Extended LP Presolve Analysis

Ramchandran Muthukumar

Email : ramcha1994@gmail.com
Github : ramcha24

Contents

1. Introduction

Considering optimization problems of the format,

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & lb \leq Ax \leq ub \\ & l \leq x \leq u \end{aligned}$$

where $x, c, l, u \in \mathbb{R}^n$ and $lb, ub \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$ and the solutions are interpreted as,
x is the primal column variable.
$y = Ax$ is the primal row variable.
Some of the simple column l_j, u_j and row bounds lb_j, ub_j may be $-\infty, \infty$.
Define $L = \{j : l_j > -\infty\}$ and $U = \{j : u_j < \infty\}$
We can re-write this problem as -

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & f(x) \preceq \vec{0} \\ & g(x) \preceq \vec{0} \\ & p(x) \preceq \vec{0} \\ & q(x) \preceq \vec{0} \end{aligned}$$

where $x \in \mathbb{R}^n$,

$$f, g : \mathbb{R}^m \to \mathbb{R}^m$$

$$p, q : \mathbb{R}^n \to \mathbb{R}^n$$

are defined as

$$\begin{aligned} [f(x)]_i &= lb[i] - \left(\sum_{j=1}^{n} a_{ij} \cdot xj \right) \\ [g(x)]_i &= \left(\sum_{j=1}^{n} a_{ij} \cdot xj \right) - ub[i] \\ [p(x)]_j &= l[j] - xj \\ [q(x)]_j &= xj - u[j] \end{aligned}$$

Constructing the Lagrangian $L : (\mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}^n) \to \mathbb{R}^n$ we get,

$$L(x, \alpha, \beta, \gamma, \delta) = c^T x + \alpha^T f(x) + \beta^T g(x) + \gamma^T p(x) + \delta^T q(x)$$

where, $\alpha, \beta \in \mathbb{R}^m, \gamma, \delta \in \mathbb{R}^n$ are the lagrange multipliers and can be interpreted as,
α = Dual variable for lower bound row inequality
β = Dual variable for upper bound row inequality
γ = Dual variable for lower bound col inequality
δ = Dual variable for upper bound col inequality

The Lagrange Dual Function is,

$$\begin{aligned} g(\alpha,\beta,\gamma,\delta) &= \inf_x \quad L \\ &= \inf_x \left(\alpha^t \cdot lb - \beta^T \cdot ub + \gamma^T \cdot l - \delta^T \cdot u\right) \\ &+ x \cdot \left(c^T + A(\beta^T - \alpha^T) + (\delta^T - \gamma^T)\right) \\ &= \left(\alpha^t \cdot lb - \beta^T \cdot ub + \gamma^T \cdot l - \delta^T \cdot u\right) \\ &, \quad \text{if } \left(c^T + A(\beta^T - \alpha^T) + (\delta^T - \gamma^T) = 0\right) \\ &- \infty, \quad \text{else} \end{aligned}$$

Therefore the Dual Problem is ,

$$\begin{aligned} \max_{\alpha,\beta,\gamma,\delta} \quad & g(\alpha,\beta,\gamma,\delta) \\ \text{subject to} \quad & c = A^t(\alpha-\beta) + (\gamma - \delta) \\ & \alpha,\beta,\gamma,\delta \succeq \vec{0} \end{aligned}$$

This gives us the dual constraint -

$$c = A^t(\alpha-\beta) + (\gamma - \delta)$$

Complimentary Slackness conditions gives us -

$$\begin{aligned} \alpha_i^\star \cdot [f(x^\star)]_i &= 0 \\ \beta_i^\star \cdot [g(x^\star)]_i &= 0 \\ \gamma_i^\star \cdot [p(x^\star)]_i &= 0 \\ \delta_i^\star \cdot [q(x^\star)]_i &= 0 \end{aligned}$$

2. States

Each inequality bound pair (l and u) has a set of states as enumerated below.

(1) $l = -\infty \quad , u = \infty$
(2) $l = -\infty \quad , u < \infty$
  (a) $u < 0$
  (b) $u = 0$
  (c) $u > 0$
(3) $l > -\infty \quad , u = \infty$
  (a) $l < 0$
  (b) $l = 0$
  (c) $l > 0$
(4) $l > -\infty \quad , u < \infty$
  (a) $l = u = b$
    (i) $b < 0$
    (ii) $b = 0$
    (iii) $b > 0$
  (b) $l! = u$
    (i) $l < 0 \quad , u < 0$
    (ii) $l < 0 \quad , u = 0$
    (iii) $l < 0 \quad , u > 0$
    (iv) $l = 0 \quad , u < 0$ (infeasible)
    (v) $l = 0 \quad , u = 0$ (already seen before)

(vi) $l = 0 \quad , u > 0$
(vii) $l > 0 \quad , u < 0$ (infeasible)
(viii) $l > 0 \quad , u = 0$ (infeasible)
(ix) $l > 0 \quad , u > 0$

Thus each inequality bound pair has 15 possible feasible states.

Typically with each presolve constraint there is one row and/or one column involved and the overall state of (lb, ub, l, u) helps us decide whether there is redundant information and what action to take

If row i is deemed to be redundant, then we can set the dual variables α_i and β_i and y_i to be a Linear Dependency and then the row can be removed.

If column j is deemed to be redundant, then we can set the dual variables γ_j and δ_j and x_j to be a Linear Dependency and then the column can be removed.

Some kinds of presolve redundancies leads to the elimination of both a row and one or more columns.

The dual variables are set either using the Complimentary Slackness condition or the dual equality constraint. By convention we can set $\beta = 0$ and $\delta = 0$ to ensure that $\alpha \geq \beta$ and $\gamma \geq \delta$.

The final reported solution after postsolving should be $(x, y, \alpha, \beta, \gamma, \delta)$ or an error status of the form - *Infeasible* or *Unbounded*.

3. General Analysis

Each Row is classified to be one of - *Empty*, *Singleton*, *Forcing*, *Normal*

Each column is classified to be one of - *Empty*, *Singleton*, *Normal*

We can think of three generic types of presolve sub-routines -

We will have a separate presolve routine for each. Earlier we had a common postsolve routine for everything but now we need to atleast pass some extra information so that we can track which kind of redundancy caused a particular addition. Some redundancies like forcing result in multiple additions to the presolve stack, so we might also have to record that certain additions were the result of a single redundancy.

At first look, it looked like row and column bounds seemed all important but while they are important to fix the dual values, we will see later that we can usually come up with an abstraction that is independent of the different states of a row or column bound and sometimes though we need to consider whether a row constraint is actually an equality or inequality.

Apart from the row bounds and column bounds , sometimes in subroutine3(), we might need to consider the status of a_{ij} and c_j.

The Presolve Stack element will have a slight difference. We only add fixed values or linear dependencies of $x_j and y_i$. I think this should be enough to later gather the details required to postsolve the other values.

We should never come across a state where the matrix element a_{ij} concerned with the presolve constraint is zero. This should be debugged as a serious error in the data setup.

4. Empty Row

This happens when $\exists i$s.t. $a_{ij} = 0 \forall j$

Here, $[f(x^\star)]_i = lb[i]$ and $[g(x^\star)]_i = -ub[i]$ and $y_i = 0$

Here, as long as the row bounds aren't infeasible or disallow $y_i = 0$, we can set $y_i = \alpha_i = \beta_i = 0$

The cases when there will be infeasibility are - when $ub_i < 0$ or when $lb_i > 0$ or when $lb_i > ub_i$

5. Singleton Row

This happens when

$$\exists(i,j) \quad \text{s.t.} \quad a_{ij}! = 0 \quad \text{and} \quad a_{ik} = 0 \quad \forall \quad k! = j$$

Here,

$$\begin{aligned} f_i(x^\star) &= \ lb[i] - a_{ij} \cdot x_j \\ g_i(x^\star) &= \ a_{ij} \cdot x_j - ub[i] \\ p_j(x^\star) &= \ l[j] - x_j \\ q_j(x^\star) &= \ x_j - u[j] \end{aligned}$$

We can define l_{new} and u_{new} as,

$$u_{\text{new}} = \begin{cases} \dfrac{ub_i}{a_{ij}}, & a_{ij} > 0 \\ \dfrac{lb_i}{a_{ij}}, & a_{ij} < 0 \end{cases}$$

$$l_{\text{new}} = \begin{cases} \dfrac{ub_i}{a_{ij}}, & a_{ij} < 0 \\ \dfrac{lb_i}{a_{ij}}, & a_{ij} > 0 \end{cases}$$

$$l_{\text{new}} \leq x_j \leq u_{\text{new}}$$

In all cases, we will update the column bounds as - $l_j^{'} := \max(l_{\text{new}}, l_j)$ and $u_j^{'} := \min(u_j, u_{\text{new}})$ and remove the row moving forward.

There are two distinct end-results when we process a singleton row (i,j) - fixing the column variable x_j and its duals, fixing the row variable y_i and its duals. The later can be done for any singleton row, but the former requires that $lb_i == ub_i$.

In the latter case, we assume that the column variable x_j and its duals are already known, either by future presolve routines or by the solver itself and use these to determine our row variables y_i and duals.

The order of postsolve is the reverse of presolve and thus this assumption makes sense.

For all the different cases, we can add $y_i = a_{ij} \cdot x_j$ as a Linear Dependency, the values of the dual variables wil depend on status of the row bounds and on the postsolved values of the column variables.

(1) $lb = -\infty \ \ , ub = \infty$

Here, $f_i, g_i < 0 \Rightarrow \alpha_i, \beta_i = 0$

We can add $y_i = a_{ij} \cdot x_j$ as a Linear Dependency

Row i can now be removed.

(2) $lb_i = -\infty \ \ , ub_i < \infty$

Here, $f_i < 0 \Rightarrow \alpha_i = 0$

We can add $y_i = a_{ij} \cdot x_j$ as a Linear Dependency

Here it doesn't matter whether ub_i is positive, zero or negative but instead we will need to consider a_{ij} to determine the actual bounds given by the row.

(I) $a_{ij} < 0$

In this case, $l_{\text{new}} = \dfrac{ub_i}{a_{ij}}$ and $u_{\text{new}} = \infty$

(A) $l_{\text{new}} > l_j$

In this case $l_j^{'} = l_{\text{new}}$ and $u_j^{'} = u_j$

Supose the postsolved value of variable x_j, we have to consider the different states of x_j.

(1) $l_j^{'} < x_j < u_j^{'}$

The row is inactive and $f_i, g_i < 0 \Rightarrow \beta_i = 0$. The column variables and dual values will remain the same as before.

(2) $l_j^{'} = x_j$

Here, the column is actually basic as $l_j < x_j < u_j$. The row on the other hand is non-basic at its lowerbound, therefore, $-\beta_i = \frac{(\gamma_j - \delta_j)}{a_{ij}}$ Now we set $\gamma_j = \delta_j = 0$ and the row has now been postsolved.

(3) $x_j = u_j^{'}$

Here, the column is nonbasic at its upperbound, and the row is basic as $lb_i < a_{ij} \cdot x_j < ub_i$, therefore $\beta_i = 0$

(B) $l_{\text{new}} <= l_j$

In this case $l_j^{'} = l_j$ and $u_j^{'} = u_j$

For all states of the postsolved value of x_j , the row i is basic and thus we can straight away set $\beta_i = 0$

(II) $a_{ij} = 0$

We should never reach this state as we are initially creating a sparse matrix of non-zero values only.

(III) $a_{ij} > 0$

In this case, $l_{\text{new}} = \infty$ and $u_{\text{new}} = \dfrac{ub_i}{a_{ij}}$

(A) $u_{\text{new}} < u_j$

In this case $l_j^{'} = l_j$ and $u_j^{'} = u_{\text{new}}$

Supose the postsolved value of variable x_j, we have to consider the different states of x_j.

(1) $l_j^{'} < x_j < u_j^{'}$

The row is inactive and $f_i, g_i < 0 \Rightarrow \beta_i = 0$. The column variables and dual values will remain the same as before.

(2) $l_j^{'} = x_j$

Here, the column is nonbasic at its lowerbound, and the row is basic as $lb_i < a_{ij} \cdot x_j < ub_i$, therefore $\beta_i = 0$

(3) $x_j = u_j^{'}$

Here, the column is actually basic as $l_j < x_j < u_j$. The row on the other hand is non-basic at its upperbound, therefore, $\beta_i = \frac{(\gamma_j - \delta_j)}{a_{ij}}$ Now we set $\gamma_j = \delta_j = 0$ and the row has now been postsolved.

(B) $u_{\text{new}} >= u_j$

In this case $l_j^{'} = l_j$ and $u_j^{'} = u_j$

For all states of the postsolved value of x_j , the row i is basic and thus we can straight away set $\beta_i = 0$

Row i has now been postsolved

(3) $lb_i > -\infty \;\; , ub_i = \infty$

Here, $g_i < 0 \Rightarrow \beta_i = 0$

We can add $y_i = a_{ij} \cdot x_j$ as a Linear Dependency

Here it doesn't matter whether lb_i is positive, zero or negative but instead we will need to consider a_{ij} to determine the actual bounds given by the row.

(I) $a_{ij} < 0$

In this case, $l_{\text{new}} = -\infty$ and $u_{\text{new}} = \dfrac{lb_i}{a_{ij}}$ and $l_j^{'} = l_j$

(A) $u_{\text{new}} < u_j$

In this case $u_j^{'} = u_{\text{new}}$

Supose the postsolved value of variable x_j, we have to consider the different states of x_j.

(1) $l_j^{'} < x_j < u_j^{'}$

The row is inactive and $f_i, g_i < 0 \Rightarrow \alpha_i = 0$. The column variables and dual values will remain the same as before.

(2) $l_j^{'} = x_j$

Here, the column is nonbasic at its lowerbound, and the row is basic as $lb_i < a_{ij} \cdot x_j < ub_i$, therefore $\alpha_i = \beta_i = 0$

(3) $x_j = u_j^{'}$

Here, the column is actually basic as $l_j < x_j < u_j$. The row on the other hand is non-basic at its lowerbound, therefore, $\alpha_i = \frac{(\gamma_j - \delta_j)}{a_{ij}}$ Now we set $\gamma_j = \delta_j = 0$ and the row has now been postsolved.

(B) $u_{\text{new}} >= l_j$

In this case $l_j^{'} = l_j$ and $u_j^{'} = u_j$

For all states of the postsolved value of x_j , the row i is basic and thus we can straight away set $\alpha_i = \beta_i = 0$

(II) $a_{ij} = 0$

We should never reach this state as we are initially creating a sparse matrix of non-zero values only.

(III) $a_{ij} > 0$

In this case, $l_{\text{new}} = \dfrac{lb_i}{a_{ij}}$ and $u_{\text{new}} = \infty$

(A) $l_{\text{new}} > l_j$

In this case $l_j^{'} = l_{\text{new}}$ and $u_j^{'} = u_j$

Supose the postsolved value of variable x_j, we have to consider the different states of x_j.

(1) $l_j^{'} < x_j < u_j^{'}$

The row is inactive and $f_i, g_i < 0 \Rightarrow \alpha_i = \beta_i = 0$. The column variables and dual values will remain the same as before.

(2) $l_j^{'} = x_j$

Here, the column is actually basic as $l_j < x_j < u_j$. The row on the other hand is non-basic at its lowerbound, therefore, $\alpha_i = \frac{(\gamma_j - \delta_j)}{a_{ij}}$ Now we set $\gamma_j = \delta_j = 0$ and the row has now been postsolved.

(3) $x_j = u_j^{'}$

Here, the column is nonbasic at its upperbound, and the row is basic as $lb_i < a_{ij} \cdot x_j < ub_i$, therefore $\alpha_i = 0$

(B) $l_{\text{new}} \leq l_j$

In this case $l_j^{'} = l_j$ and $u_j^{'} = u_j$

For all states of the postsolved value of x_j , the row i is basic and thus we can straight away set $\alpha_i = \beta_i = 0$

Row i has now been postsolved

(4) $lb_i > -\infty \;\; , ub_i < \infty$

(I) $lb_i != ub_i$

Here again the different internal bound status aren't of direct relevance except for those that imply infeasible constraint independent of the singleton row.

(A) $l = 0 \quad , u < 0$ (infeasible)

(B) $l > 0 \quad , u < 0$ (infeasible)

(C) $l > 0 \quad , u = 0$ (infeasible)

(D) All other cases,
$lb_i \leq a_{ij} \cdot x_j \leq ub_i$ and equivalently, $l_{\text{new}} \leq x_j \leq u_{\text{new}}$ and we can calculate the new bounds based on the state of a_{ij}. For all cases, $y_i = a_{ij} \cdot x_j$ can be added as a Linear Dependency.
The following assumes $a_{ij} > 0$, if instead $a_{ij} < 0$, then we can switch α_i and β_i.

(i) $l_{\text{new}} \leq l_j \quad , u_j \leq u_{\text{new}}$
In this case $l_j^{'} = l_j$ and $u_j^{'} = u_j$
For all states of the postsolved value of x_j , the row i is basic and thus we can straight away set $\alpha_i = \beta_i = 0$

(ii) $l_{\text{new}} \leq l_j \quad , u_j > u_{\text{new}}$
Here $l_j^{'} = l_j$ and $u_j^{'} = u_{\text{new}}$ Supose the postsolved value of variable x_j, we have to consider the different states of x_j.

(1) $l_j^{'} < x_j < u_j^{'}$
The row is inactive and $f_i, g_i < 0 \Rightarrow \alpha_i = 0$. The column variables and dual values will remain the same as before.

(2) $l_j^{'} = x_j$
Here, the column is nonbasic at its lowerbound, and the row is basic as $lb_i < a_{ij} \cdot x_j < ub_i$, therefore $\alpha_i = \beta_i = 0$

(3) $x_j = u_j^{'}$
Here, the column is actually basic as $l_j < x_j < u_j$. The row on the other hand is nonbasic at its upperbound, therefore, $\alpha_i = \frac{(\gamma_j - \delta_j)}{a_{ij}}$ Now we set $\gamma_j = \delta_j = 0$ and the row has now been postsolved.

(iii) $l_{\text{new}} > l_j \quad , u_j \leq u_{\text{new}}$
Here $l_j^{'} = l_{\text{new}}$ and $u_j^{'} = u_j$ Supose the postsolved value of variable x_j, we have to consider the different states of x_j.

(1) $l_j^{'} < x_j < u_j^{'}$
The row is inactive and $f_i, g_i < 0 \Rightarrow \alpha_i = \beta_i = 0$. The column variables and dual values will remain the same as before.

(2) $l_j^{'} = x_j$
Here, the column is actually basic as $l_j < x_j < u_j$. The row on the other hand is nonbasic at its lowerbound, therefore, $\alpha_i = \frac{(\gamma_j - \delta_j)}{a_{ij}}$ Now we set $\gamma_j = \delta_j = 0$ and the row has now been postsolved.

(3) $x_j = u_j^{'}$
Here, the column is nonbasic at its upperbound, and the row is basic as $lb_i < a_{ij} \cdot x_j < ub_i$, therefore $\alpha_i = 0$

(iv) $l_{\text{new}} > l_j \quad , u_j > u_{\text{new}}$
Here $l_j^{'} = l_{\text{new}}$ and $u_j^{'} = u_{\text{new}}$ Supose the postsolved value of variable x_j, we have to consider the different states of x_j.

(1) $l_j^{'} < x_j < u_j^{'}$
The row is inactive and $f_i, g_i < 0 \Rightarrow \alpha_i = \beta_i = 0$. The column variables and dual values will remain the same as before.

(2) $l_j^{'} = x_j$
Here, the column is actually basic as $l_j < x_j < u_j$. The row on the other hand is non-basic at its lowerbound, therefore, $\alpha_i = \frac{(\gamma_j - \delta_j)}{a_{ij}}$ Now we set $\gamma_j = \delta_j = 0$ and the row has now been postsolved.

(3) $x_j = u_j^{'}$
Here, the column is actually basic as $l_j < x_j < u_j$. The row on the other hand is non-basic at its upperbound, therefore, $\alpha_i = \frac{(\gamma_j - \delta_j)}{a_{ij}}$ Now we set $\gamma_j = \delta_j = 0$ and the row has now been postsolved.

Row i can now be removed

(E) $lb_i = ub_i = b$

Here, $f_i = g_i = 0$ and $y_i = b_i$. The column variable x_j is fixed to the value $\frac{b_i}{a_{ij}}$.

If x_j isn't finite we can immediately give the problem an *Unbounded* status and exit the entire presolve operation.

The column j is always basic and its dual variables can be set to 0.

The row dual variables are set as, $\alpha_i - \beta_i = \frac{c_j - \sum_{k!=i}(\alpha_k - \beta_k) \cdot a_{kj}}{a_{ij}}$

The above analysis is tedious but revealing. It can be put in a more succint form that envelops more than one case. The table below summarizes the analysis for the case $lb_i != ub_i$.

Each cell in the table is a tuple of (column status,row status,row dual type) and the possible states are - Basic (B), Non-basic on the lowerbound (L), Non-basic on the upperbound (U).

The two possible row dual types are 1 - $\alpha_i = \beta_i = 0$ and 2 - $\alpha_i - \beta_i = \frac{\gamma_j - \delta_j}{a_{ij}}$

This table holds when $a_{ij} > 0$ and for all possible states of the bounds of row and column.

TABLE 1. Singleton Row Presolve Update when $lb_i != ub_i$

	$l_j^{'} < x_j^* < u_j^{'}$	$l_j^{'} = x_j$	$x_j = u_j^{'}$
$l_{\text{new}} \leq l_j$ && $u_j \leq u_{\text{new}}$	(B,B,1)	(B,B,1)	(B,B,1)
$l_{\text{new}} \leq l_j$ && $u_j > u_{\text{new}}$	(B,B,1)	(L,B,1)	(B,U,2)
$l_{\text{new}} > l_j$ && $u_j \leq u_{\text{new}}$	(B,B,1)	(L,B,2)	(U,B,1)
$l_{\text{new}} > l_j$ && $u_j > u_{\text{new}}$	(B,B,1)	(L,B,1)	(B,U,2))

6. Forcing Constraint

Let $P_i = \{i : a_{ij} > 0\}$ and $M_i = \{i : a_{ij} < 0\}$. Let's compute the quantities:

$$g_i = \sum_{j \in P_i} a_{ij} l_j - \sum_{j \in M_i} a_{ij} u_j$$

$$h_i = \sum_{j \in M_i} a_{ij} l_j - \sum_{j \in P_i} a_{ij} u_j$$

Clearly, $g_i \leq \sum a_{ij} x_j \leq h_i$ for any solution x. For efficiency purposes we will only consider the case when both g_i and h_i are finite. Now if $h_i < lb_i$ or $g_i > ub_i$, we have an infeasible constraint.

Suppose $lb_i = ub_i = b_i$ then a forcing constraint is one where $g_i = b_i$ or $h_i = b_i$. Here, the value of x_j is fixed at its bounds according to the sign of a_{ij} and thus we can fix all variables that occur in the i^{th} constraint.

Removal of Forcing Constraints is highly advantageous as they remove all variables that are structurally degenerate.

But if $lb_i != ub_i$,then we can still find a redundancy under some conditions. If $lb_i \leq g_i$ then the lower bound is redundant, similarly for the upper bound. A basic requirement for a row to not be redundant is $lb_i > g_i$ and $ub_i < h_i$.

TABLE 2. Forcing Constraint Analysis when $lb_i = ub_i = b_i$

	$h_i > b_i$	$h_i = b_i$
$g_i < b_i$	Not redundant	$x_j = \begin{cases} u_j & , j \in P_i \\ l_j & , j \in M_i \end{cases}$
$g_i = b_i$	$x_j = \begin{cases} l_j & , j \in P_i \\ u_j & , j \in M_i \end{cases}$	All variables are fixed variables

TABLE 3. Forcing Constraint Analysis when $lb_i != ub_i$

	$h_i > ub_i$	$h_i \leq ub_i$
$g_i < lb_i$	Not redundant	Not redundant
$g_i \geq lb_i$	Not redundant	Free row

Postsolving a free row is dealt with later.

Postsolving a forcing constraint still needs to be dealt with precisely. The basic outline should be to make all the column variables in the row i an artificial fixed column variable.

When we enter the postsolve routine for forcing constraint we have the dual values of these artificial fixed columns. Since actually the column variable j is fixed at one of its bounds, we have either $\gamma_j = 0$ or $\delta_j = 0$ and this with $\gamma_j, \delta_j \geq 0$ gives us conditions that the dual values should satisfy. We should adjust the row dual value $(\alpha_i - \beta_i)$ until this is satisfied.

7. Normal Row

This happens when a row is neither of the previous 3 types The only redundancy here is when the row is "free" or $lb_i = -\infty$ and $ub_i = \infty$. In this case the row is removed. Both the dual variables are set to zero and y_i can be computed at the end after the other variables are fixed while postsolving.

8. Empty Column

This happens when $\exists j$s.t. $a_{ij} = 0 \ \ \forall i$. Here, $[p(x^\star)]_j = l[j]$ and $[q(x^\star)]_j = -u[j]$

Here, the variable x_j is fixed at either u_j or l_j depending on whether $c_j > 0$ or < 0 respectively. The duals need to be computed according to the dual constraint here, $c_j = \gamma_j - \delta_j$. Individually one of the two column duals is zero depending on which bound it is fixed at.

9. Singleton Column

This happens when $\exists i, j$s.t. $a_{ij}! = 0$ & $\&a_{kj} = 0 \forall k! = i$

Define $s_{ij} = \sum_{k!=j} a_{ik} \cdot x_k$

Here, $lb_i \leq y_i = s_{ij} + a_{ij} \cdot x_j \leq ub_i$.

In this section we are going to ignore whether $lb_i == ub_i$ as the equations and implications are the same regardless and we are in fact considering a more general case.

A free column singleton is when $l_j = -\infty$ and $u_j = \infty$. In this case, the only constraint on column variable x_j is the row i and since x_j can literally be any value, we only need to worry about making sure that the other variables and other constraints are satisfied and optimal. In the end, we can fix the value of x_j such that it minimizes the objective function.

Thus we can remove both the column j and row i. Later while post solving we can give x_j an appropriate value that would satisfy the original row constraint i while being optimal. This is possible because x_j has no bounds on it thus we irrespective of the values of the other variables we can fix x_j such that $s_{ij} + a_{ij}x_j$ is within the row bounds. This post solve fixing can be done according to the following table.

TABLE 4. Free Column Singleton Presolve Update when $l_j = -\infty$ and $u_j = \infty$

	$c_j > 0$	$c_j < 0$
$a_j > 0$	$x_j = \frac{lb_i - s_{ij}}{a_{ij}}$	$x_j = \frac{ub_i - s_{ij}}{a_{ij}}$
$a_j < 0$	$x_j = \frac{ub_i - s_{ij}}{a_{ij}}$	$x_j = \frac{lb_i - s_{ij}}{a_{ij}}$

The table above can actually be condensed further to when $a_{ij} \cdot c_j > 0$ or < 0. I have left it in a expanded form simply because I felt its easier to follow the logic. Depending on whether $c_j > 0$ or not, we need to fix x_j to either its lower bound or upper bound. And its lower bound/upper bound is also decided by the sign of a_{ij} . We can also see that the column duals are always going to be 0. The row duals can be fixed based on the dual constraint.

This is not the only case where column singletons are useful though. The original row constraint i effectively gives us the bounds $l_{\text{new}} \leq x_j \leq u_{\text{new}}$ defined as -

$$u_{\text{new}} = \begin{cases} \frac{ub_i - (\sum_{k \in P_{ij}} a_{ik} \cdot l_k) - (\sum_{k \in M_{ij}} a_{ik} \cdot u_k)}{a_{ij}}, & a_{ij} > 0 \\ \frac{lb_i - (\sum_{k \in P_{ij}} a_{ik} \cdot u_k) - (\sum_{k \in M_{ij}} a_{ik} \cdot l_k)}{a_{ij}}, & a_{ij} < 0 \end{cases}$$

$$l_{\text{new}} = \begin{cases} \frac{lb_i - (\sum_{k \in P_{ij}} a_{ik} \cdot u_k) - (\sum_{k \in M_{ij}} a_{ik} \cdot l_k)}{a_{ij}}, & a_{ij} > 0 \\ \frac{ub_i - (\sum_{k \in P_{ij}} a_{ik} \cdot l_k) - (\sum_{k \in M_{ij}} a_{ik} \cdot u_k)}{a_{ij}}, & a_{ij} < 0 \end{cases}$$

where, $P_{ij} = \{k : a_{ik} > 0, k! = j\}$ and $M_{ij} = \{k : a_{ik} < 0, k! = j\}$. We can also recognize the structure and say ,

$$s_{ij,\min} = (\sum_{k \in P_{ij}} a_{ik} \cdot l_k) + (\sum_{k \in M_{ij}} a_{ik} \cdot u_k)$$

$$s_{ij,\max} = (\sum_{k \in P_{ij}} a_{ik} \cdot u_k) + (\sum_{k \in M_{ij}} a_{ik} \cdot l_k)$$

Now suppose $l_j \leq l_{\text{new}} \leq u_{\text{new}} \leq u_j$ then the column constraint are still redundant information on top of the row constraint. Thus we can again treat the column as if it was free.

The important difference now is that we need to ensure that the value we fix for x_j has to be between l_{new} and u_{new} and this means we need to be careful about the value of s_{ij}.

That is suppose $s_{ij,\min} < lb_i - a_{ij}u_{\text{new}}$ and suppose the other variables x_k are fixed such that $s_{ij} = s_{ij,\min}$ then no matter what value of x_j we fix, we will always have $y_i = s_{ij} + a_{ij} \cdot x_j < lb_i - a_{ij}u_{\text{new}} + a_{ij} \cdot x_j < lb_i$ which is infeasible. Thus we cannot remove the row i as before.

The row should now satisfy the bounds lb_{new} , ub_{new} :

$$lb_{\text{new}} = \max\{s_{ij,\min}, lb_i - a_{ij} \cdot u_{new}\}$$

$$ub_{\text{new}} = \min\{s_{ij,\max}, ub_i - a_{ij} \cdot l_{new}\}$$

Now suppose, both $lb_{\text{new}} = s_{ij,\min}$ and $ub_{\text{new}} = s_{ij,\max}$, then the row bounds are redundant as they offer no additional information to the column bounds on all the other variables , thus in this case we can actually remove the row.

Note that if we had computed these new row bounds for the previous case, we would get that $lb_{\text{new}} = s_{ij,\min}$ and $ub_{\text{new}} = s_{ij,\max}$ as the column bounds are not finite and we removed the row.

Thus Free Column Singleton is effectively a subset of Implied Column Singleton.

We already know how to postsolve the dual variables when both the row and column is removed. Let us consider the case when the row i is not removed and it has the new bounds.

When we enter the postsolve procedure, we already know the primal and dual values of row i - $y_i^{'}$ or s_{ij} , α_i , $\beta_i^{'}$. We can again set the value of x_j according to the table above. The column duals are set as $(\gamma_j - \delta_j) = -a_{ij} \cdot (\alpha_i^{'} - \beta_i^{'})$ and the actual row duals are set as -

$$\alpha_i - \beta_i = \frac{c_j - (\gamma_j - \delta_j)}{a_{ij}}$$

Individually they are set based on whether y_i is at its lower bound or upper bound.

The above analysis is mostly diverging on the two cases -

(1) Row and Column both removed
(2) Only Column removed.

Both Free and Implied Column Singletons can be treated the same in an abstract sense. If a column singleton is neither of these two types, then it is not redundant and we leave it be. We can succinctly summarize the discussion in the following two tables.

The column dual variables in the second case need to be fixed based on whether x_j is basic or non-basic at one of its bounds.

TABLE 5. Column Singleton Presolve Update when both row and column removed

	Column Variables	Row Variables
$a_j \cdot c_j > 0$	$x_j = \frac{lb_i - s_{ij}}{a_{ij}}$, $\gamma_j = \delta_j = 0$	$y_i = lb_i$, $\alpha_i = \frac{c_j}{a_{ij}}$, $\beta_i = 0$
$a_j \cdot c_j < 0$	$x_j = \frac{ub_i - s_{ij}}{a_{ij}}$, $\gamma_j = \delta_j = 0$	$y_i = ub_i$, $\alpha_i = 0$, $\beta_i = \frac{-c_j}{a_{ij}}$

TABLE 6. Column Singleton Presolve Update when only column removed

	Column Variables	Row Variables
$a_j \cdot c_j > 0$	$x_j = \frac{lb_i - s_{ij}}{a_{ij}}$, $(\gamma_j - \delta_j) = -a_{ij} \cdot (\alpha_i^{'} - \beta_i^{'})$	$y_i = lb_i$, $\alpha_i = \frac{c_j - (\gamma_j - \delta_j)}{a_{ij}}$, $\beta_i = 0$
$a_j \cdot c_j < 0$	$x_j = \frac{ub_i - s_{ij}}{a_{ij}}$, $(\gamma_j - \delta_j) = -a_{ij} \cdot (\alpha_i^{'} - \beta_i^{'})$	$y_i = ub_i$, $\alpha_i = 0$, $\beta_i = \frac{-c_j + (\gamma_j - \delta_j)}{a_{ij}}$

10. Normal Column

This happens when a column is neither of the previous 2 types.

The only redundancy here is when the column is "fixed" or $l_j == u_j$. In this case the col is removed. While postsolving, the dual variables are calculated by the dual constraint -

$$\gamma_j - \delta_j = c_j - \sum(\alpha_i - \beta_i) \cdot a_{ij}$$

and arbitrarily δ_j is chosen to be zero.

11. Future Work and Conclusions

The biggest chunk of the work that went on behind the scenes was understanding exactly which conditions mattered and which didn't and systematically eliminating and reducing the number of cases. The work somehow still feels fragile as very often it got tedious with the number of cases growing exponentially like a tree.

A lot of the complication is brought on by the well-meaning intentions at the start of having inequality constraint for the row rather than equality constraints only.

The theoretical analysis doesn't seem to be covered explicitly in literature. In either partial or some other form might be incorporated in code libraries like GLPK,Clp but that by no means makes it transparent (even though its open source!).

I think documentation wrt open source packages should have a very different meaning in the context of scientific computation without restricting to just API and function calls and architecture. There is a need for flowcharts and sequence diagrams that both identifies and specifies the state of a system.

This would also result in logically identified dead states and danger states hopefully making debugging easier.

The future work would now involve making some flowcharts for my own sanity to give an abstract view of what the design choices and micro execution aspects wrt code and optimization at a language level.