Integer Linear Programming (ILP)

Zdeněk Hanzálek, Přemysl Šůcha
zdenek.hanzalek@cvut.cz

CTU in Prague

April 9, 2018

Problem Statement

Integer Linear Programming (ILP)

The ILP problem is given by matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ and vectors $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^n$. The goal is to find a vector $\mathbf{x} \in \mathbb{Z}^n$ such that $\mathbf{A} \cdot x \leq b$ and $c^T \cdot x$ is the maximum.

Usually, the problem is given as $\max \left\{ c^T \cdot x : \mathbf{A} \cdot x \leq b, x \in \mathbb{Z}^n \right\}$.

- A large number of practical optimization problems can be modeled and solved using Integer Linear Programming - ILP.

Comparison of ILP and LP

- The LP problem solution space is convex, since $x \in \mathbb{R}^n$
- The ILP problem differs from the LP problem in allowing integer-valued variables. If some variables can contain real numbers, the problem is called Mixed Integer Programming - MIP. Often MIP is also called ILP, and **we will use the term ILP when at least one variable has integer domain**.
- If we solve the ILP problem by an LP algorithm and then **just round the solution**, we could not only get the suboptimal solution, we can also obtain a solution which is not feasible.
- While the LP is solvable in polynomial time, **ILP is NP-hard**, i.e. there is no known algorithm which can solve it in polynomial time.
- Since the **ILP solution space is not a convex set**, we cannot use convex optimization techniques.

Example ILP1a: 2-Partition Problem

2-Partition Problem

- **Instance:** Number of banknotes $n \in \mathbb{Z}^+$ and their values $p_1, \dots, p_n$, where $p_{i \in 1..n} \in \mathbb{Z}^+$.
- **Decision:** Is there a subset $S \subseteq \{1, \dots, n\}$ such that $\sum_{i \in S} p_i = \sum_{i \notin S} p_i$?

The decision problem, which can be written while using the equation above as an ILP constraint (but we write it differently).

- $x_i = 1$ iff $i \in S$

This is one of the "easiest" NP-complete problems.

min $\quad 0$

subject to:

$$\sum_{i \in 1..n} x_i * p_i = 0.5 * \sum_{i \in 1..n} p_i$$

parameters: $\quad n \in \mathbb{Z}^+,\ p_{i \in 1..n} \in \mathbb{Z}^+$

variables: $\quad \mathbf{x_{i \in 1..n}} \in \{\mathbf{0}, \mathbf{1}\}$

Example ILP1b: Fractional Variant of the 2-Part. Prob.

We allow division of banknotes such that $x_{i \in 1..n} \in \langle 0, 1 \rangle$. The solution space is a convex set - the problem can be formulated by LP:

$$\boxed{\begin{array}{lrll} \text{min} & 0 & & \\ \text{subject to:} & & & \\ & \sum_{i \in 1..n} x_i * p_i & = 0.5 * \sum_{i \in 1..n} p_i & \\ & \mathbf{x_i} & \leq \mathbf{1} & i \in 1..n \\ \text{parameters:} & \multicolumn{3}{l}{n \in \mathbb{Z}_0^+,\ p_{i \in 1..n} \in \mathbb{Z}_0^+} \\ \text{variables:} & \multicolumn{3}{l}{\mathbf{x_{i \in 1..n}} \in \mathbb{R}_\mathbf{0}^+} \end{array}}$$

- For example: $p = [100, 50, 50, 50, 20, 20, 10, 10]$ the fractional variant allows for $x = [0, 0, 0.9, 1, 1, 1, 1, 1]$ and thus divides the banknotes into equal halves $100 + 50 + 5 = 45 + 50 + 20 + 20 + 10 + 10$, but this instance does not have a non-fractional solution.
- For some non-fractional instances we can easily find that they cannot be partitioned (e.g. when the sum of all values divided by the greatest common divisor is not an even number), however we do not know any alg that can do it in polynomial time for any non-fractional instance.

Example ILP1c: 2-Partition Prob. - Optimization Version

- The decision problem can be solved by an optimization algorithm while using a **threshold** value (here $0.5 * \sum_{i \in 1..n} p_i$) and comparing the optimal solution with the threshold.
- Moreover, when the decision problem has no solution, the optimization algorithm returns a value that is closest to the threshold.

$$\begin{array}{ll}
\text{min} & C_{max} \\
\text{subject to:} & \\
 & \sum_{i \in 1..n} x_i * p_i \leq C_{max} \\
 & \sum_{i \in 1..n} (1 - x_i) * p_i \leq C_{max} \\
\text{parameters:} & n \in \mathbb{Z}_0^+,\ p_{i \in 1..n} \in \mathbb{Z}_0^+ \\
\text{variables:} & x_{i \in 1..n} \in \{0,1\},\ C_{max} \in \mathbb{R}_0^+
\end{array}$$

Application: the scheduling of nonpreemptive tasks $\{T_1, T_2, ..., T_n\}$ with processing times $[p_1, p_2, ..., p_n]$ on two parallel identical processors and minimization of the completion time of the last task (i.e. maximum completion time C_{max}) - $P2 || C_{max}$. The fractional variant of 2-partition problem corresponds to the preemptive scheduling problem.

Example ILP2a: Shortest Paths

Shortest Path in directed graph

- **Instance:** digraph G with n nodes, distance matrix $c : V \times V \to \mathbb{R}_0^+$ and two nodes $s, t \in V$.
- **Goal:** find the shortest path from s to t or decide that t is unreachable from s.

LP formulation using a physical analogy:

- node = ball
- edge = string (we consider a symmetric distance matrix c)
- node s is fixed, other nodes are pulled by gravity
- tightened string = shortest path

$$
\begin{array}{ll}
\text{max} & l_t \\
\text{subject to:} & \\
\quad l_s = 0 & \\
\quad l_j \leq l_i + c_{i,j} \quad i \in 1..n, j \in 1..n & \\
\text{parameters:} & n \in \mathbb{Z}_0^+,\ c_{i \in 1..n, j \in 1..n} \in \mathbb{R}_0^+ \\
\text{variables:} & l_{i \in 1..n} \in \mathbb{R}_0^+
\end{array}
$$

Is it a polynomial problem?

Example ILP3: Traveling Salesman Problem

Asymmetric Traveling Salesman Problem

- **Instance:** complete digraph K_n, distance matrix $c : V \times V \to \mathbb{Q}^+$.
- **Goal:** find the shortest Hamiltonian cycle. Cycle is a subgraph $(v_1, ..., v_k, e_1, ..., e_k)$ such that the sequence $v_1, e_1, v_2, ..., v_k, e_k, v_1$ is a closed directed walk (tah) and $v_i \neq v_j$ for $1 \leq i < j \leq k$.

$x_{i,j} = 1$ iff node i is in the cycle just before node j
The enter and leave constraints do not capture the TSP completely, since any disjoint cycle (i.e. consisting of several sub-tours) will satisfy them. We use s_i, the "time" of entering node i, to **eliminate the sub-tours**.

$$
\begin{array}{llll}
\text{min} & \sum_{i\in 1..n}\sum_{j\in 1..n} c_{i,j} * x_{i,j} & & \\
\text{subject to:} & x_{i,i} = 0 & i \in 1..n & \text{avoid self-loop} \\
 & \sum_{i\in 1..n} x_{i,j} = 1 & j \in 1..n & \text{enter once} \\
 & \sum_{j\in 1..n} x_{i,j} = 1 & i \in 1..n & \text{leave once} \\
 & s_i + c_{i,j} - (1 - x_{i,j}) * M \leq s_j & i \in 1..n, j \in 2..n & \text{cycle indivisibility} \\
\text{parameters:} & M \in \mathbb{Z}_0^+,\ n \in \mathbb{Z}_0^+,\ c_{i\in 1..n, j\in 1..n} \in \mathbb{Q}^+ & & \\
\text{variables:} & x_{i\in 1..n, j\in 1..n} \in \{0,1\},\ s_{i\in 1..n} \in \mathbb{R}_0^+ & &
\end{array}
$$

Branch and Bound Method

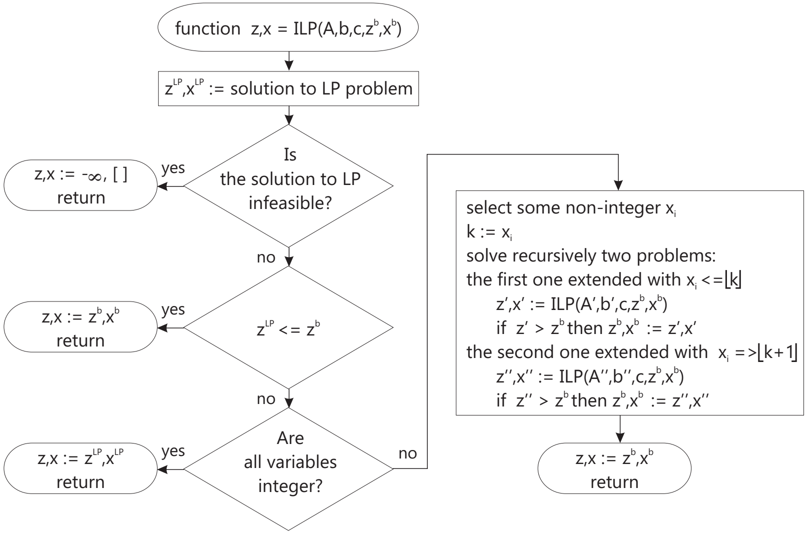
- The method is based on **splitting** the solution space into disjoint sets.
- It starts by relaxing on the integrality of the variables and **solves the LP problem**.
- If all variables x_i are **integers, the computation ends**. Otherwise one variable $x_i \notin \mathbb{Z}$ is chosen and its value is assigned to k.
- Then the solution space is **divided into two sets** - in the first one we consider $x_i \leq \lfloor k \rfloor$ and in the second one $x_i \geq \lfloor k \rfloor + 1$.
- The algorithm **recursively repeats** computation for the both new sets till feasible integer solution is found.

Branch and Bound Method

- By *branching* the algorithm creates a solution space which can be depicted as a **tree**.
- A node represents the **partial solution**.
- A **leaf** determines some (integer) solution or “bounded” branch (infeasible solution or the solution which does not give a better value than the best solution found up to now)
- As soon as the algorithm finds an integer solution, its objective function value can be used for *bounding*
    - The **node is discarded** whenever z, its (integer or real) objective function value, is worse than z^*, the value of the best known solution

The ILP algorithm often uses an LP **simplex method** because after adding a new constraint it is not needed to start the algorithm again, but it allows one to continue the previous LP computation while solving the dual simplex method.

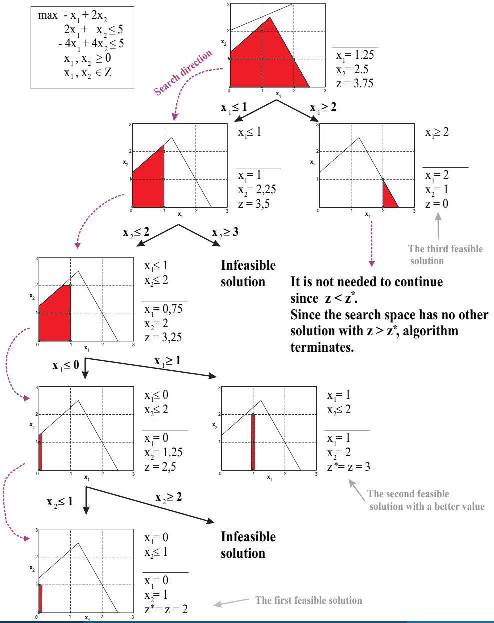
Branch and Bound Algorithm - ILP maximization problem



function z,x = ILP(A,b,c,z^b,x^b)
z^LP,x^LP := solution to LP problem
Is the solution to LP infeasible?
yes
z,x := -∞, [ ] return
no
z^LP <= z^b
yes
z,x := z^b,x^b return
no
Are all variables integer?
yes
z,x := z^LP,x^LP return
no
select some non-integer x_i
k := x_i
solve recursively two problems:
the first one extended with x_i <=⌊k⌋
z',x' := ILP(A',b',c,z^b,x^b)
if z' > z^b then z^b,x^b := z',x'
the second one extended with x_i =>⌊k+1⌋
z'',x'' := ILP(A'',b'',c,z^b,x^b)
if z'' > z^b then z^b,x^b := z'',x''
z,x := z^b,x^b return

Branch and Bound - Example

$$\begin{aligned} \max\ & -x_1 + 2x_2 \\ & 2x_1 + x_2 \le 5 \\ & -4x_1 + 4x_2 \le 5 \\ & x_1, x_2 \ge 0 \\ & x_1, x_2 \in Z \end{aligned}$$

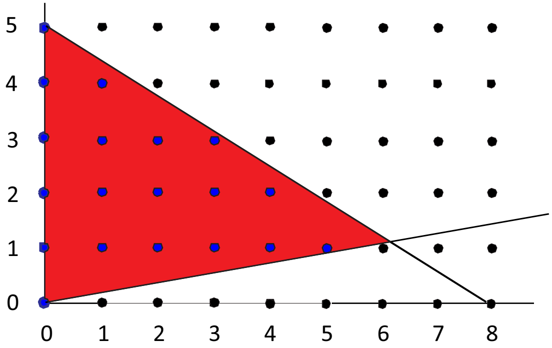


It is not needed to continue since $z < z^*$. Since the search space has no other solution with $z > z^*$, algorithm terminates.

ILP Solution Space

$$\begin{aligned} \max z = \quad & 3x_1 + 4x_2 \\ s.t. \quad & 5x_1 + 8x_2 \leq 40 \\ & x_1 - 5x_2 \leq 0 \\ & x_1, x_2 \in \mathbb{Z}_0^+ \end{aligned}$$

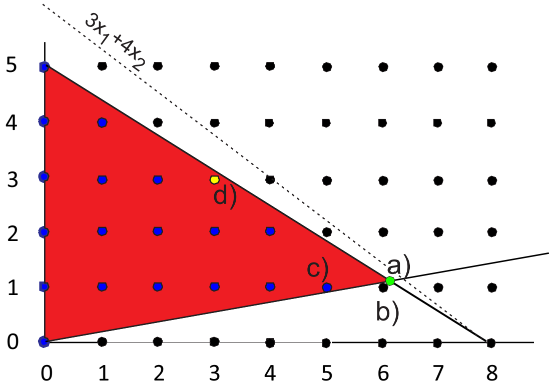
- What is optimal solution?
- Can we use LP to solve ILP problem?



Rounding is not always good choice

$$\max z = 3x_1 + 4x_2$$

- a) LP solution $z = 23.03$ for $x_1 = 4.8, x_2 = 0$
- b) Rounding leads to infeasible solution $x_1 = 6.06, x_2 = 1.21$
- c) Nearest feasible integer is not optimal $z = 19$ for $x_1 = 5, x_2 = 1$
- d) Optimal solution is $z = 21$ for $x_1 = 3, x_2 = 3$



Why integer programming?

Advantages of using integer variables

- more realistic (it does not make sense to produce 4.3 cars)
- flexible - e.g. binary variable can be used to model the decision (logical expression)
- we can formulate NP-hard problems

Drawbacks

- harder to create a model
- usually suited to solve the problems with less than 1000 integer variables

Special Cases of ILP - Example ILP2b: Shortest Paths

Shortest path in a graph

- **Instance:** digraph G given by incidence matrix $W : V \times E \to \{-1, 0, +1\}$ (such that $w_{ij} = +1$ when edge e_j leaves vertex i and $w_{kj} = -1$ when edge e_j enters vertex k), distance vector $c \in \mathbb{R}_0^+$ and two nodes $s, t \in V$.
- **Goal:** find the shortest path from s to t or decide that t is unreachable from s.

LP formulation:

- $x_j = 1$ iff edge j is chosen
- For every node except s and t we enter the node as many times as we leave it

$$\begin{array}{lll}
\text{min} & \sum_{j\in 1..m} c_j * x_j & \\
\text{subject to:} & & \\
\sum_{j\in 1..m} w_{s,j} * x_j & = 1 & \text{source } s \\
\sum_{j\in 1..m} w_{i,j} * x_j & = 0 & i \in V \setminus \{s, t\} \\
\sum_{j\in 1..m} w_{t,j} * x_j & = -1 & \text{sink } t \\
\text{pars:} & w_{i\in 1..n, j\in 1..m} \in \{-1, 0, 1\},\ c_{j\in 1..m} \in \mathbb{R}_0^+ & \\
\text{vars:} & x_{j\in 1..m} \in \mathbb{R}_0^+ &
\end{array}$$

The returned values of x_j are integers (binary) even though it is LP. Why?

Totally Unimodular Matrix leads to Integral Polyhedron

Polynomial time algorithm for general ILP is not known, however there are special cases which can be solved in polynomial time.

Definition - Totally unimodular matrix

Matrix $\mathbf{A} = [a_{ij}]$ of size m/n is totally unimodular if the determinant of every square submatrix of matrix $\mathbf{A}$ is equal 0, $+1$ or -1.

Necessary condition: if $\mathbf{A}$ is totally unimodular then $a_{ij} \in \{0, 1, -1\}\ \forall i, j$.

Lemma - Integral Polyhedron

Let A be a totally unimodular m/n matrix and let $b \in \mathbb{Z}^m$. Then each vertex of the polyhedron $P := \{x\,;\, \mathbf{A}x \leq b\}$ is an integer vector.

Proof: [Schrijver] Theorem 8.1.

Integer Solution by Polynomial Algorithm

Lemma - Integer solution by simplex algorithm

If the ILP problem is given by a totally unimodular matrix $\mathbf{A}$ and integer vector b then every feasible solution by a simplex algorithm is an integer vector.

Proof: From the Lemma on previous slide - the simplex algorithm inspects vertices that are integer vectors.
Unfortunately, the simplex algorithm does not have polynomial complexity.

Fortunately, there are polynomial algorithms able to solve the LP problems and to find the vertex in the facet with optimal solutions. But this subject is studied in Linear Programming and Polyhedral Computation.

Sufficient Condition for Totally Unimodular Matrix

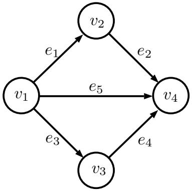
Lemma - Sufficient Condition

Let **A** be matrix of size m/n such that

1. $a_{ij} \in \{0, 1, -1\}$, $i = 1, ..., m$, $j = 1, ..., n$
2. each column in **A** contains one non-zero element or exactly two non-zero elements $+1$ and -1

Then matrix **A** is totally unimodular.

Proof: [Ahuja] Theorem 11.12. [KorteVygen] Theorem 5.26.
Example: ILP constraints of the Shortest Paths problem are: $W * x = b$



$$W = \begin{array}{c|ccccc} & e_1 & e_2 & e_3 & e_4 & e_5 \\ \hline v_1 & 1 & 0 & 1 & 0 & 1 \\ v_2 & -1 & 1 & 0 & 0 & 0 \\ v_3 & 0 & 0 & -1 & 1 & 0 \\ v_4 & 0 & -1 & 0 & -1 & -1 \end{array} \qquad x = \begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{array} \qquad b = \begin{array}{r} 1 \\ 0 \\ 0 \\ -1 \end{array}$$

Problem Formulation Using ILP - Real Estate Investment

We consider 6 buildings for investment.

The price and rental income for each of them are listed in the table.

building	1	2	3	4	5	6
price[mil. Kč]	5	7	4	3	4	6
income[thousands Kč]	16	22	12	8	11	19

Goal:

- maximize income

Constraints:

- investment budget is 14 mil Kč
- each building can be bought only once

Problem Formulation Using ILP - Real Estate Investment

We consider 6 buildings for investment.
The price and rental income for each of them are listed in the table.

building	1	2	3	4	5	6
price[mil. Kč]	5	7	4	3	4	6
income[thousands Kč]	16	22	12	8	11	19

Goal:

- maximize income

Constraints:

- investment budget is 14 mil Kč
- each building can be bought only once

Formulation

- $x_i = 1$ if we buy building i

$$\boxed{\begin{array}{ll} \max & z = 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6 \\ s.t. & \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14 \\ x_{i\in 1\cdots 6} \in \{0,1\} & \end{array}}$$

Adding Logical Formula $x_1 \Rightarrow x_2$

Another constraint:

- if building 1 is selected, then building 2 is selected too

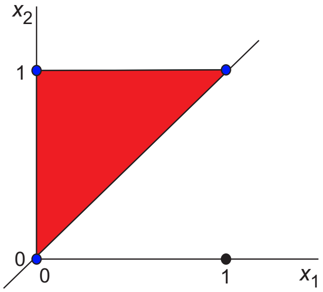
x_1	x_2	$x_1 \Rightarrow x_2$
0	0	1
0	1	1
1	0	0
1	1	1

Adding Logical Formula $x_1 \Rightarrow x_2$

Another constraint:

- if building 1 is selected, then building 2 is selected too

x_1	x_2	$x_1 \Rightarrow x_2$
0	0	1
0	1	1
1	0	0
1	1	1

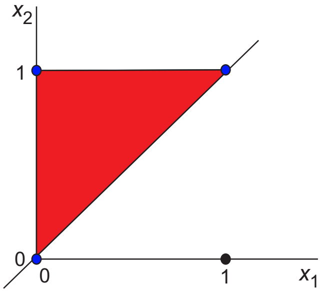


Adding Logical Formula $x_1 \Rightarrow x_2$

Another constraint:

- if building 1 is selected, then building 2 is selected too

x_1	x_2	$x_1 \Rightarrow x_2$
0	0	1
0	1	1
1	0	0
1	1	1



$$
\begin{array}{ll}
\max & z = 16x_1+22x_2+12x_3+8x_4+11x_5+19x_6 \\
s.t. & 5x_1+7x_2+4x_3+3x_4+4x_5+6x_6 \leq 14 \\
 & x_2 \geq x_1 \\
x_{i\in 1\cdots 6} \in \{0,1\} &
\end{array}
$$

Adding Logical Formula $x_3 \Rightarrow \overline{x_4}$

Another constraint:

- If building 3 is selected, then building 4 is not selected.

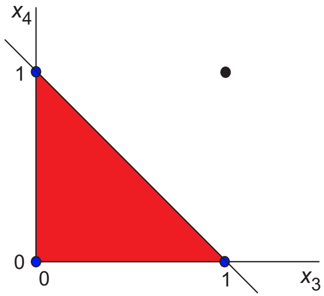
x_3	x_4	$x_3 \Rightarrow \overline{x_4}$
0	0	1
0	1	1
1	0	1
1	1	0

Adding Logical Formula $x_3 \Rightarrow \overline{x_4}$

Another constraint:

- If building 3 is selected, then building 4 is not selected.

x_3	x_4	$x_3 \Rightarrow \overline{x_4}$
0	0	1
0	1	1
1	0	1
1	1	0

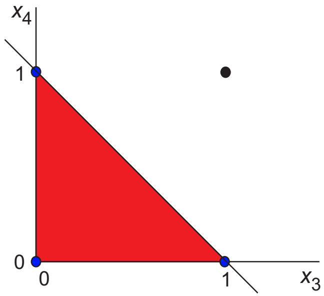


Adding Logical Formula $x_3 \Rightarrow \overline{x_4}$

Another constraint:

- If building 3 is selected, then building 4 is not selected.

x_3	x_4	$x_3 \Rightarrow \overline{x_4}$
0	0	1
0	1	1
1	0	1
1	1	0



$$\begin{array}{ll} \max & z = 16x_1+22x_2+12x_3+8x_4+11x_5+19x_6 \\ s.t. & 5x_1+7x_2+4x_3+3x_4+4x_5+6x_6 \le 14 \\ & x_3 + x_4 \le 1 \\ x_{i\in 1\cdots 6} \in \{0,1\} & \end{array}$$

Adding Logical Formula x_5 XOR x_6

Another constraint:

- either building 5 is chosen or building 6 is chosen, but not both

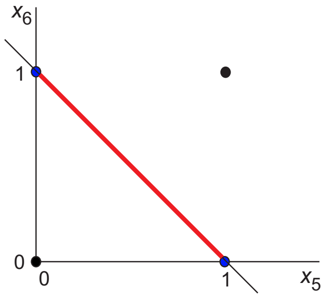
x_5	x_6	x_5 XOR x_6
0	0	0
0	1	1
1	0	1
1	1	0

Adding Logical Formula x_5 XOR x_6

Another constraint:

- either building 5 is chosen or building 6 is chosen, but not both

x_5	x_6	x_5 XOR x_6
0	0	0
0	1	1
1	0	1
1	1	0

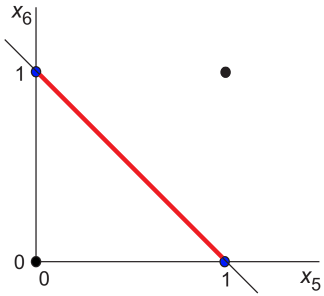


Adding Logical Formula x_5 XOR x_6

Another constraint:

- either building 5 is chosen or building 6 is chosen, but not both

x_5	x_6	x_5 XOR x_6
0	0	0
0	1	1
1	0	1
1	1	0



$$
\begin{array}{ll}
\max & z = 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6 \\
s.t. & 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14 \\
 & x_5 + x_6 = 1 \\
x_{i\in 1\cdots 6} \in \{0,1\} &
\end{array}
$$

Adding Logical Formula - Homework

Formulate:

- building 1 must be chosen but building 2 can not
- at least 3 estates must be chosen
- exactly 3 estates must be chosen
- if estates 1 and 2 have been chosen, then estate 3 must be chosen too $(x_1 \text{ AND } x_2) \Rightarrow x_3$
- exactly 2 estates can not be chosen

At least One of Two Constraints Must be Valid

While modeling problems using ILP, we often need to express that the first, the second or both constraints hold. For example, $x_{i \in 1 \ldots 4} \in \langle 0, 5 \rangle$, $x_{i \in 1 \ldots 4} \in \mathcal{R}$

$$\begin{array}{cc} \text{holds} & 2x_1 + 2x_2 \leq 8 \\ \text{or} & 2x_3 - 2x_4 \leq 2 \\ & \text{or both} \end{array}$$

This can be modeled by a big M, i.e. big positive number (here 15), and variable $y \in \{0, 1\}$ so it can “switch off” one of the inequalities.

$$2x_1 + 2x_2 \leq 8 + M \cdot y$$
$$2x_3 - 2x_4 \leq 2 + M \cdot (1 - y)$$

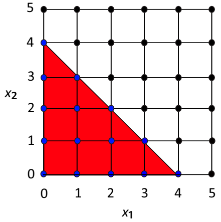
At least One of Two Constraints Must be Valid

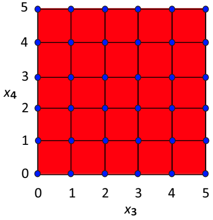
for $y = 0$ inequalities:

$$2x_1 + 2x_2 \leq 8 + M \cdot y$$
$$2x_3 - 2x_4 \leq 2 + M \cdot (1 - y)$$

reduce to:

$$2x_1 + 2x_2 \leq 8$$





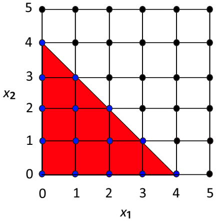
At least One of Two Constraints Must be Valid

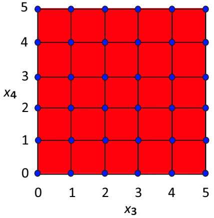
for $y = 0$ inequalities:

$$2x_1 + 2x_2 \leq 8 + M \cdot y$$
$$2x_3 - 2x_4 \leq 2 + M \cdot (1 - y)$$

reduce to:

$$2x_1 + 2x_2 \leq 8$$



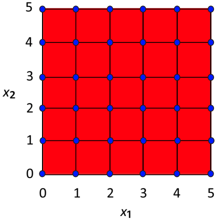


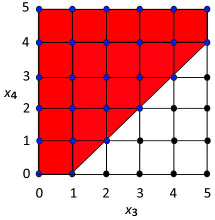
for $y = 1$ inequalities:

$$2x_1 + 2x_2 \leq 8 + M \cdot y$$
$$2x_3 - 2x_4 \leq 2 + M \cdot (1 - y)$$

reduce to:

$$2x_3 - 2x_4 \leq 2$$





At least One of Two Constraints Must be Valid - Homework

- In 2D draw the solution space of the system of inequalities:

$$\begin{gathered} 2x_1 + x_2 \leq 5 + M \cdot y \\ 2x_1 - x_2 \leq 2 + M \cdot (1 - y) \\ y \in \{0, 1\} \end{gathered}$$

- In 2D draw the solution space of the system of inequalities. Note that the equations correspond to parallel lines. Is it possible to find x_1, x_2 such that both equations are valid simultaneously?

$$\begin{gathered} 2x_1 + x_2 \leq 5 + M \cdot y \\ M \cdot (1 - y) + 2x_1 + x_2 \geq 10 \\ y \in \{0, 1\} \end{gathered}$$

At least One of Two Constraints Must be Valid Example: Non-preemptive Scheduling

$1 \left| r_j, \widetilde{d}_j \right| C_{max}$ ... NP-hard problem

- **Instance:** A set of non-preemptive tasks $\mathcal{T} = \{T_1, \ldots, T_i, \ldots T_n\}$ with release date r and deadline $\widetilde{d}$ should be executed on one processor. The processing times are given by vector p.
- **Goal:** Find a feasible schedule represented by start times s that minimizes completion time $C_{max} = \max_{i \in \langle 1,n \rangle} s_i + p_i$ or decide that it does not exist.

Example:

- T_i - chair to be produced by a joiner
- r_i - time, when the material is available
- $\widetilde{d}_i$ - time when the chair must be completed
- s_i - time when the chair production starts
- $s_i + p_i$ - time when the chair production ends

At least One of Two Constraints Must be Valid Example: Non-preemptive Scheduling

Since at the given moment, at most, one task is running on a given resource, therefore, for all task pairs T_i, T_j it must hold:

1. T_i precedes T_j ($s_j \geq s_i + p_i$)
2. or T_j precedes T_i ($s_i \geq s_j + p_j$)

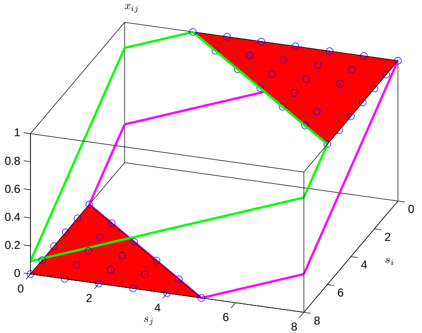
Note that (for $p_i > 0$) both inequalities can't hold simultaneously.
We need to formulate that at least one inequality holds. We will use variable $x_{ij} \in \{0, 1\}$ such that $x_{ij} = 1$ if T_i preceds T_j.
For every pair T_i, T_j we introduce inequalities:

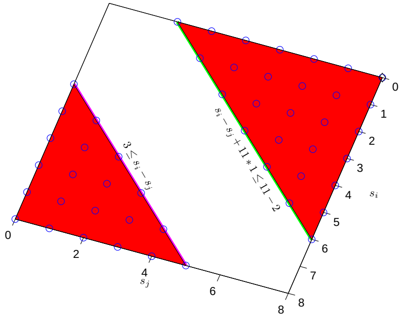
$$s_j + M \cdot (1 - x_{ij}) \geq s_i + p_i \qquad \text{“switched off” when } x_{ij} = 0$$
$$s_i + M \cdot x_{ij} \geq s_j + p_j \qquad \text{“switched off” when } x_{ij} = 1$$

Scheduling - Illustration of Non-convex Space

$$\begin{array}{rll} s_i \geq r_i & i \in 1..n & \text{release date} \\ \widetilde{d}_i \geq s_i + p_i & i \in 1..n & \text{deadline} \\ s_j + M \cdot (1 - x_{ij}) \geq s_i + p_i & i \in 1..n, j < i & T_i \text{ precedes } T_j \text{ GREEN} \\ s_i + M \cdot x_{ij} \geq s_j + p_j & i \in 1..n, j < i & T_j \text{ precedes } T_i \text{ VIOLET} \end{array}$$

For example: $p_i = 2, p_j = 3, r_i = r_j = 0, \widetilde{d}_i = 10, \widetilde{d}_j = 11, M = 11$





Non-convex 2D space is a projection of two cuts of a 3D polytope (determined by the set of inequalities) in planes $x_{ij} = 0$ and $x_{ij} = 1$.

At least K of N Constraints Must Hold

We have N constraints and we need at least K of them to hold.
Constraints are of type:

$$\begin{gathered} f(x_1, x_2, \dots, x_n) \leq b_1 \\ f(x_1, x_2, \dots, x_n) \leq b_2 \\ \vdots \\ f(x_1, x_2, \dots, x_n) \leq b_N \end{gathered}$$

Can be solved by introducing N variables $y_{i \in 1 \dots N} \in \{0, 1\}$ such that

$$\begin{gathered} f(x_1, x_2, \dots, x_n) \leq b_1 + M \cdot y_1 \\ f(x_1, x_2, \dots, x_n) \leq b_2 + M \cdot y_2 \\ \vdots \\ f(x_1, x_2, \dots, x_n) \leq b_N + M \cdot y_N \\ \sum_{i=1}^{N} y_i = N - K \end{gathered}$$

If $K = 1$ and $N = 2$ we can use just one variable y_i and represent its negation as a $(1 - y_i)$, see above slides for details.

ILP Solvers

- CPLEX - proprietary IBM `http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud`
- MOSEK - proprietary `http://www.mosek.com/`
- GLPK - free `http://www.gnu.org/software/glpk/`
- LP_SOLVE - free `http://groups.yahoo.com/group/lp_solve/`
- GUROBI - proprietary `http://www.gurobi.com/`

YALMIP - Matlab toolbox for modelling ILP problems
CVX - modeling framework

ILP - Conclusion

- NP-hard problem.
- Used to formulate majority of combinatorial problems.
- Often solved by branch and bound method.

References

Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin.
Network Flows: Theory, Algorithms, and Applications.
Prentice Hall, 1993.

B. H. Korte and Jens Vygen.
Combinatorial Optimization: Theory and Algorithms.
Springer, fourth edition, 2008.

James B. Orlin.
15.053 Optimization Methods in Management Science.
MIT OpenCourseWare, 2007.

Alexander Schrijver.
A Course in Combinatorial Optimization.
2006.