Implementation of Simplex Algorithm for Optimization of Linear Programs

Shannon Whalen and Arijit Ghosh

April 3, 2013

Abstract

This report describes the design and implementation of the *Simplex* algorithm for optimizing Linear Programming problems. We demonstrate a solution by solving the *Diet Problem.*

1 Simplex Algorithm

Simplex is an efficient algorithm for solving Linear Programming problems. It is implemented here using a data structure of a 2 dimensional array of type double, A, to represent the coefficient matrix of the constraints, the constraints vector b and the objective function coefficients c, each of type double array, the basic variable vector B and the non-basic variable vector N both of type integer array, a double value for the current solution to the problem, and two integer variables to hold the entering and leaving variable indexes. The leaving index is representative of a variable $\in B$, the basic vector, while the entering index corresponds to a non-basic variable $\in N$. These variables are identified and swapped until an optimal solution is found, or the problem is discovered to be un-bounded. If all the coefficients in the objective function are negative, this indicates an optimal solution has been found as there is no variable that will increase the value of the objective function. If one of the constraints exhibits a negative value, an auxiliary program is constructed to determine if the problem is feasible with the given constraints. The condition of infeasibility occurs when a solution to this auxiliary problem does not equate to zero. A problem is considered un-bounded if it is found that all the coefficients of the possible entering variables in the constraints are less than or equal to 0, causing the minimum ratio to be undefined or negative.

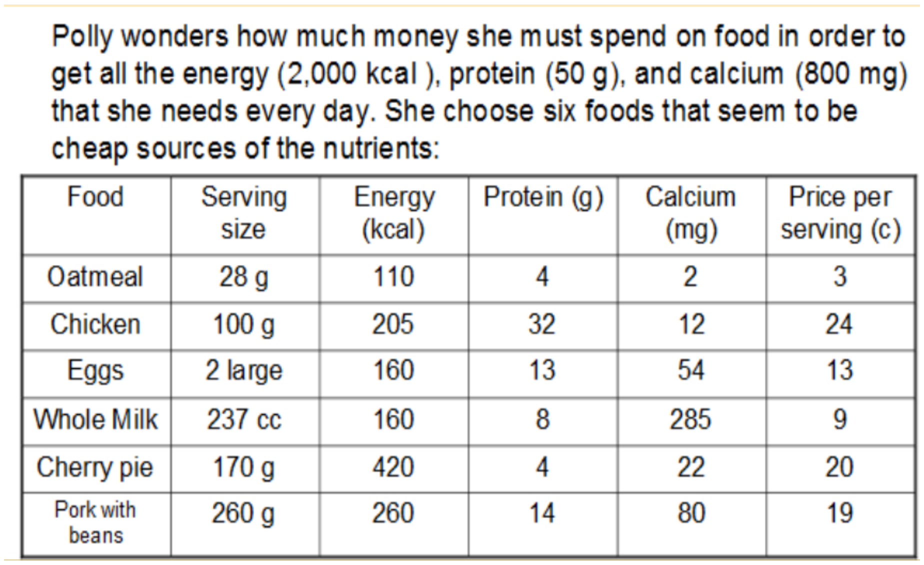
2 The Diet Problem

The simplex algorithm demonstrated here is implemented to solve maximization problems, trying to maximize the value of the linear program. However, sometimes a problem requires the minimum value solution to a linear programming problem. This occurs when the objective is to discern the best value, or minimum cost, of a problem. The *Diet Problem* is such an example. Given a set of ingredients, we are to find the smallest amount that will satisfy our needs. Figures 1 and 2 show a real-world diet problem and the linear program created to solve it. The problem is put into an input format that the algorithm knows how to solve, shown in figure 3. The input file is read in and the linear program is transposed to become a maximization problem. This problem is sent to the simplex algorithm and is solved for maximization. From this result, the diet problem can calculate its solutions using the final objective function from the solution of the maximization problem and the following equation:

$$y_i = \begin{cases} -c_{n+i}, & \text{if n+i} \in \text{N.} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

3 Implementation and Analysis

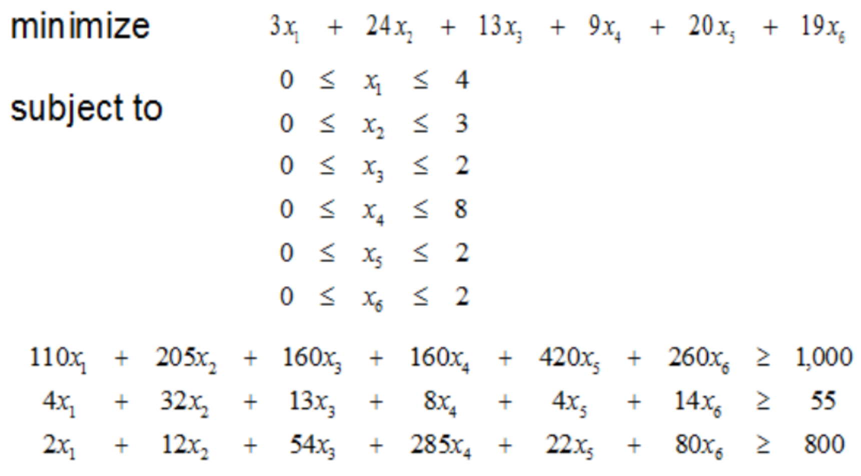
The simplex algorithm is broken up into three parts, the initialization, the main loop and the pivot. Each method in the algorithm is dependent on the others. Hence the resulting theoretical analysis of simplex complexity,which is $0(2^n)$, considers all the running times of its individual pieces.



Polly wonders how much money she must spend on food in order to get all the energy (2,000 kcal), protein (50 g), and calcium (800 mg) that she needs every day. She choose six foods that seem to be cheap sources of the nutrients:

Food	Serving size	Energy (kcal)	Protein (g)	Calcium (mg)	Price per serving (c)
Oatmeal	28 g	110	4	2	3
Chicken	100 g	205	32	12	24
Eggs	2 large	160	13	54	13
Whole Milk	237 cc	160	8	285	9
Cherry pie	170 g	420	4	22	20
Pork with beans	260 g	260	14	80	19

Figure 1: This is the real-world formulation of a diet problem



minimize $3x_1 + 24x_2 + 13x_3 + 9x_4 + 20x_5 + 19x_6$

subject to

$$\begin{aligned}
0 \le x_1 &\le 4\\
0 \le x_2 &\le 3\\
0 \le x_3 &\le 2\\
0 \le x_4 &\le 8\\
0 \le x_5 &\le 2\\
0 \le x_6 &\le 2
\end{aligned}$$

$$\begin{aligned}
110x_1 + 205x_2 + 160x_3 + 160x_4 + 420x_5 + 260x_6 &\ge 1{,}000\\
4x_1 + 32x_2 + 13x_3 + 8x_4 + 4x_5 + 14x_6 &\ge 55\\
2x_1 + 12x_2 + 54x_3 + 285x_4 + 22x_5 + 80x_6 &\ge 800
\end{aligned}$$

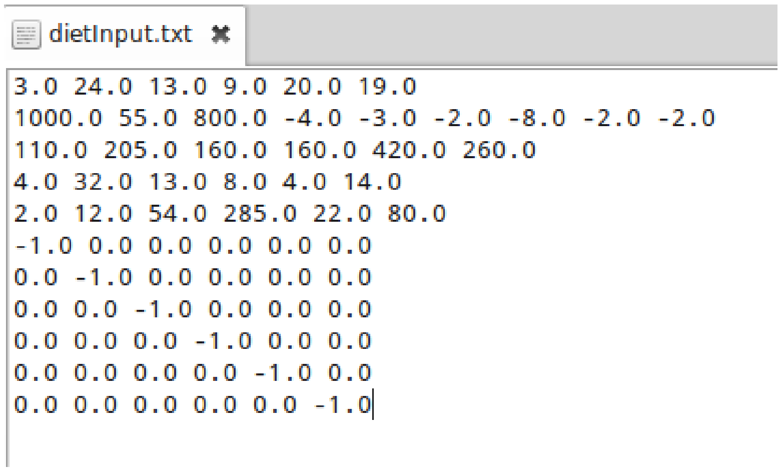
Figure 2: Linear Program formulated from the diet problem

3.1 initSimplex

The initSimplex method takes the coefficient matrix of the constraints A, the constraints vector b, the objective function coefficients c and two empty arrays for the basic and non basic variables of size m and n respectively. It initializes these arrays as follows:

- the first n variables of the problem are assigned to N, the non-basic vector. It is initialized with the subscripts of these variables, (1...n)
- the last m variables are assigned to the B, the basic vector. It is initialized with the subscripts from (n+1...n+m).

The cost for vector initialization is $\Theta(N)$. The method then finds the lowest constraint from the b vector. If this value is greater than zero, we are done. To find this value, the complexity is $\Theta(N)$ as it has to look at every value in b. If the minimum constraint value is found to be less than zero, an auxiliary linear program must be constructed by changing the objective function to $-x_0$ and adding this variable to all the constraints. This procedure involves altering the matrix by copying it to a new data structure, yielding a $\Theta(M * N)$ complexity. A pivot is performed and simplexLoop is enacted. The complexity of these procedures is $O(M * N)$. If the basic solution to the auxiliary program is found to be 0, the variable x_0 is removed from the matrix, taking $O(M * N)$ time as it is copied to a new matrix. The new constraint is then substituted back into the original objective function, which at worst performance



```
3.0 24.0 13.0 9.0 20.0 19.0
1000.0 55.0 800.0 -4.0 -3.0 -2.0 -8.0 -2.0 -2.0
110.0 205.0 160.0 160.0 420.0 260.0
4.0 32.0 13.0 8.0 4.0 14.0
2.0 12.0 54.0 285.0 22.0 80.0
-1.0 0.0 0.0 0.0 0.0 0.0
0.0 -1.0 0.0 0.0 0.0 0.0
0.0 0.0 -1.0 0.0 0.0 0.0
0.0 0.0 0.0 -1.0 0.0 0.0
0.0 0.0 0.0 0.0 -1.0 0.0
0.0 0.0 0.0 0.0 0.0 -1.0
```

Figure 3: Diet Problem input format for the Simplex Algorithm

is $O(M * N)$. If the optimal solution is not 0, the problem is infeasible. Resulting complexity for initSimplex is $O(M * N)$.

Algorithm 1 initSimplex

```
Find the minimum constraint $k \in b$
if $k \geq 0$ then
  return $(N, B, A, b, c, 0)$
end if
form auxiliary linear program
while optimal solution not found do
  search for optimal solution to $L_{aux}$
end while
if solution to $L_{aux} == 0$ then
  if $x_0 \in B$ then
    pivot and remove $x_0$
  end if
  return $(N, B, A, b, c, v)$
else
  return ”infeasible”
end if
```

3.2 simplexLoop

The method to loop in search of finding an optimal solution is simplexLoop. It takes as arguments the variable vectors B and N, the matrix A, the constraints b and the objective function c. It returns the optimally maximized solution to the linear program, if one exists. For each iteration, it searches the objective function for a coefficient that is greater than 0 to maximize the solution. Anything less than or equal to zero will not increase the value. If it is found, this variable becomes the *entering* pivot value. This step takes $O(N)$ time because it may look at every coefficient in the objective function to find such a value. Once the value is found, it computes all the minimum ratios for the b constraint vector, dividing the b-value by its coefficient value of the variable found in the previous operation. The minimum, positive result of this operation will identify the *leaving* pivot variable. If all of these ratios are negative, the solution is un-bounded. To find the leaving variable, the complexity is $O(N)$ because it examines every constraint value. A pivot is performed, which completes in $\Theta(M * N)$. This loop in the simplex algorithm will terminate in $\binom{m+n}{m}$ iterations, leading to an exponential theoretical complexity. Simplex accomplishes this by maintaining the N and B vectors so that they are individual. A variable cannot be both basic and non-basic, though it might have the same value as a variable in the opposite vector. The algorithm avoids cycles by always choosing the smallest index from B in the event of a tie. This is known as *Bland's rule.* The resulting complexity for simplexLoop is $O(2^n)$.

Algorithm 2 simplexLoop

```
while optimal solution not found do
  search objective function c for a positive coefficient leaving variable
  if none found then
    finished
  else
    calculate all the minimum ratio's to identify entering variable
    if no positive minimum ratio found then
      return "un-bounded"
    else
      pivot with leaving and entering variables
    end if
  end if
end while
return $(N, B, A, b, c, v)$
```

3.3 pivot

The pivot method takes a basic variable and makes it non-basic in search for an optimal solution to the linear programming problem. Its arguments are the variable vectors B and N, the matrix A, the constraints b and the objective function c, the current value of the linear program, v, and the index of the leaving and entering variable, l and e respectively. First the pivot works on the leaving variable, computing the new b-value and the coefficients in the A matrix. It then does the same for every other constraint, completing in $\Theta(M * N)$. Computing the new objective function and updating the N and B vectors all take $\Theta(M * N)$. The resulting complexity of the pivot method is $\Theta(M * N)$ because it must calculate the full matrix anew.

Algorithm 3 pivot

```
compute the new constraint value for the leaving variable
for all leaving variable $\in A$ do
  compute new coefficient
end for
for all remaining basic variables $\in A$ do
  compute the new constraint value
  compute new coefficients
end for
update optimal value for z
for all variables in the objective function do
  compute the objective function coefficients
end for
update B vector
update N vector
return $(N, B, A, b, c, v)$
```

4 Conclusion

Our project team comes from different backgrounds, one from Mathematics and one from Computer Science. We had some confusion about the best way to implement our algorithm, but came up with a strong solution. We both learned to think about problems differently and were more equipped to handle technical problems when they arose. We found the simplex algorithm complicated because there were many in-progress versions of the data structures during the execution and a lot of factors to keep straight when calculating new pivots. The actual analysis of the algorithm shows that it performs linearly on average, due to the variability of the number of loops. It rarely loops an exponential number of times to find an optimal solution as shown in figures 4 and 5.

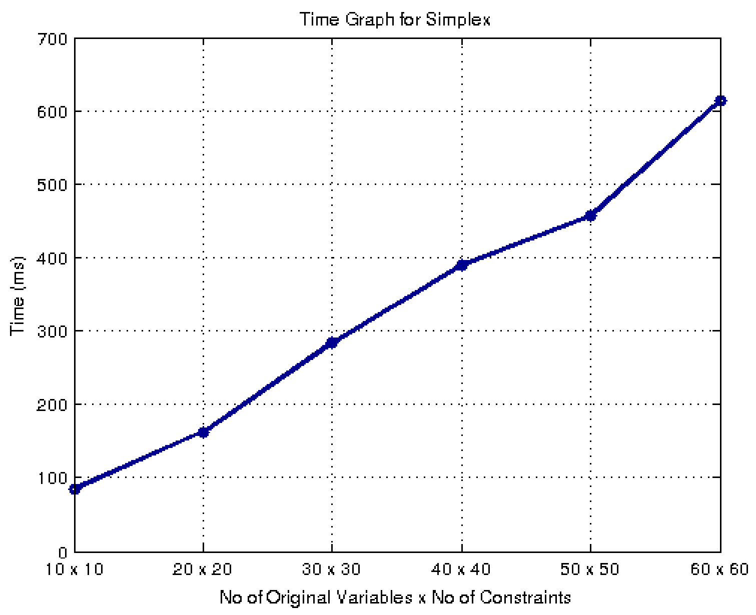


Figure 4: Execution time of Simplex is grows linear as the problem gets larger

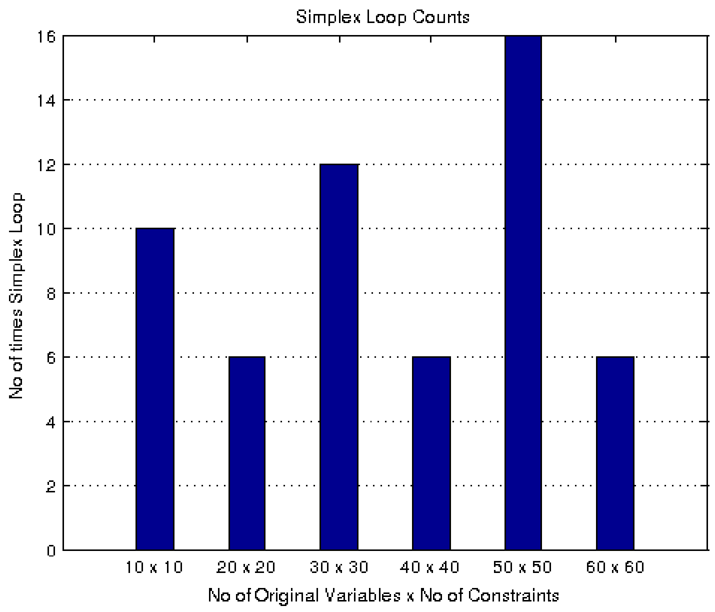


Figure 5: The number of loops to calculate the optimal solution doesn't appear to be related to the size of the problem.