Evaluation of Adaptive Routing using Dijkstra's Algorithm

ECGR 5187 Data Communication and Networking II

Prasanth Reddy Pallerla

ID: 801033619

Email: ppallerl@uncc.edu

Abstract- **The purpose of the project is to evaluate and know the effect of adaptive link costs on the traffic in the links and the total number of transmissions of the network. Also, the project demonstrates how the packets flow with respect to the changing costs of the links. This report contains the introduction to the project and explanation to the code that I implemented. This report is organized in the following manner: First a brief introduction of project, explanation to the code, discussion on the results obtained, code and finally conclusions.**

I. Introduction

In this project, the code for the Dijkstra's algorithm is given which calculates the least cost path for the network. Now we need to adapt the costs of the links based on the current load on the links using the given formula. The assumption in the given code is that the link costs are same in both the directions.
The code adaptation formula used in the project is shown below:

$$\text{Cost}(i,j) = (1-p) \times \text{Cost}(i,j) + p \times (W(i,j) + 5 \times L(i,j))$$

Where, p is a variable parameter. P= {0.2, 0.4, 0.6, 0.8, 1.0}

$W(i,j) =$ Initial cost matrix.

$\text{L}(i,j) =$ Traffic or Load matrix.

(In LHS) Cost(i, j) = Updated Cost matrix

(In RHS) Cost(i, j) = Previous Cost matrix

Now we need to develop a code which goes through the following steps in a loop

1)calculate the shortest paths, 2) update the costs based on the load matrix 3) Again calculate the least cost paths.

In each iteration of the loop we need to find the following values:

1) Number of transmissions on links 3-2 and 5-4.

2) Calculated costs on links 3-2 and 5-4

3) Total number of transmissions in the network.

The above loop should be repeated for different values of p provided in the above set.

II. Code Briefing

1) **Lines 1-4** are the include and #define sections.

2) **Lines 6-19** contains a user defined function which calculates the number of transmissions on links 3-2 and 5-4, and this also calculates the total number of transmissions in the network.

3) **Lines 21-34** is another user defined function which re-initializes the preced, distance and L matrix to zero before each iteration of the loop.

4) **Lines 36-51** contains the initialization of the cost matrix and copying the initial cost matrix to the W(i, j) matrix and declaration of some variables and arrays needed for the program.

5) In **line 53**, I have used a while loop which iterates for 15 times. **Lines 55-60** contains a call to the reset function to set the values of matrices to zero before each iteration, printing the iteration number of the loop and costs of links 3-2 and 5-4. And finally calling the shortpath function which calculates the least cost paths in the networks.

6) **Lines 62-71** is the part of the code that calculates the traffic or load on each of the links. I have made L(i, j)=L(j, i) to make the calculated link costs to remain same in both the directions in all the iterations of the loop. **Line 72** is a call to the printL function which is discussed in step 2.

7) **Lines 74-83** is the section of the code that calculates the updated cost matrix based on the above formula.

8) **Lines 84-87** contains the increment to the count variable to iterate the loop for 15 times and finally return 0 to exit the main function.

9) Repeat the code for different values p

10) Finally plot the results obtained.

III. Results

After running the code, the following graphs are plotted

1) Cost of link 3->2 for different values of p

2) Cost of link 5->4 for different values of p

3) Load on link 3->2 for different values of p

4) Load on link 5->4 for different values of p

5) Number of transmissions in the network.

The graphs are shown below.

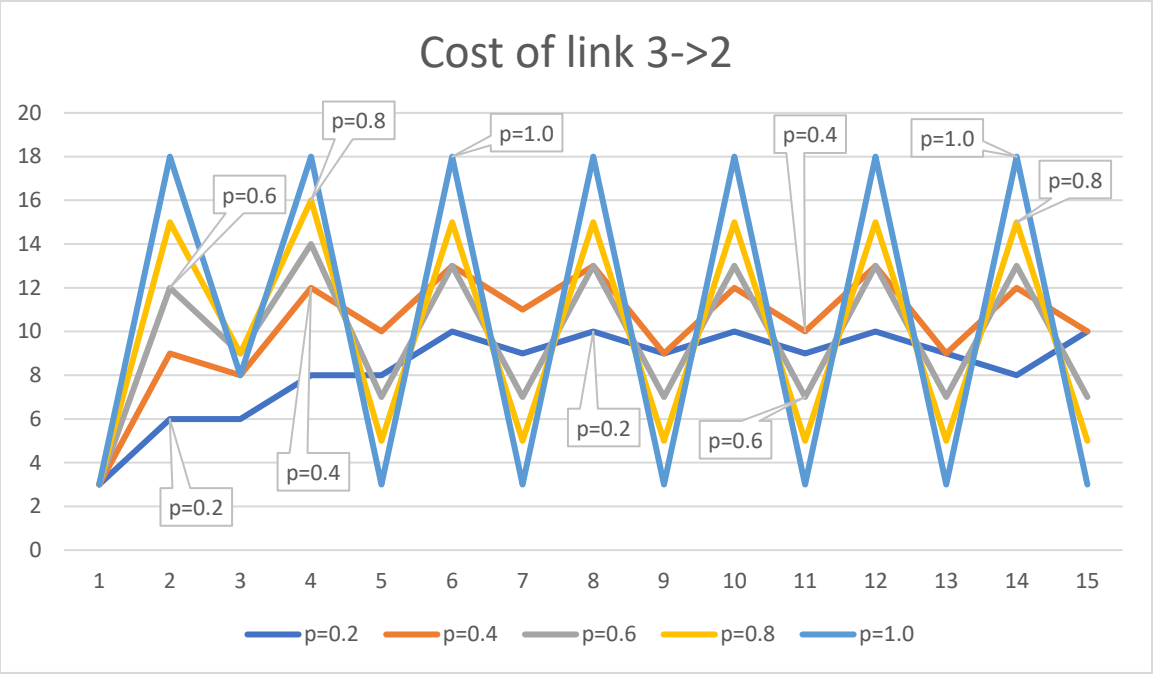
Cost of link 3->2
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
p=0.8
p=1.0
p=0.4
p=1.0
p=0.6
p=0.8
p=0.2
p=0.6
p=0.4
p=0.2
p=0.2
p=0.4
p=0.6
p=0.8
p=1.0

fig. 1

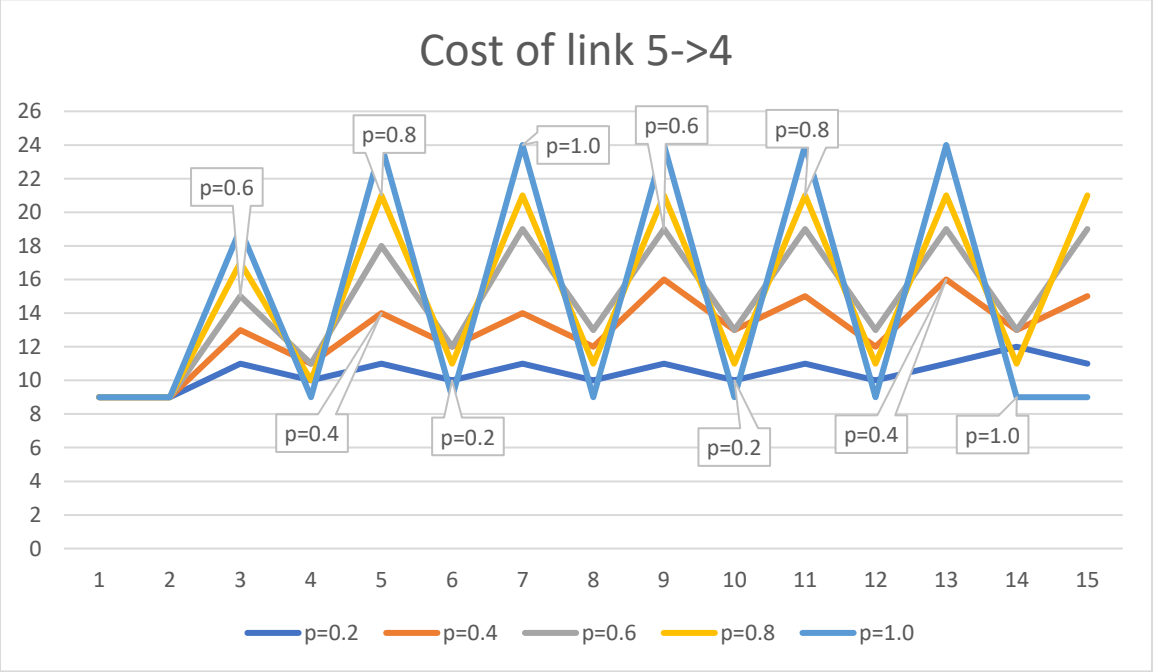
Cost of link 5->4
26
24
22
20
18
16
14
12
10
8
6
4
2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
p=0.6
p=0.8
p=1.0
p=0.6
p=0.8
p=0.4
p=0.2
p=0.2
p=0.4
p=1.0
p=0.2
p=0.4
p=0.6
p=0.8
p=1.0

fig. 2

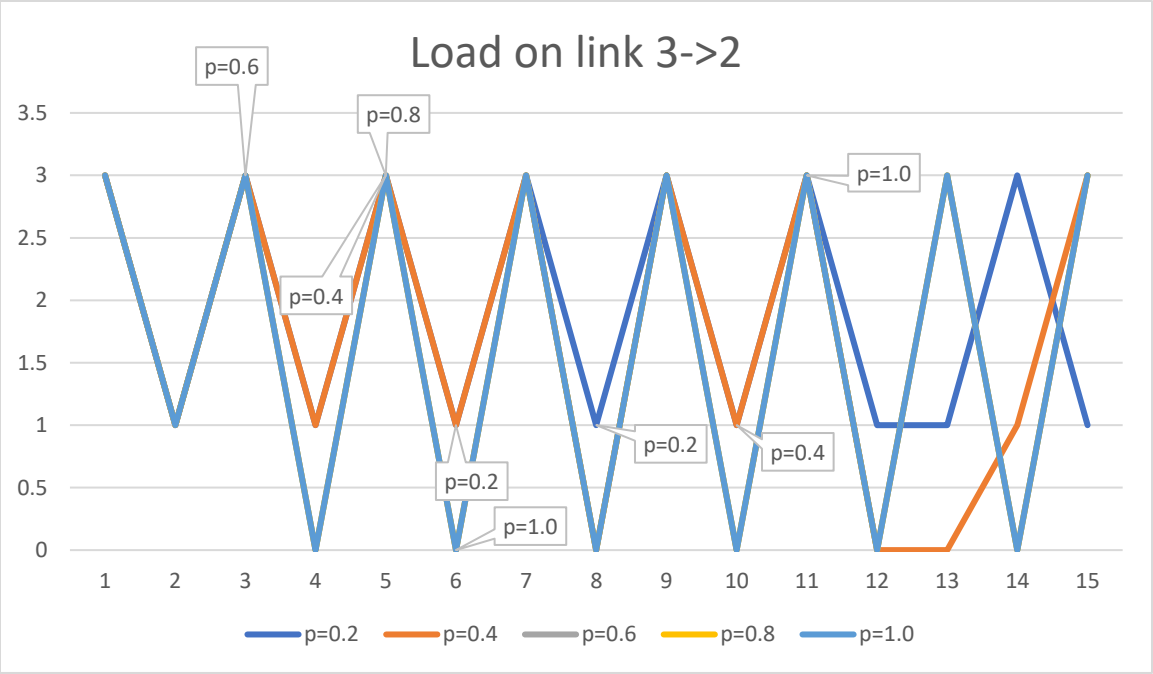
Load on link 3->2
p=0.6
p=0.8
p=1.0
p=0.4
p=0.2
p=0.4
p=0.2
p=1.0
3.5
3
2.5
2
1.5
1
0.5
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
p=0.2 p=0.4 p=0.6 p=0.8 p=1.0

fig. 3

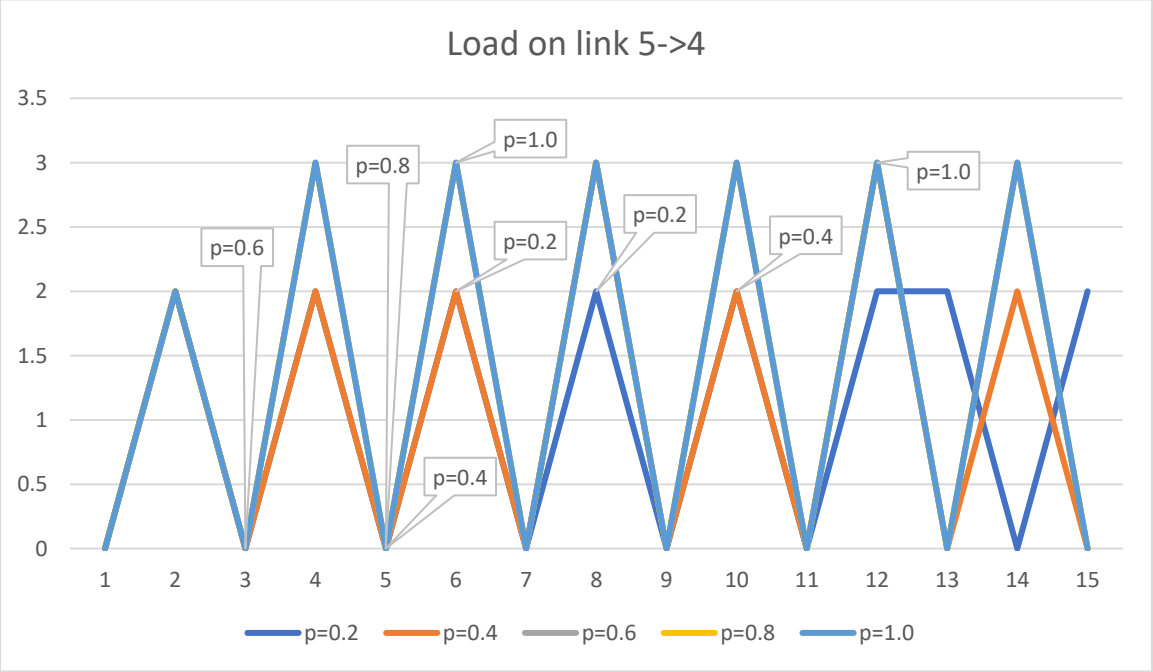
Load on link 5->4
p=1.0
p=0.8
p=1.0
p=0.6
p=0.2
p=0.2
p=0.4
p=0.4
3.5
3
2.5
2
1.5
1
0.5
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
p=0.2 p=0.4 p=0.6 p=0.8 p=1.0

fig.4

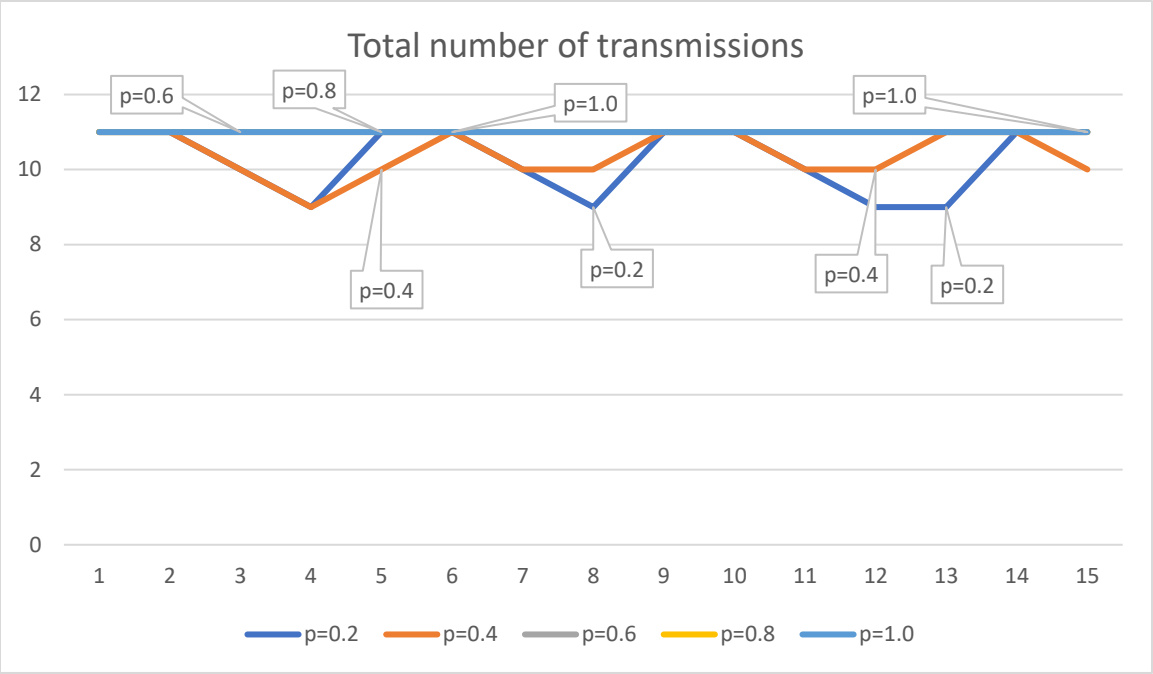

fig. 5

Let us discuss on p=1.0 case. We can see from graphs 1 and 2, in 7th and 8th iterations of the loop the costs on the links 3-2 are 3,18 respectively and costs on the links 5-4 are 24,9 respectively. In 9th and 10th iterations of the loop the costs on the links 3-2 are 3,18 respectively and costs on the links 5-4 are 24,9 respectively. As you can see that in the 7th iteration, cost of link 3-2 is low then the cost of link 5-4 is high and in this case the packets form node 3,5,6 flows on the link 3-2. Consider the 8th or 10th iteration, the cost of links 3-2 is high whereas the cost of link 5-4 is low and the packets from node 3,5,6 flows on the link 5-4. Also, we can see that the link costs are repeating after every 2 iterations.

Also, at p=1.0, we can see from graphs 3 and 4, in 7th and 8th iterations of the loop the number of transmissions on the links 3-2 are 3,0 respectively and costs on the links 5-4 are 0,3 respectively. In 9th and 10th iterations of the loop the costs on the links 3-2 are 3,0 respectively and costs on the links 5-4 are 0,3 respectively. From this we can infer that since in 7th and 9th iteration the cost on link 3-2 is low, hence the traffic on link 3-2 is high as all the packets form nodes 3,5,6 flows on this link. Also, in 8th and 10th iterations the cost of link 5-4 is low and hence the traffic on link 5-4 is high as all the packets form nodes 3,5,6 flows on this link. Thus, we can see that always the packets flow on the least cost paths only.

Also, from the 5th graph we can see that the number of transmissions in the network for p=0.6, 0.8, 1.0 remains constant with all the iterations of the loop. We can say that as the value of the p is high the network adapted quickly to the changing costs.

We can observe from the graphs 1 and 2, for higher values of p i.e. when p=0.6,0.8,1.0 the cost on links 3-2 and 5-4 has a more difference, hence the packets flow in any on of the link. That is the reason why we can say number of transmissions on those links are either 3 or 0 according to the link costs. Whereas when p=0.2,0.4 the cost on links 3-2 and 5-4 has a little difference and hence the packets can flow in any one of them or both. Hence the number of transmissions on links 3-2 and 5-4 in these cases are 0,1,2,3. By this we can say that, with lower values of p the adaptation is slower and with higher values of p we have aggressive adaptation because we are multiplying the higher value of p with 5 times the L(i, j) and hence the link costs vary greatly with each iteration.

IV. Conclusions

The adaptation is aggressively done with higher values of p and the number of transmissions in the network remains constant in all the iterations with aggressive adaption. With lower values of p, the adaptation is slower.

Q) **Can you suggest additional adaptation** ideas **to improve stability and performance?**
A) With higher values of p, the degree of oscillations is increased and the total number of transmissions in the network remains constant. Also to increase the stability we should also consider adding the L matrix to the Cost matrix in the 1st term of the RHS in the equation i.e. the equation should be like $\text{Cost}(i, j) = (1-p) \times (\text{Cost}(i, j) + 3*\text{L(I, j)}) + p \times (W(i, j) + 2 \times L(i,j))$ so that the equation the costs does not change when compared to the previous ones. Moreover, we should also consider adding the smoothing factor to the equation so that the oscillations will decrease.

Q) **How would you need to change your code if the link costs were not symmetrical, i.e. Cost(i, j)!=Cost(j, i).**
A) In this project, we are supposed to send packets from all other nodes to node 1, i.e. we must find the least cost paths form other nodes to node1. Since the given code is calculating the least cost paths to other nodes form node 1 only, I have made the Load matrix symmetrical to maintain the updated cost matrix symmetrical.

By carefully examining the given shortest.c code, I came to know that since in line 18, the current is initialized to 0 and in line 21 we have used selected[current]=1, hence the given code is calculating the least cost paths form node 1 to all other nodes.

So, to find the least cost paths from other nodes to node 1 we can alternatively change the current to 1,2,3,4,5 in line 18 so that we can obtain the least cost paths from node 2,3,4,6 to node 1 respectively. And by doing this, there is no need to maintain the cost matrix symmetrical all the time.

```
#include"shortest.c"
#include <stdio.h>
#define MAX 6  // Number of nodes
#define INFINITE 998  //infinity

void printL(int L[][MAX])
{      int sum=0;
    for(int i=0;i<MAX;i++)
    {
     for(int j=0;j<i;j++)
     {
      if(L[i][j]>0)
      sum+=L[i][j];
     }
    }
    printf("\nNumber of transmission on links 3-2 and 5-4 are %d, %d
respectively\n\n",L[2][1],L[4][3]);
    printf("\nTotal number of transmissions in all the links is: %d\n",sum);
}

void reset(int preced[MAX],int distance[MAX],int L[][MAX])
{
        for(int i=0;i<MAX;i++)
        {
         for(int j=0;j<MAX;j++)
              L[i][j]=0;
        }

        for(int i=0;i<MAX;i++)
        {
              preced[i]=0;
              distance[i]=0;
        }
}

int main()
{
 int cost[MAX][MAX]=
     {{INFINITE,2,INFINITE,1,INFINITE,INFINITE},
      {2,INFINITE,3,2,INFINITE,INFINITE},
      {INFINITE,3,INFINITE,INFINITE,1,5},
      {1,2,INFINITE,INFINITE,9,INFINITE},
      {INFINITE,INFINITE,1,9,INFINITE,2},
      {INFINITE,INFINITE,5,INFINITE,2,INFINITE}};

  int i,preced[MAX],distance[MAX],L[MAX][MAX], W[MAX][MAX],count=1,j,temp;
  float p=0.2;

   for(int i=0; i<MAX; i++)
    for(int j=0; j<MAX; j++)
     W[i][j]=cost[i][j];

 while(count<=15)
 {
       reset(preced,distance,L);
       printf("\n\n*************************Iteration
NUmber=%d*************************\n\n",count);
       printf("\nCosts on links 3-2 and 5-4 are %d, %d
respectively\n\n",cost[2][1],cost[4][3]);
   shortpath(cost,preced,distance);

```

```
    for(i=1;i<MAX;i++)
    {     j=i;
          while(j!=0)
          {
                 temp=preced[j];
                 L[j][temp]+=1;
                 L[temp][j]+=1;
                 j=temp;
          }
    }
          printL(L);

          for(int i=0;i<MAX;i++)
          {
           for(int j=0;j<MAX;j++)
           {
                 if(W[i][j]!=INFINITE)
                 cost[i][j]=(1-p)*cost[i][j]+p*(W[i][j]+5*L[i][j]);
             else
                 cost[i][j]==INFINITE;
           }
          }
     count++;
  }
   return 0;
  }
```