Network Architectures and Services (NAS) Group  
Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS)  
Delft University of Technology

Link-Disjoint Quality of Service Routing

Student: Xia GUO  
Supervisor: Prof. dr. ir. P. Van Mieghem  
Committee Members:  
   Advisor: Dr.ir. F.A. Kuipers  
   Member: Prof. dr. ir. N.H.G. Baken  
   Member: Dr.ir. A. Lo  
Codenumber: PVM 2005-034  
Date: August 2005
Abstract

In a network, sometimes a path may fail during the course of transferring information along the path. Ideally, we expect that a backup path can be used so that the information data will be quickly directed on this backup path. Consequently, it is necessary to construct a routing algorithm which can build a pair of link-disjoint paths between a specified source and destination node. If the primary path is blocked, another backup path can quickly be used.

With the development of Internet, many new applications appear, like multi-media applications, users still need more service. Meantime, Internet also needs more guarantees on the reliability of the data it transports. Therefore just finding a pair of link-disjoint paths is not enough, more constraints on the pair of paths are necessary. It is the so-called MCLPP (multiple constrained link-disjoint path pair) problem. Because the information will be flooded in the Internet, it is better to find the pair of link-disjoint paths with minimal total length, which is the MCOLPP (multiple constrained optimal link-disjoint path pair) problem. Actually, the MCOLPP problem is the optimal solution of MCLPP problem.

To solve the MCLPP problem, a heuristic routing algorithm, DIMCRA (link-disjoint multiple constraints routing algorithm) is proposed. It is used to improve routing reliability and traffic engineering.

This research focuses on the implementation and evaluation of DIMCRA, and developing possible new methods to improve its current performance. In order to measure its current performance in solving the MCLPP problem, the RF (remove-find) algorithm is employed to examine its performance. The performance of DIMCRA and RF are measured not only in finding a solution, but also in finding an optimal solution. By analyzing the results of a lot of simulations, it is concluded that DIMCRA is always better than RF. Once RF can find a solution, then DIMCRA certainly can find it. In addition, whether DIMCRA can find a solution depends on four factors: the components of DIMCRA’s solution; the number of overlapping links, link density $p$; and correlation $r$. Although currently DIMCRA is the best link-disjoint multiple constraints routing algorithm, it is still can be improve it. Consequently, in the end of this thesis, three possible means of improvements are proposed. Method 1 is simple and does improve the success rate in finding an optimal solution with 3.7% average. Method 2's efficiency is prominent, but the cost is high. Method 3 is a good choice to improve DIMCRA’s performance, because its cost is less than Method 2 at the same success rate of Method 2.

Keywords: MCLPP, MCOLPP, DIMCRA, RF
Acknowledgement

First my special thank to my mentor dr.ir. F.A. Kuipers (Delft University of Technology) who has led me to the splendid shrine of scientific research in routing algorithm.

I would like to thank Prof. dr. ir. Piet Van Mieghem (Delft University of Technology) for his insightful supervisions and suggestions during this work.

I would also like to acknowledge Yuchun GUO who gave me some good advises.

To my parents and sisters, I wish to express my appreciations for their unconditional support.

Finally, thanks go to my husband for his love.
## Contents

List of Figures ................................................................. IX

List of Tables ................................................................. XI

1. **Introduction** ............................................................. 1
   1.1 Link-disjoint QoS Routing Problem ................................. 1
       1.1.1 Background ......................................................... 1
       1.1.2 Motivation and Problem definition ............................. 2
   1.2 Structure of the Thesis ............................................... 3

2. **Link-disjoint QoS Routing Algorithms** ........................... 5
   2.1 Introduction ................................................................... 5
   2.2 RF Algorithm .................................................................. 5
       2.2.1 RF’s steps ............................................................. 5
       2.2.2 RF’s code .............................................................. 6
       2.2.3 RF’s complexity ..................................................... 7
       2.2.4 RF’s examples ....................................................... 8
       2.2.5 RF’s property ....................................................... 11
   2.3 DIMCRA Algorithm ..................................................... 11
       2.3.1 DIMCRA’s steps ..................................................... 11
       2.3.2 DIMCRA’s concept ................................................. 12
       2.3.3 DIMCRA’s code .................................................... 13
       2.3.4 DIMCRA’s complexity ............................................ 16
       2.3.5 DIMCRA’s examples .............................................. 17
       2.3.6 DIMCRA’s property .............................................. 24
   2.4 K-shortest Paths Algorithm ........................................... 24

3. **DIMCRA’s Performance** ............................................. 27
   3.1 Motivation and Problem Definition ................................ 27
   3.2 Results and Analysis of Simulation for the Random Graphs .... 27
       3.2.1 Comparison of DIMCRA with RF in finding a solution ... 28
           3.2.1.1 Simulation description ....................................... 28
           3.2.1.2 \( p \) (link density) is changed .......................... 28
           3.2.1.3 \( r \) (correlation between link weights) is changed ... 30
           3.2.1.4 \( N \) (number of nodes) is changed ...................... 31
           3.2.1.5 Summary ....................................................... 32
       3.2.2 Comparison of DIMCRA with RF in finding an optimal solution 33
           3.2.2.1 Simulation description ....................................... 33
# CONTENTS

3.2.2.2 Success rate .........................................................33
3.2.2.3 Length ................................................................35
3.2.2.4 Summary..........................................................37
3.3 Results and Analysis of Simulation for the Lattice Graphs........37
  3.3.1 Comparison of DIMCRA with RF in finding a solution.....37
    3.3.1.1 Simulation description........................................37
    3.3.1.2 $r$ (correlation between link weights) is changed...38
    3.3.1.3 $N$ (number of nodes) is changed.........................39
    3.3.1.4 Summary........................................................40
  3.3.2 Comparison of DIMCRA with RF in finding an optimal solution 40
    3.3.2.1 Simulation description........................................40
    3.3.2.2 Success rate....................................................40
    3.3.2.3 Length ................................................................43
    3.3.2.4 Summary........................................................44
3.4 Overview .....................................................................45

4. **DIMCRA’s Performance** ........................................47
  4.1 Motivation and Problem Definition...............................47
  4.2 DIMCRA’s Solutions ..................................................47
    4.2.1 DIMCRA’s solutions classifying...............................47
    4.2.2 Solution Class 1: node-disjoint .................................48
      4.2.2.1 Description....................................................48
      4.2.2.2 Evaluation....................................................48
    4.2.3 Solution Class 2: link-no-overlapping-disjoint........49
      4.2.3.1 Description....................................................49
      4.2.3.2 Evaluation....................................................50
    4.2.4 Solution Class 3: link-overlapping-disjoint............50
      4.2.4.1 Description....................................................50
      4.2.4.2 Evaluation....................................................51
    4.2.5 Summary............................................................52
  4.3 DIMCRA Limitations....................................................52
  4.4 Overview ...................................................................52

5. **Improving DIMCRA** .................................................53
  5.1 Motivation and Problem Definition...............................53
  5.2 Method 1....................................................................53
    5.2.1 Description.......................................................53
    5.2.2 Evaluation.........................................................55
    5.2.3 Summary..........................................................57
  5.3 Method 2....................................................................57
    5.3.1 Description.......................................................58
    5.3.2 Evaluation.........................................................60
    5.3.3 Summary..........................................................61
  5.4 Method 3....................................................................61
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.1</td>
<td>Description</td>
<td>61</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Evaluation</td>
<td>64</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Summary</td>
<td>65</td>
</tr>
<tr>
<td>6.</td>
<td>Conclusions</td>
<td>67</td>
</tr>
<tr>
<td>6.1</td>
<td>Summary</td>
<td>67</td>
</tr>
<tr>
<td>6.2</td>
<td>Main Contributions</td>
<td>68</td>
</tr>
<tr>
<td>6.3</td>
<td>Potential Future Extensions</td>
<td>69</td>
</tr>
<tr>
<td></td>
<td>Reference</td>
<td>71</td>
</tr>
<tr>
<td></td>
<td>About the Author</td>
<td>73</td>
</tr>
</tbody>
</table>
List of Figures

Fig.2.1 RF algorithm code ........................................................................................................... 7
Fig.2.2 A bi-direction graph ........................................................................................................ 7
Fig.2.3 Example 1 of RF ............................................................................................................ 8
Fig.2.4 Example 2 of RF ............................................................................................................ 10
Fig.2.5 Example 3 of RF ............................................................................................................ 10
Fig.2.6 DIMCRA’s main function code ...................................................................................... 14
Fig.2.7 UNION function code .................................................................................................... 15
Fig.2.8 UPDATE function code .................................................................................................. 15
Fig.2.9 Example 1 of DIMCRA ................................................................................................ 19
Fig.2.10 Example 2 of DIMCRA .............................................................................................. 21
Fig.2.11 Example 3 of DIMCRA .............................................................................................. 22
Fig.2.12 Example 4 of DIMCRA .............................................................................................. 23

Fig.3.1 For m=2, N=100, the comparison of success rate and execution time between
DIMCRA and RF in finding a set of two link-disjoint paths (solution) for the
class of random graphs as a function of the link density p ........................................... 30
Fig.3.2 For m=2, N=100, the comparison of success rate and execution time between
DIMCRA and RF in finding a solution for the class of random graphs as a
function r ..................................................................................................................................... 31
Fig.3.3 For m=2, the comparison of success rate and execution time between
DIMCRA and RF in finding a solution for the class of random graphs as a
function of N .................................................................................................................................. 32
Fig.3.4. For m=2, N=9, the average length of DIMCRA’s solution, RF’s solutions
and the optimal solutions for the random graphs ................................................................. 36
Fig.3.5 For m=2, N=9, the average of length difference of DIMCRA’s solution and
RF’s solution with the optimal solution for the random graphs ........................................ 37
Fig.3.6 For m=2, N=49, the comparison of success rate and execution time between
DIMCRA and RF in finding a solution for the class of lattice graphs as a
function of r .................................................................................................................................. 38
Fig.3.7 For m=2, the comparison of success rate and execution time between
DIMCRA and RF in finding a solution for the class of lattice graphs as a
function of N .................................................................................................................................. 39
Fig.3.8 For m=2, N=16, the success rate comparison of DIMCRA and RF in finding
an optimal solution for the class of lattice graphs as a function of r ............................... 42
Fig.3.9 For m=2, the success rate comparison of DIMCRA and RF in finding an
optimal solution for the class of lattice graphs as a function of N ..................................... 43
Fig.3.10 For m=2, N=16, the average of DIMCRA’s solution and RF’s solution with
the optimal solution for the lattice graphs ............................................................................ 44
Fig.3.11 For m=2, N=16, the average of length difference of DIMCRA’s solution
and the optimal solution for the lattice graphs ..................................................................... 44
Fig. 4.1 An example of Class 1.................................................................................. 48
Fig. 4.2 For m=2, N=16, the success rate of DIMCRA’s solution Class 1: node-disjoint for the lattices as a function of r .................................................. 49
Fig. 4.3 An example of Class 2.................................................................................. 49
Fig. 4.4 For m=2, N=16, the success rate of DIMCRA’s solution Class 2: link-no-overlapping-disjoint for the class of lattice graphs as a function of r... 50
Fig. 4.5 An example of Class 3.................................................................................. 51
Fig. 4.6 For m=2, N=16, the success rate of DIMCRA’s solution Class 3: link-overlapping-disjoint for the class of lattice graphs as a function of r ....... 51

Fig. 5.1 Example 1 of Method 1 ............................................................................. 54
Fig. 5.2 Example 2 of Method 1 ............................................................................. 55
Fig. 5.3 For m=2, N=16, the success rate and execution time comparison of DIMCRA and improved DIMCRA in finding an optimal solution for the class of lattice graphs as a function of r .................................................. 56
Fig. 5.4 The concept of new algorithm .................................................................. 57
Fig. 5.5 An example of Method 2 .......................................................................... 60
Fig. 5.6 For m=2, N=16, execution time comparison of DIMCRA and improved DIMCRA in finding an optimal solution for the class of lattice graphs as a function of r .................................................. 60
Fig. 5.7 An example of Method 3 ........................................................................... 63
Fig. 5.8 A search tree ............................................................................................. 64
Fig. 5.9 For m=2, N=16, execution time comparison of DIMCRA and improved DIMCRA in finding an optimal solution for the class of lattice graphs as a function of r .................................................. 64
List of Tables

Table 3.1 For $m=2$, $N=9$ and $r=-1.0$, the success rate comparison of DIMCRA and RF in finding an optimal solution for the random graphs as a function of $p$ ........... 34
Table 3.2 For $m=2$, $p=0.2$ and $N=9$, the success rate comparison of DIMCRA and RF in finding an optimal solution for the class of random graphs as a function of $r$ .... 34
Table 3.3 For $m=2$, $p=0.2$ and $r=-1.0$, the success rate comparison of DICMRA and RF in finding an optimal solution for the class of random graphs as a function of $N$ ........................................................................................................................ 35
Table 3.4 Overview of DIMCRA’s and RF’s performance in finding a solution .... 45
Table 3.5 Overview of DIMCRA’s and RF’s performance in finding an optimal solution ......................................................................................................................... 45
1.1 Link-disjoint QoS Routing Problem

1.1.1 Background

In Internet, data packets will be transferred from source node to its destination node. The process of finding paths that the packets must follow from the source to the destination is called routing.

With the rapid development of Internet, the Internet research community has made great efforts to define more efficient network management and controlling functions. The motive forces of these efforts are specific requirements to performance relating to new applications. For instance, multi-media applications need better quality of service, and the Internet also needs more guarantees regarding the reliability of the data it transports.

Here we take the definition of Quality of Service from the ITU (International Telecommunication Union).

Definition: “Quality of Service --- the collective effect of service performance which determines the degree of satisfaction of a user of the service.”

It tells that QoS could enhance the performance of operational networks.

For example, in the Internet, QoS determines whether a multimedia application is productive and a videoconference is normal. There are many situations where QoS is needed. New applications are increasingly introducing more demands than before. QoS could lead to a better balance among these demands in a network, and could optimize the usage of the network’s resources. It will thus be more efficient.
Consequently, both finding shortest path and the path itself must obey multiple demands, which meet the user’s requirements. Finding paths that can meet such demands is called Quality of Services (QoS) routing.

However, the operation pursuing QoS efficiency will cause more cost which has to be beared by the network providers. This urges us to think about an interesting question: “how can we find a QoS routing algorithm which can not only keep the same level of cost, but also improve its reliability?”

This thesis is thus to focus on QoS routing algorithms. In this study we assume that the network-state information is temporarily static.

1.1.2 Motivation and Problem definition

In a network, a path sometimes can be blocked when information data is transferred. Consequently, we expect that a backup path can be used so that the information data will be quickly retransferred on this backup path from the source to its destination node. For this purpose, it is necessary to construct a QoS routing algorithm which can find a pair of link-disjoint paths between a specified source and its destination node. If one is blocked, another backup path can quickly start-up. Meantime, this path also has to obey multiple constraints. It is the so-called the MCLPP (multiple constrained link-disjoint path pair) problem. Before we investigate the MCLPP problem, we have to give the definition of path length.

The path length: Given a graph $G(V,E)$ with $M$ metrics per link, the non-linear length of a path $P$ from source node $s$ to destination node $t$ is defined as:

$$l(P) = \max(w_m(P) / L_m)$$

Here weights $w_m(P) = \sum_{(u \rightarrow v) \in P} w_m(u \rightarrow v)$.

We define the total length of two paths as:

$$l(P_1) + l(P_2)$$

for $M \geq 1$.

**Multiple constrained link-disjoint path pair (MCLPP) problem [1].** Given a graph $G(V,E)$ with multiple metric per link ($M \geq 2$) and a constraint vector $L$, for a

---

1 Paths between a given pair of source and destination nodes in a network are called link disjoint if they have no common (i.e. overlapping) links, and node disjoint if, besides the source and destination nodes, they have no common nodes [2].
source-destination pair \((s, t)\), find a pair of link-disjoint paths \(P_1\) and \(P_2\), such that path \(P_1\) is link-disjoint with \(P_2\), i.e. \(P_1 \cap P_2 = \emptyset\), and both paths obey the constraint vector \(L\).

\[
\sum_{u \in P} w_m(u \rightarrow v) \leq L_m \quad \text{for} \quad m = 1, \ldots, M
\]

Here \(w_m(P) = \sum_{(u \rightarrow v) \in P} w_m(u \rightarrow v)\), for \(m = 1, \ldots, M\).

MCLPP is NP-complete [2].

The goal of MCLPP is actually to find a pair of two link-disjoint paths that both obey multiple constraints. If we want to find this pair with the total length \(l(P_1) + l(P_2)\) which is minimized, then we name the MCLPP problem as the MCOLPP (Multiple constrained optimal link-disjoint path pair) problem. If \(M = 1\), the MCLPP problem will be changed to the LPP problem. In the past, much attention was addressed on this level. However, in this paper, we will focus on \(MCLPP\) \((M \geq 2)\).

To solve the MCLPP problem, an algorithm DIMCRA (link-disjoint multiple constraints routing algorithm) is proposed. It can improve the routing reliability and the traffic engineering. It will be introduced in details in Chapter 2.

In this thesis, we will first investigate the performance of DIMCRA in solving the MCLPP problem. We want to understand the property of DIMCRA for MCLPP. The final goal is to find some new methods to improve DIMCRA’s current performance. In this thesis, we thus mainly focus on DIMCRA’s performance and possible means to improve it.

1.2 Structure of the Thesis

In this thesis, we will first investigate DIMCRA’s current performance and later we will try to find some new methods to improve its performance.

In Chapter 2, we will review the RF algorithm, DIMCRA and the K-shortest paths algorithm [2]. The simulation will be operated based on these three algorithms. The RF algorithm is proposed to solve the LPP problem. However we also can use the RF algorithm to solve the MCLPP problem. The RF algorithm is employed to measure DIMCRA’s performance. The K-shortest paths algorithm is not for solving the MCLPP

\[\text{Link-disjoint path pair (LPP) problem. Given a graph } (G, V), \text{ with 1 metric per link } (M = 1); \text{ for a source-destination pair } (s, t), \text{ find a set of two paths } P_1 \text{ and } P_2; \text{ such that } P_1 \cap P_2 = \emptyset; \text{ and the total length } l(P_1) + l(P_2) \text{ is minimized [2].}\]
problem but for the MCP\(^3\) problem. We can get a pair of link-disjoint paths with the minimal total length (an optimal solution) by this algorithm.

In Chapter 3, in order to identify the properties of RF and DIMCRA in solving the MCLPP problem, first we will evaluate the performance of both RF and DIMCRA in finding a pair of link-disjoint paths which is probably not optimal. Secondly, we will evaluate the solution of RF and of DIMCRA with the optimal solution set to see which one is closer to the optimal solution. This optimal solution can be returned by the K-shortest paths algorithm. From Chapter 3 we can understand that DIMCRA is better than RF solving the MCLPP problem but it is also not exact.

In Chapter 4, we intensively analyze DIMCRA’s solution in order to find: under what situation DIMCRA can find a solution; under what situation DIMCRA can find an optimal solution; under what situation DIMCRA can not find a solution at all. Then we can identify reasons which cause DIMCRA’s limitations. We evaluate DIMCRA’s performance and find this performance can still be improved.

In Chapter 5, we will propose three methods to improve DIMCRA’s current performance. It is not easy to improve DIMCRA drastically, because we search for a pair of paths with the minimal total length instead of the first shortest path and the second shortest one between the source node and the destination node. This makes the research a tough issue to be solved. However, the methods proposed in this thesis do provide better solutions to solve the tough problem.

Finally, conclusions and extension of the research are presented in Chapter 6.

\(^3\) **Multiple constrained path (MCP) problem.** Given a graph \(G(V, E)\) with \(M \geq 1\) metrics per link and a constraint vector \(L\); for a source-destination pair \((s, t)\); find a path that obeys the constraint vector \(L\);

\[ w_m(P) \leq L_m \quad \text{for } m = 1, \cdots, M. \]

where \( w_m(P) = \sum_{u \rightarrow v \in P} w_m(u \rightarrow v) \), for \( m = 1, \cdots, M. \)
2.1 Introduction

In the past, much attention about link-disjoint QoS routing algorithm lied on link-disjoint paths in one dimension. In this thesis, we focus on link-disjoint paths in multiple dimensions.

A lot of such routing algorithms have been proposed, for instance, an intuitive algorithm RF (remove-find) and a simplified variant of Bhandari’s algorithm LBA (link-disjoint version of Bhandari’s algorithm) [3]. Normally, link-disjoint QoS routing algorithms about link-disjoint paths in one dimension can be extended to multiple dimensions. We can extend RF and LBA from one dimension to multiple dimensions by using SAMCRA [4] algorithm.

In this chapter we mainly introduce the RF algorithm and DIMCRA algorithm. In addition, another algorithm (K-shortest path algorithm), which does not solve the MCLPP problem, is still mentioned. It can solve specific instances of the MCLPP problem.

2.2 RF Algorithm

The RF algorithm is an intuitive method to determine a pair of link-disjoint paths between a pair of source and destination nodes.

2.2.1 RF’s steps

The RF algorithm consists of three steps:

Step 1: To find the shortest path $P_1$ from source node $s$ to destination node $t$ in the graph $G$;

Step 2: To remove all the links of that path $P_1$ from graph $G$ to create a pruned new graph $G'$;
Step 3: To find the shortest path $P_2$ from source node $s$ to destination node $t$ in the pruned graph $G'$;

RF is direct and simple, which can provide two link-disjoint paths. But it leads to that the second link-disjoint shortest path may have a significantly larger length.

### 2.2.2 RF's code

Figure 2.1 presents the code of RF algorithm.

```c
/* m: number of QoS measures
L: Constraints
N: number of nodes in a graph G
Adj: total number of adjoining nodes
Numadj: serial number of adjoining nodes in a graph G
Datadj: tensor n*n*m with QoS values
START: source node
END: destination node
P_1', P_2': the final solution (a pair of link-disjoint paths obey constraints)
*/

VAR RF (m, L, G(N, Adj, Numadj, Datadj), START, END, P_1', P_2')

Step 1. /*Try to find the shortest path1 $P_1$ in a graph G*/
if  (P_1 ← SAMCRA() obeying L in a graph G)
    continue;
else  stop;

Step 2. /*Remove all links in the path1 to create a new graph G'*/
for  i = 1 to N
    for  j = 1 to numadj [P_1[i]]
    {
        if  (P_1[i-1] = adj [P_1[i] [ j ]]) /*remove link(P_1[i-1],j)*/
            for  k = j to numadj[P_1[i]]
                adj[P_1[i]][k] = adj[P_1[i]][k+1];
                    for  q = 1 to m
                        datadj[q][ P_1[i]][k] = datadj[q][ P_1[i]][k+1];
        numadj[P_1[i]]--;
    }
    if  (P_1[i+1] = adj[P_1[i]][j]) /*remove link(P_1[i+1],j)*/
        for  k = j to numadj[P_1[i]]
```
A few words to clarify this code:

Step 1 is to compute the shortest path \( P_1 \) by using SAMCRA algorithm. If \( P_1 \) exists, then RF continues the follow steps, otherwise RF stops. Step 2 is to remove all links in \( P_1 \). Because our graphs have bi-direction (see Figure 2.2) instead of single direction, we remove links twice. Every time we will search the adjoining nodes \( P_1[i-1] \) and \( P_1[i+1] \) of node \( P_1[i] \) from 1 to N, then remove these correlation links between node \( P_1[i] \) and its adjoining nodes \( P_1[i-1] \) and \( P_1[i+1] \) from the graph. After Step 2, a new graph \( G' \) is produced. Step 3 is to find the shortest path \( P_2 \) like Step 1. Finally, in Step 4, if RF can find a pair of feasible link-disjoint paths, then return them.

```
{  
    adj[P_1[i]][k] = adj[P_1[i]][k+1];
    for  q = 1 to m
        datadj[q][P_1[i]][k] = datadj[q][P_1[i]][k+1];
    }
    numadj[P_1[i]]--;  
}

Step 3. /*Try to find the shortest path 2 P_2 in a new graph G'*/
if  (P_2 ← SAMCRA() obeying L in the new graph G')
    continue;
else  stop;

Step 4. /*Return path1 and path2*/
return P_1 and P_2;
```

Figure 2.1 RF algorithm code

Figure 2.2 A bi-direction graph

2.2.3 RF's complexity

As we know that SAMCRA has a worst-case complexity of \( O(kN \log(kN) + k^2 ME) \). Here \( k \) is the number of stored paths that SAMCRA, \( N \) is the number of nodes of a graph. And \( M \) is the number of constraints \( (M = 2, 4, 6, and 8) \) under uniformly distributed link weights. \( E \) is the number of links. For single constraint \( (M = 1 and k = 1) \), SAMCRA’s complexity reduces to the complexity of Dijkstra algorithm [9]: \( O(N * \log N + E) \).
According to the operations of Step 2, Step 2' phase of RF leads to $O(MN^3)$.

Combining these three contributions yields a total worse-case complexity of RF of $O(kN \log(kN) + k^2 ME + MN^3 + kN \log(kN) + k^2 ME)$ or $O(kN \log(kN) + k^2 ME + MN^3)$. For single constraint ($M = 1$ and $k = 1$), RF’s complexity reduces to $O(N \log N + E + N^3) = O(N^3)$.

### 2.2.4 RF’s examples

In this section we will illustrate the operations of RF. Here our graphs have bi-direction in two dimensions.

- **Example 1**

![Figure 2.3 Example 1 of RF: (a) Step 1; (b) Step 2 & Step 3](image)

Figure 2.3 Example 1 of RF: (a) Step 1; (b) Step 2 & Step 3
We will illustrate the operation of RF with the example topology showed in Figure 2.3. For the sake of simplicity, we have assigned each link with a two-dimensional weight vector.

In this example, to solve the MCLPP problem, we are asked to find a set of two link-disjoint paths from source node 1 to destination node 9, which both obey the constraints vector \( \mathbf{L} = (9.0, 9.0) \) with the minimum total length. The optimal set of two shortest link-disjoint paths (according to the K-shortest path algorithm) in this topology is \{1-4-5-8-9, 1-2-3-6-9\}, with path vectors \((2.02, 1.98)\) and \((2.27, 1.73)\) respectively. Its minimum total length is \(2.02/9.0 + 2.27/9.0 = 0.48\).

The execution of RF on this topology follows: in Step 1, the shortest path \(P_1 = 1-4-5-8-9\) is found. In Step 2, removing all the links of \(P_1\) modifies the original graph. In Step 3, the shortest path in the modified graph, found with SAMCRA, is \(P_2 = 1-2-3-6-9\), with path weight vector \((0.55, 0.45) + (0.05, 0.95) + (0.93, 0.07) + (0.74, 0.26) = (2.27, 1.73)\). In Step 4, the solution set is found. Then the RF’s solution set \{1-4-5-8-9, 1-2-3-6-9\} is returned. In this example, RF can return the optimal solution. However, in some other cases, RF perhaps cannot return the optimal solution. The next example will investigate the situation that RF cannot find the optimal solution set.

**Example 2**

Considering the graph in Figure 2.4(a), which is the same as in the previous example, we suppose some links weights are different. The constraints remain the same as it is in the previous case. In this example the optimal set of two link-disjoint multiple-constrained paths is still the set \{1-4-7-8-9, 1-2-5-6-9\} with path vector \((1.75, 2.25)\) and \((1.7, 2.3)\) respectively, and the minimum total length \(2.25/9.0 + 2.3/9.0 = 0.51\). Running RF, we obtain the results as \{1-4-5-6-9, 1-2-5-8-9\} with paths vectors \((2.14, 1.86)\) and \((1.24, 2.76)\) respectively, and the minimum total length \(2.14/9.0 + 2.76/9.0 = 0.54\). In this example, RF failed to return the optimal solution set, but RF’s solution is close to the optimal set.
This example illustrates that the RF method cannot guarantee finding a pair of link-disjoint path. In the graph shown in Figure 2.5(a), although there exist two link-disjoint paths between 1 and 4 which obey the constraints vector $L = (4.0, 4.0)$, RF cannot find the second path in Step3 (see Figure 2.5(b)). RF failed to return a solution set.
2.2.5 RF’s property

RF has two advantages and two disadvantages, which relate to the fact that RF is direct and simple. Its advantages can be summarized in the below two points:

- Because the concept of remove-find, RF can guarantee its solution that it must be two disjoint paths.

- It is simple and direct, so its cost is very low while running it.

The two disadvantages can also be summarized as:

- Because RF directly removes all links in $P_1$ in Step 2 of its operation, actually it cannot guarantee that one of the links in $P_1$ probably appears in $P_2$ of the optimal solution set. RF cannot guarantee two constrained paths. This is illustrated in Section 2.2.4.

- Although RF can find a pair of link-disjoint path, there is no mechanism in RF to guarantee the situation that two paths must be optimal. This is also illustrated in Section 2.2.4.

2.3 DIMCRA Algorithm

In this section we will introduce DIMCRA algorithm, which can produce an optimal solution for the MCLPP problem.

2.3.1 DIMCRA’s steps

Given a directed graph $G = (V, E)$, a constraint vector $L$ and a source-destination pair $(s, t)$, DIMCRA carries out the following steps:

Step 1: Using SAMCRA to search the shortest path $P_1$ obeying $L$; if $P_1$ does not exist, then stop.

Step 2: To reverse the direction of all the links on the shortest path $P_1$, and set the value of their link weights zero, $w_m(u, v) = 0$, $\forall (u \rightarrow v) \in P_1$ and $m = 1, \ldots, M$. A modified graph $G'$ is created.

Step 3: Using SAMCRA to search the shortest path $P_2$ constrained by $2L$ in the modified graph $G'$; if $P_2$ does not exist, then stop.
Step 4: To make the union of $P_1$ and $P_2$, remove from the union the $P_1$ links whose reversed links appear on $P_2$, and vice versa. Then group the remaining links into a set of two paths $\{P_1', P_2'\}$, i.e. $P_1' \cup P_2' = (P_1 \cup P_2) \setminus (P_1 \cap P_2)$.

Step 5: To check the length of each path in the set of $\{P_1', P_2'\}$. If the $P_i (1 \leq i \leq 2)$ violates the constraints, then update the modified graph $G'$ by removing the link set $P_i \setminus (P_i \cap P_1)$ from it, and go to Step 3. Otherwise stop and return the current solution set $\{P_1', P_2'\}$.

We will illustrate the operations of DIMCRA in section 2.3.5 with some concrete examples.

### 2.3.2 DIMCRA’s concept

DIMCRA is based on four concepts: (1) A new transformation to create the modified graph; (2) The concept of using “2L” as the constraints vector; (3) The concept of checking the path length; (4) The SAMCRA algorithm.

1. A new transformation to create the modified graph. Comparing with the other algorithms, DIMCRA uses a new transformation to create the modified graph. In Step 2 of DIMCRA, the shortest path links are still reversed in direction. However, the corresponding direction—reversed links are assigned zero-value link weight vectors, instead of being negative ones [4] or removing all. By this way, negative cycles cannot occur.

2. The concept of using “2L” as the constraints vector. In Step 3 of DIMCRA, the constraints checked on path $P_2$ in SAMCRA are performed with 2L as the constraints vector, otherwise a feasible solution set may not be found.

3. The concept of checking the path length. We have also added an extra step, Step 5 of DIMCRA, to the concept of checking the path length to examine whether the constraints are obeyed. With only the condition $w(P_1', P_2') \leq 2L$, DIMCRA cannot ensure both paths with constraints, i.e. $w(P_1') \leq L$ and $w(P_2') \leq L$. Hence, Step 5 of DIMCRA checks both paths in the solution set returned by Step 4. If each of them obeys the constraints, DIMCRA will return the solution set and stop. On the other hand, if none of them obey the constraints, DIMCRA is redirected to Step 3 to continue the search for a feasible set.

4. The SAMCRA algorithm. The DIMCRA in Step 1 and Step 3 both uses SAMCRA from the source node $s$ to the destination node $t$ to compute the shortest path. SAMCRA guarantees to find the shortest path within the constraints, it reveals that such a path exists. This is similar in RF.
2.3.3 DIMCRA's code

The main code of DIMCRA is given in Figure 2.6. The sub function “UNION” of Step 4 and “UPDATE” of Step 5 see Figure 2.7 and Figure 2.8.

```c
/* m: number of QoS measures
L: Constraints
N: number of nodes in a graph G
Adj: total number of adjoining nodes
Numadj: serial number of adjoining nodes in a graph G
Datadj: tensor n*n*m with QoS values
START: source node
END: destination node
P_1', P_2': the final solution (a pair of link-disjoint paths obey constraints)
*/

DIMCRA (m, L, G(N, Adj, Numadj, Datadj), START, END, P_1', P_2')

Step 1. /*Try to find the shortest path1 P_1. */
if (P_1 ← SAMCRA( ) obeying L in G)
    continue;
else stop;

Step 2. /*Create new graph G'←(reverse links in P_1 and set the value of reverse links weights in P_1 zero. */
for i = 1 to N
    for j = 1 to numadj[P_1[i] ]
        {
            if (P_1[i-1] = adj[P_1[i]][j])
                remove links in P_1;
            if (P_1[i+1] = adj[P_1[i]][j]) /* set zero*/
                {
                    for n = 1 to m
                        datadj[n][path[i]][j] = 0.0;
                }
        }

Step 3. /*Try to find the shortest path2 P_2. */
if (P_2 ← SAMCRA( ) obeying 2L in G')
    continue;
else stop;

Step 4. /*Make solution of {P_1, P_2} to get new union {P_1', P_2'}. */
UNION (N, P_1, P_2, P_1', P_2');

Step 5. /*Check the length of each path in the set {P_1', P_2'}. */
```
if \( ((\text{length}_1 \leq 1.0) \& (\text{length}_2 \leq 1.0)) \)
return \( \{ \text{P}_1', \text{P}_2' \} \);
else
{
    \textbf{UPDATE} \( (\text{G}', \text{P}_1, \text{P}_1', \text{P}_2') \); /* remove links \( \in \text{P}_1' \backslash (\text{P}_1 \cap \text{P}_2) \)*/ 
  return 3;
}

Figure 2.6 DIMCRA’s main function code

/* if \( \text{P}_1 \text{ links = the versa of } \text{P}_2 \text{ links} \) */
{ remove these links in \( \text{P}_1 \) and \( \text{P}_2 \); union the rest of links into a set of two paths solution \( \{ \text{P}_1', \text{P}_2' \} \), i.e. \( \{ \text{P}_1', \text{P}_2' \} \leftarrow (\text{P}_1 \cup \text{P}_2) \backslash (\text{P}_1 \cap \text{P}_2); \} 
else \( \{ \text{P}_1', \text{P}_2' \} \leftarrow \{ \text{P}_1 \cup \text{P}_2 \} \) */

\textbf{UNION} \( (\text{N}, \text{P}_1, \text{P}_2, \text{P}_1', \text{P}_2') \)

\textbf{Substep 1.} /*Check whether there exist the overlapping links. */ 
OverlapLinksNumber = 0;
for \( i = 1 \) to \( \text{N} \)
for \( j = \text{N} \) to 1
if \( ((\text{P}_1[i] = \text{P}_2[j]) \& (\text{P}_1[i+1] = \text{P}_2[j+1])) \)
{ overlap links \( \leftarrow (\text{P}_1[i], \text{P}_1[i+1]); \) 
    OverlapLinksNumber++;
}

\textbf{Substep 2.} /* if \( (\text{OverlapLinksNumber}) \) 
remove these links in \( \text{P}_1 \) and \( \text{P}_2 \), then make union the solution links to \( \{ \text{P}_1', \text{P}_2' \} \); 
else \{ do not do anything, return \( \text{P}_1' = \text{P}_1 \) and \( \text{P}_2' = \text{P}_2; \} */ 
while \( (\text{OverlapLinksNumber}!=0) \)
{ for \( i = 1 \) to \( \text{N} \)
for \( j = 1 \) to \( \text{N} \)
    new \( \text{P}_1' \) links \( \leftarrow \) remove overlap links \( (i,j) \) in \( \text{P}_1 \); 
    new \( \text{P}_2' \) links \( \leftarrow \) remove overlap links \( (j,i) \) in \( \text{P}_2 \);
}
while \( (\text{OverlapLinksNumber}=0) \)
{ for \( i = 1 \) to \( \text{N} \)
    \( \text{P}_1'[i] = \text{P}_1[i]; \) 
    \( \text{P}_2'[i] = \text{P}_2[i]; \) 
}
Substep 3. /*Compute the length of $P_1$ and $P_2$.*/
for i = 1 to N
{ for j = 1 to Numadj[P_1[i]]
  if (P_1[i+1] = Adj(P_1[i])[j])
  { for k = 1 to m
    Weight_1[k] = Weight_1[k] + Datadj[k][P_1[i]][j];
  }
  for j = 1 to Numadj[P_1[i]]
  if (P_2[i+1] = Adj(P_2[i])[j])
  { for k = 1 to m
    Weight_2[k] = Weight_2[k] + Datadj[k][P_2[i]][j];
  }
}
for q = 1 to m
Length_1 = MAX(Weight_1[q])/L;
Length_2 = MAX(Weight_2[q])/L;
Substep 4. return { P_1', P_2', length1, length2};

Figure 2.7 UNION function code

/* If each of path $\{P_1', P_2'\}$ violates the constraints $L$, then update $G'$ by removing the links $\in P'_i \setminus (P_i \cap P_i)$.*/

UPDATE (G', P_1, P_1', P_2', length1, length2)
{ case 1: /*Remove links $P_1' \setminus (P_1' \cap P_i)$.*/
  for i = 1 to N
  for j = 1 to N
  if (P_1' != P_i)
    movelinks[i][j] = P_1'[i][i+1];
  for i = 1 to N
  for j = 1 to N
    remove movelinks[i][j];
  case 2: remove links $P_2' \setminus (P_2' \cap P_i)$
  case 3: remove links $P_1' \setminus (P_1' \cap P_i)$ and $P_2' \setminus (P_2' \cap P_i)$
}

Figure 2.8 UPDATE function code

We clarify DIMCRA’s codes in some words.
Like RF’s Step 1, DIMCRA’s Step 1 is to compute the shortest path $P_1$ in $G$. If $P_1$ exists then continues otherwise DIMCRA stops.

DIMCRA’s Step 2 is to reverse all links in $P_1$ and then set these links’ value zero instead of removing these links directly like RF. Here, the operation of removing links is same as that of RF. After this step, a modified new graph $G'$ is created.

As Step 3 of RF, DIMCRA’s Step 3 is to compute the shortest path $P_2$ in $G'$.

In “UNION” function of DIMCRA’s Step 4, we take four substeps to complete this function. In Substep 1, we check whether there exist some overlapping links in $P_1$ and $P_2$. Because the maximum number of $P_1$ or $P_2$ nodes is $N$, we set two loops from 1 to $N$ to check it. In Substep 2, if there exist overlapping links, then remove all links in $P_1$ and $P_2$ to group the remaining links to two new paths $P_1'$ and $P_2'$, otherwise the result of Sunbstep 2 is direct from $P_1$ and $P_2$. Here the operation of removing links is same as that of RF. In Substep 3, we compute the lengths of $P_1'$ and $P_2'$. In Substep 4, return $P_1'$ and $P_2'$ are returned.

In DIMCRA’s Step 5, we check the two new paths $P_1'$ and $P_2'$, if they obey constraints, then return DIMCRA’s solution, otherwise UPDATE graph $G'$. In “UPDATE” function, normally there three cases: $P_1'$ does not meet constraints but $P_2'$ meets the constraints; $P_2'$ does not meet constraints but $P_1$ meets the constraints; both $P_1$ and $P_2$ do not meet constraints. Whatever case, at first we need to know what links will be removed, and then remove them. Here, we state that the removed links appear on $P_1'$ or $P_2'$ instead of appearing on $P_1$. After “UPDATE” function, we obtain a modified new graph $G'$, then return Step 3 of main function to continue DIMCRA’s operations until the end.

### 2.3.4 DIMCRA’s complexity

According to the operations of DIMCRA, we can compute the time complexity of DIMCRA.

- **Step 1**

  The time complexity of SAMCRA is $O(kN \log(kN) + k^2 ME)$.

- **Step 2**

  Because the time complexity of removing links is $O(MN)$, the time complexity of this step is $O(MN^3)$.

- **Step 3**

  Like Step 1, the time complexity of this step is $O(kN \log(kN) + k^2 ME)$. 
• Step 4

The time complexity of Substep 1 is $O(MN^3)$. The time complexity of Substep 2 is $O(qMN^3 + N) = O(qMN^3)$, here $q(q \geq 1)$ is the number of overlapping links between $P_1$ and $P_2$. Normally, $q \leq (N-1)$ therefore the time complexity of this step is $O(qMN^3) \leq O(MN^4)$. The time complexity of Substep 3 is $O(MN^2)$. The time complexity of Substep 4 is $O(1)$. Adding all time complexity of four substeps is

$$O(MN^3 + MN^4 + MN^2 + 1) = O(MN^4).$$

• Step 5

The time complexity of Step 5 depends on “UPDATE” function. The time complexity of “UPDATE” function is $O(MN^3)$. Therefore the time complexity of Step 5 is $O(MN^3)$.

Combining all five steps’ contributions yields a total worse-case complexity of DIMCRA of $O(kN \log(kN) + k^2ME + MN^3 + kN \log(kN) + k^2ME + MN^4 + MN^3)$ or $O(kN \log(kN) + k^2ME + MN^4)$. For single constraint ($M = 1$ and $k = 1$), DIMCRA’s complexity reduces to $O(N \log N + E + N^4) = O(N^4)$.

Comparing with the time complexity of RF, DIMCRA has larger time complexity than RF.

### 2.3.5 DIMCRA’s examples

We are asked to find a set of two link-disjoint paths between two nodes with the minimal total length. In this section, we will employ some examples to explain the operations of DIMCRA. Like RF’s examples, here our graphs also have bi-directions.

• Example 1

Considering the example graph in Figure 2.9(a), we are asked to find two link-disjoint paths from node 1 to node 9 with the minimal total length and the constrains vector $L = (9.0, 9.0)$. For the sake of simplicity, we have assigned each link a two-dimensional weight vector, but it is still possible to use an $m$-dimensional weight vector. In Step 1, the shortest path $P_1 = 1-3-6-2-9$ with the path vector $(0.28, 0.48) + (0.11, 0.03) + (0.14, 0.01) + (0.45, 0.68) = (0.98, 1.2)$ (see Figure 2.9(a)) is found by SAMCRA. In Step 2, each $P_1$ link is reversed and is assigned with zero link weights. In Step 3, the shortest path in the modified graph $G'$ is found as $P_2 = 1-7-5-6-3-9$, with the path vector $(0.37, 0.27) + (0.60,$
0.91) + (0.03, 0.06) + (0.14, 0.01) + (0.45, 0.68) = (1.59, 1.93), as shown with bold lines in Figure 2.9 (c). In Step 4 only for the $P_1$ link 3→6, its reversed link 6→3 appears on $P_2$ and vice versa. Thus these two links are removed from the union of $P_1$ and $P_2$, and the remaining links are grouped into a set of two paths $\{P'_1, P'_2\} = \{1-3-9, 1-7-5-6-2-9\}$ with the path vectors $(1.13, 1.40)$ and $(1.59, 1.93)$, shown in Figure 2.9 (d). In Step 5, the constraints are checked on both paths $P'_1$ and $P'_2$. As each of them obeys the constraints, DIMCRA stops.

In this example, the optimal solution set of $\{1-3-9, 1-7-5-6-2-9\}$ is returned by DIMCRA. RF would not have returned the solution set.
In this example, DIMCRA can find a pair of link-disjoint paths but they are not optimal. Considering the graph in Figure 2.10(a), the constraints remain the same as in Example 1. In Step 1, the shortest path is found as $P_1 = 1-2-5-9$ with vector $(1.88, 1.75)$, shown in Figure 2.10(a). The same operation in Step 2 as the example 1 is shown in Figure 2.10(b). In Step 3, the shortest path in the modified graph is found as $P_2 = 1-8-6-9$ with path vector $(1.78, 2.09)$, shown in Figure 2.10(c). In Step 4, as for each $P_i$ link, its reversed link does not appear on $P_2$, or vice versa. The solution set $\{P_1', P_2'\}$ is constructed as $\{1-2-5-9, 1-8-6-9\}$, exactly the same as $P_1$ and $P_2$ themselves. The total length $\{P_1', P_2'\}$ is $1.88/9 + 2.09/9 = 0.44$.

In this example, DIMCRA failed to return the optimal solution set because the optimal set of two link-disjoint multiple-constrained paths is the set of $\{1-2-6-9,
with path vectors (1.93, 1.7) and (1.77, 1.98), and the minimum total length of $1.93/9 + 1.98/9 = 0.43$. But DIMCRA’s solution set is close to the optimal one, and both paths obey the constraints. RF returns the same solution.

(a) Step 1

(b) Step 2

(c) Step 3
We again consider Example 2, with different constrains (2.0, 2.0). Running DICMRA, we obtain the same results as in Example 2 (for Step 1-4). But in Step 5, when the constraints check is made on each path in the solution set \( P_1', P_2' \) = \{1-2-5-9, 1-8-6-9\}, the longer \( P_2' \) with path vector (1.78, 2.09) does not obey the constraints (2.0, 2.0). This means that the currently built solution set is not feasible. The links that only appear on \( P_2' = 1-8-6-9 \), i.e. link 1 \( \rightarrow \) 8, 8 \( \rightarrow \) 6 and 6 \( \rightarrow \) 9, are removed form the modified graphs shown in Figure 2.10(b). The updated and modified graph is shown in Figure 2.11(a). DIMCRA is redirected to Step 3. In Step 3, the shortest path in the updated modified graph is not found as \( P_2' \). DIMCRA thus stops. DIMCRA failed to return the solution set. Nevertheless, in this example the optimal set of two link-disjoint multiple-constrained paths is still the same set of \{1-2-6-9, 1-8-6-5-9\} with the minimal total length 0.43 as the same as in Example 2. RF has failed to return a solution.

In Step 3 the constrains are set to 2L. With these modified constraints, if the shortest path \( P_2 \) in the modified graph violates the constraints L, but obeys 2L, it can be returned by SAMCRA in Step3. Sometimes a feasible set related to such kind of \( P_2 \) cannot be ignored. However, if a path \( P_2 \) does not exist in the updated modified graph, DIMCRA will stop, as illustrated in this example. With the constraints checked on each path in the solution set, Step 5 guarantees that DIMCRA returns a feasible set of two link-disjoint paths under multiple-constraints, as illustrated in Example 3.

In Example 2, both DIMCRA and RF can find a pair of link-disjoint paths but their solutions probably are not optimal. In this case, DIMCRA is still working, but RF cannot find the solution.
Example 4

The above Example 1, 2 and 3, all are about DIMCRA, which can find one or two link-disjoint paths. But sometimes DIMCRA cannot return a path even if the solution exists there, as we have illustrated in Figure 2.12.
Considering the graph in Figure 2.12(a), we are asked to find a set of two link-disjoint paths from node 1 to node 5 with the minimal total length, each are covered by the constraints $L = (1.0, 1.0)$. In this example, the optimal solution set of two links-disjoint under multiple-constraints paths is the set \{1-2-3-5, 1-3-4-5\} with paths vectors $(1.0, 0.3)$ and $(0.8, 0.5)$. Running DIMCRA, in Step 1 the shortest path is found
as \( P_1 = 1-2-3-4-5 \) with path vector \((0.7, 0.6)\). In Step 3 we find the shortest path by SAMCRA is identified as \( P_2 = 1-3-5 \) with path vector \((1.1, 0.2)\). Then in Step 4 the solution set \( \{P_1', P_2'\} \) is constructed as \( \{1-2-3-4-5, 1-3-5\} \). In Step 5 the constraints check is executed on both paths \( \{P_1', P_2'\} \). But the longer path \( P_2' = 1-3-5 \) with path vector \((1.1, 0.2)\) does not obey the constraints. So we have to update the modified graph shown in Figure 2.12(e) by removing links that appear on \( P_2' \), i.e. 1→3 and 3→5 from the modified graph shown in Figure 2.12(b) and return to Step 3. Then again in Step 3 the shortest path is not found as \( P_2 \). DIMCRA stops. In this example, DIMCRA failed to return the solution set, so did RF.

### 2.3.6 DIMCRA’s property

As shown in DIMCRA’s operation in previous examples, comparing with the RF algorithm, DIMCRA has two advantages:

- Firstly, by setting the reversed path \( P_1 \) links as zero link weights guarantees that there is no loop while finding the shortest path. This point has been clarified clearly in Reference [1].

- Secondly, by reversing the shortest path \( P_1 \) to find another shortest path \( P_2 \) in the modified graph, we reach the solution set based on these two shortest paths, which guarantee two paths in the solution set are disjoint.

From Example 2, 3 and 4, we know that sometimes DIMCRA can not find the solution set or can find a solution that is not optimal. That is to say, DIMCRA also has a disadvantage.

- DIMCRA does not always find a set of feasible link-disjoint paths and can not guarantee that its solution is optimal. Hence DIMCRA can be optimized further.

### 2.4 K-shortest Paths Algorithm

The K-shortest paths algorithm is not for solving the MCLPP problem but for the MCP problem. It is similar to Dijkstra’s algorithm [4]. Running the K-shortest paths algorithm, we can compute the shortest, the second shortest, the third shortest, \ldots, up to the K-th shortest path together with the corresponding path weight from the source node \( s \) to destination node \( t \). It is possible to get less than K paths between a pair of nodes \( (s,t) \), but not more.

In the project of this thesis, we will employ the K-shortest paths algorithm to obtain K-shortest paths. And later we will compare them to compute a pair of link-disjoint paths with the minimal total length. It is actually the optimal solution set of a pair of two link-disjoint paths. Although we can get a pair of link-disjoint paths with the minimal total length by this algorithm, it costs much more time. So it is necessary to just introduce
this algorithm to evaluate DIMCRA’s performance. In practice, we will not use this algorithm as a link-disjoint multiple constraints routing algorithm.
In Chapter 2 we have reviewed some link-disjoint routing algorithms. In this chapter we will evaluate the performance of these algorithms for solving the MCLPP problem, especially for DIMCRA algorithm. It is based on simulation results and complexity analysis.

3.1 Motivation and Problem Definition

In order to evaluate the performance of DIMCRA for solving the MCLPP problem, we will make some simulations to compare DIMCRA with RF, and with the K-shortest paths algorithm. Through evaluations we can examine which algorithm is better for solving the MCLPP problem. And through simulation, it may help solving the difficult task of selecting the proper algorithm for a QoS-capable network.

In this chapter we will make:

I. comparison of DIMCRA with RF in finding a solution based on simulation results and time complexity.
II. comparison of DIMCRA with RF in finding an optimal solution based on simulation results.

Based on these comparative results, we will identify property of DIMCRA and RF for solving the MCLPP problem.

3.2 Results and Analysis of Simulation for the Random Graphs

In this section the simulation results will be based on the class of random graphs [2]. We choose the class of random graphs is the type \( G_p(N) \) [5].

Three main parameters affect a random graph. They are number of nodes \( N \), link-density \( p \) and correlation \( r \). The link density \( p \) is probability that there exists a link between two nodes in a graph. On average \( G_p(N) \) therefore contains \( pN(N-1)/2 \) links. If \( p = 1.0 \), we have the complete graph with the maximum number of links \( N(N-1)/2 \).
CHAPTER 3  DIMCRA’s PERFORMANCE I

The parameter \( r \) is the correlation between two links weights. In this thesis, we choose uniformly distributed link weights in range \((0, 1.0]\). When \( r \) is equal to 1.0, the two links weights \( w_1 \) and \( w_2 \) are same; when \( r \) is close to 1.0, \( w_2 \) is close to \( w_1 \); when \( r \) is equal to –1.0, \( w_1 + w_2 = 1.0 \) (the maximum range of the two links weights); when \( r \) is close to –1.0, \( w_2 \) is close to \( (1-w_1) \); when \( r \) is equal to 0.0, \( w_1 \) and \( w_2 \) are distributed freedom in range \((0,1.0]\).

3.2.1 Comparison of DIMCRA with RF in finding a solution

We will evaluate the ability of DIMCRA and RF in finding a set of two link-disjoint paths under multiple constraints. We will further compare the success rate and execution time of DIMCRA, with those of RF in finding a solution set for the random graphs. The success rate of an algorithm is defined as the number of times that an algorithm returned a feasible pair of link-disjoint paths (a solution) divided by the total number of iterations. The execution time of an algorithm is defined as the running time of the algorithm (over all iterations) divided by the running time of an algorithm. It has to be stated here that the solution set returned by DIMCRA and RF may not be optimal.

3.2.1.1 Simulation description

Because a random graph depends on three parameters: \( N, p, r \), we set different parameters \(( p \in (0,1.0], r \in (-1.0,1.0], N )\) to examine the performance of DIMCRA and RF. We have made simulations to examine the performance of the algorithms as a function of the number of constraints \( m(m=2) \) under uniformly distributed link weights. The procedure was repeated \( 10^4 \) times.

3.2.1.2 \( p \) (link density) is changed

Figure 3.1(a) gives the success rate of DIMCRA and RF for the class of the random graphs when \( p \) is changed, with \( m=2 \) and \( N=100 \). The algorithms DIMCRA and RF always give a success rate that is close to 1. Especially when \( p \) is larger than 0.2, the algorithms DIMCRA and RF can always give the success rate = 1. And when \( p \) is smaller than 0.2, their success rate decreases a little, which are not beyond 0.06% comparing with the success rate = 1. In these simulated cases, DIMCRA and RF have same success rate for the random graphs.

Figure 3.1(b) gives the execution time of DIMCRA and RF. DIMCRA always takes more time than RF. And their execution time is almost linear increased while \( p \) increased.

Our simulations revealed that when \( p \) is changed, DIMCRA and RF could find a solution using the same success rate and RF save more time than DIMCRA.
(a) Success rate
Figure 3.1 For $m = 2$, $N = 100$, the comparison of success rate and execution time between DIMCRA and RF in finding a set of two link-disjoint paths (solution) for the class of random graphs as a function of the link density $p$.

3.2.1.3 $r$ (correlation between link weights) is changed

In this section we discuss the success rate and execution time for the random graph when $r$ is changed. We choose $p = 0.2$ for the random graphs.

The success rate of DIMCRA and RF are not equal to 1, but very close to 1. This result is consistent with the result presented in Figure 3.1.
(a) Success rate  

(b) Execution time

Figure 3.2 For $m = 2$, $N = 100$, the comparison of success rate and execution time between DIMCRA and RF in finding a solution for the class of random graphs as a function of $r$

For the execution time, like as Section 3.2.1.2, when $r$ changed the execution time of DIMCRA always take more time than RF. They have similar change.

### 3.2.1.4 $N$ (number of nodes) is changed

Like in Section 3.2.1.2 and Section 3.2.1.3, in this section we will discuss $N$. For random graphs, from Figure 3.3(a) we have seen that when $p$ is larger than 0.2, whatever $N$ is changed, the success rate of DIMCRA and RF is equal to 1. But when $p$ is smaller than 0.2 and $N$ is smaller than 49, the success rate of DIMCRA and RF decreased much more. However, the success rate of DIMCRA is still more than that of RF a little. It is hard to see clear from Figure 3.3(a) because the difference is very little. In addition, the same thing happens to the execution time of DIMCRA and RF as in Section 3.2.1.2 and Section 3.2.1.3 irrespective of $r$. DIMCRA and RF have similar change in execution time.
Figure 3.3 For $m = 2$, the comparison of success rate and execution time between DIMCRA and RF in finding a solution for the class of random graphs as a function of $N$

3.2.1.5 Summary

From the simulation study, we know that for the random graphs, DIMCRA and RF usually have the similar success rate in finding a solution. When $p$ is larger than 0.2, DIMCRA and RF can always find a solution. When $p$ is smaller than 0.2, they decrease a little. Parameter $r$ hardly affects the success rates of DIMCRA and RF. Parameter $N$ also does not affect too much the success rate of DIMCRA and RF. Nevertheless, when $p$ is smaller than 0.2 and $N$ is smaller than 49, the success rates of DIMCRA and RF decrease sharply, but the success rate of DIMCRA is still more than RF for the random graphs a
little. In addition, the execution time of DIMCRA always larger than RF.

In conclusion, DIMCRA is more efficient than RF in finding a solution for the random graphs because DIMCRA has more success rate than RF a little.

3.2.2 Comparison of DIMCRA with RF in finding an optimal solution

In Section 3.2.1 we have evaluated the performance of DIMCRA and RF, by comparing the solution returned by DIMCRA and RF, but the solution may not be the optimal one. In this section we will go on making some simulations to evaluate the performance of DIMCRA and RF. We will compare the solution returned by DCMRA with the solution returned by RF, to see which one is more close to the optimal solution. The K-shortest paths algorithm can compute and tell the optimal solution set. We want to understand how different DCMRA’s and RF’s solutions are from the optimal solution set. So the lengths of DIMCRA’s and RF’s solutions will be discussed. In addition, the success rates of DIMCRA and RF in finding an optimal solution are evaluated.

3.2.2.1 Simulation description

As in Section 3.2.1, same simulations are repeated 10^4 times.

3.2.2.2 Success rate

In order to measure the performance of DIMCRA and RF in finding an optimal solution roundly, we set different \( p, r \) and \( N \) respectively to observer the result (see Table 3.1, 3.2 and 3.3). We usually discuss three situations: 1. DIMCRA and RF can find an optimal solution; 2. DIMCRA and RF can find a solution but not optimal; 3. DIMCRA and RF cannot find a solution if it exists. Here, the success rate of an algorithm is defined as the number of times that an algorithm returned respectively three types solution (an optimal solution; a solution but not optimal; an unfound solution if it exists) divided by the total number of iterations. It is stated that the precondition of discussion above is that there exists a solution. If there does not exist the solution, our discussing is meaningless.

From Table 3.1, 3.2 and 3.3, we can see the same result. The result is the success rate of DIMCRA in finding an optimal solution is better than that of RF a little, and DIMCRA can always find a solution, if it exists. However RF sometimes can not find a solution even if this solution exists. In a word, for random graphs, DCMRA is better than RF in finding an optimal solution for the random graphs.
### DIMCRA

<table>
<thead>
<tr>
<th>$p$</th>
<th>Success rate (%)</th>
<th>The ratio of Solutions</th>
<th>Situation 1</th>
<th>Situation 2</th>
<th>Situation 3</th>
<th>The ratio of Solutions do not exist</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>16.68</td>
<td>16.46</td>
<td>0.22</td>
<td>0.00</td>
<td>83.32</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>91.80</td>
<td>90.40</td>
<td>1.40</td>
<td>0.00</td>
<td>8.20</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>100.00</td>
<td>98.80</td>
<td>0.20</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>100.00</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

### RF

<table>
<thead>
<tr>
<th>$p$</th>
<th>Success rate (%)</th>
<th>The ratio of Solutions</th>
<th>Situation 1</th>
<th>Situation 2</th>
<th>Situation 3</th>
<th>The ratio of Solutions do not exist</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>16.68</td>
<td>16.39</td>
<td>0.21</td>
<td>0.08</td>
<td>83.32</td>
<td></td>
</tr>
<tr>
<td>0.5</td>
<td>91.80</td>
<td>90.30</td>
<td>1.50</td>
<td>0.00</td>
<td>8.20</td>
<td></td>
</tr>
<tr>
<td>0.8</td>
<td>100.00</td>
<td>98.60</td>
<td>0.40</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>100.00</td>
<td>100.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.1 For $m = 2, N = 9$ and $r = -1.0$, the *success rate* comparison of DIMCRA and RF in finding an optimal solution for the class of random graphs as a function of $p$.

### DIMCRA

<table>
<thead>
<tr>
<th>$r$</th>
<th>Success rate (%)</th>
<th>The ratio of Solutions</th>
<th>Situation 1</th>
<th>Situation 2</th>
<th>Situation 3</th>
<th>The ratio of Solutions do not exist</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.0</td>
<td>16.68</td>
<td>16.46</td>
<td>0.22</td>
<td>0.00</td>
<td>83.32</td>
<td></td>
</tr>
<tr>
<td>-0.4</td>
<td>16.72</td>
<td>16.43</td>
<td>0.29</td>
<td>0.00</td>
<td>83.28</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>16.72</td>
<td>16.42</td>
<td>0.30</td>
<td>0.00</td>
<td>83.28</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>16.72</td>
<td>16.33</td>
<td>0.39</td>
<td>0.00</td>
<td>83.28</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>16.68</td>
<td>16.64</td>
<td>0.04</td>
<td>0.00</td>
<td>83.32</td>
<td></td>
</tr>
</tbody>
</table>

### RF

<table>
<thead>
<tr>
<th>$r$</th>
<th>Success rate (%)</th>
<th>The ratio of Solutions</th>
<th>Situation 1</th>
<th>Situation 2</th>
<th>Situation 3</th>
<th>The ratio of Solutions do not exist</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1.0</td>
<td>16.68</td>
<td>16.39</td>
<td>0.21</td>
<td>0.08</td>
<td>83.32</td>
<td></td>
</tr>
<tr>
<td>-0.4</td>
<td>16.72</td>
<td>16.25</td>
<td>0.35</td>
<td>0.12</td>
<td>83.28</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>16.72</td>
<td>16.27</td>
<td>0.37</td>
<td>0.08</td>
<td>83.28</td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>16.72</td>
<td>16.15</td>
<td>0.48</td>
<td>0.09</td>
<td>83.28</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>16.68</td>
<td>16.38</td>
<td>0.17</td>
<td>0.13</td>
<td>83.32</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.2 For $m = 2, p = 0.2$ and $N = 9$, the *success rate* comparison of DIMCRA and RF in finding an optimal solution for the class of random graphs as a function of $r$. 


### Table 3.3

<table>
<thead>
<tr>
<th>N</th>
<th>Success rate (%)</th>
<th>The ratio of Solutions exist</th>
<th>Situation 1</th>
<th>Situation 2</th>
<th>Situation 3</th>
<th>The ratio of Solutions do not exist</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.86</td>
<td>1.86</td>
<td>0.00</td>
<td>0.00</td>
<td>98.14</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>16.68</td>
<td>16.46</td>
<td>0.22</td>
<td>0.00</td>
<td>83.32</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>65.00</td>
<td>61.20</td>
<td>3.80</td>
<td>0.00</td>
<td>35.00</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>N</th>
<th>Success rate (%)</th>
<th>The ratio of Solutions exist</th>
<th>Situation 1</th>
<th>Situation 2</th>
<th>Situation 3</th>
<th>The ratio of Solutions do not exist</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1.86</td>
<td>1.86</td>
<td>0.00</td>
<td>1.86</td>
<td>98.14</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>16.68</td>
<td>16.39</td>
<td>0.21</td>
<td>16.39</td>
<td>83.32</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>65.00</td>
<td>59.20</td>
<td>5.5</td>
<td>59.20</td>
<td>35.00</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.3 For $m = 2$, $p = 0.2$ and $r = -1.0$, the success rate comparison of DIMCRA and RF in finding an optimal solution for the class of random graphs as a function of $N$.

#### 3.2.2.3 Length

The performance measure is based on the average lengths of both algorithms,

$$E[l_{(p)}] = \frac{1}{iter_{(DIMCRA)}} \sum_{i=1}^{iter} l_{i}^{(DIMCRA(i))}$$

$$E[l_{(p)}] = \frac{1}{iter_{(RF)}} \sum_{i=1}^{iter} l_{i}^{(RF(i))}$$

$$E[l_{(p)}] = \frac{1}{iter_{(OPTIMAL)}} \sum_{i=1}^{iter} l_{i}^{(OPTIMAL(i))}$$

where $iter$ refers to the total number of examined random graphs when an algorithm can find a solution, $l_{i}^{(DIMCRA(i))}$, $l_{i}^{(RF(i))}$ and $l_{i}^{(OPTIMAL(i))}$ refer to the length of solution computed by DICMRA, RF and the K-shortest paths algorithm, respectively. Figure 3.4 gives the result for different $r$ after running $iter = 10^4$ times.

From Figure 3.4, we observe that the average of $l_{i}^{(DIMCRA(i))}$ is always below that of $l_{i}^{(RF(i))}$, therefore DIMCRA’s solution is closer to the optimal solution than RF’s solution to the optimal solution.
Figure 3.4 For \( m = 2, N = 9 \), the average length of DIMCRA’s solutions, RF’s solutions and the optimal solutions for the random graphs

In order to know the average difference length of DIMCRA’s solution and RF’s solution with the optimal solution, we use the following formulations.

\[
\frac{\Delta l}{l_{(\text{OPTIMAL})}} = \frac{l_{(\text{DIMCRA})} - l_{(\text{OPTIMAL})}}{l_{(\text{OPTIMAL})}} = c \Rightarrow l_{(\text{DIMCRA})} = (1 + c)l_{(\text{OPTIMAL})}
\]

here \( c \) is a constant. According it, we plotted Figure 3.5. It gives the average difference length of DIMCRA’s solutions and RF’s solution with the optimal solutions for different \( r \). For instance, when \( r = 0.0 \), we can observe that

\[
l_{(\text{DIMCRA}(r=0.0))} = (1 + 0.0811)l_{(\text{OPTIMAL})}
\]

\[
l_{(\text{RF}(r=0.0))} = (1 + 0.1008)l_{(\text{OPTIMAL})}
\]

Obviously, the length of DIMCRA is more close to the optimal solution than that of RF.
3.2.2.4 Summary

Generally, for the random graphs the performance of DIMCRA is better than that of RF in finding an optimal solution for solving the MCLPP problem, but the difference is not so sharp. Once RF can find a solution, DIMCRA can certainly find its solution. But if RF cannot find a solution, DIMCRA can still probably find a solution.

3.3 Results and Analysis of Simulation for the Lattice Graphs

The simulation is similar to Section 3.2, except that we select square lattice graphs. The class of square lattices is extremely regular. A square lattice graph contains \(2(N - \sqrt{N})\) links, \(N\) is the number of nodes in this graph. The link weight is uniformly distributed in range \((0, 1.0]\). In the following parts, the result will also be analyzed.

3.3.1 Comparison of DIMCRA with RF in finding a solution

In this section the performance measure is still based on the success rate and execution time of DIMCRA and RF.

3.3.1.1 Simulation description

Because the parameter \(p\) does affect the lattice graphs, we observe success rate and execution time change as a function of \(r\) and \(N\) with \(m = 2\). This procedure was repeated \(10^4\) times.
3.3.1.2 $r$ (correlation between link weights) is changed

From Figure 3.6, we observe that for the lattice graphs, DIMCRA can always find a solution irrespective of $r$, but the success rate of RF decreases when $r$ is larger than -0.3. In addition, the execution time of DIMCRA and RF decrease when $r$ is becoming larger.

Figure 3.6 For $m = 2$, $N = 49$, the comparison of success rate and execution time between DIMCRA and RF in finding a solution for the class of lattice graphs as a function of $r$. 
3.3.1.3 \( N \) (number of nodes) is changed

Irrespective of \( N \) is changed, DIMCRA always find a solution (see Figure 3.7(a)), but RF can not do it. The reason is that in a square lattice graph, when we run the RF algorithm to compute a pair of link-disjoint paths, if the first shortest path \( P_1 \) is following the longest-hop path from source node to destination node, then the second shortest path \( P_2 \) certainly can not be found. But DIMCRA does not follow that. Therefore, RF can not always find a solution. In addition, because the lattice graph is very regular, DIMCRA and RF almost have similar running time (see Figure 3.7 (b)).

Figure 3.7 For \( m = 2 \), the comparison of success rate and execution time between DIMCRA and RF in finding a solution for the class of lattice graphs as a function of \( N \)
3.3.1.4 Summary

From the simulation study, we know that for the lattices, DIMCRA can always find a solution even if $r$ and $N$ are changed. But for RF, sometimes it can not find a solution.

The difference in the success rate of the heuristics under multiple constraints is significant. Because RF directly deletes the first path in Step2, there are some links that will still be needed in the final solution. But DIMCRA can avoid this situation.

In conclusion, DIMCRA is better than RF in finding a solution for the lattice graphs.

3.3.2 Comparison of DIMCRA with RF in finding an optimal solution

We make similar evaluation as Section 3.2.2 for the lattices graphs. For each square lattice graph, there certainly exists an optimal solution.

3.3.2.1 Simulation description

Like in Section 3.2.2, we want to see which one (DIMCRA or RF) is closer to the optimal solution for the lattices graphs. The success rates of DIMCRA and RF in finding an optimal solution and the length of DIMCRA’s solution and RF’s solution are evaluated. This procedure was repeated $10^4$ times.

3.3.2.2 Success rate

In order to measure the performance of DIMCRA and RF in finding an optimal solution roundly for the lattice graphs, we still set different $r$ and $N$ respectively to observer the simulations result in three situations (1. DIMCRA and RF can find an optimal solution; 2. DIMCRA and RF can find a solution but not optimal; 3. DIMCRA and RF cannot find a solution.).

From Figure 3.8 we know that the success rate of DIMCRA in finding an optimal solution is between 66.91% and 79.48%, and that of RF is between 66.34% and 66.85%. Generally DIMCRA is better than RF in finding an optimal solution. If there exists a solution, DIMCRA surely can find it. However RF can not guarantee it. In addition, when $r$ is equal 1.0, DIMCRA’s performance is the best.

From Figure 3.9 we know that the success rate of DIMCRA decreases while $N$ increases. However, DIMCRA is always better than RF, especially when $N$ is larger. In addition, DIMCRA always can find a solution if the solution exists.
CHAPTER 3  DIMCRA's PERFORMANCE

(a) DIMCRA in three situations

(b) RF in three situations
Figure 3.8 For \( m = 2, N = 16 \), the success rate comparison of DIMCRA and RF in finding an optimal solution for the class of lattice graphs as a function of \( r \).

(a) DIMCRA in three situations

(c) DIMCRA and RF in one situation: the optimal solution
 CHAPTER 3  DIMCRA’s PERFORMANCE I

(b) RF in three situations

Figure 3.9 For $m = 2$, the success rate comparison of DIMCRA and RF in finding an optimal solution for the class of lattice graphs as a function of $N$.

3.3.2.3 Length

Figure 3.10 and 3.11 respectively give the average length $l_{\text{DIMCRA}(i)}$, $l_{\text{RF}(i)}$ and...
Like Section 3.2.2.3, for the lattice graphs DIMCRA’s solutions are more close to the optimal solutions than that of RF.

Figure 3.10 For $m = 2$, $N = 16$, the average length of DIMCRA’s solution and RF’s solution with the optimal solution for the lattice graphs

Figure 3.11 For $m = 2$, $N = 16$, the average of length difference of DIMCRA’s solution and the optimal solution for the lattice graphs

3.3.2.4 Summary
The results from above simulations again validate that the performance of DIMCRA is better than that of RF in any case for solving the MCLPP problem. Its success rate is between 66.91% and 79.48%. Once RF can find a solution, DIMCRA can certainly find its solution too. But if RF cannot find a solution, DIMCRA maybe still find a solution.

### 3.4 Overview

In this chapter, we make a mass of simulations to measure the performance of DIMCRA and RF roundly. In order to identify their properties clearly, we sum up in Table 3.4 and 3.5.

In addition, from the above simulations, we can find that the parameter $p$ and $r$ affect the performance of DIMCRA.

In a conclusion, DIMCRA’s performance is better than RF, especially for the lattices graphs. However DIMCRA is still not exact. Therefore, on the next step, we will analyze the reasons causing DIMCRA’s disadvantage.

<table>
<thead>
<tr>
<th>For the Random Graphs</th>
<th>DIMCRA</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success rate</td>
<td>higher than RF a little</td>
<td>lower than DIMCRA a little</td>
</tr>
<tr>
<td>Execution time</td>
<td>more than RF</td>
<td>less than DIMCRA</td>
</tr>
<tr>
<td>Performance</td>
<td>DIMCRA is better than RF.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For the Lattice Graphs</th>
<th>DIMCRA</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success rate</td>
<td>higher than RF</td>
<td>lower than DIMCRA</td>
</tr>
<tr>
<td>Execution time</td>
<td>More than RF a little</td>
<td>less than DIMCRA a little</td>
</tr>
<tr>
<td>Performance</td>
<td>DIMCRA is better than RF.</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4 Overview of DIMCRA’s and RF’s performance in finding a solution

<table>
<thead>
<tr>
<th>For the Random Graphs</th>
<th>DIMCRA</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success rate</td>
<td>higher than RF a little</td>
<td>lower than DIMCRA a little</td>
</tr>
<tr>
<td>Length difference</td>
<td>has a little difference with the optimal solution</td>
<td>has much difference with the optimal solution</td>
</tr>
<tr>
<td>Performance</td>
<td>DIMCRA is better than RF.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>For the Lattice Graphs</th>
<th>DIMCRA</th>
<th>RF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success rate</td>
<td>higher than RF</td>
<td>lower than RF</td>
</tr>
<tr>
<td>Length difference</td>
<td>has a little difference with the optimal solution</td>
<td>has much difference with the optimal solution</td>
</tr>
<tr>
<td>performance</td>
<td>DIMCRA is better than RF.</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.5 Overview of DIMCRA’s and RF’s performance in finding an optimal solution
Although DIMCRA’s performance is better than RF for solving the MCLPP problem, it is not exact. In this chapter, we will analyze what reasons lead to DIMCRA’s disadvantage.

4.1 Motivation and Problem Definition

Whether DIMCRA can find an optimal solution depends on the components of the optimal solution. In this chapter, we will analyze the solution returned by DIMCRA under uniformly distributed link weights for the two-dimensional lattices. And then we will identify the reasons causing DIMCRA’s disadvantage.

4.2 DIMCRA’s Solutions

Generally, we divide the DIMCRA’s solution into three classes. To make it clear, here DIMCRA’s solution is namely the set of two paths after Step 3 in the operation of DIMCRA. Because in Step 4, DIMCRA makes a new union of solution, probably the original two paths returned by DIMCRA are arranged. In order to analyze the DIMCRA’s solution exactly, here the solution is not the final one.

4.2.1 DIMCRA’s solutions classifying

Generally, we categorize these solutions into 3 classes: node-disjoint; link-no-overlapping-disjoint; link-overlapping-disjoint. Node-disjoint means that the two paths do not have a common node. Link-no-overlapping-disjoint means there are no overlapped links in the two paths of the solution, but probably they have common nodes. Link-overlapping-disjoint means there are overlapping links in the two paths of the solution.

Next, we observe the success rate of DIMCRA in three classes respectively in order to know that in what situation DIMCRA’s performance is the best.

In order to know these three classes more clearly, we first employ some concrete examples to explain them, and then we will evaluate DIMCRA’s performance in the three classes.
4.2.2 Solution Class 1: node-disjoint

4.2.2.1 Description

The problem is to find two link-disjoint paths between 1 and 9, each within the constraints \( L = (9, 9) \) and with minimum total length. In the example, the optimal solution set that has been returned by the K-shortest paths algorithm is \{1-4-5-8-9, 1-2-3-6-9\}. The solution set returned by DIMCRA is \{1-4-5-8-9, 1-2-3-6-9\} after Step 3. After Step 5, the solution set returned by DIMCRA is also \{1-4-5-8-9, 1-2-3-6-9\}. We can find that the solution returned by DIMCRA is optimal. The two paths after Step 3 are node disjoint.

![Figure 4.1 An example of Class 1](image)

4.2.2.2 Evaluation

Our simulation is based on DIMCRA’s performance as a function of \( r \) with two-dimensional constraints under uniformly distributed link weights. The success rate of DIMCRA for the class of the lattices \( (N = 16) \) after running \( 10^4 \) times is represented in Figure 4.2.

In Figure 4.2, it shows that when the type of DIMCRA’s solution is node-disjoint, it is easy for DIMCRA to find an optimal solution set. Especially for \( r = 1.0 \), it almost can find all optimal solutions.
Figure 4.2 For \( m = 2, N = 16 \), the success rate of DIMCRA’s solution Class 1: node-disjoint for the lattices as a function of \( r \).

4.2.3 Solution Class 2: link-no-overlapping-disjoint

4.2.3.1 Description

Figure 4.3 An example of Class 2
The constrains are the same as the example in Figure 4.1. In this example (see Figure 4.3), the optimal solution set that has been returned by the K-shortest paths algorithm is \{1-2-5-8-9, 1-4-5-6-9\}. The solution set returned by DIMCRA is \{1-2-5-8-9, 1-4-5-6-9\} after Step 3. After Step 5 the solution set returned by DIMCRA is also \{1-2-5-8-9, 1-4-5-6-9\}, which is optimal. Because the two paths after Step 3 have a common node (node 5) and have no overlapping links, we name this example as link-no-overlapping-disjoint in the optimal solution.

4.2.3.2 Evaluation

Like Section 4.2.2.2 we have also made similar simulations to observe the success rate of DIMCRA in finding an optimal solution, when the class of DIMCRA’s solution is link-no-overlapping-disjoint. From Figure 4.4 we can observe that when the class of DIMCRA’s solution is link-no-overlapping-disjoint, DIMCRA can find an optimal solution most of the time. Comparing with nodes-disjoint, it is worse.

Figure 4.4 For \(m = 2, N = 16\), the success rate of DIMCRA’s solution Class 2: link-no-overlapping-disjoint for the class of lattice graphs as a function of \(r\)

4.2.4 Solution Class 3: link-overlapping-disjoint

4.2.4.1 Description

In this example, the optimal solution set that has been returned by the K-shortest paths algorithm is \{1-2-5-6-9, 1-4-7-8-9\}. The solution set returned by DIMCRA is \{1-2-5-8-9, 1-4-7-8-5-6-9\} after Step 3. After Step 5 the solution set returned by DIMCRA is also
{1-2-5-6-9, 1-4-7-8-9} that is the optimal. In the two paths after Step 3, there is an overlapping link (link 5->8). So we name this example as \textit{link-overlapping-disjoint}.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{example_class_3}
\caption{An example of Class 3}
\end{figure}

\subsection{Evaluation}

From Figure 4.6 we can see that when the class of DIMCRA’s solution is \textit{link-overlapping-disjoint}, it is hard to find an optimal solution set. However, it always finds a solution.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{success_rate_graph}
\caption{For \(m = 2, N = 16\), the success rate of DIMCRA’s solution Class 3: \textit{link-overlapping-disjoint} for the class of lattice graphs as a function of \(r\).}
\end{figure}
4.2.5 Summary

From the above three classes of DIMCRA’s solutions, we can find that whether DIMCRA can find an optimal solution depends on two factors: the components of DIMCRA’s solution set; the numbers of overlapping links.

Summary of our findings:

- When the class of DIMCRA’s solution is node-disjoint, it is better that DIMCRA can often find an optimal solution, especially when $r$ is increased. And its success rate in this situation is between 84.04% and 97.71%.

- When the class is link-overlapping-disjoint, it is hard for DIMCRA to find an optimal solution, especially when $r$ decreases. The highest success rate in finding an optimal solution is 74.44% and the lowest one is down to 12.70%. But it still can find a solution, if it exists.

- When the class is link-no-overlapping-disjoint, DIMCRA can usually find an optimal solution, but no as better than Class 1. Its success rate in finding an optimal solution is between 61.44% and 63.38%.

4.3 DIMCRA Limitations

In this section we discuss why DIMCRA cannot find an optimal solution sometimes. An important reason to explain its failure in finding the optimal solution, is that the first path returned by Step 1 in the operation of DIMCRA deviates from the optimal solution set. DIMCRA can guarantee its solution must be disjoint. However, it may not be optimal. We require two link-disjoint paths with the minimum total length instead of two orderly shortest paths. Finally, probably in the optimal solution set there are some nodes that do not exist in the DIMCRA’s solution. All are the limitation of DIMCRA inherent. That is to say, if we do not change DIMCRA, its limitation never disappears.

4.4 Overview

Summing up Chapter 3 and Chapter 4, we can find that whether DIMCRA can find an optimal solution depends on four factors: the components of DIMCRA’s solution; the number of overlapping links, link density of $p$; correlation $r$. In addition, although currently DIMCRA is the best for the link-disjoint multiple constraints routing algorithms, but it is still can be improved. Therefore, in the next chapter, we will propose three methods to improve DIMCRA’s performance.
In this chapter we will propose three methods to improve the performance of DIMCRA.

5.1 Motivation and Problem Definition

DIMCRA can guarantee that its solution must be disjoint; however DIMCRA’s solution is not necessarily optimal in terms of the minimal total length. Hence, it is possible to further optimize DIMCRA, such that it can always guarantee finding a pair of optimal link-disjoint paths, if it does exist.

5.2 Method 1

5.2.1 Description

From this example (see Figure 5.1(a)), we can get some tips to improve DIMCRA’s success rate. The operations of DIMCRA are shown in Figure 5.1(b).

(a) Constraints L (9.0, 9.0)
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Operations</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIMCRA</td>
<td>Step 1</td>
<td>( P_1 = 1-4-5-6-9 )</td>
</tr>
<tr>
<td></td>
<td>Step 3</td>
<td>( P_2 = 1-2-5-8-9 )</td>
</tr>
<tr>
<td></td>
<td>Step 4</td>
<td>( { P_1', P_2' } = {1-4-5-6-9, 1-2-5-8-9} )</td>
</tr>
<tr>
<td>DIMCRA's solution</td>
<td>Step 5</td>
<td>( \text{DIMCRA's solution} = {1-4-5-6-9, 1-2-5-8-9} )</td>
</tr>
<tr>
<td>K-shortest paths algorithm</td>
<td></td>
<td>The optimal solution = {1-4-5-8-9, 1-2-5-6-9}</td>
</tr>
</tbody>
</table>

(b) The operations of DIMCRA and the K-shortest paths algorithm

Figure 5.1 Example 1 of Method 1

In this example, DIMCRA’s solution is close to the optimal solution, but DIMCRA fails to find the optimal solution. We observe DIMCRA’s solution carefully and find that in the solution node 5 is common. If we exchange the remaining links \{5-6-9, 5-8-9\} after common node 5, then we can get the optimal solution \{1-4-5-8-9, 1-2-5-6-9\}. Consequently, in some graphs if we make such permutation and combination in the final solution set, then probably we get a bigger chance to find the optimal solution. It is necessary that we compare a new pair of link-disjoint paths obtained by such permutation and combination with the original solution of DIMCRA, because the new pair is probably not smaller than the original solution. If there are several common nodes, probably we can get more than one new pair, among which we also make comparison. Finally we can find the one with the smallest length.

We must make this exchange after Step 4, because one of the paths in the original solution (after Step 5) may not obey constraints. According to the operation of original DIMCRA, this path will be removed. If we introduce a new step before Step 5, we probably have a bigger chance to find the optimal solution.

Using Method 1, we can also solve the problem of Example 4 in Section 2.3.5 of Chapter 2. The operating process of this example is shown in Figure 5.2.

(a) Constraints \( L = (1.0, 1.0) \)
Figure 5.2 Example 2 of Method 1

In this example, we found that two pairs in the set \( \{P_1', P_2'\} \) of Step 5 have the same length 0.8. But \( P_2'' = \{1-3-5\} \) in the set \( \{P_1'', P_2''\} = \{\{1-2-3-4-5, 1-3-5\}, \{1-2-3-5, 1-3-4-5\}\} \) does not obey the constraints \( L = (1.0, 1.0) \). Therefore we remove this pair to choose another a pair \{1-2-3-5, 1-3-4-5\} to be the final solution. It is necessary for us to check each path length in \( \{P_1'', P_2''\} \). Step 5 of the original DIMCRA is indeed needed.

Next, we will make simulation to examine whether this method is efficient.

### 5.2.2 Evaluation

We have simulated for the improved DICMRA like before. We set \( N = 16 \) and different correlation \( r \) ([-1.0, 1.0]) to obtain different success rate and execution time for original DIMCRA and improved DIMCRA. We improved DIMCRA in finding an optimal solution.
Figure 5.3 For $m = 2, N = 16$, the success rate and execution time comparison of original DIMCRA and improved DIMCRA in finding an optimal solution for the class of lattice graphs as a function of $r$.

From Figure 5.3(a) we find that although the changed DIMCRA certainly improved the success rate in finding the optimal solution, generally, the average ratio of improving is 3.37%. It just improves the success rate in finding an optimal solution instead of finding more solutions. Or in another word, in the success rate of finding a solution, there is no change between improved DIMCRA and original DIMCRA. In addition, we find
that although improved DICMRA take more time than original DIMCRA, their time difference is just a little bit because the quantity level of time is in $10^{-3}$s.

5.2.3 Summary

The operation of Method 1 is simple. We just add a new step after Step 4, and revise Step 5 in the original DIMCRA. New operations of improved DIMCRA are summarized as below:

- Step 1, 2, 3 and 4 are the same as in the original DIMCRA.

- Step 5. If there are common nodes in the set $\{P'_1, P'_2\}$ except the source node and destination node, then we exchange the rest path after the common nodes to arrange new pairs. We put the new pairs to the set $\{P'_1, P'_2\}$ then get a new union $\{P''_1, P''_2\}$. Comparing these pairs in the new union $\{P''_1, P''_2\}$ to compute the smallest length pairs set $\{P''''_1, P''''_2\}$. Probably, there are several pairs with the same shortest length.

- Step 6. This step is the same as Step 5 of the original DIMCRA. We shall remove the pairs which do not obey the constraints.

We just improve the success rate of DIMCRA of finding an optimal solution instead of finding more feasible solutions.

In fact, the improved DIMCRA does not essentially change the scheme of the original DIMCRA. There are still limitations in the improved DIMCRA which are the same as in the original DIMCRA. So we think it is possible that we can first get a paths pool. Secondly, we can successfully combine an optimal solution set in this paths pool. However, it is crucial to know how we can obtain such an optimal solution paths pool exactly.

![Figure 5.4 The concept of new algorithm](image)

Actually, Method 1 still cannot guarantee finding all optimal solutions, therefore, we examine a new method, Method 2.

5.3 Method 2
Based on the limitation of original DIMCRA, we find that sometimes there are some links in the optimal solution which do not appear in DIMCRA’s solution. So it is a problem how to find these disappeared links to construct the optimal solution paths pool.

### 5.3.1 Description

In Method 2, we add a step after Step 4 in the original DIMCRA and revise Step 5 in the original DIMCRA. The other steps are the same as in the original DIMCRA. The substeps of the new step for a graph $G(V, E)$ are stated below:

1’. Put each disappeared node $n \in [V - \text{all nodes in } P'_i]$ into a disappeared node set $D_i$.

2’. For each $n \in D_i$, construct a path $P''_i(s, t) = P(s, n) + P(n, t)$ under constraints. Put all $P''_i(s, t)$ and $P'_i$ into a path pool $H$. If $P''_i(s, t)$ is a duplicate path in $H$, or contains a loop for $P'_i$, then we do not put it into $H$.

3’. We group all paths in $H$ to make union of the pairs of link-disjoint paths, comparing these pairs to obtain the DIMCRA’s solution $\{P'''_1, P'''_2\}$ with the minimal total length.

In Substep 3’, we group all paths in $H$, therefore, we need to check each path length in $\{P'''_1, P'''_2\}$. If they obey the constraints, then we return this solution, otherwise DIMCRA stops. In addition, Method 2 can find whether there exists an optimal solution. DIMCRA stopped in Substep 3’, which means that there no an optimal solution exist there.

We employ Example 1 to clarify Method 2 (see Figure 5.5).

(a) Constraints $L = (9.0, 9.0)$
Algorithm | Operations | Result
---|---|---
DIMCRA | Step 1 | $P_1 = \{1-4-5-6-9\}$
 | Step 3 | $P_2 = \{1-2-5-4-7-8-9\}$
 | Step 4 | $\{P_1', P_2'\} = \{1-4-7-8-9, 1-2-5-6-9\}$
 | Step 5 | The DIMCRA’s solution = $\{1-4-7-8-9, 1-2-5-6-9\}$

K-shortest paths algorithm | | The optimal solution = $\{1-2-3-6-9, 1-4-7-8-9\}$

(b) The operations of original DIMCRA

<table>
<thead>
<tr>
<th>$P_1' = 1-4-7-8-9$</th>
<th>$P_2' = 1-2-5-6-9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disappeared node set $D_1$</td>
<td>Path</td>
</tr>
<tr>
<td>2</td>
<td>$1-2-5-6-9$</td>
</tr>
<tr>
<td>3</td>
<td>$1-2-3-6-9$</td>
</tr>
<tr>
<td>5</td>
<td>$1-4-5-6-9$</td>
</tr>
<tr>
<td>6</td>
<td>$1-2-3-6-9$</td>
</tr>
</tbody>
</table>

(c) Construct the alternate paths individually for $P_1'$ and $P_2'$

<table>
<thead>
<tr>
<th>No.</th>
<th>Path Pool H</th>
<th>Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-4-7-8-9</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>1-2-5-6-9</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>$1-2-3-6-9$</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>$1-2-3-6-9$</td>
<td>No(duplicate)</td>
</tr>
<tr>
<td>5</td>
<td>$1-4-5-8-9$</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>$1-4-5-6-9$</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>$1-2-3-6-9$</td>
<td>No(duplicate)</td>
</tr>
<tr>
<td>8</td>
<td>$1-4-5-6-9$</td>
<td>No(duplicate)</td>
</tr>
<tr>
<td>9</td>
<td>$1-4-7-8-9$</td>
<td>No(duplicate)</td>
</tr>
<tr>
<td>10</td>
<td>$1-4-7-8-9$</td>
<td>No(duplicate)</td>
</tr>
</tbody>
</table>

(d) Clear up the alternate paths

<table>
<thead>
<tr>
<th>No.</th>
<th>Link-disjoint path pool</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>${1-4-7-8-9, 1-2-5-6-9}$</td>
<td>5.05/9.0</td>
</tr>
<tr>
<td>2</td>
<td>${1-4-7-8-9, 1-2-3-6-9}$</td>
<td>4.94/9.0</td>
</tr>
<tr>
<td>3</td>
<td>${1-2-5-6-9, 1-4-5-8-9}$</td>
<td>4.99/9.0</td>
</tr>
<tr>
<td>4</td>
<td>${1-2-3-6-9, 1-4-5-8-9}$</td>
<td>4.99/9.0</td>
</tr>
</tbody>
</table>

(e) Get DIMCRA’s solution
### Table 5.1

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Operations</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved DIMCRA</td>
<td>Step 1</td>
<td>Same</td>
</tr>
<tr>
<td></td>
<td>Step 3</td>
<td>Same</td>
</tr>
<tr>
<td></td>
<td>Step 4</td>
<td>Same</td>
</tr>
<tr>
<td></td>
<td>Step 5</td>
<td>( { P_1', P_2' } = { 1-2-3-6-9, 1-4-7-8-9 } )</td>
</tr>
<tr>
<td></td>
<td>Step 6</td>
<td>The DIMCRA’s solution = { 1-2-3-6-9, 1-4-7-8-9 }</td>
</tr>
</tbody>
</table>

(f) The operations of improved DIMCRA

Figure 5.5 An example of Method 2

In the set \( \{ P_1', P_2' \} \) of Step 4, there is not a common node, except the start node 1 and end node 9. Method 1 does not work in this case. We observe the optimal solution, there is only one path = \{ 1-4-7-8-9 \} of the DIMCRA’s solution. In addition, node 3 in the optimal solution does not appear in the DIMCRA’s solution. Method 2 will try to find the disappeared node 3, as well as its interrelated links (2→3, 3→6). Figure 4.5 shows the steps of Method 2.

#### 5.3.2 Evaluation

Generally, in my simulations, for lattice graphs Method 2 can always find an optimal solution. But Method 2 still has a disadvantage, it costs more time. In Figure 5.6, we can see it. In Substep 2’, we use SAMCRA to construct the path \( P_i''(s,t) \). Usually if we use SAMCRA to construct all \( P_i''(s,t) \) in multiple dimensions, it takes longer time. This point causes the disadvantage of Method 2. However, if the number of nodes is not large, Method 2 is still fine.

![Execution time comparison](image)

Figure 5.6 For \( m = 2, N = 16 \), execution time comparison of DIMCRA and improved DIMCRA in finding an optimal solution for the class of lattice graphs as a function of \( r \)
5.3.3 Summary

We sum up the new operations of improved DIMCRA as follows:

- Step 1, 2, 3 and 4 are the same as in the original DIMCA.

- Step 5.
  - 1’. Put each disappeared node \( n \in [V - \text{all nodes in } P'_i] \) into a disappeared node set \( D_i \).
  
  - 2’. For each \( n \in D_i \), construct a path \( P''_i(s, t) = P(s, n) + P(n, t) \) under constraints. Put all \( P''_i(s, t) \) and \( P'_i \) into a path pool \( H \). If \( P''_i(s, t) \) is a duplicated path in \( H \), or contains a loop for \( P'_i \), then we do not put it into \( H \).
  
  - 3’. We group all paths in \( H \) to make union of the pairs of link-disjoint paths, comparing these pairs to obtain the DIMCRA’s solution \( \{P''_1, P''_2\} \) with the minimal total length.

- Step 6. To check each path length in \( \{P''_1, P''_2\} \). If they obey constraints, then return this solution, otherwise DIMCRA stop.

5.4 Method 3

To avoid the disadvantages of Method 1 and Method 2, we propose Method 3. Method 3 combines the advantages of Method 1 and Method 2. Its simple operations lead to lower cost and its precise operations lead to higher success rate.

5.4.1 Description

In Substep 2’ of Step 5 in Method 2, we use SAMCRA to construct a path \( P''_i(s, t) = P(s, n) + P(n, t) \), it is expensive. In this method, we use a new way with lower cost to replace SAMCRA.

In Method 3, we modify Substep 2’ of Step 5 in the Method 2, the other steps are same as in the Method 2. The substeps of the new step 5 for a graph \( G(V, E) \) are stated below:

- 1’. Put each disappeared node \( n \in [V - \text{all nodes in } P'_i] \) into a disappeared node set \( D_i \). It is same as Method 2.

- 2’. For each \( n \in D_i \), construct a path
\[ P_i''(s, t) = \sum_{i=N_1}^{2} \max(w_m(P(s_i, s_{i-1}))/L_m) + \sum_{j=1}^{N_2} \max(w_m(P(t_j, t_{j+1}))/L_m) \]

by using all nodes appearing in the set of two paths \( \{P_1', P_2'\} \). In the path \( P_i''(s, t) \), \( \forall s_{N_i} = n, s_1 = s, t_1 = n, t_{N_2} = t \) and weights \( w_m(P) = \sum_{(u \rightarrow v) \in P} w_m(u \rightarrow v) \). Here, \( N_1 \) is the numbers of all nodes which are smaller than \( n \) in the set \( \{P_1', P_2'\} \), \( N_2 \) is the numbers of all nodes which are larger than \( n \) in the set \( \{P_1', P_2'\} \). Put all \( P_i''(s, t) \) and \( P_i' \) into a path pool \( H \). If \( P_i''(s, t) \) is a duplicate path in \( H \), or contains a loop for \( P_i' \), then we do not put it into \( H \).

3’. We group all paths in \( H \) to make union of the pairs of link-disjoint paths, comparing these pairs to obtain the DIMCRA’s solution \( \{P_1''', P_2'''\} \) with the minimal total length.

Actually, Method 3 is improving Method 2. In this method, we use a balance tree to construct a path \( P_i''''(s, t) \). By using the nodes from original DIMCRA’s solution, we reduce the constructing range for the optimal solution. Finally comparing these new paths with the original DIMCRA’s solution, we obtain the final solution.

We still use the example in Figure 5.5 to clarify Method 3.

(a) Same as Figure 5.5(a)

(b) Same as Figure 5.5(b)

<table>
<thead>
<tr>
<th></th>
<th>Disappeared node set D1</th>
<th>Path</th>
<th>Disappeared node set D2</th>
<th>Path</th>
<th>Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>{1,2}, {4,5,6,7,8,9}</td>
<td>1-2-5-6-9</td>
<td>{1,2}, {4,5,6,7,8,9}</td>
<td>1-2-3-6-9</td>
<td>No (duplicate)</td>
</tr>
<tr>
<td>3</td>
<td>{1,2}, {4,5,6,7,8,9}</td>
<td>1-2-3-6-9</td>
<td>{1,2}, {4,5,6,7,8,9}</td>
<td>1-4-7-8-9</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>{1,2}, {4,5,6,7,8,9}</td>
<td>1-2-3-6-9</td>
<td>{1,2,4}, {5,6,7,8,9}</td>
<td>1-4-7-8-9</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>{1,2,4,5}, {6,7,8,9}</td>
<td>1-4-5-6-9</td>
<td>{1,2,4,5,6,7}, {8,9}</td>
<td>1-4-7-8-9</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>{1,2,4,5,6}, {7,8,9}</td>
<td>1-2-3-6-9</td>
<td>{1,2,4,5,6,8,9}, {9}</td>
<td>1-4-5-8-9</td>
<td>Yes</td>
</tr>
</tbody>
</table>

(c) Construct the alternate paths individually for \( P_1' \) and \( P_2' \)
<table>
<thead>
<tr>
<th>No.</th>
<th>Path Pool H</th>
<th>Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1-4-7-8-9</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>1-2-5-6-9</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>1-2-5-6-9</td>
<td>No (duplicate)</td>
</tr>
<tr>
<td>4</td>
<td>1-2-3-6-9</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>1-4-5-6-9</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>1-2-3-6-9</td>
<td>No (duplicate)</td>
</tr>
<tr>
<td>7</td>
<td>1-4-7-8-9</td>
<td>No (duplicate)</td>
</tr>
<tr>
<td>8</td>
<td>1-4-7-8-9</td>
<td>No (duplicate)</td>
</tr>
<tr>
<td>9</td>
<td>1-4-5-8-9</td>
<td>Yes</td>
</tr>
</tbody>
</table>

(d) Clear up the alternate paths

<table>
<thead>
<tr>
<th>No.</th>
<th>Link-disjoint paths pool</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{1-4-7-8-9, 1-2-5-6-9}</td>
<td>5.05/9.0</td>
</tr>
<tr>
<td>2</td>
<td>{1-4-7-8-9, 1-2-3-6-9}</td>
<td>4.94/9.0</td>
</tr>
<tr>
<td>3</td>
<td>{1-2-5-6-9, 1-4-5-8-9}</td>
<td>5.10/9.0</td>
</tr>
<tr>
<td>4</td>
<td>{1-2-3-6-9, 1-4-5-8-9}</td>
<td>4.99/9.0</td>
</tr>
</tbody>
</table>

(e) Get DIMCRA’s solution

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Operations</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved DIMCRA</td>
<td>Step 1</td>
<td>Same</td>
</tr>
<tr>
<td></td>
<td>Step 3</td>
<td>Same</td>
</tr>
<tr>
<td></td>
<td>Step 4</td>
<td>Same</td>
</tr>
<tr>
<td></td>
<td>Step 5</td>
<td>{ P_1', P_1'' } = {1-2-3-6-9, 1-4-7-8-9}</td>
</tr>
<tr>
<td></td>
<td>Step 6</td>
<td>The DIMCRA’s solution = {1-2-3-6-9, 1-4-7-8-9}</td>
</tr>
<tr>
<td>K-shortest paths</td>
<td></td>
<td>The optimal solution = {1-2-3-6-9, 1-4-7-8-9}</td>
</tr>
<tr>
<td>algorithm</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(f) All operations of improved DIMCRA

Figure 5.7 An example of Method 3

In order to clarify how to construct a path \( P_i''(s,t) \), we make disappeared node 2 in \( D_i \) to be an example to explain this constructing way. In Figure 5.8, our final goal is to construct a new shortest path from source node 1 to destination node 9 by node 2. Therefore, the shortest path \( P_{[2\rightarrow 1]} \) and \( P_{[2\rightarrow 9]} \) is required. To any root node, its left node is always smaller and its right node is always larger. For example, there are eight nodes \{1,4,5,6,7,8,9\} appearing on the \( P_i \), node 1 is smaller than 2 and nodes \{4,5,6,7,8,9\} are larger than node 2. The shortest path from node 2 to node 1 is \( P_{[2\rightarrow 1]} \), the shortest path from node 2 to nodes \{4,5,6,7,8,9\} is \( P_{[2\rightarrow 5]} \). Through the same operation, we can get the shortest path from node 5 to nodes \{6,7,8,9\} is node 6. Finally we combine all root nodes to create a new path \( P_{[1\rightarrow 2\rightarrow 9]} = 1-2-5-6-9 \). Normally using this way to search the shortest path to construct a tree, it will be faster.
5.4.2 Evaluation

We also make the same simulation as Method 2. For the square lattices Method 3 always found an optimal solution. And the execution time is less than in Method 2.
5.4.3 Summary

We sum up the new operations of improved DIMCRA as follows:

1. Step 1, 2, 3 and 4 are the same as in the original DIMCA.

2. Step 5.
   1’. Put each disappeared node $n \in \{V\}$ into a disappeared node set $D_1$.
   2’. For each $n \in D_1$, construct a path
   \[ P_i''(s, t) = \sum_{i=1}^{N_1} \max(w_m(P(s_0, s_{i-1}))/L_m) + \sum_{j=1}^{N_2} \max(w_m(P(t_j, t_{j+1}))/L_m) \]
   using all nodes in the set of two paths $\{P_1', P_2'\}$. In the path $P_i''(s, t)$, $\forall s_{N_1} = n, s_1 = s, t_1 = n, t_{N_2} = t$ and weights $w_m(P) = \sum_{(u \rightarrow v) \in P} w_m(u \rightarrow v)$. In addition, $N_1$ is the numbers of all nodes which are smaller than $n$ in the set $\{P_1', P_2'\}$, $N_2$ is the numbers of all nodes which are larger than $n$ in the set $\{P_1', P_2'\}$. Put all $P_i''(s, t)$ and $P_i'$ into a path pool $H$. If $P_i''(s, t)$ is a duplicate path in $H$, or contains a loop for $P_i'$, then we do not put it into $H$.

3’. We group all paths in $H$ to make union of the pairs of link-disjoint paths, comparing these pairs to obtain the DIMCRA’s solution $\{P_1'''', P_2'''\}$ with the minimal total length.

3. Step 6. To check each path length in $\{P_1'''', P_2'''\}$, if they obey constraints, and then return this solution, otherwise DIMCRA stop.

Because of the time problem, we only tested Method 3 for the square lattice graphs.
In this chapter, we will present the main conclusions and contributions of this thesis, as well as possible extension for future research.

6.1 Summary

Chapter 1 presented the background and problem of this thesis. With the rapid development of Internet, the Internet research community has made great efforts to define more efficient network management. The motive forces of these efforts are specific requirements for performance relating to new applications. For instance, multi-media applications need better quality of service, and the Internet also needs more guarantees regarding the reliability of the data it transports. Therefore, when a path is blocked and information cannot be transferred properly, we expect a backup path can be used to retransfer this information quickly. For this purpose, we hope to develop a QoS routing algorithm which can find two link-disjoint paths that meet the users’ needs. It is named the MCLPP (multiple constrained link-disjoint path pair) problem. Because the information will be flooded in the Internet, it is better to find the pair of link-disjoint paths with minimal total length, which is defined as the MCOLPP (multiple constrained optimal link-disjoint path pair) problem. Actually, the MCOLPP problem is the optimal solution of MCLPP problem. To solve the MCLPP problem, algorithm DIMCRA (link-disjoint multiple constraints routing algorithm), is proposed. The goal of this thesis is to implement, evaluate and improve DIMCRA in solving the MCLPP problem.

Chapter 2 first reviewed the concepts, examples, implementation code, complexity and properties of DIMCRA in details. In order to measure the performance of DIMCRA in solving the MCLPP problem, we introduced another algorithm RF (remove-find). This algorithm is originally proposed to solve the LPP (link-disjoint path pair) problem. However we can also use the RF algorithm to solve the MCLPP problem. The RF algorithm is simple and direct.

Chapter 3 first evaluated the performance of DIMCRA and RF in solving the MCLPP problem. We made many simulations and analyzed the results to identify the behavior of DIMCRA and RF for MCLPP. Our simulation is operated on general level. We chose two classes of graphs: the random graphs and the lattice graphs to measure DIMCRA and RF. For every class of graphs, we measured the performance of DIMCRA
and RF not only in finding a solution, but also in finding an optimal solution. The conclusion is that DIMCRA’s performance is always better than RF. Once RF can find a solution, DIMCRA certainly can find it. In addition, the parameter $p$ (link density) and $r$ (correlation between links weights) both have influences on the success rate of DIMCRA.

From chapter 3, we see that DIMCRA may not find all the solutions. Therefore we want to understand the underlying reasons which cause the failure. Chapter 4 detailedly analyzed DIMCRA’s solution. It is concluded that whether DIMCRA can find an optimal solution depends on two factors: the components of DIMCRA’s solution set and the numbers of overlapping links.

- When the class of DIMCRA’s solution is node-disjoint, DIMCRA can often find an optimal solution, especially when $r$ increases. And its success rate in this situation is between 84.04% and 97.71%.

- When the class is link-overlapping-disjoint, it is hard for DIMCRA to find an optimal solution, especially when $r$ decreases. The highest success rate in finding an optimal solution is 74.44% and the lowest one is down to 12.7%.

- When the class is link-no-overlapping-disjoint, DIMCRA can usually find an optimal solution, but not as often as node-disjoint does. Its success rate in finding an optimal solution is between 61.44% and 63.38%.

Starting from both Chapters 3 and 4, we find that whether DIMCRA can find a solution depends on four factors: the components of DIMCRA’s solution; the number of overlapping links, link density $p$; and correlation $r$. In addition, although currently DIMCRA is the best link-disjoint multiple constraints routing algorithm, it is still can be improved. We thus propose three possible improvements. Method 1 is simple and improves the success rate in finding an optimal solution with 3.7% average. Method 2's efficiency is prominent, but the cost is high. Method 3 is a good choice to improve DIMCRA’s performance, because its cost is less than Method 2 at the same success rate of Method 2.

### 6.2 Main Contributions

The main contributions of this thesis can be summed up as the below points:

- We have implemented the code of the DIMCRA algorithm in solving the MCLPP problem. In order to measure DIMCRA, we have also created the RF algorithm function. In addition, to compute the optimal solution (MCOLPP), we have modified the K-shortest paths algorithm. Many code have been also provided in this thesis.

- We have evaluated the performance of DIMCRA. We have also performed many simulations to compare DIMCRA and RF with the optimal solution
• returned by the K-shortest paths algorithm. Additionally, many concrete first-hand data are carefully collected in this thesis.

• By investigating DIMCRA, we found that it can still be improved. Therefore, we have proposed three methods to improve DIMCRA. We have even evaluated the improved DIMCRA. These methods are indeed efficient, especially we have proposed Method 3 which takes less cost to achieve higher success rate in solving the MCLPP problem.

### 6.3 Potential Future Extensions

Two possible extensions of this research are suggested:

- Although, in Chapter 5, Method 3 is efficient in those simulations, it is suggested to make more simulations on different graphs to measure it.

- In addition, it is possible to propose a better solution with lower cost to construct path $P^*_i (s, t)$ in Method 3.
Reference


About the Author

Xia GUO was born on February 21, 1978, in Urumchi, China. In July 1998, she completed her Bachelor of Engineering degree at the School of Electrical Engineering, Xi’an University of Technology. After working at Tianjin Commerce University as an engineer for five years, she started to study for her master degree in the Computer Engineering (CE) group at Delft University of Technology (TUD), in the Netherlands.

In October 2004, she started working on her MSc thesis under the guidance of Dr. Ir. Fernando A. Kuipers in the NAS group. The NAS group is part of the faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS) of Delft University of Technology. The NAS group belongs to the Department of Telecommunications and participates in the Telecom colloquium series. The NAS group is chaired by Professor Van Mieghem. Xia GUO’s thesis is entitled: “Link-disjoint Quality of Service Routing”. Her research interests include: wireless network, data communication, embedded systems design.