij} \geqslant 1.$$

- **Subtour elimination constraint**[1]: There are no circuits smaller than n. Whatever the subset S with $2 \leqslant |S| \leqslant n - 1$ is

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leqslant |S| - 1.$$

  - ensures that no proper subset S can form a sub-tour, so the solution returned is a single tour and not the union of smaller tours.
  - leads to an exponential number of possible constraints (why?).

There are other, more effective formulations such the *Miller–Tucker–Zemlin formulation.*

[1] AKA the Dantzig–Fulkerson–Johnson formulation

Enumeration

All these problems can be solved by *enumeration*, since the set of feasible solutions is finite in each case. It is finite, but it is large...

[2] $(n-1)!/2$ can also be found in the literature by removing equivalent circuits of opposite directions.

Enumeration

All these problems can be solved by *enumeration*, since the set of feasible solutions is finite in each case. It is finite, but it is large...

- **Assignment Problem**: There are n people and n tasks. There are therefore $n!$ assignments possible.

[2] $(n-1)!/2$ can also be found in the literature by removing equivalent circuits of opposite directions.

Enumeration

All these problems can be solved by *enumeration*, since the set of feasible solutions is finite in each case. It is finite, but it is large...

- **Assignment Problem**: There are n people and n tasks. There are therefore $n!$ assignments possible.
- **Knapsack Problem**: There are n objects and each object can be chosen or left out. There are 2^n possible choices.

[2] $(n-1)!/2$ can also be found in the literature by removing equivalent circuits of opposite directions.

Enumeration

All these problems can be solved by *enumeration*, since the set of feasible solutions is finite in each case. It is finite, but it is large...

- **Assignment Problem**: There are n people and n tasks. There are therefore $n!$ assignments possible.
- **Knapsack Problem**: There are n objects and each object can be chosen or left out. There are 2^n possible choices.
- **TSP Problem**: Starting from city 1, there are $n-1$ possible choices, then $n-2$, and so on. So there are $(n-1)!$[2] possible circuits in all. In a complete graph with 101 nodes, there are $100! \approx 9.3310^{157}$ possible circuits.

[2] $(n-1)!/2$ can also be found in the literature by removing equivalent circuits of opposite directions.

Relaxation

Consider the problem with integers

$$\begin{aligned} \max_x \quad & c^T x \\ \text{such that} \quad & Ax \leqslant b \\ & x \geqslant 0 \text{ and } x \text{ integer.} \end{aligned}$$

We construct the relaxation of this problem by eliminating the 'x integer' constraint:

$$\begin{aligned} \max_x \quad & c^T x \\ \text{such that} \quad & Ax \leqslant b \\ & x \geqslant 0. \end{aligned}$$

Crucial observations:

- If the relaxed problem has no feasible solution, **neither** does the original problem.
- An optimal solution of the relaxed problem provides an **upper bound**[3] on the optimal objective of the initial problem.
- If the optimal solution of the relaxed problem is integer, then this solution **is also** optimal for the original problem.

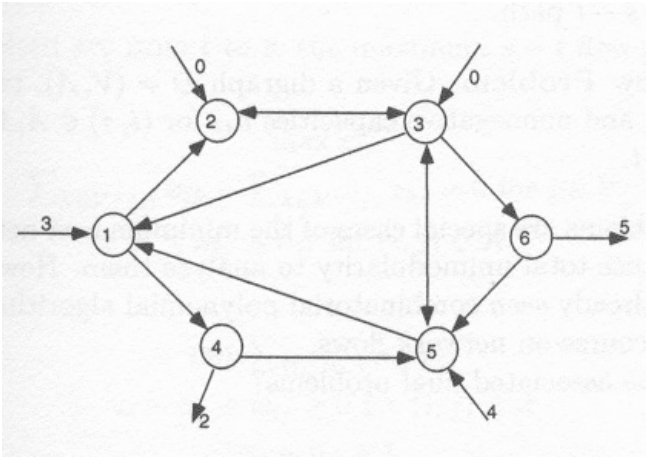
[3] for a maximization problem, otherwise it is a lower bound

Min cost flow

Transmission in a communication network.

- Consider a directed graph given by a set of nodes V and edges E.
- At each node i, a quantity b_i enters ($b_i > 0$) or leaves ($b_i < 0$).
- We assume that incoming and outgoing quantities balance out, that is $\sum_i b_i = 0$.
- The arc (i, j) has a maximum capacity of h_{ij} and unit cost c_{ij}. We wish to find a feasible stream with minimum cost.

Min cost flow



Min cost flow

We define:

$V^+(i) = \{k | (i,k) \in E\}$. (nodes whose edge leaves i)
$V^-(i) = \{k | (k,i) \in E\}$. (nodes whose edge goes to i)

- **Choice of variables**:

Min cost flow

We define:

$V^+(i) = \{k|(i,k) \in E\}$. (nodes whose edge leaves i)
$V^-(i) = \{k|(k,i) \in E\}$. (nodes whose edge goes to i)

- **Choice of variables**: x_{ij} is the flow in the edge (i,j).

Min cost flow

We define:

$V^+(i) = \{k|(i,k) \in E\}$. (nodes whose edge leaves i)
$V^-(i) = \{k|(k,i) \in E\}$. (nodes whose edge goes to i)

- **Choice of variables**: x_{ij} is the flow in the edge (i,j).
- **Constraints**: The flow is preserved at each node:

Min cost flow

We define:

$V^+(i) = \{k | (i,k) \in E\}$. (nodes whose edge leaves i)
$V^-(i) = \{k | (k,i) \in E\}$. (nodes whose edge goes to i)

- **Choice of variables**: x_{ij} is the flow in the edge (i,j).
- **Constraints**: The flow is preserved at each node:

$$\sum_{k \in V^+(i)}^{n} x_{ik} - \sum_{k \in V^-(i)}^{n} x_{ki} = b_i \; \forall \, i \in V.$$

Min cost flow

We define:

$V^+(i) = \{k|(i,k) \in E\}$. (nodes whose edge leaves i)
$V^-(i) = \{k|(k,i) \in E\}$. (nodes whose edge goes to i)

- **Choice of variables**: x_{ij} is the flow in the edge (i,j).
- **Constraints**: The flow is preserved at each node:

$$\sum_{k \in V^+(i)}^{n} x_{ik} - \sum_{k \in V^-(i)}^{n} x_{ki} = b_i \; \forall \, i \in V.$$

Maximum capacities are not exceeded:

$$0 \leqslant x_{ij} \leqslant h_{ij} \quad \forall \, (i,j) \in E.$$

Min cost flow

We define:

$V^+(i) = \{k|(i,k) \in E\}$. (nodes whose edge leaves i)
$V^-(i) = \{k|(k,i) \in E\}$. (nodes whose edge goes to i)

- **Choice of variables**: x_{ij} is the flow in the edge (i,j).
- **Constraints**: The flow is preserved at each node:

$$\sum_{k \in V^+(i)}^{n} x_{ik} - \sum_{k \in V^-(i)}^{n} x_{ki} = b_i \; \forall \, i \in V.$$

  Maximum capacities are not exceeded:

$$0 \leqslant x_{ij} \leqslant h_{ij} \quad \forall \, (i,j) \in E.$$

- **Objective Function**: total cost is minimized

$$\min_x \sum_{(i,j) \in E} c_{ij} x_{ij}.$$

Min cost flow

Theorem

If the demands b_i and capacities h_{ij} of a minimum-cost flow problem are integer, then there is an integer optimal solution.

Proof sketch The problem can be written as

$$\min_x \quad c^T x \quad \text{such that} \quad \begin{array}{l} Ax \leqslant b, \\ x \geqslant 0 \text{ and } x \text{ integer}, \end{array}$$

where the entries of A and b are integers. The relaxation of this problem is

$$\min_x \quad c^T x \quad \text{such that} \quad \begin{array}{l} Ax \leqslant b, \\ x \geqslant 0. \end{array}$$

The basic feasible solutions of this problem can be written as (after permutation)
$x = (x_B, x_N) = (A_B^{-1} b, 0)$ where A_B is a submatrix of $[A, I]$.
These sub-matrices all have a determinant equal to ± 1. (These are unmodular matrices; see for example `http://fr.wikipedia.org/wiki/Matrice_unimodulaire`).

Optimal transport problem

- A product is transported from m origins to n destinations.
- The product is available in quantities $a_1, a_2, \ldots, a_m$ at origins and the demands at destinations are $b_1, b_2, \ldots, b_n$.
- The cost of transporting one unit of product from origin i to destination j is c_{ij} (for example, the distance between i and j).
- We want to determine the quantities of product to be transported from i to j so as to satisfy demand while minimizing total transport costs.

Assignment problem

- Optimal task allocation: There are n people available and n tasks to be completed.
- Each person is assigned exactly one task. We want all tasks to be assigned.
- The cost of performing task j by person i is c_{ij}.
- We're looking for a task assignment that minimizes the total cost.

This problem is a transport problem with a bipartite graph ! Why ? for which a_i and b_i ?

The assignment Problem can be seen as special case of minimum-flow problems;

- What is the value for each supply and demand[4] ?
- What should be the edge capacity ?

[4] Still consider a bipartite graph

Shortest path problem

The length of the shortest path between two nodes in a graph can be obtained by solving a minimum-cost flow problem.

- Let $G = (V, E)$ be a graph. We are looking for the length of the shortest path between nodes k and l.
- Let $b_k = 1$, $b_l = -1$ and $b_i = 0$ for $i \neq k, l$.[5]
- Give all edges infinite capacity.
- In the case we define unit costs equal to 1 for all edges, then the cost of the minimum-cost flow is equal to the length of the shortest path from k to l.

[5] i.e., we send one unit of flow from a designated source k to a designated sink l.

Complexity

If the data is integer, for problems:

- Minimum cost flow problem;
- Transport problem;
- Assignment problem;
- Shortest path problem;
- Maximum flow problem;

there is always an optimal solution to the relaxed problem which is integer !
However, this is not always the case with problems:

- Backpack problem;
- Partitioning/clustering problem;
- Covering problem;
- travelling salesman problem; etc.

What to do about them?

Branch and Bound

Upper and Lower-bounds

Consider the problem in integers

$$\begin{aligned} \max_{x} \quad & c^T x \\ \text{such that} \quad & Ax \leqslant b, \\ & x \geqslant 0 \text{ and } x \text{ integer}. \end{aligned}$$

Any feasible solution $x^\dagger$ provides a bound on the optimum: $\underline{f} = c^T x^\dagger \leqslant f^*$.

Consider now the relaxed problem

$$\begin{aligned} \max_{x} \quad & c^T x \\ \text{such that} \quad & Ax \leqslant b, \\ & x \geqslant 0. \end{aligned}$$

Any optimal solution x^* provides an upper-bound of the optimum: $\bar{f} = c^T x^* \geqslant f^*$.

Divide and Conquer

- Let $z = \max_{x \in S} c^T x$.

Divide and Conquer

- Let $z = \max_{x \in S} c^T x$.
- To find the solution, we divide the set S into two subsets $S = S_1 \cup S_2$, and try to solve the separate problems:

$$z_1 = \max_{x \in S_1} c^T x \quad \text{and} \quad z_2 = \max_{x \in S_2} c^T x.$$

Since $S = S_1 \cup S_2$, we have $z = \max(z_1, z_2)$.

Divide and Conquer

- Let $z = \max_{x \in S} c^T x$.
- To find the solution, we divide the set S into two subsets $S = S_1 \cup S_2$, and try to solve the separate problems:

$$z_1 = \max_{x \in S_1} c^T x \quad \text{and} \quad z_2 = \max_{x \in S_2} c^T x.$$

  Since $S = S_1 \cup S_2$, we have $z = \max(z_1, z_2)$.
- In general, we don't have exact expressions for z_1 and z_2, but we do have upper and lower bounds: $\underline{z_1} \leqslant z_1 \leqslant \bar{z_1}$ and $\underline{z_2} \leqslant z_2 \leqslant \bar{z_2}$.

Divide and Conquer

- Let $z = \max_{x \in S} c^T x$.
- To find the solution, we divide the set S into two subsets $S = S_1 \cup S_2$, and try to solve the separate problems:

$$z_1 = \max_{x \in S_1} c^T x \quad \text{and} \quad z_2 = \max_{x \in S_2} c^T x.$$

  Since $S = S_1 \cup S_2$, we have $z = \max(z_1, z_2)$.
- In general, we don't have exact expressions for z_1 and z_2, but we do have upper and lower bounds: $\underline{z_1} \leqslant z_1 \leqslant \bar{z_1}$ and $\underline{z_2} \leqslant z_2 \leqslant \bar{z_2}$.
- Lower bounds are obtained by feasible solutions, upper bounds by relaxation.

Divide and Conquer

- Let $z = \max_{x \in S} c^T x$.
- To find the solution, we divide the set S into two subsets $S = S_1 \cup S_2$, and try to solve the separate problems:

$$z_1 = \max_{x \in S_1} c^T x \quad \text{and} \quad z_2 = \max_{x \in S_2} c^T x.$$

  Since $S = S_1 \cup S_2$, we have $z = \max(z_1, z_2)$.
- In general, we don't have exact expressions for z_1 and z_2, but we do have upper and lower bounds: $\underline{z_1} \leqslant z_1 \leqslant \bar{z_1}$ and $\underline{z_2} \leqslant z_2 \leqslant \bar{z_2}$.
- Lower bounds are obtained by feasible solutions, upper bounds by relaxation.
- Since $z = \max(z_1, z_2)$, we have

$$\max(\underline{z_1}, \underline{z_2}) \leqslant z \leqslant \max(\bar{z_1}, \bar{z_2}).$$

Branch and Bound: Example 1

How to solve the problem

$$\begin{aligned} \max_{x} \quad & x_1 + 10x_2 \\ \text{such that} \quad & 10x_1 + 3x_2 \leqslant 27, \\ & -x_1 + 3x_2 \leqslant 12, \\ & x_1, x_2 \geqslant 0 \text{ and integer?} \end{aligned}$$

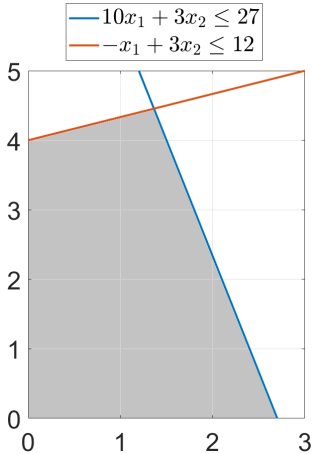
Branch and Bound: Example 1

How to solve the problem

$$\begin{aligned} \max_{x} \quad & x_1 + 10x_2 \\ \text{such that} \quad & 10x_1 + 3x_2 \leqslant 27, \\ & -x_1 + 3x_2 \leqslant 12, \\ & x_1, x_2 \geqslant 0 \text{ and integer?} \end{aligned}$$

and with the objective of $9x_1 + 3x_2$?

Branch and Bound: Example 1

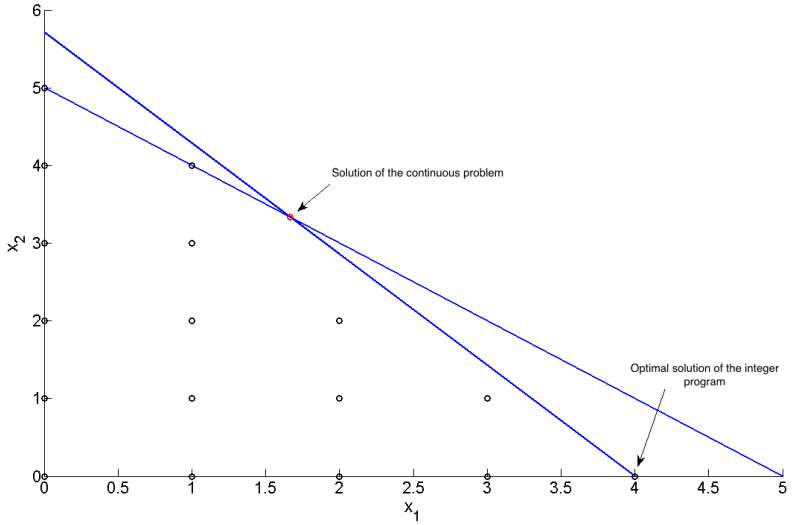


Branch and Bound: Example 2

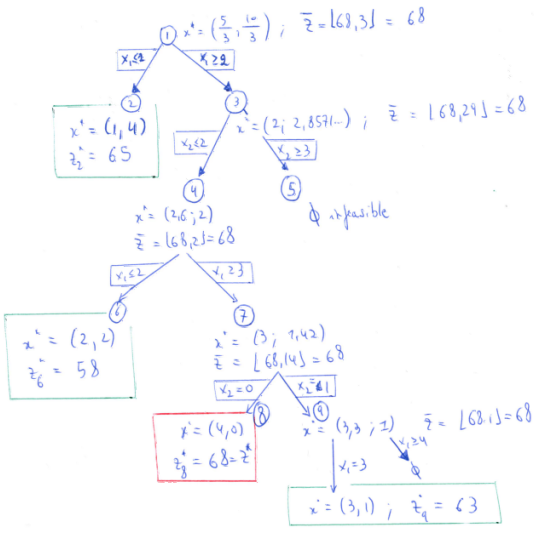
How to solve the problem

$$\begin{aligned} \max_{x} \quad & 17x_1 + 12x_2 \\ \text{such that} \quad & 10x_1 + 7x_2 \leqslant 40, \\ & x_1 + x_2 \leqslant 5, \\ & x_1, x_2 \geqslant 0 \text{ and integer?} \end{aligned}$$

Branch and Bound: Example 2



Branch and Bound: Example 2



see colab file - Section 2

Branch and Bound

- The process can be continued, leading to the construction of an enumeration tree.

Branch and Bound

- The process can be continued, leading to the construction of an enumeration tree.
- The tree-building process does not have to be continued at every node.

Branch and Bound

- The process can be continued, leading to the construction of an enumeration tree.
- The tree-building process does not have to be continued at every node. There are a number of situations where simplification is required.
  1. Optimality. If for a given node we have $\underline{z} = \bar{z}$, the optimal cost for the node is known and the division process stops.
  2. Sub-Optimality. If an upper bound at node X is lower than the lower bound at another node, then the process stops at node X.
  3. Empty feasible set. After decomposing a set S into two sets, one of the sets may be empty. The process stops at the associated empty set node.

Building the tree

How to build the tree?

Building the tree

How to build the tree?

1. How to choose the splittings $S = S_1 \cup S_2$?
   A simple strategy is to choose the variable whose fractional part is closest to $1/2$.

Building the tree

How to build the tree?

1. How to choose the splittings $S = S_1 \cup S_2$?
   A simple strategy is to choose the variable whose fractional part is closest to $1/2$.
2. In what order should you choose the nodes to explore?

Choice of node order

1. Width search. Solve all problems on a given level before continuing.

Choice of node order

1. Width search. Solve all problems on a given level before continuing.

Choice of node order

1. Width search. Solve all problems on a given level before continuing.
2. In-depth search.
   - We go down the tree as quickly as possible to obtain a feasible solution.
   - There's an obvious interest in obtaining a feasible solution *quickly*, since such a solution allows us to prune all branches with weak upper bounds !
   - Moreover, a feasible solution allows us to stop the process at any point and propose something :).
   - Finally, in a depth-first search, successive linear programs are obtained by adding or refining bounds for a variable. The dual simplex method is particularly effective in dealing with this kind of modification.

Choice of node order

1. Width search. Solve all problems on a given level before continuing.
2. In-depth search.
   - We go down the tree as quickly as possible to obtain a feasible solution.
   - There's an obvious interest in obtaining a feasible solution *quickly*, since such a solution allows us to prune all branches with weak upper bounds !
   - Moreover, a feasible solution allows us to stop the process at any point and propose something :).
   - Finally, in a depth-first search, successive linear programs are obtained by adding or refining bounds for a variable. The dual simplex method is particularly effective in dealing with this kind of modification.
3. Best node strategy. Choose a node for which the upper bound is the highest. In this way, we never develop a node whose upper bound is lower than the optimum.

Example: backpack problem

- **Context**: We have a bag capable of supporting a maximum weight of 8kg and a set of objects, each with a weight (in kg) and a value (in rubles).
- **Aim**: to fill the bag with objects without exceeding the maximum weight, while maximizing the sum of the values of the objects it contains.
- You have 5 objects weighing 4, 3, 3, 2 and 2 kg respectively and with values of 3, 5, 2, 2 and 3 rubles respectively.

Example: backpack problem

- **Context**: We have a bag capable of supporting a maximum weight of 8kg and a set of objects, each with a weight (in kg) and a value (in rubles).
- **Aim**: to fill the bag with objects without exceeding the maximum weight, while maximizing the sum of the values of the objects it contains.
- You have 5 objects weighing 4, 3, 3, 2 and 2 kg respectively and with values of 3, 5, 2, 2 and 3 rubles respectively.
- A binary variable is introduced for each object ($x_i = 1$ if object i is used, otherwise $x_i = 0$):

$$\begin{aligned} \max_{x \in \mathbb{R}^5} \quad & 3x_1 + 5x_2 + 2x_3 + 2x_4 + 3x_5 \\ \text{such that} \quad & 4x_1 + 3x_2 + 3x_3 + 2x_4 + 2x_5 \leqslant 8, \\ & x_i \in \{0, 1\} \text{ pour } 1 \leqslant i \leqslant 5. \end{aligned}$$

Linear relaxation (example)

- We relax $x_i \in \{0,1\}$ by $0 \leqslant x_i \leqslant 1$ for all i :

$$\begin{aligned} f^* \;=\; \max_{x \in \mathbb{R}^5} \quad & 3x_1 + 5x_2 + 2x_3 + 2x_4 + 3x_5 \\ \text{such that} \quad & 4x_1 + 3x_2 + 3x_3 + 2x_4 + 2x_5 \leqslant 8, \\ & 0 \leqslant x_i \leqslant 1 \text{ pour } 1 \leqslant i \leqslant 5. \end{aligned}$$

- To calculate the solution, we need to rank the objects in order of preference. This order is given by the value per unit weight: $2 \succ 5 \succ 4 \succ 1 \succ 3$.
- Indeed, we have:

$$\frac{5}{3} \geqslant \frac{3}{2} \geqslant \frac{2}{2} \geqslant \frac{3}{4} \geqslant \frac{2}{3}.$$

It is therefore in our interest to first take object 2, then 5, then 4, then 1, then 3. The bag is filled in this order until it is full: the solution to the relaxed problem is

$$x^* = (1/4, 1, 0, 1, 1).$$

Branch and Bound (example)

- The solution to the relaxed problem is

$$x^* = (1/4, 1, 0, 1, 1),$$

  with $f(x^*) = \bar{f} = 10.75$.

- This implies that the optimal value f^* of the problem in binary numbers will be smaller (because the problem is more constrained, and we're maximizing): $\bar{f} = 10.75 \geqslant f^*$.
- Moreover, since f^* must be an integer, we actually know that $f^* \leqslant 10$.
- We can now begin the Branch and Bound procedure: we solve two linear relaxations in order to 'get rid of' the non-integer x_1^* variable:
  - Node $x_1 = 0$: optimal solution (0 , 1 , 1/3 , 1 , 1) with $f^* = 10.66$.
    - Node $x_1 = 0, x_3 = 0$: (0,1,0,1,1) with $f^* = 10 \Rightarrow$ optimal solution.
    - Node $x_1 = 0, x_3 = 1$: (0,1,1,0,1) with $f^* = 10 \Rightarrow$ optimal solution.
  - Node $x_1 = 1$: optimal solution (1 , 1 , 0 , 0 , 1/2) with $f^* = 9.5$: we can stop, we've already found a strictly better solution.

Exercise

- You have 96 rubles at your disposal. You would like to buy beverages, available in various packagings:

Beverages	A	B	C	D	E
Unit volume (liters)	15	16	17	18	19
Unit price (Rubles)	16	17	18	19	20

- You want to buy as large a volume as possible, without your budget :
  1. Formulate this problem as an integer linear program.
  2. How many packages of each type will you buy? ? Detail your reasoning.

Conclusions

Summary

We have seen

- What is the impact of *integer* constraints on the optimal solutions before and after their integration $\Rightarrow$ solutions are very different !
- How to model different kind of constraints for $x_i \in \{0, 1\}$: *Exclusive*($x_i + x_j \leqslant 1$), *Alternativity* constraints (x_i or x_j), and *Contingency* constraints ($x_j \leqslant x_i$).
- A set of important problems involving integer variables:
  - a subset of problems that can be solved "efficiently"[6] thanks to the nature of the data (integers) and a fundamental feature of the sub-matrices A_B: they are **unimodular** : Min-cost flow, transport problem, assignment problem, shortest path problem, etc.
  - a subset of hardcore problems :): knapsack, clustering, TSP, etc.
- The *Branch and Bound* Algorithm: build an enumeration tree with 3 rules for efficient pruning, and at each node we have a *relaxed* problem to solve.

[6] there is always an optimal solution to the relaxed problem which is integer

Preparations for the next lecture

- Review the lecture :).
- Nothing else, this lecture was the last one for the first part of the course dedicated to linear programming: congratulations !

Goodbye, So Soon

THANKS FOR THE ATTENTION

- v.leplat@innopolis.ru
- sites.google.com/view/valentinleplat/