LO Assignment Report

Group - 14

Nakul Periwal - 2020316
Parmesh Yadav - 2020319
Sahil Sahu - 2020327

Q1.

Our code is implementing the revised simplex method for solving LPPs. The user is prompted to enter the number of variables and constraints, the coefficients of the objective function, the coefficients of the constraints, and the right hand side value of the constraints.

After this the program creates a table representing the LPP and applies the revised simplex method to find the optimal solution and objective value.

The Revised simplex method works by iteratively pivoting the table to improve the objective value until a solution is found that satisfies all the constraints. It will iteratively pivot the table until an optimal solution is found, or until it is determined that the problem is unbounded.

The assumption has been taken that all the input values are valid and the input problem is also in standard form i.e all constraints are in the form Ax <= b.

If we need to solve a maximization problem, then we are multiplying all the coefficients of the cost function by -1 and then solving the minimization problem.

The unboundedness of the problem has also been covered.

Q2.

Firstly, we will formulate our problem statement as a linear program :-

Network:

[
  [0 1 0 0]
  [1 0 1 1]
  [0 1 0 1]
  [0 1 1 0]

]

There are a total of 4 edges in the network: e1, e2, e3, e4

Let xi = 1 if edge ei is used, 0 otherwise

maximize z = x1 + x2 + x3 + x4 + 0s1 + 0s2 + 0s3 + 0s4

subject to:

x1 + s1 = 1
x1 + x2 + x4 + s2 = 1
x2 + x3 + s3 = 1
x3 + x4 + s4 = 1

x1, x2, x3, x4, s1, s2, s3, s4 >= 0

Since our revised simplex method works for standard form, and here the objective function is to maximize. We will multiply all the cost coefficients by -1 and then minimize our objective function. At last, we will multiply the objective value by -1.

In this way, we have solved the maximum matching problem.

Q3.

Firstly, we will formulate our problem statement as a linear program :-

Network :-

[
  [1 1 0 0]
  [0 1 1 0]
  [0 0 1 1]
  [0 1 0 1]
]

minimize z = x1 + x2 + x3 + x4

subject to:

x1 + s1 = 1
x1 + x2 + x3 + s2 = 1
x2 + x3 + s3 = 1
x3 + x4 + s4 = 1

x1, x2, x3, x4 >= 0
x1, x2, x3, x4 are integers

The above formulated problem is in standard form, so we can call the revised simplex method directly without altering the objective function.

In this way, we are solving the minimum size vertex cover for G.

Q4.

A.

Our main function consists of two methods -

1. Run_max_flow
2. Run_min_cut

The run_max_flow method is implementing the primal-dual algorithm to solve a maximum flow problems whereas the run_min_cut is using the same algorithm to solve a minimum cut problem.

The run_max_flow method is calling the ‘get_primal’ function which takes a network as input and returns a set of linear constraints and a cost function for the primal LP problem.

The run_min_cut method is similar to ‘run_max_flow’, except that it calls the ‘get_dual’ function to obtain the set of linear constraints and cost function ‘c’ for the dual problem.

The ‘get_primal’ function is creating the primal LP problem by constructing the constraints for the flow conservation and flow capacity, as well as the objective function. It is doing so by iterating through every edge in the adjacency matrix. It checks if the edges are connecting the source, sink, or two intermediate vertices. The constraints are created based on the vertex type and the edge and slack counters are incremented accordingly.

The ‘get_dual’ function is creating the dual LP problem by iterating through each edge in the adjacency matrix and constructing the appropriate constraints for the dual problem. Firstly, it creates the constraint corresponding to the edge capacity and then creates constraints corresponding to the flow conservation. Finally, the constrains for the slack variables and the objective function are created.

In contrast to the simplex methods that iterate through boundary points (vertices) of the feasible region, the Newton barrier method iterates through solutions in the interior of the feasible region and will typically find a close approximation of an optimal solution. Consequently, the number of barrier iterations required to complete the method on a problem is determined more so by the required proximity to the optimal solution than the number of decision variables in the problem.

Unlike the simplex method, therefore, the barrier often completes in a similar number of iterations regardless of the problem size.

B.

The LPP for flow network 2 to find out the maximum flow from S to T can be formulated in the following way :-

maximize z : xsa + xsb + xsc;

[These are all the capacity constraints]

s.t. c1: 0<=xsa<=11;
s.t. c2: 0<=xsb <= 15;
s.t. c3: 0<=xsc <=10;
s.t. c4: 0<=xae<=18;
s.t. c5: 0<=xaf <=4;
s.t. c6: 0<=xba <=3;
s.t. c7: 0<=xbb<=8;
s.t. c8: 0<=xbc<=5;
s.t. c9: 0<=xcd<=6;
s.t. c10: 0<=xcg <= 3;
s.t. c11: 0<=xch <=11;
s.t. c12: 0<=xdc <= 4;
s.t. c13: 0<=xdg <= 17;
s.t. c14: 0<=xdh <= 6;
s.t. c15: 0<=xed<=3;
s.t. c16: 0<=xee<= 16;
s.t. c17: 0<=xei <= 13;
s.t. c18: 0<=xfa<=12;
s.t. c19: 0<=xfd<=4;
s.t. c20: 0<=xft<=21;
s.t. c21: 0<=xgh<=4;
s.t. c22: 0<=xgi<=9;
s.t. c23: 0<=xgj<=4;
s.t. c24: 0<=xgt<=3;
s.t. c25: 0<=xhg<=4;
s.t. c26: 0<=xhj<=5;
s.t. c27: 0<=xht<=4;
s.t. c28: 0<=xij<=7;
s.t. c29: 0<=xit<=9;
s.t. c30: 0<=xjh<=2;
s.t. c31: 0<=xjt<=15;

[These are all the conservation constraintsts]

s.t. c32: xsa=xae+xaf;
s.t. c33: xbb+xsb = xba+xbb+xbc;
s.t. c34: xsc= xcd+xcg+xch;
s.t. c35: xcd+xed+xfd= xdh+xdg+xdc;
s.t. c36: xae+xee = xee+xed+xei;
s.t. c37: xaf= xfa+xfd+xft;
s.t. c38: xcg+xhg+xdg = xgh+xgi+xgj+xgt;
s.t. c39: xch+xdh+xgh+xjh = xhg+xhj+xht;
s.t. c40: xgj+xhj+xij = xjh+xjt;

C.

For obtaining the LPP for S-T min cut, we have taken the dual of the max-flow problem.

The LPP for flow network 2 to find out the S−T Min-Cut can be formulated in the following way :-

minimize w : 11*y1 + 15*y2 + 10*y3 + 18*y4 + 4*y5 + 3*y6 + 8*y7 + 5*y8 + 6*y9 + 3*y10 + 16*y11 + 13*y12 + 12*y18 + 4*y19 + 21*y20 + 4*y21 + 9*y22 + 4*y23 + 3*y24 + 4*y25 + 5*y26 + 4*y27 + 7*y28 + 9*y29 + 2*y30 + 15*y31;

s.t.

c1: y1 + y32 >= 1;
c2: y2 + y7 + y11 + y22 + y31 >= 1;
c3: y3 + y10 + y14 + y20 + y23 + y25 >= 1;
c4: y4 + y5 >= 1;
c5: y4 + y6 + y12 + y18 + y26 >= 1;
c6: y5 + y27 >= 1;
c7: y7 + y8 + y9 >= 1;
c8: y8 + y21 + y28 >= 1;
c9: y9 + y13 + y15 + y19 + y24 >= 1;
c10: y10 + y11 + y12 >= 1;
c11: y13 + y16 + y29 >= 1;
c12: y14 + y15 + y17 >= 1;
c13: y16 + y17 + y30 >= 1;
c14: y18 + y19 + y20 >= 1;
c15: y21 + y22 + y23 + y24 >= 1;
c16: y26 + y27 + y28 >= 1;
c17: y29 + y30 + y31 + y32 >= 1;

Q5.

We have added all the codes in AMPL to solve all LPs. You can find all the codes in the zipped folder.