Integer Linear Programming

Solution : cutting planes and Branch and Bound

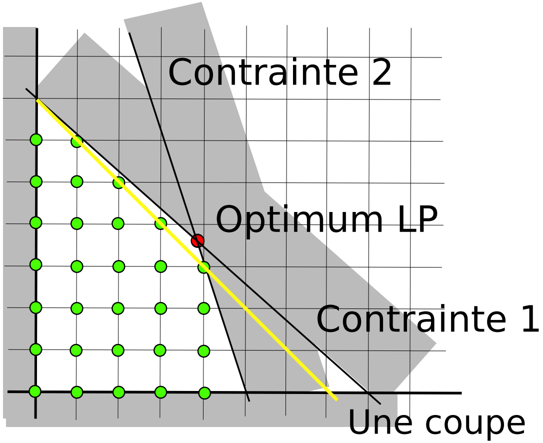
Hugues Talbot

Laboratoire CVN

April 13, 2018

Plan

Principle



Initial formulation

- Consider a linear “relaxation” of an IP problem. This is the corresponding LP for which the optimal solution is X^*.
  - If X^* is integer, this is the optimal solution.
  - If it is not, X^* is fractionnary .
- Consider the basis corresponding to the LP optimal solution

$$AX = BX_b + EX_e = b \Rightarrow X_b = B^{-1}[b - EX_e]; \bar{b} = B^{-1}b.$$

with $\bar{b}_i = B^{-1}b$ fractionnary.

- We have

$$X_{b_i} = e_i^t B^{-1}[b - EX_e]$$

où $e_i^t = [000 \ldots \underset{\text{pos}\, i}{1} \ldots 00]$

Formulation (next)

- So :

$$\begin{aligned} X_{b_i} &= e_i^t B^{-1} b - e_i^t B^{-1} E X_e \\ &= \bar{b}_i - \alpha^i X_e \end{aligned}$$

  interpreting :

$$\alpha^i = e_i^t B^{-1} E \text{ (i.e: line } i \text{ of } B^{-1}E\text{)}$$

- By reordering:

$$\begin{aligned} X_{b_i} + \alpha^i X_e &= \bar{b}_i \\ X_{b_i} + \textstyle\sum_j \alpha_j^i X_{e_j} &= \bar{b}_i \qquad [1]) \end{aligned}$$

Integer part/ fractionnary part

- Definition : let a be a non-integer
  - int(a) = greatest integer $\leq a$.
  - frac(a) = $a -$ int(a)(therefore ≥ 0).
- From [1] we have :

$$\begin{array}{ll} X_{b_i} + \sum_j [\text{int}(\alpha_j^i) + \text{frac}(\alpha_j^i)] X_{e_j} & = \text{int}(\bar{b}_i) + \text{frac}(\bar{b}_i) \\ X_{b_i} + \sum_j \text{int}(\alpha_j^i) X_{e_j} + \sum_j \text{frac}(\alpha_j^i) X_{e_j} & = \text{int}(\bar{b}_i) + \text{frac}(\bar{b}_i) \quad \text{[2])} \end{array}$$

with frac$(\alpha_j^i) \geq 0$ and $X_{e_j} \geq 0$.

- so

$$X_{b_i} + \sum_j \text{int}(\alpha_j^i) X_{e_j} \leq \text{int}(\bar{b}_i) + \text{frac}(\bar{b}_i)$$

Gomory cut

- However X_{b_i} is integer and so is X_{e_j}.
- therefore

$$X_{b_i} \sum_j \text{int}(\alpha^i_j) X_{e_j} \leq \text{int}(\bar{b}_i) \qquad [3]$$

- from [3]–[2] we deduce

$$\begin{array}{rcrl} -\sum_j \text{frac}(\alpha^i_j) X_{e_j} & \leq & - & \text{frac}(\bar{b}_i) \\ \sum_j \text{frac}(\alpha^i_j) X_{e_j} & \geq & & \text{frac}(\bar{b}_i) \qquad [4] \end{array}$$

[4] is a *Gomory cut*

Example

- Consider the following problem :

$$\begin{array}{llcll} \max & x_1 & + & x_2 & \\ & 3x_1 & + & 4x_2 & \leq 15 \\ & x_1 & - & 4x_2 & \leq 0 \end{array}$$

- In standard form :

$$\begin{array}{lllll} 3x_1 & +4x_2 & +x_3 & & = 15 \\ x_1 & -4x_2 & & +x_4 & = 0 \end{array}$$

Cut 1

- Optimal solution: basis $X_b = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, $B = \begin{bmatrix} 3 & 4 \\ 1 & -4 \end{bmatrix}$
- Inverse : $B^{-1} = -\frac{1}{16} \begin{bmatrix} -4 & -4 \\ -1 & 3 \end{bmatrix} = \begin{bmatrix} 1/4 & 1/4 \\ 1/16 & -3/16 \end{bmatrix}$
- Solution :
  $\bar{b} = B^{-1}b = \begin{bmatrix} 1/4 & 1/4 \\ 1/16 & -3/16 \end{bmatrix} \begin{bmatrix} 15 \\ 0 \end{bmatrix} = \begin{bmatrix} 15/4 \\ 15/16 \end{bmatrix}$
- Cut (1) : $\alpha^1 = [\frac{1}{4} \frac{1}{4}], \bar{b}_1 = \frac{15}{4}$
  $\frac{1}{4}x_3 + \frac{1}{4}x_4 \geq \frac{3}{4}$

$$x_3 + x_4 \geq 3 \qquad \text{cut (1)}$$

Cut 2

- Cut (2) :
  $\alpha^2 = [\frac{1}{16} - \frac{3}{16}], \bar{b}_2 = \frac{15}{16}$, Careful! frac$(\alpha^2) = [\frac{1}{16} \frac{13}{16}]$
  $\frac{1}{16}x_3 + \frac{13}{16}x_4 \geq \frac{15}{16}$

$$x_3 + 13x_4 \geq 15 \qquad \text{cut (2)}$$

- These are not very interesting, now we express the out-of-basis variables in terms of the basis variables:

$$\begin{array}{rclll} x_3 & = & 15 & -3x_1 & -4x_2 \\ x_4 & = & & -x_1 & +4x_2 \end{array}$$

Final cut

- By substituting in cut 1:

$$\begin{aligned} 15 - 3x_1 - 4x_2 - x_1 + 4x_2 &\geq 3 \\ 12 &\geq 4x_1 \\ x_1 &\leq 3 \end{aligned}$$

- By substituting in cut 2 :

$$\begin{aligned} 15 - 3x_1 - 4x_2 - 13x_1 + 52x_2 &\geq 15 \\ -16x_1 + 48x_2 &\geq 0 \\ x_1 - 3x_2 &\leq 0 \end{aligned}$$

- We add the cuts to the original problem, and this time we obtain :
  $\bar{b} = [3, \frac{3}{2}]$, we need another iteration.

Note on the Gomory cuts

- It is possible to show that the method converges rapidly. In practice one component of the solution becomes integer at each iteration
- However, the number of variables double every iteration, so the complexity is exponential.

Solution by branch and bound

- Gomory's cut provide an algorithmic solution to IP problems.
- In the case where both constraints and variables are rational (or integer), Gomory's algorithm converges in finite time.
- Cuts attack the problem from the outside, which may be inefficient in some configurations.
- It is also possible to attack the problem from the inside: we will separate the domains and evaluate which domain to explore first. This is the branch-and-bound method.

Tables and chairs

- A company produces tables and chairs;
- A table requires 1h of work and 9 m^2 of wood ;
- A chair requires 1h of work et $5m^2$ of wood ;
- Our resources are 6h of work and 45 m^2 of wood ;
- Each table generates a profit of 8 €, and each chair 5 €;
- Formulate and solve the associated IP problem.

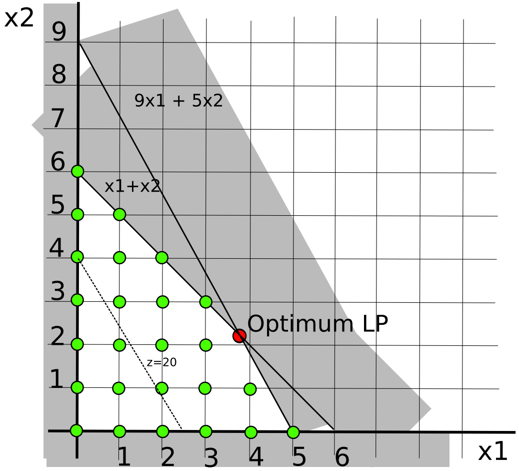
Formulation

This is a possible formulation:

$$\begin{array}{rcccl} \max z = & 8x_1 & + & 5x_2 & \\ & x_1 & + & x_2 & \leq 6 \\ & 9x_1 & + & 5x_2 & \leq 45 \end{array}$$

$x_i \geq 0, x_i \in \mathbb{N}$.

Representation



Probleme initial (1)

General Method

- We start from solving the LP relaxation as before, we obtain:

$$\begin{aligned} z &= \frac{165}{4} &= 41.25 \\ x_1 &= \frac{15}{4} \\ x_2 &= \frac{9}{4} \end{aligned}$$

- If the solution is integer, we stop, this is optimal. If not, the z value obtained is an upper bound for the integer solution (why?).

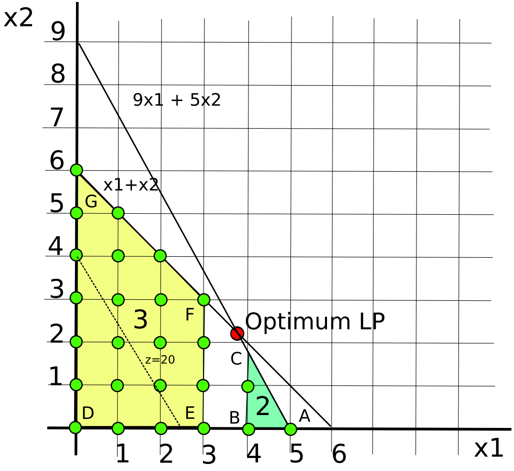
General method (next)

- If the solution is not integer, we partition the domain. We choose arbitrarily one of the variables that is still fractionary, for instance here x_1.
- We impose extra constraints due to the nature of the variables, here either $x_1 \leq 3$, or $x_1 \geq 4$ (Question : why can we eliminate the solution $3 < x_1 < 4$?)

Separation

- We now have two subproblems:
  2. Initial problem + constraint $x_1 \geq 4$
  3. Initial problem + constraint $x_1 \leq 3$.

Representation (2)



Sous-problemes 2+3

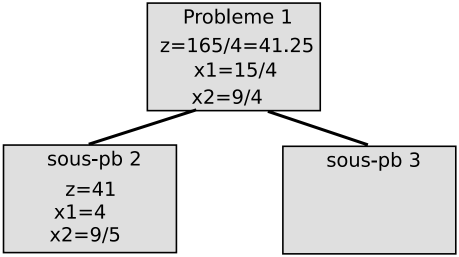
Recursion

- We now have eliminated the region that used to contain the LP solution, however we have retained all the integer parts of the domain
- We now have two sub problems we can try to solve with LP.
- We choose one of them arbitrarily, for instance problem 2.
- The solution of the LP relaxation for region 2 is

$$\begin{aligned} z &= 41 \\ x_1 &= 4 \\ x_2 &= \frac{9}{5} \end{aligned}$$

  This is point C.
- This is the first branch of a *tree*.
- Since x_2 is still fractionnary, we can decide to separate on this variable. We split region 2 into two zones : The one for $x_2 \geq 2$ and the one for $x_2 \leq 1$.

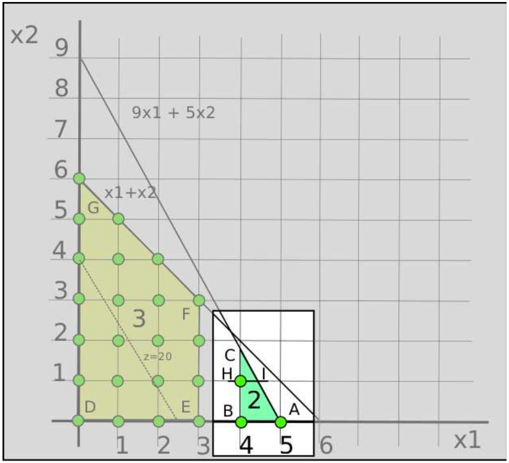
Tree (1+2)


Probleme 1
z=165/4=41.25
x1=15/4
x2=9/4
sous-pb 2
z=41
x1=4
x2=9/5
sous-pb 3

New branch (4 + 5)

- We now have two new sub-problems
  4. Problem 2 + constraint $x_2 \geq 2$
  5. Problme 2 + constraint $x_2 \leq 1$.
- It is preferable to explore the tree depth first (we will see later why). We choose one of these new sub-problems, for instance problem 4. However this problem is not feasible in integers.

Representation (4 + 5)



Sous-problemes 4+5

Bounding (4 + 5)

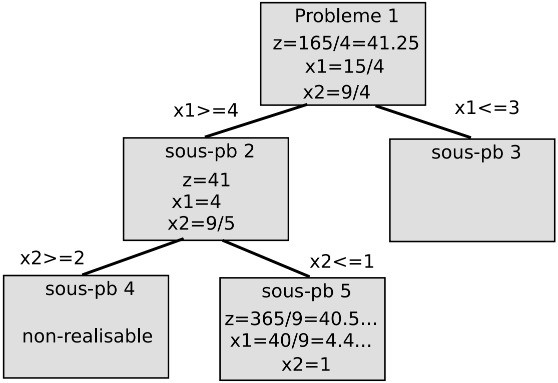
- We realize that sub-problem 4 (région $x_2 \geq 2$) is not feasible in integers.
- By solving the relaxed LP associated with sub-problem 5, we find the optimum I with

$$\begin{aligned} z &= 365/9 = 40.555\ldots \\ x_1 &= \frac{40}{9} = 4.444\ldots \\ x_2 &= 1 \end{aligned}$$

- we must branch on x_1, with the constraints

  6. Problem 5 + constraint $x_1 \geq 5$
  7. Problem 5 + constraint $x_1 \leq 4$.

Tree for (4+5)


Probleme 1
z=165/4=41.25
x1=15/4
x2=9/4
x1>=4
x1<=3
sous-pb 2
z=41
x1=4
x2=9/5
sous-pb 3
x2>=2
x2<=1
sous-pb 4
non-realisable
sous-pb 5
z=365/9=40.5...
x1=40/9=4.4...
x2=1

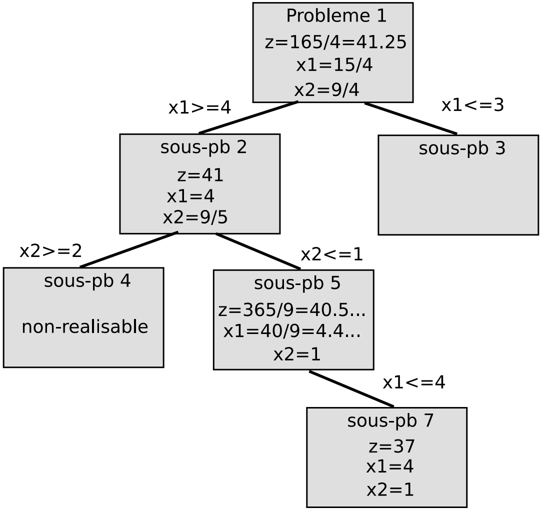
Branch (6+7)

- We are faced with two new problems. As usual, we eliminate the fractionnary solution by branching it.
- We keep trying to scan the tree depth first, by testing one of the new cases (6 et 7). Arbitrarily we choose 7.
- The relaxed LP solution is now:

$$\begin{aligned} z &= 37 \\ x_1 &= 4 \\ x_2 &= 1 \end{aligned}$$

  This solution is feasible in integer, this corresponds to point H. This is a *candidate solution*, that provides us with a lower bound on the final result.
- We stop branching from here.

Tree (7)


Probleme 1
z=165/4=41.25
x1=15/4
x2=9/4
x1>=4
x1<=3
sous-pb 2
z=41
x1=4
x2=9/5
sous-pb 3
x2>=2
x2<=1
sous-pb 4
non-realisable
sous-pb 5
z=365/9=40.5...
x1=40/9=4.4...
x2=1
x1<=4
sous-pb 7
z=37
x1=4
x2=1

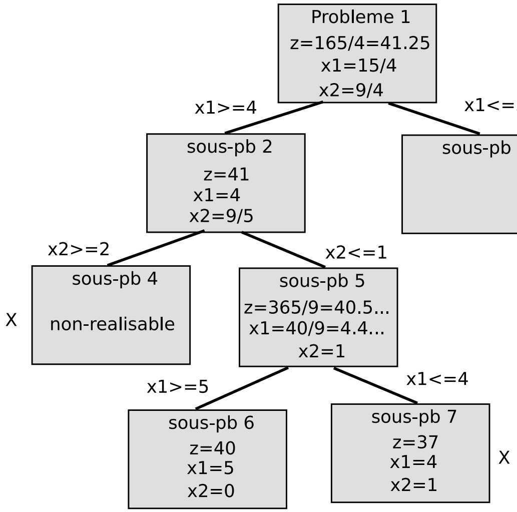
Bounding (6)

- Continuing depth first, we evaluate sub-problem 6.
- We find the following solution (point A).

$$\begin{aligned} z &= 40 \\ x_1 &= 5 \\ x_2 &= 0 \end{aligned}$$

  This is also a candidate solution. Now our lower bound is improved to 40.
- Therefore candidate solution of problem 7 is not optimal
- We do not need to branch further from 6.

Tree (6)


Probleme 1
z=165/4=41.25
x1=15/4
x2=9/4
x1>=4
x1<=3
sous-pb 2
z=41
x1=4
x2=9/5
sous-pb 3
x2>=2
x2<=1
sous-pb 4
non-realisable
X
sous-pb 5
z=365/9=40.5...
x1=40/9=4.4...
x2=1
x1>=5
x1<=4
sous-pb 6
z=40
x1=5
x2=0
sous-pb 7
z=37
x1=4
x2=1
X

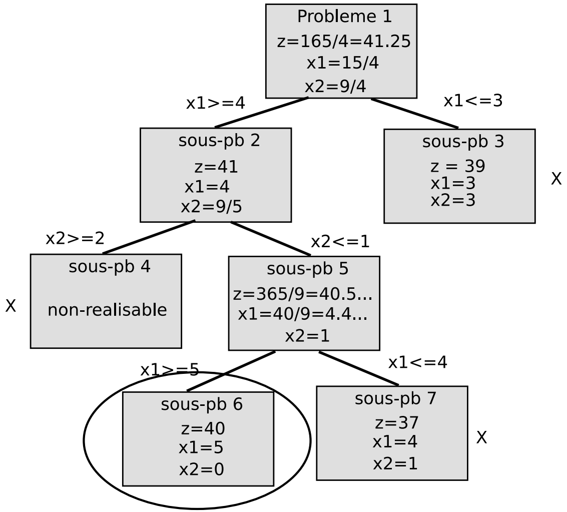
Bounding (3)

- We backtrack to the solution for (3). We find the solution for F :

$$\begin{aligned} x_1 &= 3 \\ x_2 &= 3 \\ z &= 39 \end{aligned}$$

- This result is less than 40, our current lower bound. This branch cannot produce a better result than the one we already have.
- We have now explored or bounded every node in the tree, so we have found our optimum: build 5 tables et 0 chairs for a profit of 40€.

Arbre (3)


Probleme 1
z=165/4=41.25
x1=15/4
x2=9/4
x1>=4
x1<=3
sous-pb 2
z=41
x1=4
x2=9/5
sous-pb 3
z = 39
x1=3
x2=3
X
x2>=2
x2<=1
sous-pb 4
non-realisable
X
sous-pb 5
z=365/9=40.5...
x1=40/9=4.4...
x2=1
x1>=5
x1<=4
sous-pb 6
z=40
x1=5
x2=0
sous-pb 7
z=37
x1=4
x2=1
X

Node labelling

- The branch-and-bound algorithm labels all the points of the feasible region, some in a explicit and other in an implicit manner.
- For instance, point ($x_1 = 2, x_2 = 3$) is included in sub-problem 3, for which the optimum is $(3, 3)$. There is no point in actually evaluating it.
- The algorithm implements "divide and conquer". It necessarily converges due to the fact that it always eliminate at least one point from the feasible region at each step , and these points are in finite numbers.

Sterilizing nodes

1. A note that is not necessarily to branch on is said to be *sterile*. There are several cases where this might happen:
   1.1 The node is associated with a candidate solution. Branching on it will not improve it.
   1.2 The node is associated with an unfeasible sub-problem. Branching on it will not make if any more feasible.
   1.3 The z-value of the node is less than that of the best-known solution so far, even if it is not a candidate solution. Branching will not improve it.

Backtracking

- We can solve branch-and-bound problems either depth or breadth first. The latter would correspond to exploring all the subproblem at one level rather than one particular branch all the way to the bottom first. E.g., in the previous problem we would explore (3) before (4) or (5).
- In practice, depth first works better because doing so quickly converges to a candidate solution or an unfeasible subproblem, allowing branches to be sterilized quicker.
- Note : branch-and-bound is not a greedy algorithm: it does backtrack.

Solving mixed problems

- On mixed problems, we only branch on the integer variables !

Generalization of branch-and-bound

- The LP relaxed problem is only an heuristic.
- Sometimes can be replaced by something more appropriate
- Example : knapsack (see TD).
- Linked with $A^\star$ in combinatorial optimization.

Knapsack

- Reminder : a knapsack (problème de sac à dos) is a problem with a single constraint

$$\begin{aligned} \max z &= \textstyle\sum_i c_i x_i \\ \textstyle\sum_i a_i x_i &\leq b \end{aligned}$$

  with $x_i \in \{0, 1\}$
- Branching on x_i yields a branch on $x_i = 0$ and a branch on $x_i = 1$.
- The LP relaxation is simplified by remarking that we can sort the variables in order of decreasing $\frac{c_i}{a_i}$. The variables with the best ratio are preferable.
- See associated TD.
- A knapsack can also be solved with dynamic programming.

Planification

- Planification problems, with deadlines, and so on are very common.
- For instance :

	Length	deadline
task 1	6	8
task 2	4	4
task 3	5	12
task 4	8	16

- The table is interpreted this way : task 1 required 6 days and must be completed by close of business on day 8.
- Each late day incurs penalties. We must minimize these.

Solution by branch and bound

- We can use the following variables :

$$x_{ij} = \begin{cases} 1 & \text{if task } i \text{ is in position } j \\ 0 & \text{otherwise} \end{cases}$$

- It is efficient to assign the *last* task first, and to bound the total minimal delay by summing the duration of all assigned and remaining tasks, minus the delay for the last task.
- For instance, if task 3 is the last, ($x_{34} = 1$), a bound for the delay is at least $6 + 4 + 8 + 5 - 12 = 23 - 12 = 11$.

Solution by B& B, next

- Then, in the tree, one branches by assigning more tasks before the last one (and so on). We bound by summing the estimated delays. For instance, if task 3 is last, and then task 4 is the one before last (penultimate), the total delay cannot be less than $11 + (6 + 4 + 8 - 16) = 11 + 2 = 13$.
- It is possible to check that the optimal scheduling is 2-1-3-4 with a total delay of 12 days.

Travelling Salesperson Problem

- Note the politically correct version...
- We have a number of cities we all want to visit at the lowest possible cost.
- Here is a cost matrix, where c_{ij} represents the cost of going from city i to city j :

	city 1	city 2	city 3	city 4	city 5
city 1	0	132	217	164	58
city 2	132	0	290	201	79
city 3	217	290	0	113	303
city 4	164	201	113	0	196
city 5	58	79	303	196	0

TSP (II)

- The minimum cost trip is a Hamiltonian cycle.
- To solve by B&B, we need some means to branch and to bound. One solution is to transform the problem into an assignment one, with cost matrix c : Let the x_{ij} be a set of binary variables. If $x_{ij} = 1$ then the trip from city i to city j is done, 0 otherwise.
- We will learn how to solve assignment problem very efficiently in the next lecture.
- In this case, if $x_{12} = x_{24} = x_{45} = x_{53} = x_{31} = 1$, then we chose the circuit 1-2-4-5-3-1.
- If the assignment solution produces a circuit then it is optimal (why ?). However an assignment may not generat circuits, for instance one may obtain $x_{15} = x_{21} = x_{34} = x_{43} = x_{52} = 1$, which has two non-connex subcircuit.

TSP (III)

- One must also impose $i \neq j$. This is achieved with $c_{ii} = M$ a large number.
- We can for instance start from the assignment problem, then split by eliminating some sub-circuit. We do so by imposing than one or the other of an invalid sub-circuit be a large M.
- We split on whether we forbid either i to j or j to i.
- For instance, from the initial problem with the cost matrix given earlier (with $c_{ii} = M$), the first assignment is $x_{15} = x_{21} = x_{34} = x_{43} = x_{52} = 1$.
- We have two sub-circuit : 1-5-2-1 et 3-4-3. we can eliminate 3-4-3, either via imposing $c_{34} = M$, or the converse $c_{43} = M$. (Q: why is this legitimate ?)

TSP (IV)

- We can relatively easily verify that the optimal solution is $x_{15} = x_{24} = x_{31} = x_{43} = x_{52} = 1$, i.e. the circuit 1-5-2-4-3-1 for a cost $z = 668$.

Binary problems by implicit enumeration

- Binary problems are a large class (BP) in IP.
- Remark : any IP is expressible in BP (via decomposition over the powers of 2)
- There exist particular BP solutions for instance, for instance the implicit enumeration method : superior branches represent fixed variables. we branch on the variable that improves the solution the most (like in knapsack), as long as it is feasible.
- Even if a solution is unfeasible it will help bounding the solutions.

Sudoku

See Matlab TP.

Conclusion

- The LP provides a relaxation for integer programming (IP) or mixed integer programming (MIP) problems
- All IP and MIP converge in finite time but complexity may be exponential.
- However the LP relaxation provides upper and lower bounds that improve as we explore branches.
- In some cases, a better bounding function can exist than the simplex, however the general formulation remains the same.
- quality and speed of result depends on this bounding function.
- Many links with heuristics $A^\star$, some graph algorithms, etc.
- In general IP problems solvable by B&B are difficult, but not always.