Introduction to Integer Linear Programming, reformulation and decomposition

Sandra Ulrich Ngueveu

CIMPA School 2021: July 6th - 8th, 2021

Université de Toulouse - Toulouse INP - Laboratory LAAS-CNRS
ngueveu@laas.fr

Prerequisites

You already know

- modeling with continuous variables and linear constraints
- basic facts about polyhedra
- (how the simplex algorithm works)
- (a bit about linear programming duality)

Goals of this course

We plan to

- introduce you to the basics of (Mixed) Integer Linear Programming (part 1)
- illustrate how MILP can be used to solve Mixed Integer Nonlinear Problems via piecewise linear approximation (part 2)
- introduce you to principles of the column generation and branch-and-price algorithms (part 3)

With the aim of

- expanding your modeling and solving capabilities

Part 1: basics

Content

Introduction

An illustrative example

From linear to integer linear programming

- Key differences
- Combinatorial explosion
- Complexity ?

Branch-and-Bound (divide and conquer)

Valid inequalities

Introduction

(Mixed) Integer linear programming

We are interested in optimization problems that can be modeled as follows:

$$\min f(x) \quad (1)$$

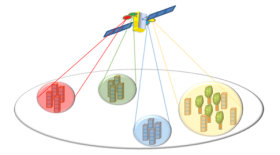
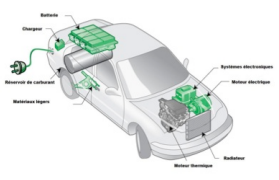
s.t.

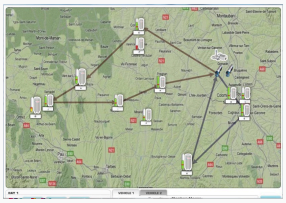
$$g(x) \leq 0 \quad (2)$$

$$x \in X, X \subset \mathbb{R}^{n1} \times \mathbb{Z}^{n2} \quad (3)$$

	$n2 \neq 0$	
	$n1 = 0$	$n1 \neq 0$
linear f, g	ILP	MILP
nonlinear f,g	INLP	MINLP

Numerous fields of application



An illustrative example

Problem (P): cement manufacturing

	Cement A	Cement B
selling price (coins)	100	110
duration per ton in the industrial oven (min)	100	120
duration per ton at the grinding station (min)	50	200

The industrial oven is available 590 minutes per day. The grinding station is available 750 minutes per day. What quantity cement of each type should we produce per day to maximise the earnings ?

Variables

Objective-function

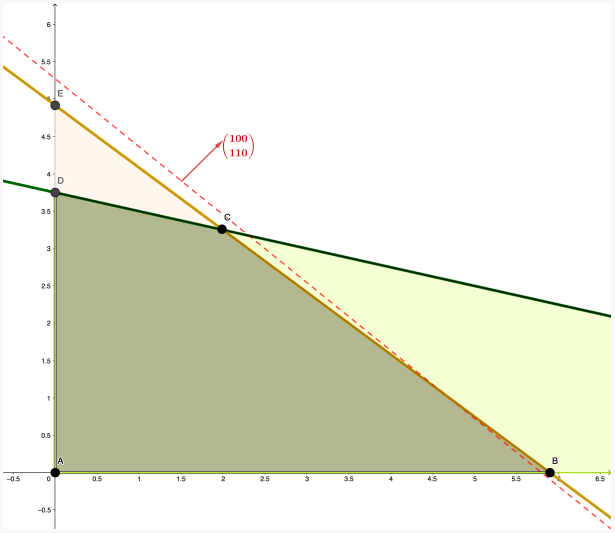
Constraints

Domain

Resulting mathematical model ?

Problem (P): cement manufacturing

Graphical representation



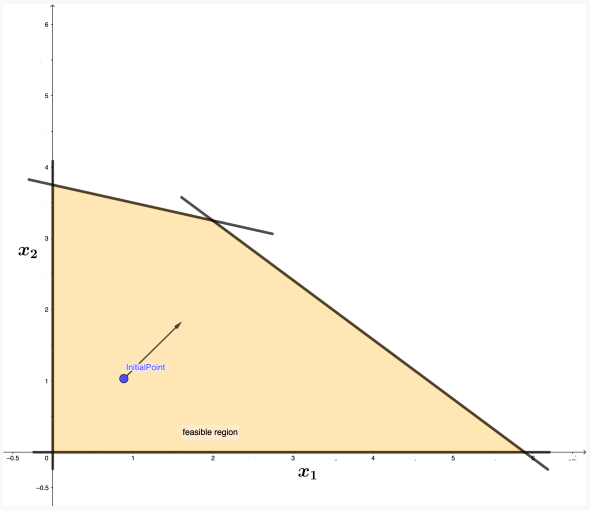
set of feasible solutions ?

polyhedron (convex) ?

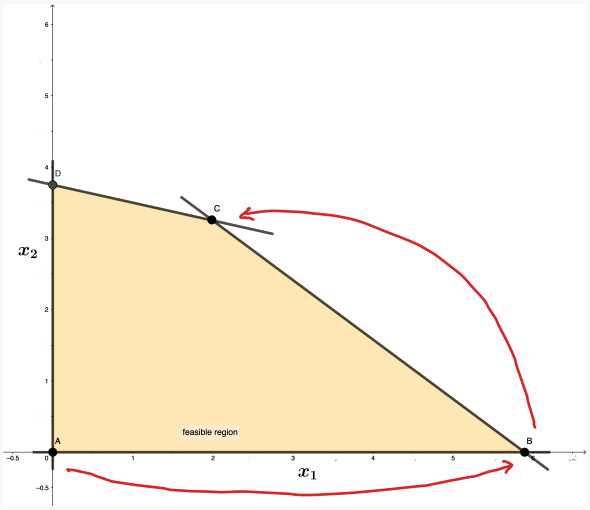
extremal vs interior points ?

optimal solution ?

Two classical families of LP solution methods



Interior point methods



Simplex method

Linear programming

$$\min \sum_{i=1}^{n} c_i x_i \tag{4}$$

s.t.

$$a_{1,j}x_1 + \ldots + a_{i,j}x_i + \ldots + a_{n,j}x_n \leq b_j, \quad \forall j \in \{1 \ldots m\} \tag{5}$$

$$x_i \in \mathbb{R}, \quad \forall i \in \{1 \ldots n\} \tag{6}$$

where $c_i, b_j, a_{i,j}$ are scalars

Convenience of LP: Local optimum = global optimum

Limits: To be modelled with an LP, the consequences of the decisions modelled by the variables have to be verify

- additivity
- proportionality
- divisibility

From linear to integer linear programming

Content

Introduction

An illustrative example

From linear to integer linear programming

- Key differences
- Combinatorial explosion
- Complexity ?

Branch-and-Bound (divide and conquer)

Valid inequalities

Difference between the discrete and the continuous case

Problem (P1) = (P) + new rules:

- cements A and B must be stored in separate bags of 1kg
- each bag used must be full at the end of each day

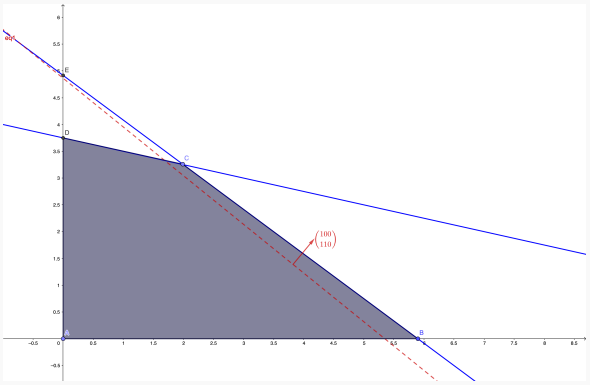
$$(P1)\ \min\ 100x_A + 110x_B \qquad (7)$$

s.t.

$$100x_A + 120x_B \leq 590 \qquad (8)$$

$$50x_A + 200x_B \leq 750 \qquad (9)$$

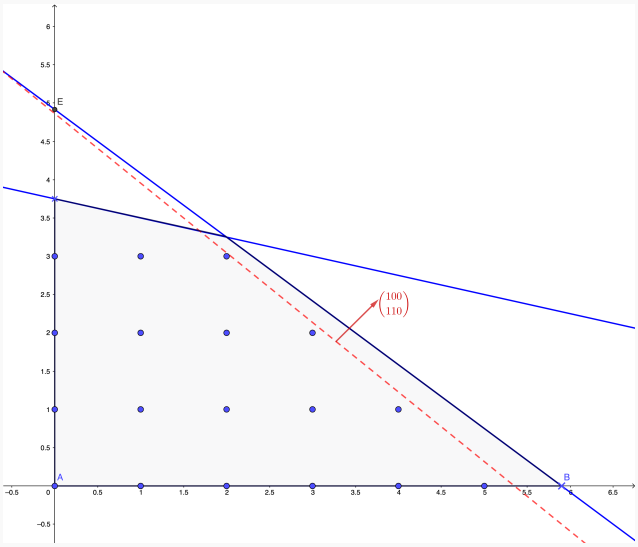
$$x_A, x_B \in \mathbb{N} \qquad (10)$$



- The optimal solution of (P) is not feasible for(P1)
- The location of the optimal solution of (P1) is not obvious

Problem (P1): cement manufacturing with storage in 1 kgs bags

Graphical representation of problem (P1)



set of feasible solutions ?

polyhedron (convex) ?

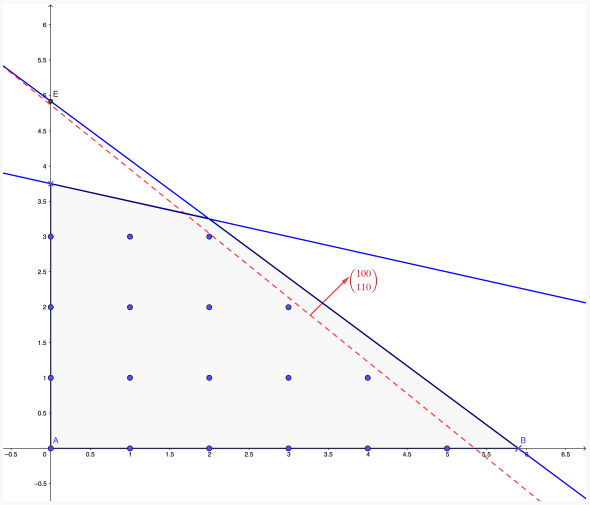
extremal vs interior points ?

optimal solution ?

Can’t we just round LP solutions to nearest integers ?

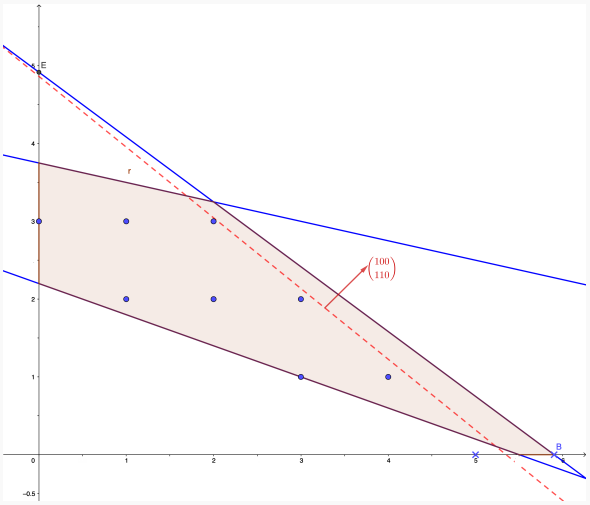
Why not solve the LP relaxation and round the fractionnal up/down to get the required integer solution?

optimality is not guaranteed



Problem (P1)

feasbility is not guaranteed

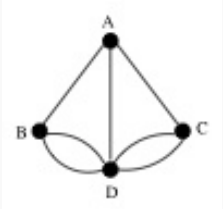


Problem (P2)
= (P1) $\cup \{x_1 + 5x_2 \geq 11\}$

Combinatorial explosion

The set of feasible solutions is finite. Can’t we simply enumerate all of them to identify the best one ?

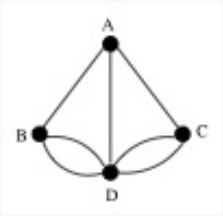
The Traveling Salesman Problem (TSP)



- For 3 nodes; how many solutions possible ?
- For 5 nodes, how many solutions possible ?
- For 20 nodes how many solutions possible ?
- How long to find the best solution for 25 task with a computer able to evaluate 1 billion solutions per second ?
- How many such computers to keep a similar computing time if one additional node is added ? two additional tasks ?

Combinatorial explosion

”Brute-Force” Method: $O(n!)$ solutions

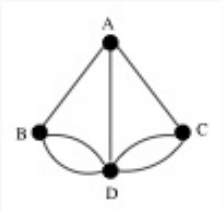


20 nodes $\approx 1e^{17}$ solutions

Nodes			

Combinatorial explosion

”Brute-Force” Method: $O(n!)$ solutions



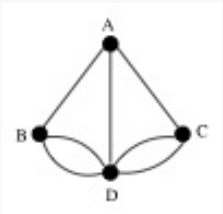
20 nodes $\approx 1e^{17}$ solutions

- Proc. 3GHz : 3 op / nano second

Nodes	Proc. 3 GHz		
10	1/100 seconds		
15	1 hour		
19	1 year		
27	8× age of univ		

Combinatorial explosion

”Brute-Force” Method: $O(n!)$ solutions



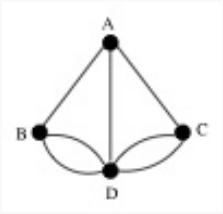
20 nodes $\approx 1e^{17}$ solutions

- Proc. 3GHz : 3 op / nano second
- Proc. Planck : 1 op / Planck time (5.39×10^{-44})s

Nodes	Proc. 3 GHz	Proc. Planck	
10	1/100 seconds		
15	1 hour		
19	1 year		
27	8× age of univ		
35	?	5/1000 seconds	
40	?	12 years	
50	?	4000× age of univ.	

Combinatorial explosion

”Brute-Force” Method: $O(n!)$ solutions



20 nodes $\approx 1e^{17}$ solutions

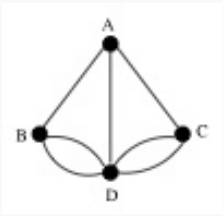
- Proc. 3GHz : 3 op / nano second
- Proc. Planck : 1 op / Planck time (5.39×10^{-44})s
- //P. Planck : fill the universe with Planck Proc. of $1mm^3$

Nodes	Proc. 3 GHz	Proc. Planck	//P. Planck
10	1/100 seconds		
15	1 hour		
19	1 year		
27	8× age of univ		
35	?	5/1000 seconds	
40	?	12 years	
50	?	4000× age of univ.	
90	?	?	

⇒Theoretical analysis of classes of problems to reduce their search space

Combinatorial explosion

"Brute-Force" Method: $O(n!)$ solutions



20 nodes $\approx 1e^{17}$ solutions

- Proc. 3GHz : 3 op / nano second
- Proc. Planck : 1 op / Planck time (5.39×10^{-44})s
- //P. Planck : fill the universe with Planck Proc. of $1mm^3$

Nodes	Proc. 3 GHz	Proc. Planck	//P. Planck
10	1/100 seconds		
15	1 hour		
19	1 year		
27	8× age of univ		
35	?	5/1000 seconds	
40	?	12 years	
50	?	4000× age of univ.	time of Planck
90	?	?	3× age of univ.

⇒Theoretical analysis of classes of problems to reduce their search space

Complexity ?

The complexity of a problem corresponds to the complexity of the best algorithm able to solve it.

Problems for which verifying the feasibility of a given solution is polynomial are generally classified into two classes:

- ones that can be solved optimally with a polynomial algorithm (textcolorblueclass P)
- ones that can require an exponential computing time to solve in the worst case (class NP)

Complexity theory, in particular the notions of NP-completeness and NP-hardness, leads to a better understanding as to why certain classes of problems are still not efficiently solved.

Example

Sometimes, adding a small change in constraints or variables is sufficient to move a problem from *polynomial* to *NP-hard*.

The assignment problem
Consider n products and n machines. Manufacturing i costs d_{ij} if i is assigned to machine j. We want to assign products to machines to build everything while minimizing the total cost.

$\Rightarrow$Polynomial

The Generalized assignment problem
Let be m products, n machines. Making i takes a_{ij} and costs d_{ij} if i is made by machine j. The maximal operating duration of a machine j is b_j. We want to assign products to machines to minimize the total production cost.

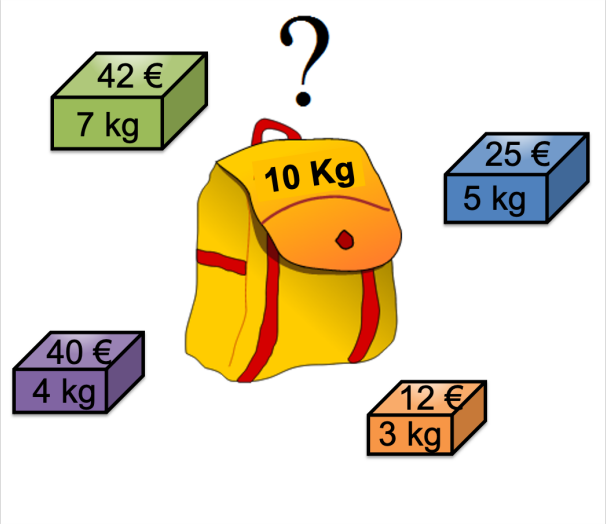
$\Rightarrow$NP-hard

How to solve NP-hard problems ?

- Complete enumeration (for very small instances !)
- Implicit enumeration (for medium-size instances)
  - branch-and-bound
  - dynamic programming
  - ...
- Heuristic methods (for very large instances): can provide very good solutions but no guarantee of optimality
  - (méta/math-)heuristics
  - local search
  - ...

Branch-and-Bound (divide and conquer)

Our case study: the knapsack problem (KP)



Data:

- knapsack of capacity Q,
- $n = 4$ items available,
- item i weights w_i and costs c_i

How to fill the knapsack with items of maximal total cost whilst respecting the capacity ?

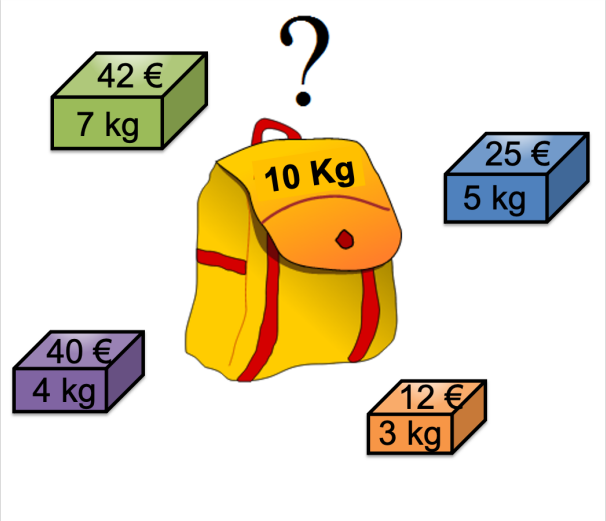
Variables
?

Objective-function
?

Constraints
?

Domain
?

Our case study: the knapsack problem (KP)



Data:

- knapsack of capacity Q,
- $n = 4$ items available,
- item i weights w_i and costs c_i

How to fill the knapsack with items of maximal total cost whilst respecting the capacity ?

$$\max 42x_1 + 40x_2 + 12x_3 + 25x_4 \tag{11}$$

s.t.

$$7x_1 + 4x_2 + 3x_3 + 5x_4 \leq 10 \tag{12}$$

$$x_1, x_2, x_3, x_4 \in \{0, 1\} \tag{13}$$

Relaxation of a problem

Let (P) and (RP) be two problems, and let $S_{(P)}$ and $S_{(RP)}$ be their solutions sets, let $s^*_{(P)}$ and $s^*_{(RP)}$ be their optimal solutions.

- Relaxation
  (RP) is a relaxation of (P) if and only if $S_{(RP)} \subset S_{(P)}$
- Linear Relaxation
  (RP) is a linear relaxation of (P) if and only if
  (P) = (RP) $\cup\{\text{integrality constraints}\}$
- Other types of relaxations
  suppression or aggregation of contraints
  Lagrangian relaxation
  ...

NB: If (P) is a minimisation problem, then $f(s^*_{(RP)}) \leq f(s^*_{(P)})$.

NB2: if the optimal solution of (RP) happens to be feasible for (P), then it is also the optimal solution of (P).

Lower bound (LB) and Upper bound (UB)

Definitions (for a maximisation problem)

- UB = any value greater or equal to the optimal value
- LB = any value lower or equal to the optimal value

How to obtain bounds for a maximisation problem ?

- LB: the cost of any feasible solution provides an upper bound
- UB: any optimal solution of a relaxation provides a lower bound

Benefit of LB: the corresponding solution is feasible

Benefits of UB

- Evaluate the quality of a feasible solution, prove its optimality
- Estimate the gap to optimality when the optimal solution is unknown
- Evaluate if it is worth allocating more computing time to the search of a better solution

Example of bound computation for the knapsack problem

Key idea: solve the linear relaxation (called the Fractionnal knapsack problem - FKP) with a dedicated algorithm

$$(FKP)\ \max 42x_1 + 40x_2 + 12x_3 + 25x_4 \tag{14}$$

s.t.

$$7x_1 + 4x_2 + 3x_3 + 5x_4 \leq 10 \tag{15}$$

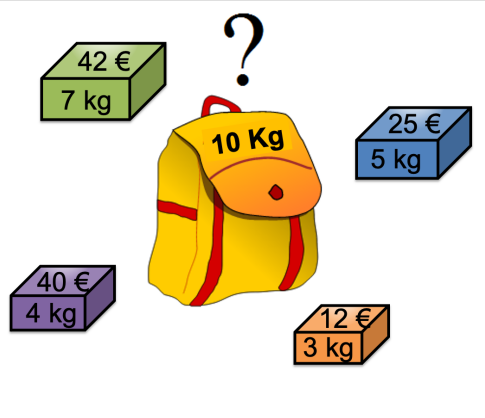
$$0 \leq x_i \leq 1, \forall i \in \{1, 2, 3, 4\} \tag{16}$$

(FKP) can be solved with a linear programming algorithm such as simplex. But it can also be solved with a polynomial algorithm based on sorting all items in decreasing order of ratio $r_i = \frac{\text{value}}{\text{weight}}$.

- Apply the algorithm to deduce an upper bound for (KP).
- What can you conclude ?

A faster upper bound computation = capacity $\times \max_i r_i$

Decision tree / enumeration tree



$$\max 42x_1 + 40x_2 + 12x_3 + 25x_4 \quad (17)$$

s.t.

$$7x_1 + 4x_2 + 3x_3 + 5x_4 \leq 10 \quad (18)$$

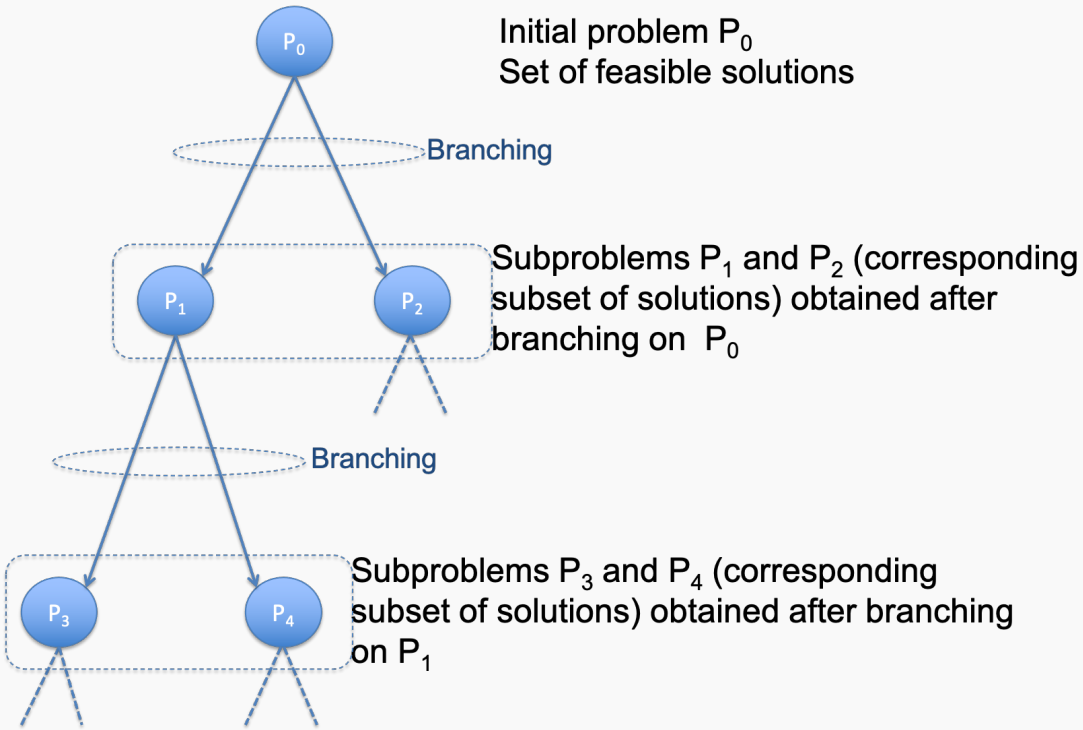
$$x_1, x_2, x_3, x_4 \in \{0, 1\} \quad (19)$$

- Build a decision tree associated to the problem
- Deduce the associated subproblem tree

The tree has 2^n leaves ($\Rightarrow$ combinatorial explosion)

But if we could identify which subtrees do contain an optimal solution, we could signficantly reduce the search space.

Branch-and-bound



Branch-and-bound

Principle

- Separate progressively the problem into subproblems easier to tackle

Objective

- Enumerate implicitly the set of feasible solutions
- Classify the subproblems in a search tree
- Prune nodes as early as possible in the search tree

3 main components

1. Branching rule
2. Pruning tests (usually based on bounds)
   - Admissibility Test, Optimality Test, Resolution Test
3. Exploration strategy

A branch-and-bound algorithm for the knapsack problem

$$(KP)\ \max 42x_1 + 40x_2 + 12x_3 + 25x_4 \tag{20}$$

s.t.

$$7x_1 + 4x_2 + 3x_3 + 5x_4 \leq 10 \tag{21}$$

$$x_1, x_2, x_3, x_4 \in \{0, 1\} \tag{22}$$

1. Branching rule
   - choose the first variable x_k in the decreasing ratio order
   - 2 subproblems : $x_k = 0$ and $x_k = 1$
2. Pruning tests : based on uppers bounds (UB)
   - Admissibility test: prune if the remaining capacity is negative
   - Optimality test: prune if the UB is worse than KP's best known solution
   - Resolution test: prune if the optimal solution of FKP is integer
3. Exploration strategy
   - priority to the node with the highest UB

Computation of the ratio

	1	2	3	4
c_i	42	40	12	~~25~~
w_i	7	4	3	5
$r_i = \frac{c_i}{w_i}$	6	10	4	5

Resulting order:
x_2, x_1, x_4, x_3

Best Known Solution

~~$LB_0 = 0$, $x_1 = x_2 = x_3 = x_4 = 0$~~

~~$LB_1 = 40$, $x_2 = 1, x_1 = x_3 = x_4 = 0$~~

$LB_4 = 65$, $x_2 = x_4 = 1$, $x_1 = x_3 = 0$

0

$\max 42x_1 + 40x_2 + 12x_3 + 25x_4$
s.t. $7x_1 + 4x_2 + 3x_3 + 5x_4 \leq 10$
$x_1, x_2, x_3, x_4 \geq 0$

$UB_0 = r_2 \times 10 = 100$
$LB_0 = 0$
$x_2 < 0$

$x_2 = 1$

1

$\max 40 + 42x_1 + 12x_3 + 25x_4$
s.t. $7x_1 + 3x_3 + 5x_4 \leq 6$
$x_1, x_3, x_4 \geq 0$

$UB_1 = 40 + r_1 \times 6 = 76$
$LB_1 = 40$

~~**2**~~

$\max 0 + 42x_1 + 12x_3 + 25x_4$
s.t. $7x_1 + 3x_3 + 5x_4 \leq 10$
$x_1, x_3, x_4 \geq 0$

$UB_2 = 0 + r_1 \times 10 = 60$
$LB_2 = 0$

≤ 65

Optimality test !!
OT

$x_1 = 1$

~~**3**~~

$\max 82 + 12x_3 + 25x_4$
s.t. $3x_3 + 5x_4 \leq -1$

Infeasible !
Admissibility test
AT

$x_1 = 0$

4

$\max 40 + 12x_3 + 25x_4$
s.t. $3x_3 + 5x_4 \leq 6$
$x_3, x_4 \geq 0$

$UB_4 = 40 + r_4 \times 6 = 64$
$LB_4 = 40$

$x_4 = 1$

5

$\max 65 + 12x_3$
s.t. $3x_3 \leq 1$
$x_3 \geq 0$

$UB_4 = 65 + r_3 \times 1 = 69$
$LB_4 = 65$

$x_4 = 0$

~~**6**~~

$\max 40 + 12x_3$
s.t. $3x_3 \leq 6$
$x_3 \geq 0$

$UB_5 = 40 + r_3 \times 6 = 64$
≤ 65

Optimality test !!
OT.

$x_3 = 1$

~~**7**~~

$\max 77$
s.t. $3 \leq 1$

infeasible !
AT

$x_3 = 0$

~~**8**~~

$\max 65$
s.t. $0 \leq 1$

Solved !
Resolution test
$LB_8 = 65$

RT

Active nodes stored

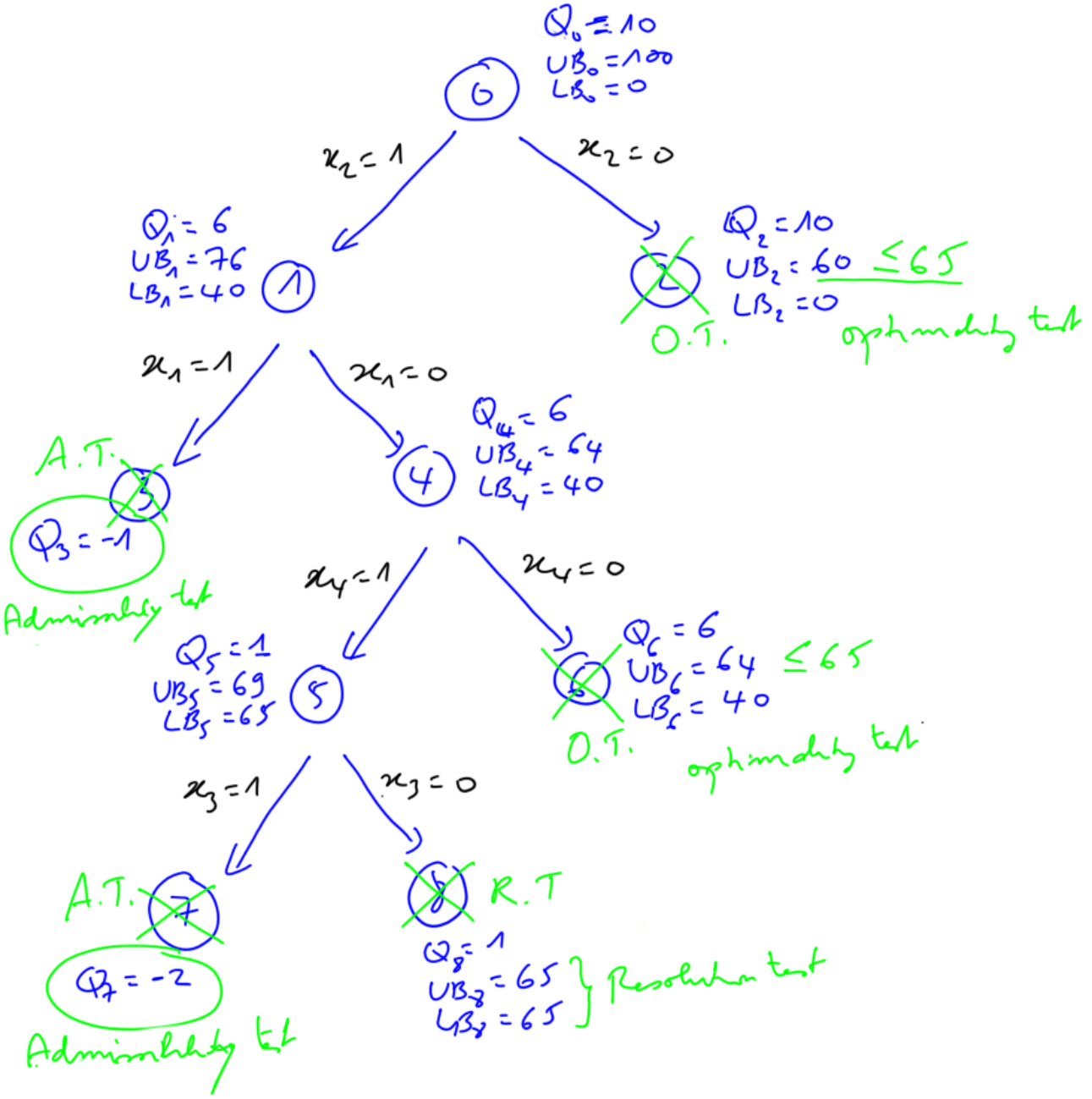
~~0 → UB = 100~~
~~1 → UB = 76~~
~~2 → UB = 60~~
~~4 → UB = 64~~
~~5 → UB = 65~~

Best Known Solution

$LB_0 = 0$, $x_1 = x_2 = x_3 = x_4 = 0$

$LB_1 = 40$, $x_2 = 1$, $x_1 = x_3 = x_4 = 0$

$LB_4 = 65$, $x_2 = x_4 = 1$, $x_1 = x_3 = 0$



Classical branch-and-bound algorithm for general MILP

1. Branching rule
   - choose the variable x_k with the most fractionnal value v^* in the optimal solution of the relaxation
   - 2 subproblems : $x_k \leq \lceil v^* \rceil$ and $x_k \geq \lfloor v^* \rfloor$
2. Pruning tests : based on solving the linear relaxation (LR) of the current node to obtain bounds
   - Admissibility test: prune if LR is infeasible
   - Optimality test: prune if LR's solution value is worse than the best known solution value
   - Resolution test: prune if LR's optimal solution is integer
3. Exploration strategy
   - priority to the node with the best bound

$\Rightarrow$To be applied on the cement problem from slide 11 for illustration.

(replace the simplex method with the graphical method to solve LR)

Tools

MILP solvers