DM872

Math Optimization at Work

Presolving Techniques in Linear Programming
Enhancing Solver Efficiency through Problem Simplification

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

Outline

Outline

Introduction to Presolving

Definition: Presolving refers to the preprocessing phase in linear programming where the problem is analyzed and simplified before applying optimization algorithms.

Purpose: To reduce problem size, eliminate redundancies, and improve solver efficiency.

Primal Reductions are solely based on reasoning applied to the feasible region, while dual reductions consider the objective function.

Side effects: asking for dual values, solution removed

Objectives of Presolving

- Simplification: Reduce the number of variables and constraints
- Detection: Identify infeasibility or unboundedness early
- Enhancement: Improve numerical stability and solution accuracy

Outline

1. Introduction

2. Techniques

Common Presolving Techniques

- Eliminating Redundant Constraints: Removing constraints that do not affect the feasible region
- Variable Fixing: Assigning fixed values to certain variables based on constraint analysis
- Constraint Aggregation: Combining multiple constraints into a single constraint to simplify the problem
- Coefficient Reduction: Simplifying coefficients to improve numerical stability

Advanced Presolving Techniques

- Probing: Testing variable assignments to deduce further reductions.
- Dominated Columns/Rows Removal: Eliminating variables or constraints that are dominated by others.
- Constraint Sparsification: Reducing the density of constraints to simplify the problem structure.

Common Presolving Techniques: Examples

Consider $S = \{\boldsymbol{x} : a_0x_0 + \sum_{j=1}^{n} a_jx_j \leq b, l_j \leq x_j \leq u_j, j = 0..n\}$

- Bounds on variables.
  If $a_0 > 0$ then:

$$x_0 \leq \left(b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j \right) / a_0$$

  and if $a_0 < 0$ then

$$x_0 \geq \left(b - \sum_{j:a_j>0} a_j l_j - \sum_{j:a_j<0} a_j u_j \right) / a_0$$

- Redundancy. The constraint $\sum_{j=0}^{n} a_jx_j \leq b$ is redundant if

$$\sum_{j:a_j>0} a_j u_j + \sum_{j:a_j<0} a_j l_j \leq b$$

- Infeasibility: $S = \emptyset$ if (swapping lower and upper bounds from previous case)

$$\sum_{j:a_j>0} a_j l_j + \sum_{j:a_j<0} a_j u_j > b$$

- Variable fixing. For a max problem in the form

$$\max\{\boldsymbol{c}^T\boldsymbol{x} : A\boldsymbol{x} \leq \boldsymbol{b}, \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}\}$$

if $\forall i = 1..m : a_{ij} \geq 0, c_j < 0$ then fix $x_j = l_j$
if $\forall i = 1..m : a_{ij} < 0, c_j > 0$ then fix $x_j = u_j$

- Integer variables:

$$\lceil l_j \rceil \leq x_j \leq \lfloor u_j \rfloor$$

Example

$$
\begin{aligned}
\max\ & 2x_1 + x_2 - x_3 \\
\mathtt{R1}:\ & 5x_1 - 2x_2 + 8x_3 \leq 15 \\
\mathtt{R2}:\ & 8x_1 + 3x_2 - x_3 \geq 9 \\
\mathtt{R3}:\ & x_1 + x_2 + x_3 \leq 6 \\
& 0 \leq x_1 \leq 3 \\
& 0 \leq x_2 \leq 1 \\
& x_3 \geq 1
\end{aligned}
$$

$$\mathtt{R1}: 5x_1 \leq 15 + 2x_2 - 8x_3 \leq 15 + 2 \cdot \overbrace{1}^{u_2} - 8 \cdot \overbrace{1}^{l_3} = 9 \qquad \rightsquigarrow x_1 \leq 9/5$$

$$8x_3 \leq 15 + 2x_2 - 5x_1 \leq 15 + 2 \cdot 1 - 5 \cdot 0 = 17 \qquad \rightsquigarrow x_3 \leq 17/8$$

$$2x_2 \geq 5x_1 + 8x_3 - 15 \geq 5 \cdot 0 + 8 \cdot 1 = -7 \qquad \rightsquigarrow x_2 \geq -7/2, x_2 \geq 0$$

$$\mathtt{R2}: 8x_1 \geq 9 - 3x_2 + x_3 \geq 9 - 3 + 1 = 7 \qquad \rightsquigarrow x_1 \geq 7/8$$

$$\mathtt{R1}: 8x_3 \geq 15 + 2x_2 - 5x_1 \leq 15 + 2 - 5 \cdot 7/8 = 101/8 \qquad \rightsquigarrow x_3 \leq 101/64$$

$\mathtt{R3}: x_1 + x_2 + x_3 \leq 9/5 + 1 + 101/64 < 6$ Hence $\mathtt{R3}$ is redundant

Example

$$
\begin{aligned}
\max\ & 2x_1 + x_2 - x_3 \\
\mathtt{R1}:\ & 5x_1 - 2x_2 + 8x_3 \leq 15 \\
\mathtt{R2}:\ & 8x_1 + 3x_2 - x_3 \geq 9 \\
& 7/8 \leq x_1 \leq 9/5 \\
& 0 \leq x_2 \leq 1 \\
& 1 \leq x_3 \leq 101/64
\end{aligned}
$$

Increasing x_2 makes constraints satisfied $\rightsquigarrow$ $x_2 = 1$
Decreasing x_3 makes constraints satisfied $\rightsquigarrow$ $x_3 = 1$

We are left with:

$$\max\{2x_1 : 7/8 \leq x_1 \leq 9/5\}$$

Probing

Set some binary variable tentatively to zero or one and derive further or stronger inequalities or better bounds

- x_k binary variable, x_j arbitrary variable with bounds $\ell_j \leq x_j \leq u_j$
- ℓ_j^0 and u_j^0 bounds of x_j deduced from setting $x_k := 0$
  ℓ_j^1 and u_j^1 bounds of x_j deduced from setting $x_k := 1$
- the following observations can be made:
  1. If setting $x_k = 0$ leads to an infeasible problem, we can fix $x_k := 1$. Conversely, if $x_k = 1$ is infeasible, we can fix $x_k := 0$.
  2. If $\ell_j^0 = u_j^0$ and $\ell_j^1 = u_j^1$, x_j can be substituted as $x_j := \ell_j^0 + (\ell_j^1 - \ell_j^0) \cdot x_k$. Note that for $\ell_j^0 = \ell_j^1$ this meas to fix $x_j := \ell_j^0$.
  3. We can deduce valid global bounds of x_j by $\ell_j := \min\{\ell_j^0, \ell_j^1\}$ and $u_j := \min\{u_j^0, u_j^1\}$
  4. otherwise we store valid implications on the bounds of x_j depending on the value of x_k

$$x_k = 0 \implies x_j \geq \ell_j^0 \qquad x_k = 1 \implies x_j \geq \ell_1$$
$$x_k = 0 \implies x_j \leq u_j^0 \qquad x_k = 1 \implies x_j \leq u_1$$

Presolving for Set Covering/Partitioning

1. if $\mathbf{e}_i^T A = \mathbf{0}$ then the ith row can never be satisfied

$$\begin{bmatrix}0 & 0 & \dots & 1 & \dots & 0\end{bmatrix}\begin{bmatrix} & & & \\ \hdashline 0 \dots 0 & & \dots & 0 \\ \hdashline & & & \end{bmatrix} = \begin{bmatrix}0\\0\\0\\\vdots\\0\\0\end{bmatrix}$$

2. if $\mathbf{e}_i^T A = \mathbf{e}_k$ then $x_k = 1$ in every feasible solution

$$\begin{bmatrix}0 & 0 & \dots & 1 & \dots & 0\end{bmatrix}\begin{bmatrix} & & & \\ \hdashline 0 & \dots & 1 & \dots & 0 \\ \hdashline & & & \end{bmatrix} = \begin{bmatrix}0\\\vdots\\1\\\vdots\\0\\0\end{bmatrix}$$

In SPP can remove all rows t with $a_{tk} = 1$ and set $x_j = 0$ (ie, remove cols) for all cols that cover t

3. if $\boldsymbol{e}_t^T A \geq \boldsymbol{e}_p^T A$ then we can remove row t, row p dominates row t (by covering p we cover t)

$$
\begin{array}{c}
t \\ \\ p \\ \\
\end{array}
\begin{bmatrix}
 & 1 & 1 & & 1 & \\
 & & & & & \\
 & 1 & & & 1 & \\
 & & & & &
\end{bmatrix}
$$

In SPP we can remove all cols j: $a_{tj} = 1, a_{pj} = 0$

4. if $\sum_{j \in S} A\boldsymbol{e}_j = A\boldsymbol{e}_k$ and $\sum_{j \in S} c_j \leq c_k$ then we can cover the rows by $A\boldsymbol{e}_k$ more cheaply with S and set $x_k = 0$
(Note, we cannot remove S if $\sum_{j \in S} c_j \geq c_k$)

$$
\begin{bmatrix}
 & & 1 & & & 1 \\
 & 1 & & & & 1 \\
 & & & 1 & & 1 \\
 & 0 & 0 & 0 & & 0 \\
 & 1 & & & & 1 \\
 & 0 & 0 & 0 & & 0
\end{bmatrix}
$$

Hot Topic

- MIP challenge 2024 on Presolving Reductions `https://github.com/dominiqs81/MIPcc24`
- Papilo, parallel presolving `https://github.com/dominiqs81/MIPcc24`

Summary

1. Introduction

2. Techniques