Project Report

CS 5310

Assignment 5

Dillon Daudert

Step-by-step Walkthrough

Declare the linear program and call simplex()

```
A = [-1 1;
    1 3;
    1 -2]
b = [3 13 1]
c = [1 2]
simplex(A, b, c)
```

simplex(A, b, c): Line 1 - Call initsimplex(A, b, c) to turn the standard form LP into slack form

initsimplex(A, b, c): Lines 2-4 - Calculate the number of basic and nonbasic variables

initsimplex(A, b, c): Line 5 - Check if the lowest constraint in *b* is nonnegative. In this case, *b[k]* is 1, so we know that the initial basic solution to this system will be feasible.

initsimplex(A, b, c): Lines 7-8 - Create *N* and *B*, the sets of basic and nonbasic variable indices

```
N = [1, 2]
B = [3, 4, 5]
```

initsimplex(A, b, c): Line 10 - Expand the standard form matrices into equivalent slack form matrices. The resulting matrix *A'* will be of size (n+m, n+m) where n = |N|, m = |B|.

```
A' = [0 0 0 0 0;
      0 0 0 0 0;
      -1 1 0 0 0;
      1 3 0 0 0;
      1 -1 0 0 0]
b' = [0 0 3 13 1]
c' = [1 2 0 0 0]
```

initsimplex(A, b, c): Line 11 - Return the slack form LP, with *v* initialized to 0

simplex(A, b, c): Lines 2-5 - Check if this linear program is unbounded or infeasible. In this case, there is a feasible solution, so the if statement evaluates to false.

simplex(A, b, c): Line 6 - Call _innersimplex(), which finds an optimal solution to the slack for linear program

__innersimplex(N, B, A, b, c, v) called - *N*, *B* are the indices of the nonbasic and basic variables. *A*, *b*, *c* is the linear program in slack form. *v* is the initial objective function value

__innersimplex(N, B, A, b, c, v): Line 2 - Create a set of the indices for the nonzero coefficients in the objective function

```
N+ = [1, 2]
```

__innersimplex(. . .): Line 4 - Loop until there are no longer nonnegative coefficients in the objective function

__innersimplex(. . .): Line 5 - Create a matrix Delta to store how much each basic variable constraints the nonbasic variables in N+

```
Delta = [Inf Inf Inf Inf Inf]
```

__innersimplex(. . .): Lines 8-13 - Pick one nonbasic variable with a nonnegative coefficient, here $en = 2$. Calculate how much each basic variable constrains en.

```
Delta = [Inf Inf 3.0 4.33 Inf]
```

__innersimplex(. . .): Lines 15-18 - Check if the solution is unbounded. Here, the minimum is 3.0, so $lv = 3$

__innersimplex(. . .): Line 20 - Call pivot on $en = 2$, $lv = 3$

pivot(N, B, A, b, c, v, en, lv) is called - en, lv are the entering and leaving variables, respectively. The rest of the arguments are the linear program

pivot(. . .): Lines 1-3 - Create matrices that will hold the new slack form

pivot(. . .): Lines 5-9 - Compute the coefficients of the equation for the new basic variable, x_en

pivot(. . .): Lines 12-18 - Compute the coefficients for the remaining constraints

pivot(. . .): Lines 21-26 - Compute the objective function with respect to the new nonbasic variables

pivot(. . .): Lines 29-30 - Compute the new sets of basic and nonbasic variables

pivot(. . .): Line 31 - Return the new slack form linear program and objective function value v

Return from Pivot - The new slack form following the pivot is:

```
N = [1, 3]
B = [2, 4, 5]
A = [0 0 0 0 0;
    -1 0 1 0 0;
    0 0 0 0 0;
    4 0 -3 0 0;
    0 0 1 0 0]
b = [0 3 0 4 4]
c = [3 0 -2 0 0]
v = 6
```

__innersimplex(...): Line 22 - Update the set of nonnegative coefficients in the objective function. Since N+ isn't empty, we loop

```
N+ = [1]
```

__innersimplex(...): Line 5-13 - Now with *en = 1*, calculate how much each basic variable constrains *en*

```
Delta = [Inf Inf Inf 1.0 Inf]
```

__innersimplex(...): Line 15-18 - Set *lv = 4*. Since *Delta[lv]* isn't Inf, solution isn't unbounded.

__innersimplex(...): Line 20 - Call pivot on the slack form, with *en = 1*, *lv = 4*

pivot(N, B, A, b, c, v, en, lv) is called - en, lv are the entering and leaving variables, respectively. The rest of the arguments are the linear program

pivot works as described before

Return from Pivot - The new slack form following the pivot is:

```
N = [3, 4]
B = [1, 2, 5]
A = [0 0 -0.75 0.25 0;
    0 0 0.25 0.25 0;
    0 0 0 0 0;
    0 0 0 0 0;
    0 0 1 0 0]
b = [1 4 0 0 4]
c = [0 0 0.25 -0.75 0]
v = 9
```

__innersimplex(...): Line 22 - Update the set of nonnegative coefficients in the objective function. Since N+ isn't empty, we loop

```
N+ = [3]
```

__innersimplex(...): Line 5-13 - Now with *en = 3*, calculate how much each basic variable constrains *en*

```
Delta = [Inf 16. Inf Inf 4.]
```

__innersimplex(. . .): Line 15-18 - Set *lv = 5*. Since *Delta[lv]* isn't Inf, solution isn't unbounded.

__innersimplex(. . .): Line 20 - Call pivot on the slack form, with *en = 3*, *lv = 5*

pivot(N, B, A, b, c, v, en, lv) is called - en, lv are the entering and leaving variables, respectively. The rest of the arguments are the linear program

pivot works as described before

Return from Pivot - The new slack form following the pivot is:

```
N = [4, 5]
B = [1, 2, 3]
A = [0 0 0 0.24 0.75;
    0 0 0 0.25 -0.25;
    0 0 0 0 1;
    0 0 0 0 0;
    0 0 0 0 0]
b = [4 3 4 0 0]
c = [0 0 0 -0.75 -0.25]
v = 10
```

__innersimplex(. . .): Lines 25-30 - Set the values of the optimal solution to the original problem, return

Return the linear program, with solution

```
x = [4 3 4 0 0]
```

simplex(. . .): Return the linear program solution - Returns the linear program, the values of x, and the objective function value v.

The final values are *v = 10*, *x_1 = 4*, *x_2 = 3*.