Preprocessing

Complements of Operations Research

Giovanni Righini

Università degli Studi di Milano

Preprocessing

Computational complexity theory classifies *problems*.

However, when we run algorithms, this is because we are confronted with specific problem *instances*.

The practical difficulty of solving a specific instance may well depend on its *data*, but it always depends on its *size*.

Hence, before running any algorithm to solve a discrete optimization problem instance, it is often useful to pre-process it to possibly reduce its size.

Goals

Pre-processing aims at:

- restricting bounds on variables;
- inserting constraints;
- fixing variables;
- removing redundant constraints;
- detecting infeasibility.

This can be done repeatedly for all *sub-problems* generated in algorithms based on decomposition and implicit enumeration.

Constraint programming is a discipline exploiting these techniques at their best.

Bounds tightening: example

If we have a constraint

$$y \leq ux$$

being y a continuous variable and x a binary variable, and we relax

$$x \in \{0, 1\}$$

into

$$0 \leq x \leq 1,$$

it is important that u take the smallest possible value, i.e. the maximum value y can take in any feasible solution.

Bounds tightening: example

$$
\begin{cases}
y_1 + y_2 = 1.46 \\
y_3 + y_4 = 0.72 \\
-y_2 - y_3 + y_5 = 0 \\
y_6 = 0.32 \\
-y_5 - y_6 + y_7 = 0 \\
0 \leq y_i \leq M x_i \quad \forall i = 1, \ldots, 7 \\
x_i \in \{0, 1\} \quad \forall i = 1, \ldots, 7
\end{cases}
$$

From this system, we can derive:

$$
\begin{cases}
y_1 \leq 1.46 x_1 \\
y_2 \leq 1.46 x_2 \\
y_3 \leq 0.72 x_3 \\
y_4 \leq 0.72 x_4 \\
y_5 \leq (1.46 + 0.72) x_5 \\
y_6 = 0.32 \\
y_7 \leq (1.46 + 0.72 + 0.32) x_7
\end{cases}
$$

Constraints insertion: example

Given the constraint

$$\sum_j a_j x_j \leq b$$

with $a_j > 0 \ \ \forall j$ and $x_j \in \{0, 1\} \ \ \forall j$, we can insert cover inequalities

$$\sum_{j \in C} x_j \leq |C| - 1 \ \ \forall C : \sum_{j \in C} a_j > b.$$

Strengthening bounds and inserting constraints yield *tighter formulations*.

Variable fixing: example

$$\begin{cases} x_1 + x_2 \leq 1 \\ x_1 + (1 - x_2) \leq 1 \\ x_1, x_2 \in \{0, 1\} \end{cases} \Rightarrow x_1 = 0$$

Redundant constraints detection: example

$$\begin{cases} -3x_2 - 2x_3 & \leq -2 \\ -4x_1 - 3x_2 - 3x_3 & \leq -6 \\ 2x_1 - 2x_2 + 6x_3 & \leq 5 \\ x \in \mathcal{B}^3 & \end{cases}$$

We define $\overline{x}_i = 1 - x_i \ \ \forall i = 1, \ldots, 3$.

$$\begin{cases} 3\overline{x}_2 + 2\overline{x}_3 & \leq 3 & \Rightarrow \overline{x}_2 + \overline{x}_3 \leq 1 \\ 4\overline{x}_1 + 3\overline{x}_2 + 3\overline{x}_3 & \leq 4 & \\ 2x_1 + 2\overline{x}_2 + 6x_3 & \leq 7 & \Rightarrow \overline{x}_2 + x_3 \leq 1 \\ x \in \mathcal{B}^3 & & \end{cases}$$

Therefore $\overline{x}_2 = 0$, i.e. $x_2 = 1$.

Redundant constraints detection: example

$$\begin{cases} +2\overline{x}_3 & \leq 3 & \text{redundant} \\ 4\overline{x}_1 + 3\overline{x}_3 & \leq 4 & \Rightarrow \overline{x}_1 + \overline{x}_3 \leq 1 \\ 2x_1 + 6x_3 & \leq 7 & \Rightarrow x_1 + x_3 \leq 1 \\ x \in \mathcal{B}^3 & & \end{cases}$$

Therefore $x_1 + x_3 = 1$.

And so on...

Fixing variables and removing redundant constraints yield *smaller instances*.

Preprocessing options

State-of-the-art solvers have powerful pre-processing capabilities.

They offer the possibility of turning pre-processing algorithms them on/off.

Domain reduction for integer variables and *constraint propagation* are two main concepts developed and exploited in *constraint programming* algorithms.

There are solvers based on constraint programming.

They are especially useful when finding a feasible solution is difficult, because of many sets of constraints.

Elementary pre-processing techniques

Given a mixed-integer problem,

$$MIP) \min\{cx + hy : Ax + Gy \leq b, x \in \mathcal{B}^n, y \in \Re^m\}$$

with bounds

$$l_j \leq y_j \leq u_j \quad \forall j = 1, \ldots, m$$

where some of the y variables can be constrained to be integer-valued, we consider, one at a time, the constraints

$$\sum_{j \in B_i^+} a_j x_j + \sum_{j \in B_i^-} a_j x_j + \sum_{j \in C_i^+} g_j y_j + \sum_{j \in C_i^-} g_j y_j \leq b$$

where

$$\begin{aligned} a_j &> 0 \quad \forall j \in B_i^+ \\ a_j &< 0 \quad \forall j \in B_i^- \\ g_j &> 0 \quad \forall j \in C_i^+ \\ g_j &< 0 \quad \forall j \in C_i^- \end{aligned}$$

We indicate the i^{th} constraint with $a^i x + g^i y \leq b_i$.

Infeasibility

If

$$\sum_{j \in B_i^+} a_j^i 0 + \sum_{j \in B_i^-} a_j^i 1 + \sum_{j \in C_i^+} g_j^i l_j + \sum_{j \in C_i^-} g_j^i u_j > b_i$$

then the problem instance is infeasible.

The minimum value the left-hand-side can take is too large.

Redundancy

If

$$\sum_{j \in B_i^+} a_j^i 1 + \sum_{j \in B_i^-} a_j^i 0 + \sum_{j \in C_i^+} g_j^i u_j + \sum_{j \in C_i^-} g_j^i l_j \leq b_i$$

then the constraint is redundant.

The maximum value the left-hand-side can take is small enough.

Bounds tightening

Let define for each $k = 1, \ldots, m$

$$\underline{z}_k^i = \min\{\sum_{j \in B_i^+} a_j^i x_j + \sum_{j \in B_i^-} a_j^i x_j + \sum_{j \in C_i^+} g_j^i y_j + \sum_{j \in C_i^-} g_j^i y_j - g_k^i y_k\}.$$

It corresponds to the minimum value the left-hand-side of constraint i can take neglecting the term $g_k^i y_k$.

If $k \in C_i^+$ we must have $\underline{z}_k^i + g_k^i y_k \leq b_i$ for all feasible values of y_k.

Therefore we can set

$$u_k := \min\{u_k, \frac{b_i - \underline{z}_k^i}{g_k^i}\}$$

Analogously, if $k \in C_i^-$ we can set

$$l_k := \max\{l_k, \frac{\underline{z}_k^i - b_i}{-g_k^i}\}$$

Elementary probing techniques

They consists in tentatively fixing a binary variable to 0 or to 1 and analyzing the consequences.

They can lead to:

- variable fixing
- coefficients improvement.

Variable fixing

Let define for each $k = 1, \ldots, n$

$$\underline{z}_k^i = \min\{\sum_{j \in B_i^+} a_j^i x_j + \sum_{j \in B_i^-} a_j^i x_j - a_k^i x_k + \sum_{j \in C_i^+} g_j^i y_j + \sum_{j \in C_i^-} g_j^i y_j\}.$$

If $k \in B_i^+$ and $\underline{z}_k^i + a_k^i > b_i$, then we can fix $x_k := 0$.

If $k \in B_i^-$ and $\underline{z}_k^i > b_i$, then we can fix $x_k := 1$.

Coefficients improvement

Consider a variable index $k \in B_i^+$ and define

$$\overline{z}_k^i = \max\{\sum_{j \in B_i^+} a_j^i x_j - a_k^i x_k + \sum_{j \in B_i^-} a_j^i x_j + \sum_{j \in C_i^+} g_j^i y_j + \sum_{j \in C_i^-} g_j^i y_j\}$$

If $\overline{z}_k^i \leq b_i$, then constraint i is redundant for $x_k = 0$.

Hence, when $x_k = 0$ we can reduce both a_k^i and b_i by an amount (slack) $\delta = b_i - \overline{z}_k^i$.

This modification is valid also when $x_k = 1$ because we subtract the same amount from both sides of the inequality.

Therefore we can make constraint i tighter by setting:

$$a_k^i := a_k^i - \delta \quad b_i := b_i - \delta$$

Analogously we can improve coefficients for $k \in B_i^-$.

Computational complexity

The bounds $\overline{z}^i$ and $\underline{z}^i$ can be computed in $O(mn)$.

- Infeasibility: $\underline{z}^i > b_i$
- Redundancy: $\overline{z}^i \leq b_i$
- Bounds tightening:
$$u_k > \frac{b_i - (\underline{z}^i - g_k^i l_k)}{g_k^i} \quad k \in C_i^+$$
$$l_k < \frac{(\underline{z}^i - g_k^i u_k) - b_i}{-g_k^i} \quad k \in C_i^-$$
- Variable fixing:
$$\underline{z}^i + a_k^i > b_i \quad k \in B_i^+$$
$$\underline{z}^i - a_k^i > b_i \quad k \in B_i^-$$
- Coefficients improvement:
$$\overline{z}^i - a_k^i < b_i \quad k \in B_i^+$$
$$\overline{z}^i + a_k^i < b_i \quad k \in B_i^-$$

They are $O(mn)$ tests. Each of them takes constant time.

Advanced preprocessing techniques

They are similar to the elementary preprocessing and probing techniques, but they consider more than one constraint at a time.

They are based on logical implications: by fixing a binary variable and running the elementary preprocessing techniques on the resulting constraint system, we can fix additional variables or strengthen the coefficients of other variables.

These logical implications can be used for:

- Advanced probing
- Constraints insertion
- Variable elimination

Advanced probing

We tentatively fix $x_k = 0$ (or $x_k = 1$) and we run the elementary preprocessing techniques.

If the resulting instance is infeasible, then we can fix $x_k = 1$ (or $x_k = 0$).

If a constraint i is redundant, then we can improve the coefficient of x_k in constraint i.

Logical inequalities

Consider a logical relation

$$\text{If } x_i = 1 \text{ then } y_j = v_j$$

where v_j is a datum. The relation can be translated into linear inequalities in this way:

$$\begin{cases} y_j \geq l_j + (v_j - l_j)x_i \\ y_j \leq u_j - (u_j - v_j)x_i \end{cases}$$

This allows for *automatic disaggregation of constraints*. From an aggregated constraint like

$$\sum_j y_j \leq (\sum_j u_j)x_i$$

we obtain logical implications

$$\text{If } x_i = 0 \text{ then } y_j = 0 \ \ \forall j$$

and therefore the disaggregated constraints

$$y_j \leq u_j x_i \ \ \forall j.$$

Clique inequalities

A logical relation between two binary variables can be expressed as

$$\text{If } x_i' = 1 \text{ then } x_j' = 0$$

where each x' can be a variable or its complement. The relation states that it is not allowed that both variables take value 1.

The *incompatibility graph* $G = (B^o, B^c, E)$ is defined as follows:

- B^o is the vertex set corresponding to the original variables,
- B^c is the vertex set corresponding to their complement,
- E has an edge for each pair of incompatible variables.

In G every clique C is made by two subsets $C^o \subseteq B^o$ and $C^c \subseteq B^c$.

Clique inequalities

Every clique C of G corresponds to a valid inequality

$$\sum_{j:B_j^o \in C^o} x_j + \sum_{j:B_j^c \in C^c} \overline{x}_j \leq 1.$$

Moreover, if there is only one variable index $k : B_k^o \in C^o \wedge B_k^c \in C^c$,

then

$$\begin{cases} x_j = 0 \ \ \forall j : B_j^o \in C^o, j \neq k \\ x_j = 1 \ \ \forall j : B_j^c \in C^c, j \neq k \end{cases}$$

If, instead, there are two or more indices k such that $B_k^o \in C^o \wedge B_k^c \in C^c$, then the instance is infeasible.

Variable elimination

If $\begin{cases} x_i = 0 & \Rightarrow y_j = v_j \\ x_i = 1 & \Rightarrow y_j = v_j \end{cases}$ then $y_j = v_j$.

If $\begin{cases} x_i = 0 & \Rightarrow y_j = l_j \\ x_i = 1 & \Rightarrow y_j = u_j \end{cases}$ then $y_j = l_j + (u_j - l_j)x_j$.

In these cases we obtain equality constraints and hence we can eliminate variables by substitution.

Probing on constraints

We tentatively fix the value of the right-hand-side of an inequality constraint and we analyze the consequences.

For instance, with a clique inequality $\sum_{j \in C} x_j \leq 1$ we can have two cases:

- either $\sum_{j \in C} x_j = 0$
- or $\sum_{j \in C} x_j = 1$.

If we fix $\sum_{j \in C} x_j = 1$ and we obtain an infeasible instance, then we can set $x_j = 0 \; \forall j \in C$.

If we fix $\sum_{j \in C} x_j = 1$ and we find a redundant constraint i, then we can improve the coefficients of $x_j \; \forall j \in C$ in constraint i.