Project 1:

Implementing Frank-Wolfe Algorithm to Solve for User Equilibrium

CIVL 6260 Transportation Network Analysis

Prof. Sean He

Abdelrahman Ismael

October 25, 2017

1. Introduction

This project aims to solve a transportation network for the user equilibrium solution by coding the network on MATLAB. In this project the code is tested using the well-known “Sioux Falls” network. The algorithm used to solve this problem is Frank-Wolfe algorithm. In this report, a brief description of the project elements, followed by the discussion of implementing the algorithm along with the analysis of results.

2. Sioux Falls Network

Sioux Falls network is a famous network that is used a lot in studies related to network systems. The network (as shown in figure 1) used is a simplified representation of the actual network of the city of Sioux Falls in the state of South Dakota. The network includes 24 nodes joined by 76 links, all these links are two-way links (allowing movement in both directions).

Figure 1. Sioux Falls Network

3. User Equilibrium

3.1. Concept

User equilibrium (UE) is a method of traffic assignment that is based on Wardrop's first principle which states "The journey times used on all routes are equal, and less than those which would be experienced by a single vehicle on any unused road", (Wardrop, 1952). In other words, user equilibrium assignment provides a solution in which the times of the roads between 2 nodes are equal provided that these roads are used, if they aren't used; then they will have a time bigger than that of the used roads. This leads to that no driver can unilaterally change his route and gain more benefit.

3.2. Assumptions

The User equilibrium assignment assumes that:

- Drivers are rational,
- Drivers have perfect perception of travel costs,
- Drivers are homogeneous,
- Link cost functions are positive and increasing,
- The cost function depends only on the link flows of that link only and not the other links.

3.3. Formulation

User equilibrium is formulated as a constrained convex optimization problem, in which the objective function is the sum of integrals of the links performance functions. However, "This objective function does not have any intuitive economic or behavioral interpretation" (Sheffi, 1985). Also, UE solution is unique with respect to link flows due to the last two mentioned assumptions; as the partial derivative of the cost function with respect to its link flow is always greater than zero, and the partial derivative of the cost function of a link with respect to other link flows is equal zero. This leads to a positive definite hessian matrix with positive diagonal elements and zero non-diagonal elements. So, along with a strictly convex objective function the solution is unique.

$$\text{Min. } z(x) = \sum_a \int_0^{x_a} t_a(\omega)\, d\omega$$

Subject to

$$x_a = \sum_r \sum_s \sum_k f_k^{rs}\, \delta_{a,k}^{rs} \quad \forall\, a$$

$$\sum_k f_k^{rs} = q_{rs} \quad \forall\, r, s$$

$$f_k^{rs} \geq 0 \;\; \forall\, k, r, s$$

Where

x_a: link flow for any link **a**

f_k^{rs}: flow on path **k** that connects origin **r** with destination **s**

$\delta_{a,k}^{rs}$: binary variable that indicates if link **a** is on the path **k** between origin **r** and destination **s**

q_{rs}: demand between origin **r** and destination **s**

4. Frank-Wolfe Algorithm

Frank-Wolfe (F-W) algorithm was proposed in 1956 by Marguerite Frank and Philip Wolfe. It is an "iterative first order optimization algorithm for constrained convex optimization" (Wikipedia) depends on linear approximation of the objective. It solves the problem by finding an extreme point and then using a linear combination to search along the line that joins the extreme point and the current point, this is done using a search method to obtain the optimal step size (e.g. bisection, golden section search methods). Then it repeats this process for multiple iterations until the convergence criterion is met.

5. Step Size

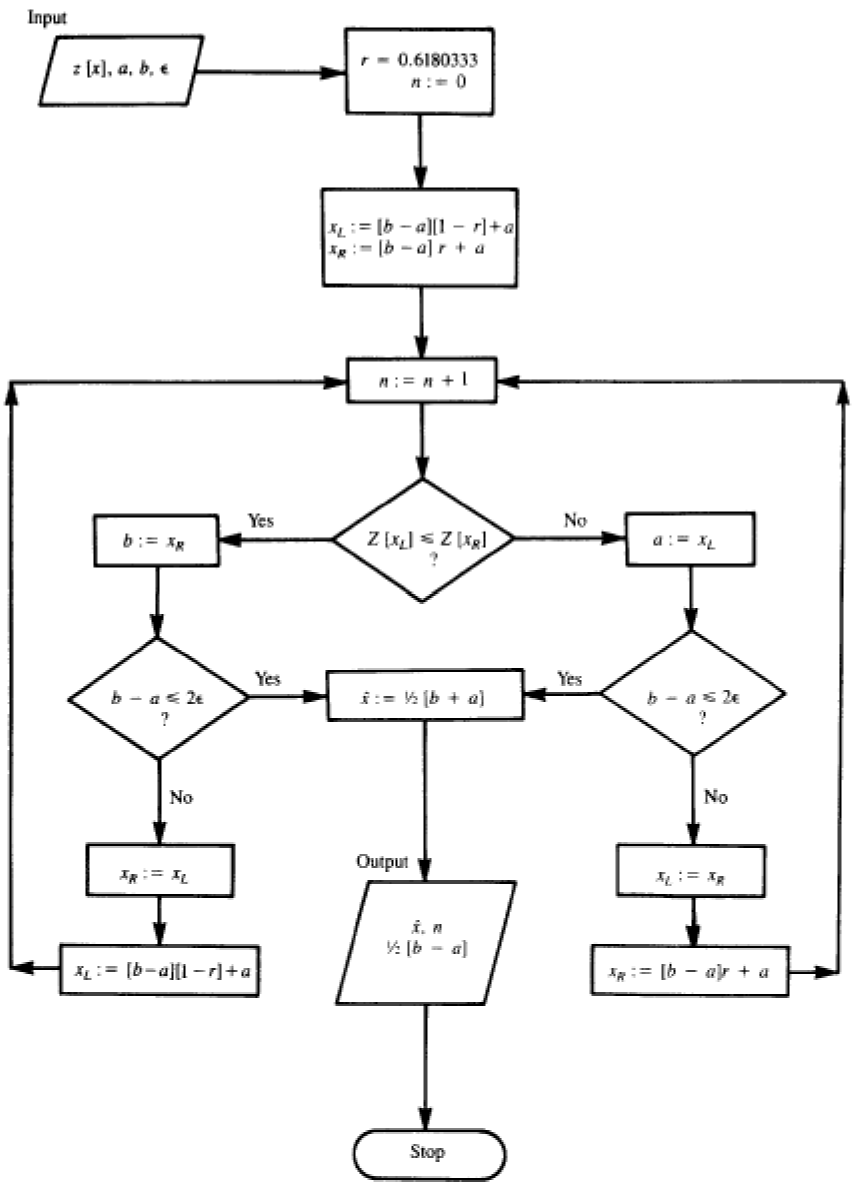
Frank-Wolfe algorithm uses a search method to determine the step size for each iteration, which can be obtained by solving a minimization problem

$$min_\alpha z[x^n + \alpha(y^n - x^n)]$$

$$subject\ to\ \ 0 \le \alpha \le 1$$

There are many famous search methods like bisection that uses three points including the boundary points and the middle point and evaluates the derivative of the objective function at the middle of the search range till it find the optimum step size, and golden section method (which is used in this project) that uses four points including boundary points and two intermediate points determined as a linear combination using a constant reduction ratio ($\frac{1}{2}(\sqrt{5}$-1)), it evaluates the objective function value at the two intermediate points and compare them to each other (as shown in the flowchart from Sheffi's book exhibited in figure 2).

Figure 2. Golden section algorithm



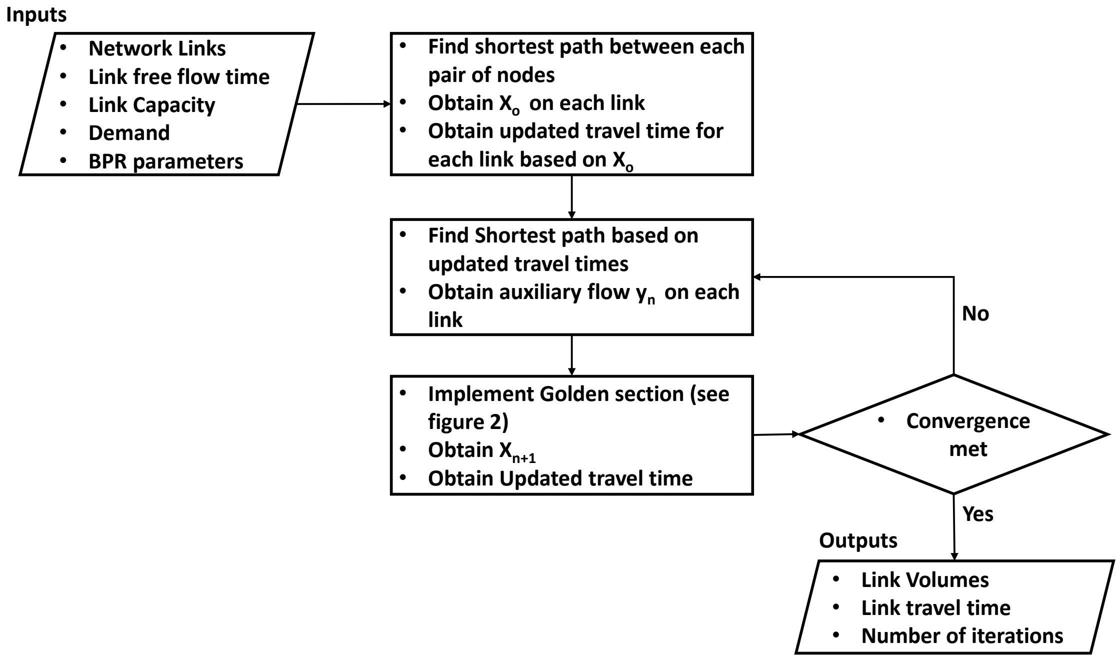
6. Code Design and Implementation

The problem was coded in MATLAB, using the input data files provided for the Sioux Falls network that included:

- Demand between each pair of nodes,
- Free flow time of each link,
- Capacity of each link,
- Parameters of the Bureau of Public Roads (BPR) formula ($\alpha = 0.15, \beta = 4.0$).

In figure 3, a flowchart is shown that describes how the code works. The code starts with initialization by implementing all or nothing and then follows by getting auxiliary flows using all or nothing on updated times (in this code it's coded by coding a function, here called "frank") after that golden section is applied (it is included in the code by coding a function, here called "gold") and then the process is repeated multiple times until convergence.

Figure 3. Code flowchart



7. Convergence

In this code, convergence was used in two different parts in the golden section and in the implementation of F-W algorithm.

For golden section search method, the convergence criteria used was the difference between the boundaries of the section being searched($b - a \leq \varepsilon$), where ε used is 5×10^{-5}.

For F-W algorithm, two convergence criteria were evaluated. The first (criteria-1) is the percentage of change in the value of objective function between each two consecutive iterations ($\frac{z(x_n) - z(x_{n+1})}{z(x_n)} \leq \varepsilon$), where ε used is 5×10^{-6}.

As for the second (criteria-2), the percentage of change of the volumes in link between two consecutive iterations was used ($\frac{|x_{a\,(n+1)} - x_{a\,n}|}{x_{a\,n}} \leq \varepsilon \;\; \forall\, a$), in other words the maximum percentage of change all volumes is less than or equal the threshold, where ε used is 5×10^{-6}.

The full code for both convergence criteria can be found in Appendix A.

8. Result Analysis

Multiple convergence thresholds have been tested, and a threshold that yields a close result to the best solution in less iterations. Appendix B shows the code results for the different convergence criteria used.

Table 1 shows number of iterations used for each convergence criteria used in F-W algorithm

Table 1. Convergence Threshold and Iterations

	Convergence Threshold	Iterations
Convergence Criteria 1	5.00E-06	134
Convergence Criteria 2	5.00E-06	85

Table 2 shows the comparison between the solutions obtained from each convergence criteria, and the best known solution of the network in terms of cost differences statistics

Table2. Differences Statistics with the best known solution

Convergence Criteria 1		Convergence Criteria 2	
Iterations	134	**Iterations**	85
Elapsed Time (sec.)	34.00	**Elapsed Time (sec.)**	20.46
Min. Absolute Difference	0.000073	**Min. Absolute Difference**	0.000062
Max. Absolute Difference	0.437	**Max. Absolute Difference**	0.386
Absolute Average	0.080	**Absolute Average**	0.054
Std. Devation	0.128	**Std. Devation**	0.090

From table 2, it's obvious that using convergence criteria-2; yield better results in less time.

References

- Sheffi, Y., Urban Transportation Networks, 1985.
- He, S., Transportation Network Analysis Lecture Notes, 2017.
- Wikipedia, Frank-Wolfe Algorithm, https://en.wikipedia.org/wiki/Frank%E2%80%93Wolfe_algorithm, accessed 23/10/2017.
- Indian Institute of Technology Bombay, Transportation Systems Engineering Lecture notes, https://www.civil.iitb.ac.in/tvm/1100_LnTse/206_lnTse/plain/plain.html, 2011.
- https://www.researchgate.net/figure/279246245_fig6_Fig-6-Mapping-of-transportation-and-power-model, accessed 23/10/2017.

Appendix A

Main Code- Convergence criteria-1

```
tic;
Net_read = readtable('Network.csv');
Network=table2array(Net_read);
Dem_read = readtable('Demand.csv');
Demand=table2array(Dem_read);
fft=[]';
fft=Network(:,5);
cap=[]';
cap=Network(:,3);
K= digraph (Network(:,1), Network(:,2), fft);

Dem_matrix =zeros (24,24);
for i = 1:24
    for j = 1 : 24
        for n= 1 : 576
            if Demand (n,1)== i && Demand (n,2)== j
                Dem_matrix (i,j) = Demand(n,3);
            end
        end
    end
end

volx=zeros(76,1);
for i = 1:24
    for j= 1:24
        [p]= shortestpath (K,i,j);
         u=length(p);
            for n= 1 :u-1
                for l=1:76
                    if Network(l,1)==p(n) && Network(l,2)==p(n+1)
                        volx(l)=volx(l)+Dem_matrix(i,j);
                    end
                end
            end
    end
end

[zold, voly]= frank(Dem_matrix, Network, volx,cap,fft);
[znew,volx]= gold(volx,voly,cap,fft);
iteration=2;
conv=abs(zold-znew)/zold;
while conv > 0.000005
   [zold, voly]= frank(Dem_matrix, Network, volx,cap,fft);
   [znew,volx]= gold(volx,voly,cap,fft);
    conv=abs(zold-znew)/zold;
    iteration=iteration+1;
end

for i = 1:76
    tt(i)=fft(i)*(1+0.15*(volx(i)/cap(i))^4);
end

display (iteration)
Solution=table;
Solution.Initial_Node= Network(:,1);
Solution.End_Node= Network(:,2);
Solution.Volume= volx;
Solution.Cost= tt';
writetable(Solution,'Solution.xlsx')
toc
```

Main Code- Convergence criteria-2

```
tic;
Net_read = readtable('Network.csv');
Network=table2array(Net_read);
Dem_read = readtable('Demand.csv');
Demand=table2array(Dem_read);
fft=[]';
fft=Network(:,5);
cap=[]';
cap=Network(:,3);
K= digraph (Network(:,1), Network(:,2), fft);

Dem_matrix =zeros (24,24);
for i = 1:24
    for j = 1 : 24
        for n= 1 : 576
            if Demand (n,1)== i && Demand (n,2)== j
                Dem_matrix (i,j) = Demand(n,3);
            end
        end
    end
end

volx=zeros(76,1);
for i = 1:24
    for j= 1:24
        [p]= shortestpath (K,i,j);
         u=length(p);
            for n= 1 :u-1
                for l=1:76
                    if Network(l,1)==p(n) && Network(l,2)==p(n+1)
                        volx(l)=volx(l)+Dem_matrix(i,j);
                    end
                end
            end
    end
end

[zol, zold, voly]= frank(Dem_matrix, Network, volx,cap,fft);
for i=1:76
    volo(i)=volx(i);
end
[zn, znew,volx]= gold(volx,voly,cap,fft);
iteration=2;

conv_crit= 0.000005*ones(76,1);
for i= 1:76
conv_2(i)=abs(volo(i)-volx(i))/volo(i);
end
while conv_2 > conv_crit
   [zol, zold, voly]= frank(Dem_matrix, Network, volx,cap,fft);
   for i=1:76
    volo(i)=volx(i);
end
   [zn, znew,volx]= gold(volx,voly,cap,fft);
    iteration=iteration+1;
    for i= 1:76
conv_2(i)=abs(volo(i)-volx(i))/volo(i);
end
end

for i = 1:76
    tt(i)=fft(i)*(1+0.15*(volx(i)/cap(i))^4);
```

```
end

display (iteration)
Solution=table;
Solution.Initial_Node= Network(:,1);
Solution.End_Node= Network(:,2);
Solution.Volume= volx;
Solution.Cost= tt';
writetable(Solution,'Solution.xlsx')
toc
```

Shortest Path and Auxiliary Flows Function

```
function [zol, zold, voly] = frank(Dem_matrix, Network, volx,cap,fft)

Time=zeros(76,1);
for l = 1:76
    for i = 1:24
        for j= 1:24
            if Network(l,1)==i && Network(l,2)==j
                Time(l)=fft(l)*(1+0.15*(volx(l)/cap(l))^4);
            end
        end
    end
end

T= digraph (Network(:,1), Network(:,2), Time);

voly = zeros(76,1);
for i = 1:24
    for j= 1:24
        [p]= shortestpath (T,i,j);
         u=length(p);
            for n= 1 :u-1
                for l=1:76
                    if Network(l,1)==p(n) && Network(l,2)==p(n+1)
                        voly(l)=voly(l)+Dem_matrix(i,j);
                    end
                end
            end
    end
end

zol=zeros(76,1);
for i=1:76
    zol(i)=fft(i)*volx(i)+0.03*(fft(i)/cap(i)^4)*(volx(i))^5;
end
zold=sum(zol);
end
```

Golden Section Function

```
function [zn,znew,volx] = gold(volx,voly,cap,fft)
r=0.5*(sqrt(5)-1);
 a=0;
 b=1;
 xl=(b-a)*(1-r)+a;
 xr=(b-a)*r+a;
 vl=zeros(76,1);
 vr=zeros(76,1);
```

```
 zvl=zeros(76,1);
 zvr=zeros(76,1);
 zn=zeros(76,1);

while (b-a) > 0.00005
     for i =1:76
         vl(i)=volx(i)+xl*(voly(i)-volx(i));
         vr(i)=volx(i)+xr*(voly(i)-volx(i));
         zvl(i)=fft(i)*vl(i)+0.03*(fft(i)/cap(i)^4)*(vl(i))^5;
         zvr(i)=fft(i)*vr(i)+0.03*(fft(i)/cap(i)^4)*(vr(i))^5;
         zn(i)=0.5*(zvl(i)+zvr(i));

     end
     Zl=sum(zvl);
     Zr=sum(zvr);
     if Zl <= Zr
         b=xr;
         xr=xl;
         xl=(b-a)*(1-r)+a;
     else
         a=xl;
         xl=xr;
         xr=(b-a)*r+a;
     end

end
alpha=(b+a)/2;
znew=(Zl+Zr)/2;

for i = 1:76
   volx(i)=(vl(i)+vr(i))/2;
   tt(i)=fft(i)*(1+0.15*(volx(i)/cap(i))^4);
 end
end
```

Appendix B

Results-Convergence Criteria-1

Initial Node	End Node	Best Known Volume	Best Known Cost	Resulting volume convergence criteria 1	Resulting Cost convergence criteria 1	Absolute Volume Difference	Absolute Cost Difference
1	2	4494.6576	6.00087322	4508.9	6.0008	14.24235354	0.0000732
1	3	8119.0799	4.00907182	8119.2	4.0087	0.120051952	0.0003718
2	1	4519.0799	6.00090487	4521	6.0008	1.920051952	0.0001049
2	6	5967.3364	6.57573046	5976.2	6.583	8.863603829	0.0072695
3	1	8094.6576	4.00889247	8107.1	4.0086	12.44235354	0.0002925
3	4	14006.371	4.29555799	14098	4.2765	91.62898014	0.0190580
3	12	10022.32	4.02288371	10137	4.0211	114.6803848	0.0017837
4	3	14030.561	4.29277459	14138	4.2797	107.4390826	0.0130746
4	5	18006.371	2.33857528	18096	2.3217	89.62898014	0.0168753
4	11	5200	7.34138474	5279.9	7.2045	79.9	0.1368847
5	4	18030.561	2.33679867	18144	2.3251	113.4390826	0.0116987
5	6	8798.2677	9.65556474	8776.3	9.9385	21.96771411	0.2829353
5	9	15780.782	9.97183974	15847	9.7303	66.21794453	0.2415397
6	2	5991.7587	6.61932384	5988.3	6.5958	3.458697763	0.0235238
6	5	8806.4987	9.92239192	8810	10.0302	3.501333185	0.1078081
6	8	12492.925	14.4207501	12480	14.6391	12.92536056	0.2183499
7	8	12101.529	5.56683144	12173	5.6131	71.47087769	0.0462686
7	18	15794.011	2.0629184	15807	2.0624	12.98939302	0.0005184
8	6	12525.579	14.9956889	12526	14.8254	0.421385138	0.1702889
8	7	12040.918	5.69595764	12070	5.5259	29.08172715	0.1700576
8	9	6882.6649	15.0883665	6880.3	15.1675	2.364912662	0.0791335
8	16	8388.7131	10.3884259	8373.8	10.6888	14.913063	0.3003741
9	5	15796.741	9.81921102	15862	9.748	65.2589997	0.0712110
9	8	6836.706	15.559115	6865	15.1219	28.29402471	0.4372150
9	10	21744.076	5.68433845	21747	5.6841	2.923919823	0.0002385
10	9	21814.076	5.74845083	21847	5.7336	32.92391236	0.0148508
10	11	17726.625	12.3580866	17721	12.3967	5.625032961	0.0386134
10	15	23125.797	13.6776018	23132	13.7313	6.202709897	0.0536982
10	16	11047.094	20.0767055	11041	20.0489	6.093881273	0.0278055
10	17	8100	16.3637937	8104.6	16.3268	4.6	0.0369937
11	4	5300	7.43288998	5372.2	7.291	72.2	0.1418900
11	10	17604.224	12.1220379	17628	12.2415	23.77646677	0.1194621
11	12	8365.2857	13.623693	8363.6	13.584	1.685653859	0.0396930
11	14	9776.1195	13.7376938	9793	13.7583	16.88046725	0.0206062
12	3	9973.7074	4.02282185	10084	4.0207	110.292584	0.0021219
12	11	8404.9346	13.591282	8410.6	13.7559	5.665376053	0.1646180
12	13	12287.605	3.02527354	12398	3.0236	110.394731	0.0016735
13	12	12378.642	3.02595453	12492	3.0244	113.35796	0.0015545
13	24	11121.358	17.5209178	11106	17.5847	15.35796002	0.0637822
14	11	9814.0691	13.9602746	9844.6	13.9658	30.53093707	0.0055254
14	15	9036.3341	12.4240914	9018.2	12.1763	18.13413403	0.2477914
14	23	8400.4368	9.13828827	8394.4	9.0648	6.036830275	0.0734883
15	10	23192.283	13.8494822	23189	13.8068	3.283359358	0.0426822
15	14	9079.8203	12.6545692	9075.3	12.3599	4.520316587	0.2946692
15	19	19083.29	4.31533113	19085	4.3268	1.710235253	0.0114689
15	22	18409.935	8.82925459	18355	9.0164	54.93502652	0.1871454
16	8	8406.7144	10.7567367	8331.8	10.5756	74.91440521	0.1811367
16	10	11073.009	20.278261	11075	20.2457	1.99068079	0.0325610
16	17	11695.003	9.29353764	11648	9.3822	47.00291653	0.0886624
16	18	15278.325	3.16811361	15414	3.1694	135.6747585	0.0012864
17	10	8100	16.4040979	8107.9	16.3403	7.9	0.0637979
17	16	11683.838	9.20846104	11638	9.3568	45.83828244	0.1483390
17	19	9953.0214	7.39193251	9933.3	7.3938	19.72143205	0.0018675
18	7	15854.621	2.06057433	15910	2.0641	55.37854356	0.0035257
18	16	15333.407	3.17730667	15416	3.1694	82.59334425	0.0079067
18	20	18976.796	4.26882093	19068	4.2644	91.2038808	0.0044209
19	15	19116.724	4.32704943	19114	4.3348	2.724279078	0.0077506
19	17	9941.8568	7.33945081	9926.6	7.3791	15.25679796	0.0396492
19	20	8688.367	9.41333465	8671	9.4155	17.36704049	0.0021654
20	18	18992.488	4.26638852	19073	4.2647	80.5116171	0.0016885
20	19	8710.6369	9.45847196	8693.2	9.4712	17.43692073	0.0127280
20	21	6302.0229	8.19158776	6307.8	8.1737	5.777125813	0.0178878
20	22	7000	7.82086045	7023.1	7.7491	23.1	0.0717605
21	20	6239.985	8.0781726	6248.9	8.0935	8.914981878	0.0153274
21	22	8619.5397	4.19639468	8622.9	4.217	3.360301898	0.0206053
21	24	10309.411	11.5118649	10296	11.8792	13.41080392	0.3673351
22	15	18386.473	8.90087026	18340	8.9961	46.472764	0.0952297
22	20	7000	7.75436095	7008.9	7.727	8.9	0.0273609
22	21	8607.3879	4.17003834	8623.9	4.218	16.51207026	0.0479617
22	23	9661.8242	12.077024	9624.7	12.238	37.12423137	0.1609760
23	14	8394.9002	9.10343657	8388.9	9.0516	6.000177898	0.0518366
23	22	9626.2102	12.0356797	9596	12.1401	30.21020048	0.1044203
23	24	7902.9839	3.79794116	7889.1	3.747	13.88392706	0.0509412
24	13	11112.395	17.4435091	11100	17.556	12.39473098	0.1124909
24	21	10259.525	11.3213716	10236	11.6743	23.52471622	0.3529284
24	23	7861.8332	3.77398518	7854.9	3.7169	6.933243796	0.0570852

Appendix B

Results-Convergence Criteria-2

Initial Node	End Node	Best Known Volume	Best Known Cost	Resulting volume convergence criteria 1	Resulting Cost convergence criteria 1	Absolute Volume Difference	Absolute Cost Difference
1	2	4494.658	6.000873	4534.6	6.0008	39.94235354	0.0000732
1	3	8119.08	4.009072	8129.4	4.0087	10.32005195	0.0003718
2	1	4519.08	6.000905	4532.4	6.0008	13.32005195	0.0001049
2	6	5967.336	6.57573	5977.3	6.5841	9.963603829	0.0083695
3	1	8094.658	4.008892	8131.6	4.0087	36.94235354	0.0001925
3	4	14006.37	4.295558	14152	4.2808	145.6289801	0.0147580
3	12	10022.32	4.022884	10201	4.0217	178.6803848	0.0011837
4	3	14030.56	4.292775	14208	4.2853	177.4390826	0.0074746
4	5	18006.37	2.338575	18149	2.3255	142.6289801	0.0130753
4	11	5200	7.341385	5334	7.2547	134	0.0866847
5	4	18030.56	2.336799	18218	2.3304	187.4390826	0.0063987
5	6	8798.268	9.655565	8736.6	9.8319	61.66771411	0.1763353
5	9	15780.78	9.97184	15889	9.7801	108.2179445	0.1917397
6	2	5991.759	6.619324	5975.1	6.5818	16.65869776	0.0375238
6	5	8806.499	9.922392	8819.7	10.0569	13.20133319	0.1345081
6	8	12492.93	14.42075	12463	14.5685	29.92536056	0.1477499
7	8	12101.53	5.566831	12085	5.538	16.52912231	0.0288314
7	18	15794.01	2.062918	15763	2.0617	31.01060698	0.0012184
8	6	12525.58	14.99569	12544	14.898	18.42138514	0.0976889
8	7	12040.92	5.695958	12075	5.5301	34.08172715	0.1658576
8	9	6882.665	15.08837	6855.6	15.0938	27.06491266	0.0054335
8	16	8388.713	10.38843	8367.6	10.6719	21.113063	0.2834741
9	5	15796.74	9.819211	15875	9.7633	78.2589997	0.0559110
9	8	6836.706	15.55911	6882.1	15.173	45.39402471	0.3861150
9	10	21744.08	5.684338	21748	5.6844	3.923919823	0.0000615
10	9	21814.08	5.748451	21860	5.7404	45.92391236	0.0080508
10	11	17726.63	12.35809	17697	12.3556	29.62503296	0.0024866
10	15	23125.8	13.6776	23121	13.7159	4.797290103	0.0382982
10	16	11047.09	20.07671	11051	20.1056	3.906118727	0.0288945
10	17	8100	16.36379	8107.7	16.3395	7.7	0.0242937
11	4	5300	7.43289	5421.1	7.3387	121.1	0.0941900
11	10	17604.22	12.12204	17606	12.2064	1.776466769	0.0843621
11	12	8365.286	13.62369	8363.3	13.5829	1.985653859	0.0407930
11	14	9776.12	13.73769	9806.7	13.8133	30.58046725	0.0756062
12	3	9973.707	4.022822	10147	4.0212	173.292584	0.0016219
12	11	8404.935	13.59128	8416.1	13.7762	11.16537605	0.1849180
12	13	12287.61	3.025274	12460	3.0241	172.394731	0.0011735
13	12	12378.64	3.025955	12558	3.0249	179.35796	0.0010545
13	24	11121.36	17.52092	11106	17.5856	15.35796002	0.0646822
14	11	9814.069	13.96027	9850.6	13.9902	36.53093707	0.0299254
14	15	9036.334	12.42409	9088.1	12.4017	51.76586597	0.0223914
14	23	8400.437	9.138288	8425.7	9.1406	25.26316973	0.0023117
15	10	23192.28	13.84948	23200	13.8224	7.716640642	0.0270822
15	14	9079.82	12.65457	9145.5	12.5902	65.67968341	0.0643692
15	19	19083.29	4.315331	19090	4.3282	6.710235253	0.0128689
15	22	18409.94	8.829255	18299	8.9422	110.9350265	0.1129454
16	8	8406.714	10.75674	8412.5	10.7949	5.785594789	0.0381633
16	10	11073.01	20.27826	11069	20.2111	4.00931921	0.0671610
16	17	11695	9.293538	11649	9.3841	46.00291653	0.0905624
16	18	15278.33	3.168114	15319	3.1652	40.67475848	0.0029136
17	10	8100	16.4041	8113.2	16.3623	13.2	0.0417979
17	16	11683.84	9.208461	11633	9.3432	50.83828244	0.1347390
17	19	9953.021	7.391933	9934.1	7.3954	18.92143205	0.0034675
18	7	15854.62	2.060574	15772	2.0619	82.62145644	0.0013257
18	16	15333.41	3.177307	15399	3.1687	65.59334425	0.0086067
18	20	18976.8	4.268821	19080	4.2651	103.2038808	0.0037209
19	15	19116.72	4.327049	19127	4.3384	10.27572092	0.0113506
19	17	9941.857	7.339451	9923.5	7.3724	18.35679796	0.0329492
19	20	8688.367	9.413335	8667.1	9.4058	21.26704049	0.0075346
20	18	18992.49	4.266389	19069	4.2644	76.5116171	0.0019885
20	19	8710.637	9.458472	8693.3	9.4715	17.33692073	0.0130280
20	21	6302.023	8.191588	6309.7	8.1762	7.677125813	0.0153878
20	22	7000	7.82086	7038.8	7.7737	38.8	0.0471605
21	20	6239.985	8.078173	6248.3	8.0927	8.314981878	0.0145274
21	22	8619.54	4.196395	8614.4	4.2082	5.139698102	0.0118053
21	24	10309.41	11.51186	10236	11.6715	73.41080392	0.1596351
22	15	18386.47	8.90087	18298	8.9421	88.472764	0.0412297
22	20	7000	7.754361	7015	7.7364	15	0.0179609
22	21	8607.388	4.170038	8585.2	4.1785	22.18792974	0.0084617
22	23	9661.824	12.07702	9613.4	12.1993	48.42423137	0.1222760
23	14	8394.9	9.103437	8412.2	9.1079	17.2998221	0.0044634
23	22	9626.21	12.03568	9560.4	12.02	65.81020048	0.0156797
23	24	7902.984	3.797941	7944.9	3.7969	41.91607294	0.0010412
24	13	11112.39	17.44351	11105	17.5794	7.394730977	0.1358909
24	21	10259.52	11.32137	10203	11.5626	56.52471622	0.2412284
24	23	7861.833	3.773985	7878.5	3.7376	16.6667562	0.0363852