Project Report

Saumitra Aditya and Michael Elliott

Problem 1

$$\mathbf{A}_\sigma \mathbf{x} = \mathbf{d}_\sigma, \qquad \sigma \in \mathcal{O}$$

$$\min_{\mathbf{x},(\mathbf{d}_\sigma)} \sum_{\sigma \in \mathcal{O}} w_\sigma \left| d_\sigma^{pr} * \mathbf{1} - \mathbf{d}_\sigma \right|$$

$$s.t. \mathbf{A}_\sigma \mathbf{x} = \mathbf{d}_\sigma, \qquad \sigma \in \mathcal{O}$$

$$\mathbf{d}_\sigma \leq U, \qquad \sigma \in \mathcal{O}$$

$$\mathbf{x} \geq 0$$

Reformulation into a Linear Program

Using the definition of L1 norm and substituting $\mathbf{d}_\sigma^{pr} = d_\sigma^{pr} * \mathbf{1}$, we get the below cost function.

$$\min_{\mathbf{x},(\mathbf{d}_\sigma)} \sum_{\sigma \in \mathcal{O}} \sum_i w_\sigma \left| \left(\mathbf{d}_\sigma^{pr} - \mathbf{d}_\sigma \right)_i \right|$$

The cost is made into a linear equation by using separate positive and negative substitutes for $\left(\mathbf{d}_\sigma^{pr} - \mathbf{d}_\sigma \right)$ to eliminate the absolute value function.

$$\left| \left(\mathbf{d}_\sigma^{pr} - \mathbf{d}_\sigma \right)_i \right| = (\mathbf{\Theta}_\sigma^+ + \mathbf{\Theta}_\sigma^-)_i$$

$$\left(\mathbf{d}_\sigma^{pr} - \mathbf{d}_\sigma \right)_i = (\mathbf{\Theta}_\sigma^+ - \mathbf{\Theta}_\sigma^-)_i$$

$$\mathbf{d}_\sigma^{pr} - \mathbf{\Theta}_\sigma^+ + \mathbf{\Theta}_\sigma^- = \mathbf{d}_\sigma$$

$$\sum_{\sigma \in \mathcal{O}} \sum_i w_\sigma (\mathbf{\Theta}_\sigma^+ + \mathbf{\Theta}_\sigma^-)_i$$

$$\text{s.t.} \quad \mathbf{A}_\sigma \mathbf{x} = \mathbf{d}_\sigma^{pr} - \mathbf{\Theta}_\sigma^+ + \mathbf{\Theta}_\sigma^- \qquad \forall \sigma \in \mathcal{O}$$

$$\mathbf{d}_\sigma^{pr} - \mathbf{\Theta}_\sigma^+ + \mathbf{\Theta}_\sigma^- \leq U \qquad \forall \sigma \in \mathcal{O}$$

$$\mathbf{x} \geq 0$$

To convert from general form to standard form, we need to change the inequality constraints into equality constraints by introducing slack variables.

$$\mathbf{d}_\sigma^{pr} - \mathbf{\Theta}_\sigma^+ + \mathbf{\Theta}_\sigma^- + \mathbf{S}_\sigma = U \qquad \forall \sigma \in \mathcal{O}$$

$$\mathbf{S}_\sigma \geq 0 \qquad \forall \sigma \in \mathcal{O}$$

The decision variables are

$$\mathbf{x}, \mathbf{\Theta}_\sigma^+, \mathbf{\Theta}_\sigma^-, \mathbf{S}_\sigma$$

Our A matrix:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{\sigma_1} & 1 & 0 & 0 & -1 & 0 & 0 & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & 0 & 0 & \ddots & 0 & \vdots & \ddots & \vdots \\ \mathbf{A}_{\sigma_{16}} & 0 & 0 & 1 & 0 & 0 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 & \ddots & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}_{524\times 948}$$

Our decision variable vector:

$$\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_{162} & \boldsymbol{\Theta}^{+}_{\sigma_1} & \cdots & \boldsymbol{\Theta}^{+}_{\sigma_{16}} & \boldsymbol{\Theta}^{-}_{\sigma_1} & \cdots & \boldsymbol{\Theta}^{-}_{\sigma_{16}} & \mathbf{S}_{\sigma_1} & \cdots & \mathbf{S}_{\sigma_{16}} \end{bmatrix}^{\mathrm{T}}_{1\times 948}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{d}^{pr}_{\sigma_1} \cdots \mathbf{d}^{pr}_{\sigma_{16}} & U - \mathbf{d}^{pr}_{\sigma_1} \cdots U - \mathbf{d}^{pr}_{\sigma_{16}} \end{bmatrix}^{\mathrm{T}}_{1\times 524}$$

$$\mathbf{c} = [\mathbf{0}_{1x162} \quad \boldsymbol{w}_{\sigma_1} \cdots \boldsymbol{w}_{\sigma_{16}} \quad \boldsymbol{w}_{\sigma_1} \cdots \boldsymbol{w}_{\sigma_{16}} \quad \mathbf{0}_{1x262}]^{\mathrm{T}}_{1\times 948}$$

Big-M formulation:

$$\mathbf{A}_{\mathrm{M}} = [\mathbf{A} \quad \mathbf{I}_{524\mathrm{x}524}]_{524\times 1472}$$

The cost vector becomes

$$\mathbf{c}_{\mathrm{M}} = [\mathbf{c} \quad \mathbf{L}]^{\mathrm{T}}_{1\times 1472}$$

where $\mathbf{L} = [M_1 \dots M_{524}]$ and M is some large number.

The decision variable matrix then becomes

$$\mathbf{x}_{\mathrm{M}} = [\mathbf{x} \quad \boldsymbol{\mu}]^{\mathrm{T}}_{1\times 1472}$$

where $\boldsymbol{\mu} = [\mu_1, \dots, \mu_{524}]$ are the Big M decision variables.

To calculate the initial basic feasible solution, we set all variables other than Big-M variables to zero and Big-M variable to the $\mathbf{b}$ vector.

Initial_bfs = $[\mathbf{0}_{1\times 948} \quad \mathbf{b}]_{1\times 1472}$

This completes the formulation for Problem 1.

Simplex Method Implementation

Our first step after compiling the raw data into A matrix and b, c vectors was to verify the feasibility of the problem. Since GAMS could not support the number of decision variables in the problem we used "scipy.opitimize" (https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html#scipy.optimize.linprog) package to validate feasibility. Using this package, the optimal cost for the problem was found to be

410.58, it is worthwhile to mention here that the package was not able to solve this problem using the 'simplex' method and we had to use 'interior-point' method to get the solution.

I followed the Full-Tableau algorithm described in the text-book for the implementation. After the implementation I tested it on the sample-examples in the text-book and verified the solutions. The algorithm worked well with all the problems and produced optimal result.

My next step was to handle the degeneracy in the initial basic feasible solution constructed using the Big-M method described earlier, to this end I decided to implement the Lexicographic pivoting rule as described in the text-book.

Before starting the iterations and after constructing the initial tableau I check if the A matrix is full rank and lexicographically positive.

In my initial implementation my numpy array was composed of float elements, owing to which I encountered severe floating-point precision issues, for example divide by zero exceptions. This situation was fixed by making use for 'numpy.longdouble' representation.

Using lexicographic pivoting rule I did not observe cycling but then some other behavior , with Large-value associated with Big-M variables set to 9.0, my initial value of the cost function was 162072 and I could see that as the iterations progressed the objective cost function would decrease in value, however after about 636 iterations the objective function value was 20467 and then the objective value becomes a large negative number. I suspect this has to do with numerical precision. I have provided a trace (problem1-trace.txt) file that records the objective function value and the iteration number in the folder for Solutions.

With lexicographic rule and using longdouble representation for floating point values, I observed that after row transformations pivot element was not exactly one but very close to it for eg. .99999999753862 and neither was reduced cost exactly zero but of the order of e^{-6}, for around 600 iterations but after that I started observing numbers with large magnitudes in the reduced cost for variable entering the basis followed by row transformation of Full tableau (please see attached trace file, behavior toward end.) This behavior is very strange because as per the code the pivot column should be 1 and reduced cost for entering basic variable should be zero after row transformation see code snippet on the next page, as per the code when zeroth row is added to a multiple of pivot row we should definitely get a zero value for reduced cost, this same formulation works for other problems in the textbook and problem-2.

Another challenge I faced was owing to the fact the reduced costs were not zero and the columns corresponding to basic variable were not identity columns I could not identify which variables were part of the basis and because of this was not able to construct a solution vector. I have provided a NumPy representation of the final full tableau (prob1FullTab.npy).

```
row_transformation_matrix = np.identity(FULL_TABLEAU.shape[0])

for i in range(0, FULL_TABLEAU.shape[0]):
    if i != pivot_row_index:
        multiplier = -pivot_column[i] / pivot_element
    else:
        multiplier = 1 / pivot_element
    row_transformation_matrix[i, pivot_row_index] = multiplier

# print(row_transformation_matrix)

# print (np.dot(row_transformation_matrix, FULL_TABLEAU))
OLD_TABLEAU = FULL_TABLEAU
FULL_TABLEAU = np.dot(row_transformation_matrix, FULL_TABLEAU)
FULL_TABLEAU = np.around(FULL_TABLEAU,decimals=6)
```

I also attempted to solve this problem using Bland's rule, this approach is very slow as the rate of decay of objective value is low with respect to iteration count, but in this approach fewer of the basic variables in the full tableau have non-zero reduced costs, 514 out of 524 basic variables have an identifiable zero reduced cost and a unit vector in the full tableau. With 3531 iterations I was able to achieve objective function value of 11647.

I have added the trace (problem1-trace-blands.txt) and full tableau(prob1FullTabBlands.npy) for this to the solution folder.

Problem 2

$$\min_{\mathbf{x}} |\mathbf{x}|$$

$$s.t. \quad \frac{1}{m}\left\|\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x}-\mathbf{A}^{\mathrm{T}}\mathbf{b}\right\|_{\infty} \leq \epsilon$$

$$\left|\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x}-\mathbf{A}^{\mathrm{T}}\mathbf{b}\right)_i\right| \leq m\epsilon$$

Reformulation into a Linear Program

Using the definition of infinity norm,

$$\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x}-\mathbf{A}^{\mathrm{T}}\mathbf{b}\right)_i \leq m\epsilon \qquad \forall i \in [0,\dots,m]$$

$$\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x}-\mathbf{A}^{\mathrm{T}}\mathbf{b}\right)_i \geq -m\epsilon \qquad \forall i \in [0,\dots,m]$$

To convert the problem into a linear program we separate x into negative parts Θ^{-} and positive parts Θ^{+}

$$|\mathbf{x}| = \mathbf{e}^{\mathrm{T}}\boldsymbol{\Theta}^{+} + \mathbf{e}^{\mathrm{T}}\boldsymbol{\Theta}^{-}$$

$$\mathbf{x} = \boldsymbol{\Theta}^{+} - \boldsymbol{\Theta}^{-}$$

The cost function then becomes

$$\min \sum_{i=1}^{n} (\Theta_i^{+} + \Theta_i^{-})$$

$$\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}(\boldsymbol{\Theta}^{+}-\boldsymbol{\Theta}^{-})-\mathbf{A}^{\mathrm{T}}\mathbf{b}\right)_i \leq m\epsilon \qquad \forall i \in [0,\dots,m]$$

$$\left(\mathbf{A}^{\mathrm{T}}\mathbf{A}(\boldsymbol{\Theta}^{+}-\boldsymbol{\Theta}^{-})-\mathbf{A}^{\mathrm{T}}\mathbf{b}\right)_i \geq -m\epsilon \qquad \forall i \in [0,\dots,m]$$

$$\boldsymbol{\Theta}^{+}, \boldsymbol{\Theta}^{-}, \mathbf{S}, \bar{\mathbf{S}} \geq 0$$

Converting the problem to standard form,

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}(\boldsymbol{\Theta}^{+}-\boldsymbol{\Theta}^{-}) + \mathbf{S} = m\epsilon + \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$-\mathbf{A}^{\mathrm{T}}\mathbf{A}(\boldsymbol{\Theta}^{+}-\boldsymbol{\Theta}^{-}) + \bar{\mathbf{S}} = m\epsilon - \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\boldsymbol{\Theta}^{+}, \boldsymbol{\Theta}^{-}, \mathbf{S}, \bar{\mathbf{S}} \geq 0$$

Our constraint equalities:

$$\begin{bmatrix} \mathbf{A}^{\mathrm{T}}\mathbf{A} & -\mathbf{A}^{\mathrm{T}}\mathbf{A} & \mathbf{I} & \mathbf{0} \\ -\mathbf{A}^{\mathrm{T}}\mathbf{A} & \mathbf{A}^{\mathrm{T}}\mathbf{A} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\Theta}^{+} \\ \boldsymbol{\Theta}^{-} \\ \mathbf{S} \\ \bar{\mathbf{S}} \end{bmatrix} = \begin{bmatrix} m\epsilon + \mathbf{A}^{\mathrm{T}}\mathbf{b} \\ m\epsilon - \mathbf{A}^{\mathrm{T}}\mathbf{b} \end{bmatrix}$$

We then add Big M decision variables:

$$\begin{bmatrix} \mathbf{A}^{\mathrm{T}}\mathbf{A} & -\mathbf{A}^{\mathrm{T}}\mathbf{A} & \mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\mathbf{A}^{\mathrm{T}}\mathbf{A} & \mathbf{A}^{\mathrm{T}}\mathbf{A} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{I} \end{bmatrix}_{2n\times 6n} \begin{bmatrix} \boldsymbol{\Theta}^{+} \\ \boldsymbol{\Theta}^{-} \\ \mathbf{S} \\ \bar{\mathbf{S}} \\ \boldsymbol{\mu} \end{bmatrix}_{6n\times 1} = \begin{bmatrix} m\epsilon + \mathbf{A}^{\mathrm{T}}\mathbf{b} \\ m\epsilon - \mathbf{A}^{\mathrm{T}}\mathbf{b} \end{bmatrix}_{2n\times 1}$$

where $\boldsymbol{\mu} = \mu_1 \dots \mu_{2n}$ are the Big M variables.

The $\mathbf{c}$ vector then becomes

$$\mathbf{c}_{\mathrm{M}} = [\mathbf{1}_{1\times 2\mathrm{n}} \quad \mathbf{0}_{1\times 2\mathrm{n}} \quad \mathbf{M}_{1\times 2\mathrm{n}}]^{\mathrm{T}}_{1\times 6\mathrm{n}}$$

Where $\mathbf{M}$ is of the form $(LargeValue \quad \dots \quad LargeValue)$.

To calculate the initial basic feasible solution, we set all decision variables other than Big-M variables to zero and Big-M variable to the requirement/right-hand side vector.

Initial_bfs = $\begin{bmatrix} \mathbf{0}_{1\times 4n} & \begin{bmatrix} m\epsilon + \mathbf{A}^{\mathrm{T}}\mathbf{b} \\ m\epsilon - \mathbf{A}^{\mathrm{T}}\mathbf{b} \end{bmatrix}^{T}_{1\times 2n} \end{bmatrix}_{1\times 6n}$

This completes the formulation for Problem 2.

Results using Implemented Simplex Method

Our simplex method implementation for Problem 2 is identical to Problem 1, except that we use Blands rule to avoid cycling.

value of objective function is 2.912907, number of iterations required to reach final solution is 254,Final solution in terms of original decision variables is [0, 0.998988, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1.853233, 0].

Results using GAMS

Using the general form and the CPLEX solver, GAMS solves the linear program in one iteration and achieves an optimal cost of 2.852315, where $x_2 = 0.9977$, $x_{35} = 1.8546$, and all others are zero. This matches closely with the results of our simplex implementation.

Model Sensitivity Analysis

The cost of the optimal solution changes nonlinearly when ϵ is perturbed. Increasing ϵ decrease the cost, where the cost becomes zero for $\epsilon > 3.5$, while decreasing ϵ increases the cost exponentially as ϵ approaches zero. This agrees with Theorem 7.1, which states "the optimal cost is a convex function of $\mathbf{b}$," since we know from our formulation that $\mathbf{b}$ is a function of ϵ. This convexity is visually apparent in Figure 1. Similarly, the number of iterations required to find the optimal solution decreases as ϵ increases, as shown in Figure 2. When $\epsilon > 3.5$, the simplex method requires zero iterations to complete. This means that it accepts the initial basic feasible solution, which is the trivial case where $\mathbf{x} = 0$.

The conclusions above are intuitive: ϵ serves as a threshold for acceptable error, so as that threshold is relaxed (i.e. ϵ is increased), the constraints become easier to satisfy linear program consequently becomes easier to solve with minimal cost.

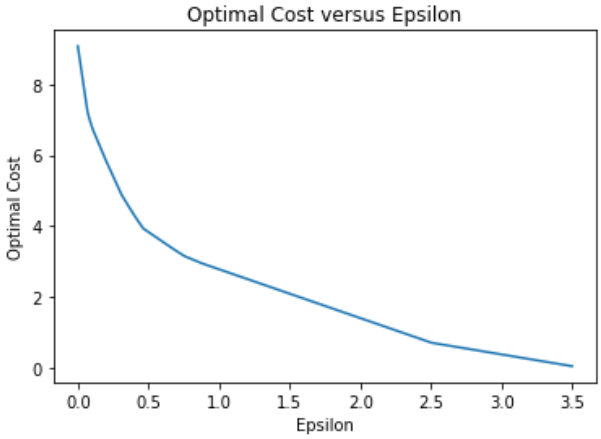

Figure 1. Sensitivity of optimal cost to changes in ϵ from 0.001 to 3.5 using the Simplex method.

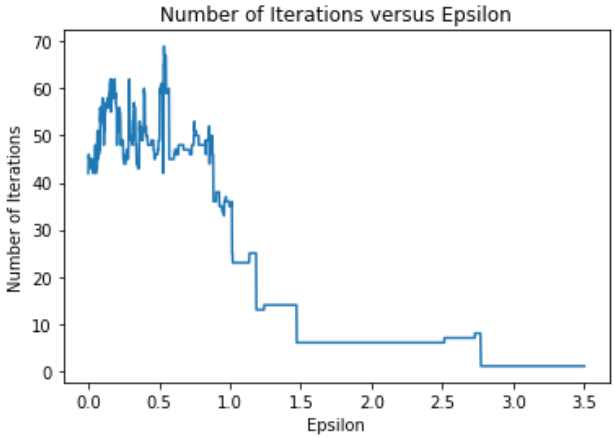

Figure 2. Sensitivity of the number of iterations to changes in ϵ from 0.001 to 3.5 using the Simplex method.

Description of source code files.

(Source files placed in respective folders for problems, folder also includes raw data)

(All python files can be executed using python3 <name_of_file>)

1. **data_loader2.py** will load the raw provided data and convert it into numpy arrays, will create the A, b, c matrices as well as initial basic feasible solution and save it as a numpy file. It also validates the formulation by calling the scipy library to check the feasibility and of the general form, standard form and BigM formulations and prints the objective value so that we can validate our inputs to Simplex method.
2. **simplex_problem2.py** will solve the problem using simplex method reading files generated by Data-loader2.py as input and will print a summary of the results on screen.
3. **data_loader1.py** reads input mat files and creates input numpy arrays in standard form formulation for the simplex implementation, I also perform validation here using the package.
4. **problem1-lexico.py** this is my attempt to solve the problem using lexicographic pivoting, this program prints the objective function value on screen along with iteration count.
5. **problem1-blands.py** simplex implementation using Blands rule.