Project Proposal

Andres Espinosa

March 31, 2025

1 Project Description

Python LP Solver: This project will focus on building an LP solver in `python` from scratch. The project will use `numpy` as the linear algebra package behind the solver, but everything else will be implemented from scratch.

2 Methodology

For me to complete this project, I think this is the best way to approach the problem.

1. First, create a `simplex_phase_2` method which will take a feasible initial $\mathbf{x}_0$ as well as $\mathbf{A}, \mathbf{b}, \mathbf{c}$ and return the optimal solution to the problem x^*. Some different variations on the entering variable algorithm are listed below. I will also investigate how these algorithms perform on different problem sizes.

    - Steepest descent - picking the value with the greatest negative value to enter the basis.
    - Bland's Rule - picking the variable with the first negative value as the entering variable.
    - Secretary's rule - I want to try using the Secretary's rule, where you do the first $\frac{1}{e}$ proportion of variables, and pick the one with first value greater than that. I expect this won't work super well but I read that it is supposed to be the most efficient way to find the optimal sequential choice.

2. Second, create a `simplex_phase_1` method which will take in any of the parameters $\mathbf{A}, \mathbf{b}, \mathbf{c}$ and return a feasible start (or an output stating that the problem is infeasible). I expect that this phase 1 simplex method will create the arbitrary variables $\mathbf{h}$ and then call the simplex phase 2 to solve it. It will identify infeasibility as a solution to the auxilary problem that is not $\mathbf{1}^\top \mathbf{h} = 0$.

3. Third, and likely the most complex part of this project, I will implement a `python` module called `lp_reductions.py` that will take any LP and turn it into standard form. In order to accomplish this, I plan to do the following:

    (a) Implement a class of Variable and Expression. `Variables` will track the variables of a problem. It will probably have some methods like `intermediate, non-negative, etc` that will be helpful for the below things. `Expression` will track the equalities and objective function of a problem. Each variable will be assumed to be a vector $\mathbb{R}^n$ and each expression be an affine matrix inequality or equality.

(b) Accept an arbitrary number of $\mathbf{A}_i\mathbf{x}_i = \mathbf{b}_i, \mathbf{x}_i \succeq 0$ equations.

Implement a `condense_standard_forms` function that will take the arbitrary number of affine matrix equalities and concatenate into one $\mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \succeq 0$ problem.

(c) Implement a function `lower_ineq_to_eq`. This function should take in the expression $\mathbf{A}\mathbf{x} \preceq \mathbf{b}$ and add slack variables to turn it into $\mathbf{A}_s\mathbf{x}_s = \mathbf{b}_s$

(d) Implement a function `greater_ineq_to_lower_ineq`. This function will turn $\mathbf{A}\mathbf{x} \succeq \mathbf{b}$ into $-\mathbf{A}\mathbf{x} \preceq -\mathbf{b}$

(e) Implement a function `convert_objective_to_standard_form`. This function will take a maximization problem $\max \mathbf{c}^\top\mathbf{x}$ and convert it into a minimization problem $\min -\mathbf{c}^\top\mathbf{x}$, which is the standard form for LP solvers.

(f) Implement a function `combine_multivar_equality` that will combine any linear combinations of matrix vector multiplications. $\mathbf{A}_0\mathbf{x}_0 + \mathbf{A}_1\mathbf{x}_1 + \ldots \mathbf{A}_n\mathbf{x}_n = \mathbf{A}_{tot}\mathbf{x}_{tot}$ where

$$\mathbf{A}_{tot} = \begin{bmatrix} \mathbf{A}_0 & \mathbf{A}_1 & \ldots & \mathbf{A}_n \end{bmatrix}, \mathbf{x}_{tot} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{bmatrix}$$

(g) Implement a function `bound_all_vars` which will accept a $\mathbf{A}\mathbf{x} = \mathbf{b}$ and if it is not already bounded by non-negativity, it will split it into x^+, x^- and make them non-negative.

4. Finally, put this altogether by testing it with some available toy and real-world LP problems with accessible data.

5. If I have time, it would be nice to make it so that variables can appear on any side of the equation and still work. Otherwise I think it is still sufficient to have to enter variables on the left side of the equation.

3 Example Logic

To test how this logic will work, I will set up a problem below and show the outline of the future algorithm I will use to solve it.

Consider a problem with variables $\mathbf{w} \in \mathbb{R}^{20}, \mathbf{y} \in \mathbb{R}^{10}, z \in \mathbb{R}$ and the parameters $\mathbf{A}_0 \in \mathbb{R}^{5\times 20}, \mathbf{A}_1 \in \mathbb{R}^{5\times 10}, \mathbf{b}_0 \in \mathbb{R}^5, \mathbf{b}_1 \in \mathbb{R}^5, \mathbf{c}_0 \in \mathbb{R}^{20}, \mathbf{c}_1 \in \mathbb{R}^{10}, c_2 \in \mathbb{R}$

$$\begin{aligned}
\text{maximize} \quad & \mathbf{c}_0^\top\mathbf{w} - \mathbf{c}_1^\top\mathbf{y} + c_2 z && (1)\\
\text{subject to} \quad & \mathbf{A}_0\mathbf{w} - \mathbf{A}_1\mathbf{y} = \mathbf{b}_0 && (2)\\
& \mathbf{A}_0\mathbf{w} \preceq \mathbf{b}_0 && (3)\\
& \mathbf{A}_1\mathbf{y} \succeq \mathbf{b}_1 && (4)\\
& z \leq 2 && (5)\\
& \mathbf{w} \succeq 0 && (6)
\end{aligned}$$

We can handle this line by line.

Objective function: call `convert_objective_to_standard_form`

$$\text{maximize} \quad \mathbf{c}_0^\top \mathbf{w} - \mathbf{c}_1^\top \mathbf{y} + c_2 z \quad \rightarrow \quad \text{minimize} \quad -\mathbf{c}_0^\top \mathbf{w} + \mathbf{c}_1^\top \mathbf{y} - c_2 z \tag{7}$$

Equality 1: call `combine_multivar_equality`

$$\mathbf{A}_0 \mathbf{w} - \mathbf{A}_1 \mathbf{y} = \mathbf{b}_0 + c_2 z \quad \rightarrow \qquad \mathbf{A}_{01} \mathbf{x}_{wy} = \mathbf{b}_0 \tag{8}$$

where $\mathbf{A}_{01}$ is a column concatenation of the parameters and $\mathbf{x}_{wy}$ is a row contenation of the variables.
Inequality 1: call `lower_ineq_to_eq`

$$\mathbf{A}_0 \mathbf{w} \preceq \mathbf{b}_0 \quad \rightarrow \quad \mathbf{A}_0 \mathbf{w} + \mathbf{s}_0 = \mathbf{b}_0, \quad \mathbf{s}_0 \succeq 0 \tag{9}$$

Inequality 2: call `greater_ineq_to_lower_ineq` and then `lower_ineq_to_eq`

$$\mathbf{A}_1 \mathbf{y} \succeq \mathbf{b}_1 \quad \rightarrow \quad -\mathbf{A}_1 \mathbf{y} \preceq -\mathbf{b}_1 \quad \rightarrow \quad -\mathbf{A}_1 \mathbf{y} + \mathbf{s}_1 = -\mathbf{b}_1, \quad \mathbf{s}_1 \succeq 0 \tag{10}$$

Inequality 3: call `lower_ineq_to_eq`

$$z \leq 2 \quad \rightarrow \quad z + s_2 = 2, \quad s_2 \succeq 0 \tag{11}$$

Final combined system:

$$\text{minimize} \quad -\mathbf{c}_0^\top \mathbf{w} + \mathbf{c}_1^\top \mathbf{y} - c_2 z \tag{12}$$

$$\text{subject to} \quad \begin{bmatrix} \mathbf{A}_0 & -\mathbf{A}_1 & 0 \\ \mathbf{A}_0 & 0 & 0 \\ 0 & -\mathbf{A}_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{y} \\ z \end{bmatrix} + \begin{bmatrix} 0 \\ \mathbf{s}_0 \\ \mathbf{s}_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_0 \\ -\mathbf{b}_1 \\ 2 \end{bmatrix}, \qquad \begin{bmatrix} \mathbf{s}_0 \\ \mathbf{s}_1 \\ s_2 \end{bmatrix} \succeq 0 \tag{13}$$

To fold the adding variables into the matrix equation, we can augment the matrix $\mathbf{A}$ and the variable vector $\mathbf{x}$ to include the slack variables. Here's how you can rewrite the final combined system:

$$\text{minimize} \begin{bmatrix} -\mathbf{c}_0^\top & \mathbf{c}_1^\top & -c_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{y} \\ z \\ \mathbf{s}_0 \\ \mathbf{s}_1 \\ s_2 \end{bmatrix} \tag{14}$$

$$\text{subject to} \quad \begin{bmatrix} \mathbf{A}_0 & -\mathbf{A}_1 & 0 & 0 & 0 & 0 \\ \mathbf{A}_0 & 0 & 0 & \mathbf{I} & 0 & 0 \\ 0 & -\mathbf{A}_1 & 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{y} \\ z \\ \mathbf{s}_0 \\ \mathbf{s}_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_0 \\ -\mathbf{b}_1 \\ 2 \end{bmatrix} \tag{15}$$

$$\begin{bmatrix} \mathbf{w} \\ \mathbf{s}_0 \\ \mathbf{s}_1 \\ s_2 \end{bmatrix} \succeq 0 \tag{16}$$

Here, the slack variables $\mathbf{s}_0, \mathbf{s}_1, s_2$ are now part of the augmented variable vector, and the identity matrices $\mathbf{I}$ are used to incorporate the slack variables into the constraints. This ensures that the system remains in the form $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{x}$ includes all original variables and slack variables. Finally, we call `bound_all_vars` to bound the variables that have not been bounded yet.

$$\text{minimize} \begin{bmatrix} -\mathbf{c}_0^\top & \mathbf{c}_1^\top & -\mathbf{c}_1^\top & -c_2 & c_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{y}^+ \\ \mathbf{y}^- \\ z^+ \\ z^- \\ \mathbf{s}_0 \\ \mathbf{s}_1 \\ s_2 \end{bmatrix} \tag{17}$$

$$\text{subject to} \quad \begin{bmatrix} \mathbf{A}_0 & -\mathbf{A}_1 & \mathbf{A}_1 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{A}_0 & 0 & 0 & 0 & 0 & \mathbf{I} & 0 & 0 \\ 0 & -\mathbf{A}_1 & \mathbf{A}_1 & 0 & 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{y}^+ \\ \mathbf{y}^- \\ z^+ \\ z^- \\ \mathbf{s}_0 \\ \mathbf{s}_1 \\ s_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_0 \\ -\mathbf{b}_1 \\ 2 \end{bmatrix} \tag{18}$$

$$\begin{bmatrix} \mathbf{w} \\ \mathbf{y}^+ \\ \mathbf{y}^- \\ z^+ \\ z^- \\ \mathbf{s}_0 \\ \mathbf{s}_1 \\ s_2 \end{bmatrix} \succeq 0 \tag{19}$$