Code

Shortest path finding function “Dijkstra.m”

```
function [pred,dist] = Dijkstra(s,A,t)

% DIJKSTRA finds shortest path from node s to all nodes in the network
% using Dijkstra Algorithm
%
%     INPUT PARAMETERS
%
%           s:          The starting node s
%           A:          Links, with direction from A(:,1) to A(:,2)
%           t:          Link travel time
%
%     OUTPUT PARAMETERS
%
%           pred:     pred(j) = Predecessor of node j
%           dist:     distance from each node to s


m = length(unique(A));      % m = number of nodes
dist = Inf*ones(m,1);
pred = zeros(m,1);

dist(s) = 0;

S = s;     % S = permanently labeled nodes
B = [1:s-1 s+1:m];     % B = temporary labeled nodes

idx = find(A(:,1) == s);     % find the links starts with node s
for j = 1:length(idx)
      node = A(idx(j),2);
      dist(node) = t(idx(j));
      pred(node) = s;
end

while length(S) ~= m
      [min_dist k] = min(dist(B));
      i = B(k);     % i = the shortest temporary node to s
      S = [S i];     % add i to permanently labeled nodes
      B = [B(1:k-1) B(k+1:length(B))];     % eliminate i from temporary labeled nodes
      idx = find(A(:,1) == i);
      for j = 1:length(idx)
            node = A(idx(j),2);
            if dist(node) > dist(i)+t(idx(j))     % update the dist and pred
                  dist(node) = dist(i)+t(idx(j));
                  pred(node) = i;
            end
      end
end
```

All-or-nothing assignment function “AllorNothing.m”

```
function x = AllorNothing(A,t,OD,o,d)

% ALLORNOTHING gets the link flows by performing all-or-nothing
% assignment
%
%    INPUT PARAMETERS
%
%         A:        Links, with direction from A(:,1) to A(:,2)
%         t:        Link travel time
%         OD:       O-D trip matrix
%         o:        Origin node index (row index of O-D trip matrix)
%         d:        Destination node index (column index of O-D trip matrix)
%
%    OUTPUT PARAMETERS
%
%         x:        Link flows by performing all-or-nothing assignment


n = size(A,1);     % n = number of links
x = zeros(n,1);

for i = 1:size(OD,1)

    s = o(i);     % s = origin (starting) node
    [pred,dist] = Dijkstra(s,A,t);

    for k = 1:size(OD,2)
        e = d(k);     % e = destination (ending) node
        while pred(e) ~= 0     % get path from s to e and assign O-D flow
            idx = find(A(:,1) == pred(e));
            for j = 1:length(idx)
                if A(idx(j),2) == e
                    x(idx(j)) = x(idx(j))+OD(i,k);
                end
            end
            e = pred(e);
        end
    end

end
```

User Equilibrium function “UserEqui.m”

```
function xue = UserEqui(A,t0,ca,OD,o,d,epsil)

% USEREQUI gets the link volumes under user equilibrium condition using
% Frank-Wolfe Algorithm
%
%     INPUT PARAMETERS
%
%          A:        Links, with direction from A(:,1) to A(:,2)
%          t0:       Link free-flow travel time
%          ca:       Link capacity
%          OD:       O-D trip matrix
%          o:        Origin node index (row index of O-D trip matrix)
%          d:        Destination node index (column index of O-D trip matrix)
%          epsil:    Stopping criteria
%
%     OUTPUT PARAMETERS
%
%          xue:      Link volumes under user equilibrium


% Initialization
n = size(A,1);     % n = number of links
x0 = zeros(n,1);
t = t0.*(1+0.15*(x0./ca).^4);     % link travel time


% Perform all-or-nothing assignment to get x1
x1 = AllorNothing(A,t,OD,o,d);


% Successive calculation to get the UE solution
ex1x2 = epsil;     % stopping criteria
delta_x1x2 = ex1x2+1;

while delta_x1x2 > ex1x2

    t1 = t0.*(1+0.15*(x1./ca).^4);     % link travel time

    % Perform all-or-nothing assignment to get y(n)
    y1 = AllorNothing(A,t1,OD,o,d);

    % Get the optimal value of alpha using bisection method
    a = 0;
    b = 1;
    eab = 1e-5;     % stopping criteria for bisection method
    delta = y1-x1;
    while (b-a)/2 > eab
        alpha = (a+b)/2;
        salpha = sum(delta.*t0.*(1+0.15*(((x1+alpha*delta)./ca).^4)));
        if salpha < 0
            a = alpha;
        else
            b = alpha;
        end
    end

    % Get x(n+1)
    x2 = x1 + alpha*(y1-x1);

    % Convergence test
    delta_x1x2 = sqrt(sum((x2-x1).^2))/sum(x1);

    % Successive x(n+1)
    x1 = x2;

end

xue = x1;
```

Lambda and Miu function “MultiplierLM.m”

```
function [lambda,miu] = MultiplierLM(AOuter,t0Outer,caOuter,...
    OD,o,d,epsil,totalueOuter,NumLink,y,CandNew,CandExp)

% MULTIPLIERLM gets lambda and miu for the outer network plan
%
%   INPUT PARAMETERS
%
%       AOuter:         Links, with direction from A(:,1) to A(:,2)
%       t0Outer:        Link free-flow travel time
%       caOuter:        Link capacity
%       OD:             O-D trip matrix
%       o:              Origin node index (row index of OD matrix)
%       d:              Destination node index (column index of OD matrix)
%       epsil:          Stopping criteria for UE calculation
%       totalueOuter:   Total system UE travel time for y
%       NumLink:        Number of links for the original network A
%       y:              Existing improvement plan
%       CandNew:        Information of new candidate links
%       CandExp:        Information of expanding candidate links
%
%   OUTPUT PARAMETERS
%
%       lambda:         Lambda as indicated in the report
%       miu:            Miu as indicated in the report


% Get the information of the candidate links
ANew = CandNew(:,1:2);
t0New = CandNew(:,3);
caNew(:,1) = CandNew(:,4);
caNew(:,2) = CandNew(:,6);

IndExp = CandExp(:,1);
caExp(:,1) = CandExp(:,5);
caExp(:,2) = CandExp(:,7);

NumNew = size(CandNew,1);
NumExp = size(CandExp,1);
NumCand = NumNew+NumExp;


% Initialization of lambda and miu
lambda = zeros(NumCand,2);
miu = zeros(NumCand,2);


% Calculation of lambda and miu
for a = 1:NumCand

    % Case of y(a,1)=0 and y(a,2)=0
    if y(a,:) == [0 0]
```

```
        if a <= NumNew      % Add new candidate link

            A10 = [AOuter;ANew(a,:)];
            t010 = [t0Outer;t0New(a)];
            ca10 = [caOuter;caNew(a,1)];

            A01= [AOuter;ANew(a,:)];
            t001 = [t0Outer;t0New(a)];
            ca01 = [caOuter;caNew(a,2)];

        else      % Change the capacity of expanding candidate link

            ind = a-NumNew;

            A10 = AOuter;
            t010 = t0Outer;
            ca10 = caOuter;
            ca10(IndExp(ind)) = ca10(IndExp(ind))+caExp(ind,1);

            A01 = AOuter;
            t001 = t0Outer;
            ca01 = caOuter;
            ca01(IndExp(ind)) = ca01(IndExp(ind))+caExp(ind,2);

        end

        % Get z00, z01, z10 values
        z00 = totalueOuter;

        x10 = UserEqui(A10,t010,ca10,OD,o,d,epsil);
        t10 = t010.*(1+0.15*(x10./ca10).^4);
        z10 = sum(t10.*x10);

        x01 = UserEqui(A01,t001,ca01,OD,o,d,epsil);
        t01 = t001.*(1+0.15*(x01./ca01).^4);
        z01 = sum(t01.*x01);

        % Calculate lambda and miu
        lambda(a,1) = z10-z00;
        lambda(a,2) = z01-z00;

        miu(a,1) = 0;
        miu(a,2) = 0;

    end


    % Case of y(a,1)=1 and y(a,2)=0
    if y(a,:) == [1 0]

        if a <= NumNew

            % Delete the candidate link
            LinkCount = 0;
            for i = 1:a
                if sum(y(i,:)) > 0
                    LinkCount = LinkCount+1;
                end
            end
            ind = NumLink+LinkCount;

            A00 = [AOuter(1:ind-1,:);AOuter(ind+1:end,:)];
            t000 = [t0Outer(1:ind-1);t0Outer(ind+1:end)];
            ca00 = [caOuter(1:ind-1);caOuter(ind+1:end)];

            % Change the capacity of the candidate link
            A01= AOuter;
            t001 = t0Outer;
            ca01 = caOuter;
            ca01(ind) = caNew(a,2);

        else     % Change the capacity of expanding candidate link

            ind = a-NumNew;

            A00 = AOuter;
            t000 = t0Outer;
            ca00 = caOuter;
            ca00(IndExp(ind)) = ca00(IndExp(ind))-caExp(ind,1);

            A01 = AOuter;
            t001 = t0Outer;
            ca01 = ca00;
            ca01(IndExp(ind)) = ca01(IndExp(ind))+caExp(ind,2);

        end

        % Get z00, z01, z10 values
        z10 = totalueOuter;

        x00 = UserEqui(A00,t000,ca00,OD,o,d,epsil);
        t00 = t000.*(1+0.15*(x00./ca00).^4);
        z00 = sum(t00.*x00);

        x01 = UserEqui(A01,t001,ca01,OD,o,d,epsil);
        t01 = t001.*(1+0.15*(x01./ca01).^4);
        z01 = sum(t01.*x01);

        % Calculate lambda and miu
        lambda(a,1) = 0;
        lambda(a,2) = z01-z00;

        miu(a,1) = z10-z00;
        miu(a,2) = 0;

    end
```

```
% Case of y(a,1)=0 and y(a,2)=1
if y(a,:) == [0 1]

    if a <= NumNew

        % Delete the candidate link
        LinkCount = 0;
        for i = 1:a
            if sum(y(i,:)) > 0
                LinkCount = LinkCount+1;
            end
        end
        ind = NumLink+LinkCount;

        A00 = [AOuter(1:ind-1,:);AOuter(ind+1:end,:)];
        t000 = [t0Outer(1:ind-1);t0Outer(ind+1:end)];
        ca00 = [caOuter(1:ind-1);caOuter(ind+1:end)];

        % Change the capacity of the candidate link
        A10= AOuter;
        t010 = t0Outer;
        ca10 = caOuter;
        ca10(ind) = caNew(a,1);

    else    % Change the capacity of expanding candidate link

        ind = a-NumNew;

        A00 = AOuter;
        t000 = t0Outer;
        ca00 = caOuter;
        ca00(IndExp(ind)) = ca00(IndExp(ind))-caExp(ind,2);

        A10 = AOuter;
        t010 = t0Outer;
        ca10 = ca00;
        ca10(IndExp(ind)) = ca10(IndExp(ind))+caExp(ind,1);

    end

    % Get z00, z01, z10 values
    z01 = totalueOuter;

    x00 = UserEqui(A00,t000,ca00,OD,o,d,epsil);
    t00 = t000.*(1+0.15*(x00./ca00).^4);
    z00 = sum(t00.*x00);

    x10 = UserEqui(A10,t010,ca10,OD,o,d,epsil);
    t10 = t010.*(1+0.15*(x10./ca10).^4);
    z10 = sum(t10.*x10);

    % Calculate lambda and miu
    lambda(a,1) = z10-z00;
```

```
            lambda(a,2) = 0;

            miu(a,1) = 0;
            miu(a,2) = z01-z00;

        end

    end
```

g and h function "DecisionGH.m"

```
function [changeSort,changeGSort,changeHSort] = ...
    DecisionGH(y,NumCand,M,Budget,lambda,miu)

% DECISIONGH gets all the possible g and h and corresponding change of
% total system UE travel time estimated using lambda and miu
%
%    INPUT PARAMETERS
%
%         y:                 Existing improvement plan
%         NumCand:           Number of candidate links
%         M:                 Construction cost for candidate links
%         Budget:            Total budget available
%         lambda:            Lambda as indicated in the report
%         miu:               Miu as indicated in the report
%
%    OUTPUT PARAMETERS
%
%         changeSort:        Change of total travel time and sorted
%         changeGSort:       Correpsonding decimal g as indicated in the report
%         changeHSort:       Correpsonding decimal h as indicated in the report


% Initialization of g and h
g = zeros(NumCand,2);
h = zeros(NumCand,2);

idxg = find(y == 0);
sizeg = length(idxg);
idxh = find(y == 1);
sizeh = length(idxh);

% Initialization of the outputs
change = [];
changeG = [];
changeH = [];

% Calculation of the outputs
for jg = 1:2^sizeg     % Loop g

    % Transfer decimal jg to binary format g
    ig = sizeg;
    DeciNumg = jg-1;
    while ig > 0
        g(idxg(ig)) = floor(DeciNumg/(2^(ig-1)));
        DeciNumg = DeciNumg - g(idxg(ig))*2^(ig-1);
        ig = ig-1;
    end

    for jh = 1:2^sizeh     % Loop h

        % Transfer decimal jh to binary format h
        ih = sizeh;
        DeciNumh = jh-1;
        while ih > 0
            h(idxh(ih)) = floor(DeciNumh/(2^(ih-1)));
            DeciNumh = DeciNumh - h(idxh(ih))*2^(ih-1);
            ih = ih-1;
        end

        % Total money spent on this (g,h)
        MoneyP = sum(sum(M.*y))+sum(sum(M.*g))-sum(sum(M.*h));
        % Check the constraint of y(a,1)+y(a,2)<=1
        validgh = zeros(NumCand,1);
        for iv = 1:NumCand
            validgh(iv) = y(iv,1)+g(iv,1)-h(iv,1)+...
                y(iv,2)+g(iv,2)-h(iv,2);
        end

        % Only calculate change for valid (g,h)
        if MoneyP <= Budget && max(validgh) <= 1

            % Change estimated using lambda and miu
            change = [change;sum(sum(lambda.*g))-sum(sum(miu.*h))];

            % Add (g,h) as valid
            changeG = [changeG;jg];
            changeH = [changeH;jh];

        end
    end
end

% Sort the outputs
[changeSort,idxSort] = sort(change);
changeGSort = changeG(idxSort);
changeHSort = changeH(idxSort);
```

Inner loop function “GetInner.m”

```
function [yInner,AInner,t0Inner,caInner,totalueInner,countInner,...
    changeMin] = GetInner(A,t0,ca,OD,o,d,epsil,y,totalueOuter,...
    changeSort,changeGSort,changeHSort,CandNew,CandExp)

% GETINNER finds the new improvement plan which can truly reduce the total
% system UE travel time
%
%    INPUT PARAMETERS
%
%        A:               Original network links
%        t0:              Original network link free-flow travel time
%        ca:              Original network link capacity
%        OD:              O-D trip matrix
%        o:               Origin node index (row index of OD matrix)
%        d:               Destination node index (column index of OD matrix)
%        epsil:           Stopping criteria for UE calculation
%        y:               Existing improvement plan
%        totalueOuter:    Total system UE travel time for y
%        changeSort:      Change of total travel time and sorted
%        changeGSort:     Decimal g as indicated in the report
%        changeHSort:     Decimal h as indicated in the report
%        CandNew:         Information of new candidate links
%        CandExp:         Information of expanding candidate links
%
%    OUTPUT PARAMETERS
%
%        yInner:          New improvement plan
%        AInner:          New network plan links
%        t0Inner:         New network plan link free-flow travel time
%        caInner:         New network plan link capacity
%        totalueInner:    Total system UE travel time for yInner
%        countInner:      Number of iterations in the inner loop
%        changeMin:       Min change of total travel time after inner loop


% Get the information of the candidate links
ANew = CandNew(:,1:2);
t0New = CandNew(:,3);
caNew(:,1) = CandNew(:,4);
caNew(:,2) = CandNew(:,6);

IndExp = CandExp(:,1);
caExp(:,1) = CandExp(:,5);
caExp(:,2) = CandExp(:,7);

NumNew = size(CandNew,1);
NumExp = size(CandExp,1);
NumCand = NumNew+NumExp;


% Initialization of g and h
```

```
g = zeros(NumCand,2);
h = zeros(NumCand,2);

idxg = find(y == 0);
sizeg = length(idxg);
idxh = find(y == 1);
sizeh = length(idxh);


% Initialization of totalueInner and index
totalueInner = inf;
iS = 1;


% Get new improvement plan which can truly reduce total travel time
while totalueInner > totalueOuter

    % Transfer decimal changeGSort(iS) to binary format g
    ig = sizeg;
    DeciNumg = changeGSort(iS)-1;
    while ig > 0
        g(idxg(ig)) = floor(DeciNumg/(2^(ig-1)));
        DeciNumg = DeciNumg - g(idxg(ig))*2^(ig-1);
        ig = ig-1;
    end

    % Transfer decimal changeHSort(iS) to binary format h
    ih = sizeh;
    DeciNumh = changeHSort(iS)-1;
    while ih > 0
        h(idxh(ih)) = floor(DeciNumh/(2^(ih-1)));
        DeciNumh = DeciNumh - h(idxh(ih))*2^(ih-1);
        ih = ih-1;
    end

    % New improvement plan
    yInner = y+g-h;

    % Get AInner,t0Inner,caInner for yInner
    AInner = A;
    t0Inner = t0;
    caInner = ca;
    for i = 1:NumNew
        if yInner(i,1)+yInner(i,2) > 0
            AInner = [AInner;ANew(i,:)];
            t0Inner = [t0Inner;t0New(i)];
            caInner = [caInner;
                caNew(i,1)*yInner(i,1)+caNew(i,2)*yInner(i,2)];
        end
    end
    for i=1:NumExp
        caInner(IndExp(i)) = ca(IndExp(i)) + ...
            caExp(i,1)*yInner(NumNew+i,1) + caExp(i,2)*yInner(NumNew+i,2);
    end
    % Calculate total system UE travel time for yInner
    xueInner = UserEqui(AInner,t0Inner,caInner,OD,o,d,epsil);
    tueInner = t0Inner.*(1+0.15*(xueInner./caInner).^4);
    totalueInner = sum(tueInner.*xueInner);

    % If the improvement plan cannot reduce total system travel time,
    % proceed to the next (g,h)
    iS = iS+1;

end

countInner = iS-1;
changeMin = changeSort(iS-1);
```

Running file “main.m”

```
close all; clear; clc

A = xlsread(‘NetworkData.xlsx','Link','B2:C77');
t0 = xlsread(‘NetworkData.xlsx','Link','D2:D77');
ca = xlsread(‘NetworkData.xlsx','Link','E2:E77');

OD = xlsread(‘NetworkData.xlsx','OD','C3:P16');
o = xlsread(‘NetworkData.xlsx','OD','B3:B16');
d = xlsread(‘NetworkData.xlsx','OD','C2:P2');

CandNew = xlsread(‘NetworkData.xlsx','Candidate','B2:H7');
CandExp = xlsread(‘NetworkData.xlsx','Candidate','A9:H10');

epsil = 1e-5;

NumLink = size(A,1);


%% Get the information of the candidate links
ANew = CandNew(:,1:2);
t0New = CandNew(:,3);
caNew(:,1) = CandNew(:,4);
caNew(:,2) = CandNew(:,6);
MNew(:,1) = CandNew(:,5);
MNew(:,2) = CandNew(:,7);

IndExp = CandExp(:,1);
caExp(:,1) = CandExp(:,5);
caExp(:,2) = CandExp(:,7);
MExp(:,1) = CandExp(:,6);
MExp(:,2) = CandExp(:,8);

NumNew = size(CandNew,1);
NumExp = size(CandExp,1);
NumCand = NumNew+NumExp;

M = [MNew;MExp];
Budget = 95;


%% Active Set Approach
y = zeros(NumCand,2);

AOuter = A;
t0Outer = t0;
caOuter = ca;

xueOuter = UserEqui(AOuter,t0Outer,caOuter,OD,o,d,epsil);
tueOuter = t0Outer.*(1+0.15*(xueOuter./caOuter).^4);
totalueOuter = sum(tueOuter.*xueOuter);
```

```
changeMin = -100;

OuterCount = 0;
InnerCount = 0;

tic
while true

    [lambda,miu] = MultiplierLM(AOuter,t0Outer,caOuter,...
        OD,o,d,epsil,totalueOuter,NumLink,y,CandNew,CandExp);

    OuterCount = OuterCount+1;

    [changeSort,changeGSort,changeHSort] = ...
        DecisionGH(y,NumCand,M,Budget,lambda,miu);

    changeMin = changeSort(1);

    if changeMin >= 0
        break;
    end

    [yInner,AInner,t0Inner,caInner,totalueInner,countInner,...
        changeMin] = GetInner(A,t0,ca,OD,o,d,epsil,y,totalueOuter,...
        changeSort,changeGSort,changeHSort,CandNew,CandExp);

    if changeMin >= 0
        break;
    end

    y = yInner;
    AOuter = AInner;
    t0Outer = t0Inner;
    caOuter = caInner;
    totalueOuter = totalueInner;

    InnerCount = InnerCount+countInner;

    Money = sum(sum(M.*y));

end

y
totalueOuter
Money = sum(sum(M.*y))
OuterCount
InnerCount
t=toc
```

Validation file “validation.m”

```
close all; clear; clc

A = xlsread(‘NetworkData.xlsx','Link','B2:C77');
t0 = xlsread(‘NetworkData.xlsx','Link','D2:D77');
ca = xlsread(‘NetworkData.xlsx','Link','E2:E77');

OD = xlsread(‘NetworkData.xlsx','OD','C3:P16');
o = xlsread(‘NetworkData.xlsx','OD','B3:B16');
d = xlsread(‘NetworkData.xlsx','OD','C2:P2');

CandNew = xlsread(‘NetworkData.xlsx','Candidate','B2:H7');
CandExp = xlsread(‘NetworkData.xlsx','Candidate','A9:H10');

epsil = 1e-5;


%% Get the information of the candidate links
ANew = CandNew(:,1:2);
t0New = CandNew(:,3);
caNew(:,1) = CandNew(:,4);
caNew(:,2) = CandNew(:,6);
MNew(:,1) = CandNew(:,5);
MNew(:,2) = CandNew(:,7);

IndExp = CandExp(:,1);
caExp(:,1) = CandExp(:,5);
caExp(:,2) = CandExp(:,7);
MExp(:,1) = CandExp(:,6);
MExp(:,2) = CandExp(:,8);

NumNew = size(CandNew,1);
NumExp = size(CandExp,1);

NumCand = NumNew+NumExp;

Budget = 95;


%% Validate by enumeration

% Enumerate all the 3^NumCand improvement plans
% First transfer plan number into ternary number
a = zeros(NumCand,3^NumCand);
for j = 1:3^NumCand
    i = NumCand;
    DeciNum = j-1;
    while i > 0
        a(i,j) = floor(DeciNum/(3^(i-1)));
        DeciNum = DeciNum - a(i,j)*3^(i-1);
        i = i-1;
    end
```

```
end

% Initialization of the outputs
totalueMin = inf;
ValidCount = 0;

tic

for count = 1:3^NumCand

    y = zeros(NumCand,2);

    % Get y from a
    for i = 1:NumCand
        if a(i,count) == 1
            y(i,:) = [1 0];
        end
        if a(i,count) == 2
            y(i,:) = [0 1];
        end
    end

    % Get AE,t0E,caE for the corresponding plan
    AE = A;
    t0E = t0;
    caE = ca;
    for i = 1:NumNew
        if a(i,count) > 0
            AE = [AE;ANew(i,:)];
            t0E = [t0E;t0New(i)];
            caE = [caE;caNew(i,1)*y(i,1)+caNew(i,2)*y(i,2)];
        end
    end
    for i=1:NumExp
        caE(IndExp(i)) = ca(IndExp(i)) + ...
            caExp(i,1)*y(NumNew+i,1) + caExp(i,2)*y(NumNew+i,2);
    end

    % Money spent on the corresponding plan
    Money = 0;
    for mn = 1:NumNew
        Money = Money + MNew(mn,1)*y(mn,1) + MNew(mn,2)*y(mn,2);
    end
    for me = 1:NumExp
        Money = Money + ...
            MExp(me,1)*y(NumNew+me,1) + MExp(me,2)*y(NumNew+me,2);
    end

    % Get UE total system travel time if budget satisfied
    if Money <= Budget
        ValidCount = ValidCount+1;
        xueE = UserEqui(AE,t0E,caE,OD,o,d,epsil);
        tueE = t0E.*(1+0.15*(xueE./caE).^4);
        totalueE = sum(tueE.*xueE);
        if totalueE < totalueMin
            yMin = y;
            totalueMin = totalueE;
            totalMoney = Money;
        end
    end

end

ValidCount
t = toc

yMin
totalueMin
totalMoney
```