Foundations of Operations Research

Marta Pascoal (marta.brazpascoal@polimi.it)

Dipartimento di Elettronica, Informazione e Bioingegneria – Politecnico di Milano

2024/25

4. Integer Linear Programming (ILP)

Definition 1

An Integer Linear Programming problem is an optimization problem of the form

$$(ILP) \quad \begin{array}{ll} \min & c^T x \\ \text{s. t.} & Ax \geq b \\ & x \geq 0 \text{ with } x \in \mathbb{Z}^n \end{array}$$

--> variables should be integers

- If $x_j \in \{0, 1\}$ for all j, binary LP.
- If not all x_j are integer, mixed integer LP.

Assumption: The parameters A, b are integer (without loss of generality).

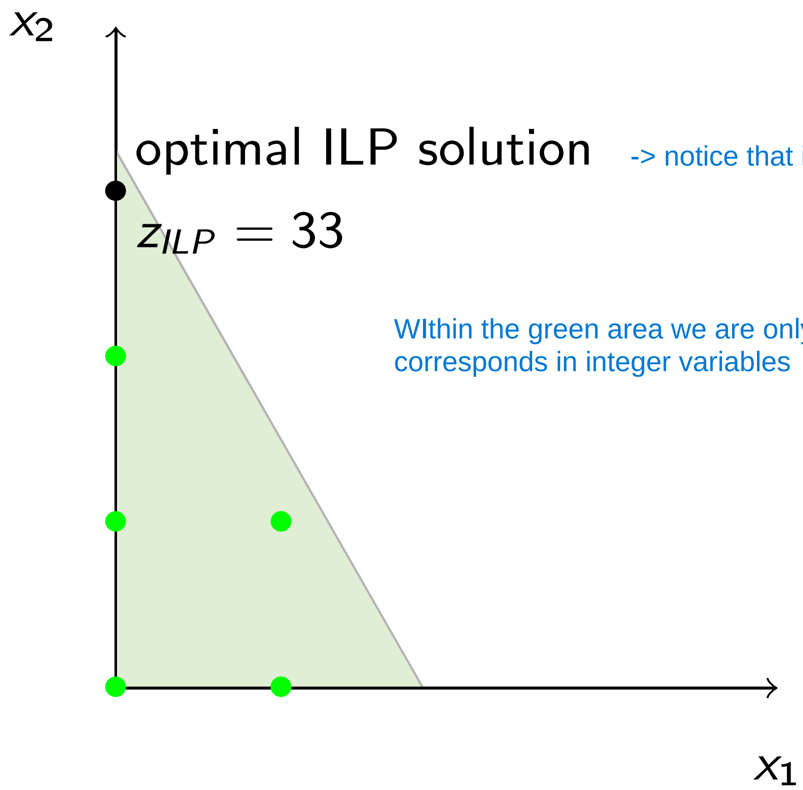
Note: The integrality condition $x_j \in \mathbb{Z}$ is non linear, since it can be expressed as $\sin(\pi x_j) = 0$.

is like we are taking a step toward non-linear problem, however the constraint and objective function are still linear.

4. Integer Linear Programming (ILP)

Example

$$
\begin{array}{lll}
z_{ILP} = \max & z = 21x_1 + 11x_2 \\
\text{s. t.} & 7x_1 + 4x_2 \le 13 \\
 & x_1, x_2 \ge 0 \text{ integer}
\end{array}
$$



-> notice that it is not a vertex of the green region

WIthin the green area we are only interested in the green points, which corresponds in integer variables

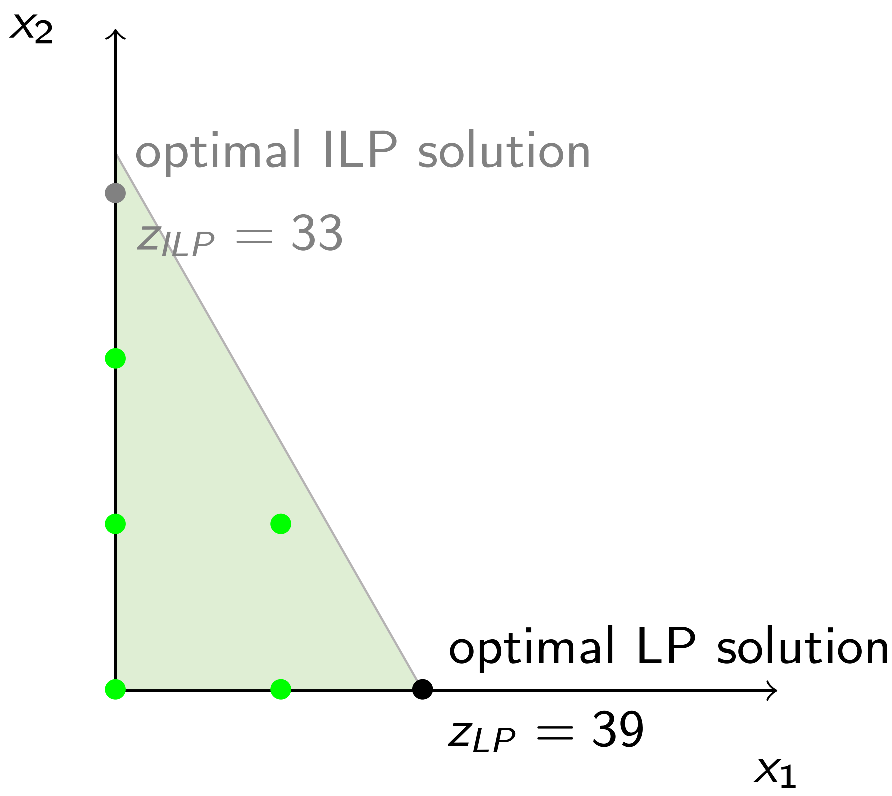
ILP feasible region $\equiv$ lattice (with a finite or infinite number of points).

4. Integer Linear Programming (ILP)

Example (cont.)

Deleting the integrality constraints we obtain the Linear Program:

$$\begin{array}{lll} z_{LP} = \max & z = 21x_1 + 11x_2 \\ \text{s. t.} & 7x_1 + 4x_2 \leq 13 \\ & x_1, x_2 \geq 0 \end{array}$$



ILP feasible region $\equiv$ lattice (with a finite or infinite number of points).

We notice that:
- that the optimal solution of the ILP and the one of the LP have nothing to do each other
- the optimal solution of LP leads to a better (In this case bigger) value of the objective function than the one of ILP --> we will see this is generalizable: LP solution gives upper bound (in case of maximization) to the optimal value of ILP

4. Integer Linear Programming (ILP)

Definition 2

Let

$$(ILP) \quad \begin{aligned} z_{ILP} := \max \quad & c^T x \\ \text{s. t.} \quad & Ax \leq b \\ & x \geq 0, x \in \mathbb{Z}^n \end{aligned}$$

The problem

$$(LP) \quad \begin{aligned} z_{LP} := \max \quad & c^T x \\ \text{s. t.} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

is the linear (continuous) relaxation of (ILP). (we get it by negleting the integrality constraint)

What is the ILP optimal objective value and the LP optimal objective value?

Property 1

For any ILP with max, we have $z_{ILP} \leq z_{LP}$, i.e., z_{LP} is an upper bound on the optimal value of (ILP). With LP we optimize on a more extended region (polyhedron) than the lattice, so we get a better value

N.B. For any ILP with min, we have $z_{ILP} \geq z_{LP}$, i.e., z_{LP} is a lower bound on the optimal value of (ILP).

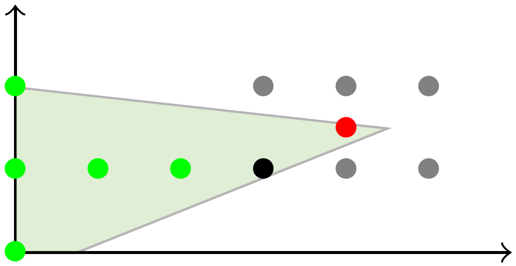
4. Integer Linear Programming (ILP)

First idea: Relax the integrality constraints of (ILP) and round up/down the optimal solution of the linear relaxation (LP).

If an optimal solution of (LP) is integer, then it is also an optimal solution of (ILP).

But often the rounded optimal solutions of (LP) are:

- Infeasible rounded solutions for (ILP).



LP optimal solution

There's a mistake since LP optimal solution should be on one of the vertex

ILP optimal solution

In this case rounding up or down will lead to take an unfeasible solution (the nearest solutions to the red one are outside of the feasible region).

- Useless rounded solutions: Very different from an optimal solution of (ILP). When the integer variables take small values at optimality.

E.g., binary assignment variables (job to machine) or activation variables (plants), . . .

- Useful rounded solutions: When the integer variables are big at optimality.

E.g., number of pieces to produce, . . .

Note: It also depends on the unit costs (coefficients of the objective function).

4. Integer Linear Programming (ILP)

Example 1: Knapsack problem

Given

n objects $j = 1, \ldots, n$
p_j profit (value) of object j
v_j volume (weight) of object j
b maximum knapsack capacity.

determine a subset of objects that maximizes the total profit, while respecting the knapsack capacity.

Variables $x_j = \begin{cases} 1, & j\text{-th object is selected} \\ 0, & \text{otherwise} \end{cases}$

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} p_j x_j \\ \text{s. t.} \quad & \sum_{j=1}^{n} v_j x_j \leq b \\ & x_j \in \{0, 1\}, \quad j = 1, \ldots, n \end{aligned}$$

4. Integer Linear Programming (ILP)

Binary knapsack problem is NP-hard

Wide range of direct and indirect applications:

- loading (containers, vehicles, CDs,. . .)
- investments (p_j is the expected return, v_j is the amount to invest, b is the available capital)
- as a suproblem, . . .

4. Integer Linear Programming (ILP)

Example 2: Assignment problem

We saw it in the maximum flow

Given

m machines, $i = 1, \ldots, m$ *(Assumed that* $n > m$.*)*
n jobs, $j = 1, \ldots, n$
c_{ij} *cost of assigning job* j *to machine* i

determine an assignment of jobs to the machines so as to minimize the total cost, while assigning at least one job per machine and at most one machine for each job.

Variables $x_{ij} = \begin{cases} 1, & \text{machine } i \text{ executes job } j \\ 0, & \text{otherwise} \end{cases}$

Binary variables are useful for select entities or to associate entities by using two indexes

$$\min \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

fix a job:

summation over machines

$$\text{s. t.} \quad \sum_{i=1}^{m} x_{ij} \leq 1, \quad j = 1, \ldots, n \qquad \text{(at most one machine for each job)}$$

$$\sum_{j=1}^{n} x_{ij} \geq 1, \quad i = 1, \ldots, m \qquad \text{(at least one job for each machine)}$$

$$x_{ij} \in \{0, 1\}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

4. Integer Linear Programming (ILP)

Example 3: Transportation problem

Given

m production plants, $i = 1, \ldots, m$
n clients, $j = 1, \ldots, n$ (assumption: $n > m$)
c_{ij} transportation cost of one unit of product from plant i to client j
p_i production capacity of plant i
d_j demand of client j
q_{ij} maximum amount to be transported from plant i to client j

determine a transportation plan that minimizes total costs while satisfying plant capacitiy and client demands.

Assumption $\sum_{i=1}^{m} p_i \geq \sum_{j=1}^{n} d_j$ otherwise there exist no feasible solution

Variables x_{ij} = amount transported from plant i to client j

4. Integer Linear Programming (ILP)

Considerando x un vettore m*n x 1, nella matrice A avremo che ogni colonna è " associata" ad un xij. Visto che in ogni constraint fissiamo un indice e iteriamo sull'altro (o iteriamo su entrambi), ogni xij apparirà una volta per constraint. Quindi significa che avremo solo 3 valori diversi da zero in ogni colonna della matrice A.
Se invece avessimo un constraint del tipo:

$$\sum_{i=1}^{m} x_{ij} + x_{11} \quad \forall j \qquad \text{oppure} \qquad x_{11} + \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}$$

allora ciò significherebbe che in ogni riga della matrice A il coefficiente nella colonna associato ad x11 è diverso da 0, quindi non sono solo tre i valori diversi da zero in quella colonna

$$\min \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

$$\text{s. t.} \ (A) \quad \sum_{j=1}^{n} x_{ij} \leq p_i, \quad i = 1, \ldots, m \qquad \text{(plant capacity)}$$

$$(B) \quad \sum_{i=1}^{m} x_{ij} \geq d_j, \quad j = 1, \ldots, n \qquad \text{(client demand)}$$

$$(C) \quad 0 \leq x_{ij} \leq q_{ij}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \qquad \text{(transportation capacity)}$$

The assignment problem is a special case of the transportation problem.

Property of the transportation and the assignment problems:

If all the right hand (pi, dj, qij) side values are integers, then the optimal solution will be integer. That is:

Optimal solution of the linear relaxation $\equiv$ optimal solution of the ILP!

How to explain this property?
We notice the special structure of the problem. In each inequality the x has a coefficient of 1 and no other coefficients.

4. Integer Linear Programming (ILP)

Theorem 3

If in a transportation problem p_i, d_{ij}, q_{ij} are integer, all the basic feasible solutions (vertices) of its linear relaxation are integer.

A B C #i × #j

- Special $(mn + n + m) \times (mn)$ integer constraint matrix A, $a_{ij} \in \{-1, 0, 1\}$ with exactly 3 nonzero coefficients per column. --> guarda spiegazione in slide precedente
- Right hand side vector b has all integer components.

whenever the matrix A of constraint is totally unimodular then the solution is a integer one

Optimal solution of the linear relaxation: is always of the type:

$$x^* = \begin{bmatrix} \overbrace{B^{-1}b}^{x_B^*} \\ \underbrace{0}_{x_N^*} \end{bmatrix} \quad \text{with} \quad B^{-1} = \frac{1}{|B|} \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1n} \\ \vdots & \ddots & \vdots \\ \alpha_{m1} & \cdots & \alpha_{mn} \end{bmatrix}$$

where $\alpha_{ij} = (-1)^{i+j} |M_{ij}|$, with M_{ij} the square sub-matrix obtained from B by eliminating row i and column j.

- B integer $\Rightarrow \alpha_{ij}$ integer since how cofactors are computed
- the determinant
- If $|B| = \pm 1 \Rightarrow B^{-1}$ integer $\Rightarrow x^*$ integer.

It can be proved that A is totally unimodular, that is $|Q| \in \{-1, 0, 1\}$, for any square sub-matrix Q of A. If that |Q|=0 then it is the N, the if |Q|!=1 is a basis B (not sure)

4. Integer Linear Programming (ILP)

Example 4: Scheduling problem

Given

m machines, $k = 1, \ldots, m$

n jobs, $j = 1, \ldots, n$

d_j deadline for job j, $j = 1, \ldots, n$

p_{jk} processing time of job j on machine k (may be $= 0$)

Assumption: Each job must be processed once on each machine following the order of the machine indices $1, 2, \ldots, m$.

Determine an optimal sequence in which to process the jobs so as to minimize the total completion time while satisfying the deadlines.

Variables

t_{jk} = *time at which the processing of job j starts on machine k*

t = *upper bound on the completion time of all jobs*

$$y_{ijk} = \begin{cases} 1, & \textit{if job } i \textit{ precedes job } j \textit{ on machine } k \\ 0, & \textit{otherwise} \end{cases}$$

Parameter $M := \sum_{j=1}^{n} d_j$

4. Integer Linear Programming (ILP)

tik + pik <= tjk : job i preceeds job j in the machine k
tjk + pjk <= tik : job j preceeds job i in machine k

N.B. Queste due condizioni sono esclusive, quindi non possono valere allo stesso tempo. Un modo per modellare l'alternativa usiamo una binary variable. Introduciamo quindi una variabile binaria yijk, definita nella slide precedente. Ora mettiamo il caso che il job i precede j nella macchina k, dobbiamo imporre la prima delle due disuguaglianze ma non la seconda, come facciamo? Non possiamo moltiplicare per la variabile binaria perchè altrimenti il problema diventa non lineare. Sommiamo alla parte destra della disuguaglianza un valore talmente grande (chiamiamolo M) che la disuguaglianza è sempre soddisfatta. Otteniamo quindi le seguenti

$$
\begin{array}{lll}
\min & t & \\
\text{s. t.} & t_{jm} + p_{jm} \leq t, \quad j = 1, \ldots, m & \text{(t is upper bound on overall completion time)} \\
& t_{jm} + p_{jm} \leq d_j, \quad j = 1, \ldots, m & \text{(satisfy deadlines)} \\
& t_{ik} + p_{ik} \leq t_{jk} + M(1 - y_{ijk}), \quad \forall i, j, k \quad i < j & (1) \\
& t_{jk} + p_{jk} \leq t_{ik} + M\, y_{ijk}, \quad \forall i, j, k \quad i < j & (2) \\
& t_{jk} + p_{jk} \leq t_{j,k+1}, \quad j, k = 1, \ldots, m-1 & \text{--> sequence of execution on different machines} \\
& t \geq 0, \quad t_{jk} \geq 0, \quad j, k = 1, \ldots, m & \\
& y_{ijk} \in \{0, 1\}, \quad i, j, k = 1, \ldots, m &
\end{array}
$$

Mixed ILP!

But how large should we choose M? It should be higher than the maximum value that could be on the left-hand sight of the inequality. We want M large enough in order to make the inequality hold, but small enough to avoid numerical problems that could occur in the linear relaxation of the. We can use M=final deadline t

- (1) and (2) make sure that 2 jobs are not simultaneously processed on the same machine
- (1) active when $y_{ijk} = 1$ (i preceeds j on machine k) and ensures that i is completed before j starts (on k)
- (2) active when $y_{ijk} = 0$ (j preceeds i on machine k) and ensures that j is completed before i starts (on k).

ILP formulation can be extended to the case where each job j must be processed on (a subset of) the m machines according to a different order.

4. Integer Linear Programming (ILP)

Most ILP problems are NP-hard.
$\nexists$ efficient algorithms to solve them and the existence of a polynomial time algorithm for any one would imply P = NP!
extremely unlikely

Type of methods

- implicit enumeration: exact methods (global optimum) --> we have a set of integer solutions, identify the optimal without enumerating all of them
- cutting planes: exact methods (global optimum) --> apply a sequence of linear programming problems by adding constraint every time.
- heuristic algorithms (“greedy”, local search, . . .): approximate methods (local optimum)

Implicit enumeration methods explore all feasible solutions explicitly or implicitly.

- “Branch-and-bound” method
- Dynamic programming (see optimal paths in acyclic graphs)