I dedicate this to my wife Ting and my daughter Eva.
ACKNOWLEDGMENTS

I would like to take this opportunity to thank my committee chair Dr. Thai for her priceless help for my doctorate program. Not only the chance that she offered me to come to United States to further my study and research on computer networking, but also her precious guidance over my 5-year doctorate study and research, are indispensable to this thesis. Her strong passion, preciseness and profound knowledge for research have been enlightening me. The financial support from her evades me from the financial problems for international students and so that I can concentrate over the researches. I would also like to thank all the professors in my committee for their time for discussing over my research topics and providing numerous constructive opinions. I would like to thank my group members, Ravi, Incheol, Yilin, Thang, Nam, Dung for their help in my study and work.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>4</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>8</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>9</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>11</td>
</tr>
<tr>
<td>CHAPTER</td>
<td></td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>13</td>
</tr>
<tr>
<td>1.1 Combinatorial Group Testing</td>
<td>13</td>
</tr>
<tr>
<td>1.2 Extending and Improving GT for Modeling Network Problems</td>
<td>14</td>
</tr>
<tr>
<td>1.3 Combinatorial Optimizations for Network Security</td>
<td>15</td>
</tr>
<tr>
<td>1.4 Combinatorial Optimizations for Network Reliability</td>
<td>16</td>
</tr>
<tr>
<td>1.5 Thesis Organization</td>
<td>17</td>
</tr>
<tr>
<td>2 A NEW CONSTRUCTION OF ERROR-TOLERANT DISJUNCT MATRIX</td>
<td>19</td>
</tr>
<tr>
<td>3 A GROUP TESTING BASED DETECTION OF APPLICATION DENIAL-OF-SERVICE</td>
<td>22</td>
</tr>
<tr>
<td>ATTACKERS</td>
<td></td>
</tr>
<tr>
<td>3.1 Related Works</td>
<td>25</td>
</tr>
<tr>
<td>3.2 Problem Models</td>
<td>26</td>
</tr>
<tr>
<td>3.2.1 Attacker Model</td>
<td>26</td>
</tr>
<tr>
<td>3.2.2 Victim/Detection Model</td>
<td>27</td>
</tr>
<tr>
<td>3.3 Strategy and Detection System</td>
<td>29</td>
</tr>
<tr>
<td>3.3.1 Size Constraint Group Testing</td>
<td>29</td>
</tr>
<tr>
<td>3.3.2 Detection System</td>
<td>30</td>
</tr>
<tr>
<td>3.3.2.1 System overview</td>
<td>30</td>
</tr>
<tr>
<td>3.3.2.2 Configuration details</td>
<td>32</td>
</tr>
<tr>
<td>3.4 Detection Algorithms and Latency Analyses</td>
<td>37</td>
</tr>
<tr>
<td>3.4.1 Sequential Detection with Packing</td>
<td>38</td>
</tr>
<tr>
<td>3.4.2 Sequential Detection without Packing</td>
<td>41</td>
</tr>
<tr>
<td>3.4.3 Partial Non-adaptive Detection</td>
<td>44</td>
</tr>
<tr>
<td>3.5 Simulation Configurations and Results</td>
<td>52</td>
</tr>
<tr>
<td>3.5.1 Configurations</td>
<td>52</td>
</tr>
<tr>
<td>3.5.2 Results</td>
<td>54</td>
</tr>
<tr>
<td>3.6 Conclusions and Discussions</td>
<td>58</td>
</tr>
<tr>
<td>3.6.1 Conclusions</td>
<td>58</td>
</tr>
<tr>
<td>3.6.2 Discussions over False Rate</td>
<td>59</td>
</tr>
</tbody>
</table>
4 A TRIGGER IDENTIFICATION SERVICE FOR DEFENDING REACTIVE JAMMERS IN WIRELESS SENSOR NETWORKS ............................................. 61

4.1 Related Works ........................................................................ 63
4.2 Problem Models and Notations .................................................. 64
  4.2.1 Network Model ..................................................................... 64
  4.2.2 Attacker Model ...................................................................... 65
    4.2.2.1 Basic attacker model ....................................................... 65
    4.2.2.2 Advanced attacker model ................................................. 66
  4.2.3 Sensor Model ......................................................................... 67
4.3 Three Kernel Techniques .......................................................... 68
  4.3.1 Minimum Disk Cover in a Simple Polygon .............................. 68
  4.3.2 Clique-Independent Set ........................................................ 69
4.4 Trigger Identification Procedure ................................................. 71
  4.4.1 Anomaly Detection ............................................................... 71
  4.4.2 Jammer Property Estimation ................................................ 73
  4.4.3 Trigger Detection ................................................................. 74
    4.4.3.1 Discovery of interference-free testing teams ......................... 77
    4.4.3.2 Estimation of trigger upperbound ..................................... 80
  4.4.4 Analysis of Time and Message Complexity ............................ 81
4.5 Advanced Solutions Toward Sophisticated Attack Models .......... 83
  4.5.1 Upperbound on the Expected Value of $z$ ............................. 84
    4.5.1.1 Probabilistic jamming response ...................................... 85
    4.5.1.2 Variant reaction time ....................................................... 85
  4.5.2 Error-tolerant Asynchronous Testing within Each Testing Team 88
4.6 Experimental Evaluation .......................................................... 88
  4.6.1 Overview ............................................................................. 88
  4.6.2 Benefits for Jamming-resistant Routing ................................. 89
  4.6.3 Improvements on Time Complexity ...................................... 91
  4.6.4 Accuracy in Estimating Jammer Properties ........................... 94
  4.6.5 Robustness to Various Jammer Models ................................. 95
4.7 Discussion and Conclusions ....................................................... 96

5 AN EFFICIENT MULTI-LINK FAILURE LOCALIZATION SCHEME IN ALL-OPTICAL NETWORKS ...................................... 97

5.1 Related Works .......................................................................... 99
5.2 Background and Problem Branches ............................................. 100
5.3 A centralized algorithm for moderate sized networks .................. 102
  5.3.1 Tree-decomposition .............................................................. 103
  5.3.2 Single-Link Failure Localization on Tree-subgraphs ................ 105
  5.3.3 Redundant Tree-Subgraphs .................................................. 106
5.4 A localized algorithm for large-scale networks ............................ 107
  5.4.1 Random Walk based Algorithm .......................................... 108
  5.4.2 Correctness and Complexity .............................................. 109
  5.4.3 Local Rarest First ............................................................... 110
### 5.5 Simulation Results

- 5.5.1 The Centralized Algorithm .................................................. 111
- 5.5.2 The Localized Algorithm .................................................. 112

### 5.6 Adaptation to Additional Constraints ........................................... 116

- 5.6.1 Case 1: $k$-par .......................................................... 116
- 5.6.2 Case 2: $q$-fail ......................................................... 117
- 5.6.3 Case 3: $d$-unknown ..................................................... 119

### 5.7 Summary ................................................................. 120

### 6 A GRAPH-THEORETIC QUALITY-OF-SERVICE AWARE VULNERABILITY ASSESSMENT FOR NETWORK TOPOLOGIES ......................................................... 122

- 6.1 Problem Model ............................................................... 124
- 6.2 Hardness ................................................................. 127
- 6.3 Exact Algorithm ............................................................ 127
- 6.4 Heuristic Solutions ......................................................... 131
  - 6.4.1 A Heuristic Solution for Small $m$ .................................. 131
  - 6.4.2 A Heuristic Solution for Large $m$ and Large Networks .... 132
- 6.5 Performance Evaluations ................................................... 133
  - 6.5.1 Dataset and Setup .................................................... 133
  - 6.5.2 Efficiency of Heuristic Solutions ................................ 134
- 6.6 Summary ................................................................. 138

### 7 CONCLUSION .............................................................. 139

### REFERENCES ................................................................. 140

### BIOGRAPHICAL SKETCH .......................................................... 147
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-1 Main Notations</td>
<td>30</td>
</tr>
<tr>
<td>4-1 Message Containing Trigger Detection Schedule</td>
<td>75</td>
</tr>
<tr>
<td>4-2 Notations</td>
<td>84</td>
</tr>
<tr>
<td>5-1 Main Notations</td>
<td>102</td>
</tr>
<tr>
<td>6-1 The time cost of MFM CSP (Max Flow Multi-Constraint Shortest Path)</td>
<td>135</td>
</tr>
<tr>
<td>6-2 The time cost of SDOP (Single Dimension Optimal Path) for large networks</td>
<td>138</td>
</tr>
<tr>
<td>6-3 The time cost of SDOP for large constraint amount</td>
<td>138</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>The Mathematical Principle of Group Testing</td>
<td>14</td>
</tr>
<tr>
<td>3-1</td>
<td>Victim/Detection model</td>
<td>28</td>
</tr>
<tr>
<td>3-2</td>
<td>2-mode diagram of the system</td>
<td>31</td>
</tr>
<tr>
<td>3-3</td>
<td>One testing round in danger mode</td>
<td>32</td>
</tr>
<tr>
<td>3-4</td>
<td>A detection example of how SDP(Sequential Detection with Packing) and S-DoP(Sequential Detection without Packing) work.</td>
<td>39</td>
</tr>
<tr>
<td>3-5</td>
<td>Status of the back-end server</td>
<td>55</td>
</tr>
<tr>
<td>3-6</td>
<td>Robustness by different number of back-end server machines $k$</td>
<td>56</td>
</tr>
<tr>
<td>3-7</td>
<td>Robustness by different number of attackers $d$</td>
<td>57</td>
</tr>
<tr>
<td>3-8</td>
<td>Robustness by different total number of clients $n$</td>
<td>58</td>
</tr>
<tr>
<td>4-1</td>
<td>Polynomial Time Reduction</td>
<td>70</td>
</tr>
<tr>
<td>4-2</td>
<td>Reactive Jamming Attack Model</td>
<td>72</td>
</tr>
<tr>
<td>4-3</td>
<td>Estimated $R$ and Jammed Area</td>
<td>74</td>
</tr>
<tr>
<td>4-4</td>
<td>Interference Teams</td>
<td>76</td>
</tr>
<tr>
<td>4-5</td>
<td>Iterative Local Refinement</td>
<td>78</td>
</tr>
<tr>
<td>4-6</td>
<td>Maximum # Interfering Cliques</td>
<td>80</td>
</tr>
<tr>
<td>4-7</td>
<td>Maximum # Jammers invoked by one team</td>
<td>80</td>
</tr>
<tr>
<td>4-8</td>
<td>Benefits for routing</td>
<td>90</td>
</tr>
<tr>
<td>4-9</td>
<td>Time and Message complexity</td>
<td>93</td>
</tr>
<tr>
<td>4-10</td>
<td>Estimation error of $R$</td>
<td>94</td>
</tr>
<tr>
<td>4-11</td>
<td>Solution Robustness</td>
<td>96</td>
</tr>
<tr>
<td>5-1</td>
<td>An example of Path-trial</td>
<td>101</td>
</tr>
<tr>
<td>5-2</td>
<td>Multiple branches of the M-LFL (Multi-Link failure localization) problem</td>
<td>101</td>
</tr>
<tr>
<td>5-3</td>
<td>Single Link Failure Localization on Trees</td>
<td>107</td>
</tr>
<tr>
<td>5-4</td>
<td>Performance of TDD (Tree-Decomposition Detection) centralized algorithm</td>
<td>113</td>
</tr>
<tr>
<td>5-5</td>
<td>Scalability of $R$-Walk localized algorithm to Network Size</td>
<td>114</td>
</tr>
</tbody>
</table>
5-6 Robustness of $R$-Walk localized algorithm to Link Failure Rate .......................... 115
5-7 Performance of adaptation to various constraints .................................................... 118
6-1 Conversion from QoSCV (Quality of Service Critical Vertex) to QoSCE (Quality of Service Critical Edge) ................................................................. 126
6-2 Integer Programming Formulation ........................................................................ 126
6-3 Combined Method of Branch-and-Bound and Pareto Optimality ......................... 130
6-4 NSFNET (National Science Foundation Networks) T1 1991 ............................... 134
6-5 Accuracy of MFMCSP (Max Flow Multi-Constraint Shortest Path) .................... 135
DISCRETE OPTIMIZATION FOR NETWORK SECURITY AND RELIABILITY

By

Ying Xuan

December 2011

Chair: My T. Thai
Major: Computer and Information Science and Engineering

Any network problems in essence equal to some principle questions in Discrete Mathematics, since all network elements can be abstracted as basic discrete structures, such as graphs, trees and permutations. One branch of discrete math, Graph Theory serves as abundant sources of theoretical support for network researches, from which people have been exploring since the last decade. However, the potential of another branch, Combinatorial Group Testing, has been overlooked, because of the intrinsic differences between its classic model and the practical network problems.

In this thesis, we attempt to fill the gap between Group Testing Theory and Network Optimization Problems, and then provide novel theoretical frameworks and efficient solutions through discrete optimizations for four network security and reliability problems. Specifically, we first provide a new size-constraint model for Group Testing, which thus can find many matches to practical network problems, and then propose an improvement over its traditional optimization solution. Then, we study two network security problems: Defending Application-Layer and Wireless Jamming Denial-of-Service Attacks and two reliability problems: Localizing All-Optical Network Link Failures and Assessing Network Topological Vulnerabilities. For each of these problems, we present a novel optimization framework, show its theoretical hardness, provide efficient algorithms with performance analysis, describe the implementation
details and feasibility/scalability, and discuss over potential improvements and future directions.
CHAPTER 1
INTRODUCTION

Discrete Optimization, which is to address optimization problems associated with a finite set of objects, has wide applications in Network Science. By mapping network nodes as vertices and links as edges, most network problems naturally fall into the categories of Graph Theory, Computational Geometry and etc. A mainstream of network theory researches focuses on applying these techniques to practical network problems, where network security and reliability are two important fields. However, another branch of combinatorial optimization, which is called Group Testing, has been neglected by the network researchers, due to the difficulties in applying its classic model to practical problems.

In this thesis, on one hand, we present a theoretical improvement over the Group Testing theory along with an extended model, which then can find many matches with practical network problems; on the other hand, we provide novel theoretical frameworks and solutions for four network security and reliability problems using a combination of Group Testing theory, Graph Theory, Set Theory and Computational Geometry.

1.1 Combinatorial Group Testing

Combinatorial Group Testing (CGT), a kernel optimization technique in Combinatorics, was proposed in WWII to speed up the detection of affected blood samples within a large sample population. Its key idea is to test items in multiple designated groups, instead of testing them individually. The traditional method of grouping items can be illustrated by a 0-1 matrix $M_{t \times n}$ where the matrix rows represent the testing group and each column refers to an item. As Fig. 1-1 shows, given a binary testing matrix $M$ and a testing outcome vector $V$. Assumed that item 1 (1st column) and item 2 (2nd column) are positive, then only the first two groups return negative outcomes, because they do not contain these two positive items. On the contrary, all the other four groups return positive outcomes. $M[i, j] = 1$ implies that the $j^{th}$ item appears in the $i^{th}$ testing group, and 0
\[ M = \begin{bmatrix}
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
\end{bmatrix} \quad \text{testing} \quad V = \begin{bmatrix}
0 \\
0 \\
1 \\
1 \\
1 \\
1 \\
\end{bmatrix} \]

Figure 1-1. The Mathematical Principle of Group Testing

otherwise. Therefore, the number of rows of the matrix denotes the number of groups tested in parallel and each entry of the result vector \( V \) refers to the test outcome of the corresponding group (row), where 1 denotes positive outcome and 0 denotes negative outcome.

Given that there are at most \( d < n \) positive items among in total \( n \) ones, all the \( d \) positive items can be efficiently and correctly identified on the condition that the testing matrix \( M \) is \( d \)-disjunct: any single column is not contained by the union of any other \( d \) columns. Owing to this property, each negative item will appear in at least one row (group) where all the positive items do not show up, therefore, by filtering all the items appearing in groups with negative outcomes, all the left ones are positive.

1.2 Extending and Improving GT for Modeling Network Problems

Although GT has been developed with a complete theoretical system and widely applied to medical testing and molecular biology during the past several decades \[16][64][62][22][14][18][21], it has plenty of room to grow in both theory and application facets.

On the one hand, despite the simple decoding method of \( d \)-disjunct matrix, is non-trivial to construct \[16][25] which may involve with complicated computations with high overhead, e.g., calculation of irreducible polynomials on Galois Field. How to either efficiently construct this matrix or adapt its concept to ease the construction hardness with testing performance guarantee, is of great interest.
On the other hand, this technique has rarely been used for network security and reliability problems due to the limitations in its conventional models and algorithms. For example, selecting networking elements into pools has to follow much more constraints (connection/location/power) than biological samples, which brings up the difficulties of applying the traditional technique models.

In this thesis, we extend the traditional GT models for network problems and provide an improved efficient construction of \(d\)-disjunct matrix, which eliminates the two shackles preventing the use of GT in network optimizations.

### 1.3 Combinatorial Optimizations for Network Security

Several variants of Denial-of-Service attacks are taken into consideration. The defense frameworks we provide use GT models as a backbone, but also involve Graph Optimization and Computation Geometry.

**Detecting Application-Layer DoS Attacks.** Application-Layer DoS aims at disrupting application service rather than depleting the network resource, has emerged as a larger threat to network services, compared to the classic DoS attack. Owing to its high similarity to legitimate traffic and much lower launching overhead than classic DDoS attack, this new assault type cannot be efficiently detected or prevented by existing detection solutions. To identify application DoS attack, we propose a GT based approach deployed on back-end servers, which not only offers a theoretical method to obtain short detection delay and low false positive/negative rate, but also provides an underlying framework against general network attacks. More specifically, we first extend classic GT model with size constraints for practice purposes, then re-distribute the client service requests to multiple virtual servers embedded within each back-end server machine, according to specific testing matrices. Base on this framework, we propose a 2-mode detection mechanism using some dynamic thresholds to efficiently identify the attackers. The focus of this work lies in the detection algorithms proposed and the corresponding theoretical complexity analysis.
Defending Reactive Wireless Jamming Attacks. During the last decade, Reactive Jamming Attack has emerged as a greatest security threat to wireless sensor networks, due to its mass destruction to legitimate sensor communications and difficulty to be disclosed and defended. Considering the specific characteristics of reactive jammer nodes, a new scheme to deactivate them by efficiently identifying all trigger nodes, whose transmissions invoke the jammer nodes, has been proposed and developed. Such a trigger-identification procedure can work as an application-layer service and benefit many existing reactive-jamming defending schemes. In our solution, on the one hand, we leverage several optimization problems to provide a complete trigger-identification service framework for unreliable wireless sensor networks. On the other hand, we provide an improved algorithm with regard to two sophisticated jamming models, in order to enhance its robustness for various network scenarios.

1.4 Combinatorial Optimizations for Network Reliability

Network reliability problems normally involve with fault tolerance of the networks. Many of them can also be abstracted as a discrete optimization problem and then addressed using GT, Graph Theory and other discrete math solutions. In this thesis, we study two classic reliability problems: faulty localization and vulnerability assessment, whose problem context are however quite novel.

Failure Localization in Optical Networks. Link failure localization has been an important and challenging problem for all-optical networks. The most general monitoring structure, called path-trial, is a light-path into which optical signals are launched and monitored. How to minimize the number of required path-trials is critical to the expense of this technique. Existing solutions are limited to localizing single link failure or handling only small networks. Moreover, some practical constraints, like lacking of knowledge of the failure quantity, are ignored. To overcome these limitations is prospective but quite challenging. To this end, we address the multi-link failure localization problem. On one hand, for moderate-size networks, we provide a tree-decomposition based centralized
algorithm; on the other hand, a random walk based localized algorithm for large-scale networks is proposed. In addition, we further adapt these two algorithms to cope with three practical constraints.

**QoS-aware Network Topological Vulnerability Assessment.** How to assess the topology vulnerability of a network has attracted more and more attentions recently. Due to the rapid growing number of real-time internet applications developed since the last decade, the discovery of topology weakness related to its quality of service (QoS) is of more interest. We provide a novel QoS-aware measurement for assessing the vulnerability of general network topologies. Specifically, we evaluate the vulnerability by detecting the minimum number of link failures that decrease the *satisfactory level* of the *QoS-Optimal* source-destination path to a given value, which means a topology with a smaller amount of such link failures is more vulnerable.

We formulate this process as a graph optimization problem called *QoSCE*, study its theoretical hardness/inapproximability and provide several exact and efficient heuristic algorithms for various QoS constraint amounts. To our best knowledge, this is the first graph-theoretical framework to evaluate QoS-aware topology vulnerability.

### 1.5 Thesis Organization

The organization of this thesis is as following.

Chapter 2 presents a new randomized construction of error-tolerant *d*-disjunct matrix, which achieves a smaller matrix size than the existing solutions and contributes to the good performances of the following group testing applications.

Chapter 3 targets at the application denial-of-service attacks, proposes a size-constraint group testing model and a 2-mode detection mechanism, which efficiently identifies the attackers by regulating the distribution of service requests to corresponding servers, according to three sequential/parallel algorithms for different network settings.

Chapter 4 illustrates a trigger-identification service which can be used to freeze and catch the sophisticated reactive jamming attackers. Besides an adapted group
testing model, this service leverages several optimization problem models and achieves theoretical performance guarantee in terms of the identification precision.

In Chapter 5, another application in localizing multiple link failures in All-Optical Networks is investigated. A graph-constrained group testing model is introduced to tackle this problem. This chapter shows a tree-decomposition centralized algorithm as well as a random walk based localized algorithm, whose theoretical correctness and efficiency is proved through theoretical and experimental evaluations.

A theoretical QoS vulnerability assessment framework is proposed in Chapter 6, where efficient algorithms are provided to investigate the robustness of any given network topologies.

As the last component, Chapter 7 sums up the whole thesis.
CHAPTER 2
A NEW CONSTRUCTION OF ERROR-TOLERANT DISJUNCT MATRIX

In order to handle errors in the testing outcomes, the error-tolerant non-adaptive group testing has been developed using \((d, z)\)-disjunct matrix, where in any \(d + 1\) columns, each column has a 1 in at least \(z\) rows where all the other \(d\) columns are 0. Therefore, a \((d, 1)\)-disjunct matrix is exactly \(d\)-disjunct. Straightforwardly, the \(d\) positive items can still be correctly identified, in the presence of at most \(z - 1\) test errors. In the literature, numerous deterministic designs for \((d, z)\)-disjunct matrix have been provided (summarized in [16]), however, these constructions often suffer from high computational complexity, thus are not efficient for practical use and distributed implementation.

On the other hand, to our best knowledge, the only randomized construction for \((d, z)\)-disjunct matrix due to Cheng’s work via \(q\)-nary matrix [11], which results in a \((d, z)\)-disjunct matrix of size \(t_1 \times n\) with probability \(p'\), where \(t_1\) is

\[
4.28d^2 \log \frac{2}{1 - p'} + 4.28d^2 \log n + 9.84dz + 3.92z^2 \ln \frac{2n - 1}{1 - p'}
\]

with time complexity \(O(n^2 \log n)\). Compared with this work, we advance a classic randomized construction for \(d\)-disjunct matrix, namely, random incidence construction [16][25], to generate \((d, z)\)-disjunct matrix which can not only generate comparably smaller \(t \times n\) matrix, but also handle the case where \(z\) is not known beforehand, instead, only the error probability of each test is bounded by some constant \(\gamma\). Although \(z\) can be quite loosely upperbounded by \(\gamma t\), yet \(t\) is not an input. The motivation of this construction lies in the real test scenarios, the error probability of each test is unknown and asymmetric, hence it is impossible to evaluate \(z\) before knowing the number of pools.

**Theorem 2.0.1.** \(M\) is \((d, z)\)-disjunct matrix with \(t = 2 \left( \frac{(d+1)^d+1}{d^d} \right) (z - 1 + \ln s + (d + 1) \ln n)\) rows with probability \((1 - \frac{1}{s})\) for a constant \(s\) where \(s\) can be arbitrarily large.
Algorithm 1 ETG construction

1: **Input:** \( n, d, z, s; \)
2: **Output:** \((d, z)\)-disjunct matrix with probability \((1 - \frac{1}{s})\)
3: Set \( p = \frac{1}{d + 1}, \)
4: Set \( t = 2 \left( \frac{1^{d+1}d+1}{d^d} \right) (z - 1 + \ln s + (d + 1) \ln n) \)
5: Construct a \( t \times n \) matrix \( M \) by letting each entry to be 1 with probability \( p \).
6: return \( M \)

Proof. \( M \) is not \((d, z)\)-disjunct matrix if for any single column \( c_0 \) and any other \( d \) columns \( c_1, \cdots c_d \), there are at most \( z - 1 \) rows where \( c_0 \) has 1 and all \( c_1, \cdots c_d \) have 0. By denoting \( p = (\frac{1}{s})' \), considering a particular column and \( d \) other columns in the matrix, the probability of such failure pattern is:

\[
\sum_{i=0}^{z-1} \binom{t}{i} [p(1 - p)^d]^i [1 - p(1 - p)^d]^{t-i}
\]

So use the union bound for all possible combinations and permutations of \((d + 1)\) columns, we have the failure possibility bounded by

\[
P_1 \leq (d + 1) \binom{n}{d + 1} \sum_{i=0}^{z-1} \binom{t}{i} [p(1 - p)^d]^i [1 - p(1 - p)^d]^{t-i}
\]

Here consider the CDF of binomial series and assume that \( z - 1 \leq tp(1 - p)^d \) (**assert 1**), we then have

\[
P_1 \leq n^{d+1} \exp\left(-\frac{(tp(1 - p)^d - z + 1)^2}{2tp(1 - p)^d}\right)
\]

by Chernoff bound. To bound this by \( \frac{1}{s} \), i.e.,

\[
P_1 \leq n^{d+1} \exp\left(-\frac{(tp(1 - p)^d - z + 1)^2}{2tp(1 - p)^d}\right) \leq \frac{1}{s}
\]

we can derive that (**assert 2**)

\[
p(1 - p)^d \leq \frac{z - 1 + \ln s + (d + 1) \ln n}{t} - \frac{\sqrt{\ln^2(sn^{d+1}) + 2(z - 1) \ln sn^{d+1}}}{t}
\]
(infeasible by assert 1)

or

\[ p(1 - p)^d \geq \frac{z - 1 + \ln s + (d + 1)\ln n}{t} + \frac{\sqrt{\ln^2(sn^{d+1}) + 2(z - 1)\ln sn^{d+1}}}{t} \]

Therefore, we can derived the lower bound

\[ t \geq 2 \left( \frac{(d + 1)^{d+1}}{d^d} \right) (z - 1 + \ln s + (d + 1)\ln n) \]

\[ \square \]

**Corollary 2.0.1.** Given that each test has an independent error probability \( \gamma \), \( M \) is \((d, z)\)-disjunct matrix with \( t = \frac{\tau \ln n(d+1)^2 - 2\tau(d+1)\ln \frac{1}{1 - p'}}{(\tau - \gamma(d+1))^2} \) with probability \((1 - \frac{1}{s})\) for arbitrary \( s \).

**Proof.** Substituting \( z \) by \( \gamma t \) in the proof above completes this proof. \[ \square \]

**Theorem 2.0.2.** The ETG algorithm produces a \((d, z)\)-disjunct matrix with probability \( p' \) where \( p' \) can be arbitrarily approaching 1.

- The worst-case number of rows of this matrix is bounded by
  \[ 3.78(d + 1)^2 \log n + 3.78(d + 1) \log \left( \frac{2}{1 - p'} \right) - 3.78(d + 1) + 5.44(d + 1)(z - 1) \]
  , much smaller than \( 4.28d^2 \log \frac{2}{1 - p'} + 4.28d^2 \log n + 9.84dz + 3.92z^2 \ln \frac{2n - 1}{1 - p'} \).

- Assume \( z \leq \gamma t \), the worst-case number of rows becomes \( t = \frac{\tau \ln n(d+1)^2 - 2\tau(d+1)\ln(1 - p')}{(\tau - \gamma(d+1))^2} \)
  where \( \tau = (d/(d + 1))^d \) and asymptotically \( t = O(d^2 \log n) \).

**Theorem 2.0.3.** The time complexity of the ETG algorithm is \( O(d^2 n \log n) \), smaller than \( O(n^2 \log n) \), provided that \( d < \sqrt{n} \).
CHAPTER 3
A GROUP TESTING BASED DETECTION OF APPLICATION DENIAL-OF-SERVICE ATTACKERS

Denial-of-Service (DoS) attack, which aims to make a service unavailable to legitimate clients, has become a severe threat to the Internet security [24]. Traditional DoS attacks mainly abuse the network bandwidth around the Internet subsystems and degrade the quality of service by generating congestions at the network [66][24]. Consequently, several network based defense methods have tried to detect these attacks by controlling traffic volume or differentiating traffic patterns at the intermediate routers [69][54]. However, with the boost in network bandwidth and application service types recently, the target of DoS attacks have shifted from network to server resources and application procedures themselves, forming a new application DoS attack [52][66].

As stated in [66], by exploiting flaws in application design and implementation, application DoS attacks exhibit three advantages over traditional DoS attacks which help evade normal detections: malicious traffic is always indistinguishable from normal traffic, adopting automated script to avoid the need for a large amount of “zombie” machines or bandwidth to launch the attack, much harder to be traced due to multiple re-directions at proxies. According to these characteristics, the malicious traffic can be classified into legitimate-like requests of two cases: (i) at a high inter-arrival rate, (ii) consuming more service resources. We refer to these two cases as “high-rate” and “high-workload” attacks respectively throughout this chapter.

Since these attacks usually do not cause congestion at the network level, thus bypass the network-based monitoring system [24], detection and mitigation at the end-system of the victim servers have been proposed [52][30][48]. Among them the DDoS shield [52] and CAPTCHA-based defenses [30] are the representatives of the two major techniques of system-based approaches: session validation based on legitimate behavior profile and authentication using human-solvable puzzles. By enhancing the accuracy of the suspicion assignment for each client session, DDoS shield can
provide efficient session schedulers for defending possible DDoS attacks. However, the overhead for per-session validation is not negligible, especially for services with dense traffic. CAPTCHA-based defenses introduce additional service delays for legitimate clients and are also restricted to human-interaction services.

A kernel observation and brief summary of our method is: the identification of attackers can be much faster if we can find them out by testing the clients in group instead of one-by-one. Thus the key problem is how to group clients and assign them to different server machines in a sophisticated way, so that if any server is found under attack, we can immediately identify and filter the attackers out of its client set. Apparently, this problem resembles the group-testing (GT) theory [16] which aims to discover defective items in a large population with the minimum number of tests where each test is applied to a subset of items, called pools, instead of testing them one by one. Therefore we apply GT theory to this network security issue and propose specific algorithms and protocols to achieve high detection performance in terms of short detection latency and low false positive/negative rate. Since the detections are merely based on the application service status of the victim servers, no individually signature-based authentications or data classifications are required, thus it may overcome the limitations of the current solutions.

To our best knowledge, the first attempts to apply GT to networking attack defense are proposed in parallel by Thai et.al [63] (which is the preliminary work of this journal) and Khattab et al [34]. The latter proposed a detection system based on “Reduced-Randomness Nonadaptive Combinatorial Group Testing”[25]. However, since this method only counts the number of incoming requests rather than monitoring the server status, it is restricted to defending high-rate DoS attacks and cannot handle high-workload ones).

In a system viewpoint, our defense scheme is to embed multiple virtual servers within each physical back-end server, and map these virtual servers to the testing
pools in GT, then assign clients into these pools by distributing their service requests to different virtual servers. By periodically monitoring some indicators (e.g., average responding time) for resource usage in each server, and comparing them with some dynamic thresholds, all the virtual servers can be judged as “safe” or “under attack”. By means of the decoding algorithm of GT, all the attacker can be identified. Therefore, the biggest challenges of this method are three-fold: (1) How to construct a testing matrix to enable prompt and accurate detection. (2) How to regulate the service requests to match the matrix, in practical system. (3) How to establish proper thresholds for server source usage indicator, to generate accurate test outcomes.

Similar to all the earlier applications of GT, this new application to network security requires modifications of the classical GT model and algorithms, so as to overcome the obstacle of applying the theoretical models to practical scenarios. Specifically, the classical GT theory assumes that each pool can have as many items as needed and the number of pools for testing is unrestricted. However, in order to provide real application services, virtual servers cannot have infinite quantity or capacity, i.e., constraints on these two parameters are required to complete our testing model.

Our main contributions in this chapter are as follows:

- Propose a new size-constrained GT model for practical DoS detection scenarios.
- Provide an end-to-end underlying system for GT-based schemes, without introducing complexity at the network core.
- Provide multiple dynamic thresholds for resource usage indicators, which help avoid error test from legitimate bursts and diagnose servers handling various amount of clients.
- Present three novel detection algorithms based on the proposed system, and show their high efficiencies in terms of detection delay and false positive/negative rate via theoretical analysis and simulations.

Besides application DoS attacks, our defense system is applicable to DoS attacks on other layers, e.g. protocol-layer attack – SYN flood [54] where victim servers are exhausted by massive half-open connections. Although these attacks occur in different
layers and of different styles, the victim machines will gradually run out of service resource and indicate anomaly. Since our mechanism only relies on the feedback of the victims, instead of monitoring the client behaviors or properties, it is promising to tackle these attack types.

The chapter is organized as follows. Section 3.2 presents the attacker model and victim/detection model of our system. In Section 3.3, we propose the detection strategy derived from the adjusted GT model, and illustrate the detailed components in the presented system. Description and latency analysis of three concrete detection algorithms are included in Section 3.4, while Section 3.5 provides the results of our simulation in terms of the detection delay and false positive rate. We reach our conclusion by summatng our contributions and providing further discussions over false positive/negative rate in Section 3.6.

3.1 Related Works

Numerous defense schemes against DoS have been proposed and developed [46]. By taking effect at different levels, these defenses can be categorized into network-based mechanisms and system-based ones.

Existing network-based mechanisms aim to identify the malicious packets at the intermediate routers or hosts [69][32], by either checking the traffic volumes or the traffic distributions. However, the application DoS attacks have no necessary deviation in terms of these metrics from the legitimate traffic statistics, therefore, network-based mechanisms cannot efficiently handle this attack type.

On the other hand, plenty of proposed system-based mechanisms tried to capture attackers at the end server by authentications [35][30] or classifications [52][32]. Honeypots [35] are used to trap attackers who attempt to evade authentications, and can efficiently mitigate flood attacks. However, this mechanism kind relies on the accuracy of authentication. Once the attacker passes the authentication, all the productive servers are exposed and targeted. Classification-based methods study on
the traffic behavior model, for example, the inter-arrival time distribution [52] and the bandwidth consumption pattern [32]. With a rigorous classification criteria, abnormal packets can be filtered out, but this requires individually checking each session and can significantly increase the overhead of the system. Besides these two defense types, Kandula et al. augments the protected service with CAPTCHA puzzles which are solvable by human clients but zombie machines. This technique can eliminate the service requests from bonnets, but it also brings in additional operations at the client ends, which delays the consumer from receiving services.

In parallel with us, Khattab et al. proposed another system-based “live-baiting” defense scheme by applying group testing theory to application DoS detection. Based on a “high probability d-disjunct” [25] matrix, their system avoided large modifications to the network topology by spanning the incoming requests to multiple virtual serves for testing, using encrypted token, which will also benefit our detection system. However, the live-baiting mechanism needs to be improved in several facets: Firstly, the number of states (pools for testing) is not as small as $O(d)$, as stated. Therefore the number of states to maintain is still approaching $O(n)$. Secondly, the static threshold High-Water-Mark(HWM), which is the arriving request aggregate at each server periodically, is not accurate for the possible asymmetric request distribution on the assigned servers to one client in round-robin. Moreover, this threshold cannot be used to servers with dynamic client set. Thirdly, the virtual servers are merely request counters, and cannot tackle the high-workload requests. In our system, virtual servers with limited service resources are mapped to testing pools and tested with dynamic thresholds, thus can overcome these limitations.

### 3.2 Problem Models

#### 3.2.1 Attacker Model

The maximum destruction caused by the attacks includes the depletion of the application service resource at the server side, the unavailability of service access to
legitimate user, and possible fatal system errors which require rebooting the server for recovery. We assume that any malicious behaviors can be discovered by monitoring the service resource usage, based on dynamic value thresholds over the monitored objects. Data manipulation and system intrusion are out of this scope.

Similar to [52], we assume that application interface presented by the servers can be readily discovered and clients communicate with the servers using HTTP/1.1 sessions on TCP connections. We consider a case that each client provides a non-spoof ID (e.g. SYN-cookie [38]), which is utilized to identify the client during our detection period. Despite that the application DoS attack is difficult to be traced, by identifying the IDs of attackers, the firewall can block the subsequent malicious requests.

As mentioned above, the attackers are assumed to launch application service requests either at high inter-arrival rate or high workload, or even both. The term “request” refers to either main request or embedded request for HTTP page. Since the detection scheme proposed will be orthogonal to the session affinity, we do not consider the repeated one-shot attack mentioned in [52].

We further assume that the number of attackers $d \ll n$ where $n$ is the total client amount. This arises from the characteristics of this attack. Due to the benefits of virtual servers we employ, this constraint can be relaxed, but we keep it for the theoretical analysis in the current work.

### 3.2.2 Victim/Detection Model

The victim model in our general framework consists of multiple back-end servers, which can be web/application servers, database servers and distributed file systems. We do not take classic multi-tier web servers as the model, since our detection scheme is deployed directly on the victim tier and identifies the attacks targeting at the same victim tier, thus multi-tier attacks should be separated into several classes to utilize this detection scheme. The victim model along with front-end proxies are shown in Fig. 3-1.
We assume that all the back-end servers provide multiple types of application services to clients using HTTP/1.1 protocol on TCP connections. Each back-end server is assumed to have the same amount of resource with the others. Moreover, the application services to clients are provided by $K$ virtual private servers ($K$ is an input parameter), which are embedded in the physical back-end server machine and operating in parallel. Each virtual server is assigned with equal amount of static service resources, e.g., CPU, storage, memory and network bandwidth. The operation of any virtual server will not affect other virtual servers in the same physical machine. The reasons for utilizing virtual servers are two-fold: first, each virtual server can reboot independently, thus is feasible for recovery from possible huge destruction; second, the state transfer overhead for moving clients among different virtual servers is much smaller than the transfer among physical server machines.

As long as the client requests arrive at the front-end proxy, they will be distributed to multiple back-end servers for load-balancing, whether session stick or not. Notice that our detection scheme is behind this front-end tier, so the load-balancing mechanism
is orthogonal to our setting. On being accepted by one physical server, one request will be simply validated according to the list of all identified attacker IDs (black-list). If it passes the authentication, it will be distributed to one virtual servers within this machine by means of virtual switch. This distribution depends on the testing matrix generated by the detection algorithm. By periodically monitoring the average response time to service requests and comparing it with specific thresholds fetched from a legitimate profile, each virtual server is associated with a “negative” or “positive” outcome. Thereout, a decision over the identities of all clients can be made among all physical servers, as discussed further in the following Section 3.3.

3.3 Strategy and Detection System

3.3.1 Size Constraint Group Testing

As mentioned in the detection model, each testing pool is mapped to a virtual server within a back-end server machine. Although the maximum number of virtual servers can be extremely huge, since each virtual server requires enough service resources to manage client requests, it is practical to have the virtual server quantity (maximum number of servers) and capacity (maximum number of clients that can be handled in parallel) constrained by two input parameters $K$ and $w$ respectively. Therefore, the traditional GT model is extended with these constraints to match our system setting.

**Definition 3.3.1.** Size Constraint Group Testing (SCGT): For any 01-matrix $M$, let

$$w_i = \sum_{j=1}^{n} M[i,j]$$

be the weight of the $i^{th}$ pool, and $t$ be the number of pools. The model is for identifying $d$ defected items in the minimum period of time, by constructing matrix $M_{t \times n}$ and conducting group testing procedures based on that, where $w_i \leq w$ for given $w$ and $t \leq K$ for given server quantity $K$.

The maximum number of attackers $d$ is assumed known beforehand. Scenarios with non-deterministic $d$ are out of the scope of this chapter. In fact these scenarios can be readily handled by first testing with an estimated $d$, then increasing $d$ if exactly $d$ positive items are found.
Table 3-1. Main Notations

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>number of clients</td>
</tr>
<tr>
<td>$K$</td>
<td>number of server machines</td>
</tr>
<tr>
<td>$d$</td>
<td>maximum number of attacks, $1 \leq d \leq n$ (practically, $d \ll n$)</td>
</tr>
<tr>
<td>$w$</td>
<td>maximum number of clients that can be handled by a virtual server in parallel</td>
</tr>
<tr>
<td>$r_{\text{leg}}$</td>
<td>the maximum inter-arrival rate of legitimate requests from one client</td>
</tr>
<tr>
<td>$P$</td>
<td>length of time of each testing round</td>
</tr>
<tr>
<td>$A$</td>
<td>the set of servers used for testing</td>
</tr>
<tr>
<td>$S$</td>
<td>the set of suspect clients</td>
</tr>
<tr>
<td>$I$</td>
<td>the set of servers which are under attack</td>
</tr>
<tr>
<td>$L$</td>
<td>the set of clients connecting to negative servers</td>
</tr>
<tr>
<td>$G$</td>
<td>the number of non-testing machines</td>
</tr>
</tbody>
</table>

3.3.2 Detection System

The implementation difficulties of our detection scheme are three-fold: *how to construct proper testing matrix $M$, how to distribute client requests based on $M$ with low overhead* and *how to generate test outcome with high accuracy*. We will address the last two in this section and leave the first one to the next section.

Table 3-1 includes the notations used throughout the chapter.

3.3.2.1 System overview

As mentioned in the detection model, each back-end server works as an independent testing domains, where all virtual servers within it serve as testing pools. In the following sections, we only discuss the operations within one back-end server, and it is similar in any other servers. The detection consists of multiple testing rounds, and each round can be sketched in four stages (Fig. 3-3):

First, generate and update matrix $M$ for testing.

Second, “assign” clients to virtual servers based on $M$. The back-end server maps each client into one distinct column in $M$ and distributes an encrypted token queue to it. Each token in the token queue corresponds to an 1-entry in the mapped column. i.e., client $j$ receives a token with destination virtual server $i$ iff $M[i,j] = 1$. By piggybacked
with one token, each request is forwarded to a virtual server by the virtual switch. In addition, requests are validated on arriving at the physical servers for faked tokens or identified malice ID. This procedure ensures that all the client requests are distributed exactly as how the matrix $M$ regulates, and prevents any attackers from accessing the virtual servers other than the ones assigned to them.

Third, all the servers are monitored for their service resource usage periodically, specifically, arriving request aggregate and average response time are recorded and compared with some dynamic thresholds to be shown later. All virtual servers are associated with positive or negative outcomes accordingly.

Fourth, decode these outcomes and identify legitimate or malicious IDs. By following the detection algorithms (presented in the next section), all the attackers can be identified within several testing rounds.

To lower the overhead and delay introduced by the mapping and piggybacking for each request, the system is exempted from this procedure in normal service state. As shown in Fig. 3-2, the back-end server cycles between two states, which we refer as NORMAL mode and DANGER mode. Once the estimated response time (ERT) of any virtual server exceeds some profile-based threshold, the whole back-end server will transfer to the DANGER mode and execute the detection scheme. Whenever the average response time (ART) of each virtual server falls below the threshold, the physical server returns to NORMAL mode.
3.3.2.2 Configuration details

Several critical issues regarding the implementation are as follows.

**Session State Transfer.** By deploying the detection service on the back-end server tier, our scheme is orthogonal with the session state transfer problem caused by the load-balancing at the reverse proxies (front-end tier). To simplify the discussion of implementation details, we assume that the front-end proxies distribute client requests strictly even to the back-end servers, i.e., without considering session stick issues. The way of distributing token queues to be mentioned later is tightly related to this assumption. However, even if the proxies conduct more sophisticated forwarding, the token queue distribution can be readily adapted by manipulating the token piggybacking mechanism at the client side accordingly.

Since the testing procedure requires distributing intra-session requests to different virtual servers, the overhead for maintaining consistent session state is incurred. Our motivation of utilizing virtual servers is to decrease such overhead to the minimum, since multiple virtual servers can retrieve the latest client state though the shared memory, which resembles the principle of Network File System (NFS). An alternative way out is to forward intra-session requests to the same virtual server, which calls for longer testing period for each round (to be further discussed in the P section), but we prefer faster detection in this chapter and thus adopt the former method.
Matrix Generations. The testing matrix $M$, which regulates distributing which client request to which server, poses as the kernel part of this chapter. All the three algorithms proposed in the next section are concerning with the design of $M$ for the purpose of shortening the testing period and decreasing the false positive/negative rate. Since the detection phase usually undergoes multiple testing rounds, $M$ is required to be re-generated at the end of each round and used for the next round. The time overhead for calculating this $M$ is quite low and will be shown via analytical proofs in Section 3.4.

Distributing Tokens. Two main purposes of utilizing tokens are associating each client with a unique, non-spoofed ID and assigning them to a set of virtual servers based on the testing matrix. On receiving the connection request from a client, each back-end server responses with a token queue where each token is of 4-tuple: (client ID, virtual server ID, matrix version, encrypted key). “client ID” refers to the unique non-spoofed index for each client, which we assume unchanged during our testing period (DANGER mode). “virtual server ID” is the index of each virtual server within the back-end server. This can be implemented as a simply index value, or through a mapping from the IP addresses of all virtual servers. The back-end server blocks out-of-date tokens by checking their “matrix version” value, to avoid messing up the request distribution with non-uniform matrices. With regard to the “encrypted key”, it is an encrypted value generated by hashing the former three values and a secured service key. This helps rule out any faked tokens generated by attackers.

Assume that the load-balancing at the proxies is strictly even for all back-end servers, the client has to agree on piggybacking each request with a token at the head of one token queue, and then the next request with the token at the head of the next token queue, when receiving this application service. Notice that there are multiple token queues released by multiple back-end servers, it is non-trivial to implement the correct request distribution.
**Length of Testing Round** $P$. Since we need to distribute the requests exactly the way $M$ regulates, it is possible that: if $P$ is too short, some clients may not have distributed their requests to all their assigned servers, i.e., not all the 1-entries in $M$ are matching with at least one request. For an attacker who launches a low-rate high-workload attack, its high-workload requests may only enter a part of its assigned servers, so false negative occurs in this case. However, if $P$ is too long, the detection latency will be significantly increased. In this chapter, we predefine a proper value for $P$, which is long enough for each client to spread their requests to all the assigned servers, i.e., each column needs to be mapped with at least $\sum_{i=1}^{t} M[i,j]$ requests. Therefore,

$$P = \max_{j=1}^{n} \sum_{i=1}^{t} M[i,j]/r_{\text{min}}$$

where $r_{\text{min}}$ denotes the minimum inter-arrival request rate, provides a theoretical lower bound of $P$.

**Legitimate Profile.** The legitimate profile does not refer to the session behavior profile for legitimate clients, but instead records the distribution of the $ART$ on a virtual server receiving only legitimate traffic. Malicious requests are certainly to generate destructions to the victim server machines, whose $ART$ will usually be much higher than that of normal cases. Therefore, $ART$ can work as an indicator of the application resource usage. However, the resource usage varies for different time intervals (peak/non-peak time) due to the change of client quantity, so we also investigate the $ART$ distributions regarding each possible number of clients, assuming that there are at most $n$ clients.

A sample construction of this profile is: The distributions of $ART$ in legitimate traffic at different time intervals for several weeks are obtained after the system is established. Legitimate traffic can be achieved by de-noising measures and ruling out the influences of potential attacks [55]. The $i^{th}$ entry of the profile records the distribution of $ART$ on a virtual server with $i \in [1, n]$ clients. Specifically, we assign $i$ clients to a virtual server,
and evenly divide the round length $P$ into 100 sub-intervals (which we refer as $P_{\text{sub}}$ throughout the chapter), and monitor the server $ART$ within each sub-interval $P_{\text{sub}}$.

**NORMAL mode and Transfer Threshold.** To decrease the length of detection period, where additional state maintenance overhead and service delay cannot be avoided, the back-end server cluster provides normal service to clients, without any special regulations. This is referred as NORMAL mode, which takes $ERT$ (estimated response time) as a monitoring object. Notice that $ERT$ is an expected value for $ART$ in the near future, and can be computed via

$$ERT = (1 - \alpha) \cdot ERT + \alpha \cdot ART$$

where $\alpha$ is decay weight for smoothing fluctuations. Since the inter-arrival rate and workload of client request can be randomized distributed, it is difficult to perfectly fit the $ART$ distribution using classic distribution functions. Considering that Normal Distribution [13] can provide an approximate fitting to $ART$, we adopt a simplified threshold based on *Empirical Rule*: If any virtual server has an $ERT > \mu + 4\sigma$ ($\mu$ and $\sigma$ denote the expected value and standard deviation of the fitted $ART$ distribution), the back-end server is probably undergoing an attack, and thus transfers to DANGER mode for detection. To enhance the accuracy of this threshold, more sophisticated distributions can be employed for fitting the samples, however, higher computation overhead will be introduced for the threshold.

**DANGER mode and Attack Threshold.** In DANGER mode, besides the $ART$ values, the back-end server simultaneously counts the arriving request aggregate within each $P_{\text{sub}}$ for each virtual server. The motivation of this strategy arises from the fact that, high-rate DoS attacks always saturate the server buffer with a large amount of malicious requests. By counting the number of arriving requests periodically, possible high-rate attacks can be detected even before depleting the service resources.
We pre-define $R$ as a maximum legitimate inter-arrival rate, and derive a upperbound of the arriving request aggregate $C_j$ for virtual server $j$ within period $P_{sub}$ as:

$$C_j = \sum M[j, i] \cdot \left[ \frac{\sum_{s=j+1}^{t} M[s, i] + R \cdot P_{sub}}{\sum_{k=1}^{t} M[k, i]} \right]$$

Once this threshold is violated in a virtual server, it is for sure undergoing high-rate attacks or flash-crowd traffic. A positive outcome can be generated for this testing round.

The outcome generation by monitoring the ART value for a virtual server consists of the following detailed steps:

1. check the ART distribution profile for the values of parameters $\mu$ and $\sigma$, as mentioned in the NORMAL mode section;

2. among all the 100 sub-intervals $P_{sub}$ (each is $P/100$) within the current testing period $P$, if no violation: $ART \geq \mu + 4\sigma$ occurs in any $P_{sub}$, the virtual server gets negative outcome for this testing round;

3. if for some sub-interval $P_{sub}$s, $ART \geq \mu + 4\sigma$ occurs, this virtual server is in danger, either under attack, or undergoing flash-crowd traffic. In this case, we wait till the end of this round to get the distribution of ART values for all $P_{sub}$s for further decision;

4. if the ratio of “danger” sub-intervals $P_{sub}$s with $ART \geq \mu + 4\sigma$, over the total sub-interval amount (100 in this case), exceeds some quantile regulated by the Empirical Rule, e.g., quantile 4% for confidence interval $[\mu + 2\sigma, \infty)$, this virtual server will be labeled as positive;

5. for the other cases, the virtual server will have a negative outcome.

After identifying up to $d$ attackers, the system remains in the DANGER mode and continues monitoring the ART for each virtual server for one more round. If all the virtual servers have negative outcomes, this back-end server is diagnosed as healthy and return to to the NORMAL mode, otherwise further detections will be executed (because of error tests).

With regard to the two objects monitored, $C_j$ provides different counting thresholds for different virtual servers, while ART profile supplies dynamic thresholds for virtual
server containing different amounts of client. The combination of these two dynamic thresholds are helps decrease the testing latency and false rate.

**Multiple Testing Domains.** With respect to the whole back-end server cluster, multiple testing domains operate during the DANGER mode. Tests over the same client within different testing domains provide a benefit for decreasing the false positive rate. In another words, due to the existence of error tests, some legitimate clients might be identified as malice in some domains, while not in the others. By analyzing the ratio of events that it is identified as malice, the system can provide a more reliable identification result. On the other hand, it is often the case that attackers aim to attack a majority of servers, so they can still be caught even if some domains fail to detect them. Therefore, both false positive and negative cases can be further alleviated.

### 3.4 Detection Algorithms and Latency Analyses

Based on the system framework above, we propose three detection algorithms $SDP$, $SDoP$ and $PND$ in this section. Note that the length of each testing round is a pre-defined constant $P$, hence we analyze the algorithm complexity in terms of the number of testing rounds for simplicity. Since all these algorithms are deterministic and introduce no false positive/negative rate (compared to the non-deterministic construction in [34]), all the test errors result from the inaccuracy of the dynamic threshold. We will discuss on this in Section 3.6.

*Based on our assumption, there is a unique *ID* for each client, thus the two items “*ID*” and “*client*” are interchangeable in this section. In addition, all the following algorithms are executed in each physical back-end server, which is an independent testing domain as mentioned. Therefore, the term “*servers*” denote the $K$ virtual servers in this section.*

For simplicity, we assume that $n \equiv 0 \pmod{K}$, $n > w \geq d \geq 1$ and $|A| \geq d + 1$. Notice that the last inequality holds in practice because of the properties of application DoS attacks indicated above.
3.4.1 Sequential Detection with Packing

This algorithm investigates the benefit of classic sequential group testing, i.e., optimizing the grouping of the subsequent tests by analyzing existing outcomes. Similar to traditional sequential testing, each client (column) only appears in one testing pool (server) at a time. However, to make full use of the available $K$ servers, we have all servers conduct test in parallel. Details can be found in Algorithm 2.

Algorithm 2 Sequential Detection with Packing (SDP)

1: while $|S| \neq 0$ do
2: for all server $i$ in $A$ do
3: $w_i \leftarrow \lceil \frac{|S|}{|A|} \rceil$ // Assign even number of distinct clients to each server.
4: end for
5: $G \leftarrow \lceil \frac{n-(|S|-(|A| \\cup L))}{w} \rceil$ // Identified legitimate IDs exempt from the following tests. Their subsequent requests are directed to and handled by $G$ non-testing servers, which are selected from the $K$ servers and only provide normal services (no testing). We call this process as “packing” the clients into non-testing servers.
6: for all server $i$ under attack do
7: $Q \leftarrow$ set of IDs on $i$
8: if $|Q| = 1$ then
9: $S \leftarrow S \setminus Q$ // This ID belongs to an attacker. Add it into the black-list.
10: end if
11: end for
12: $S \leftarrow S \setminus L$ // All IDs on safe (with negative test outcome) servers are identified as legitimate.
13: $A \leftarrow \{\text{server } 1, \ldots, \text{ server } (K - G)\}$ // Select the first $G$ servers as non-testing machines.
14: end while

Algorithm Description. The basic idea of SDP algorithm can be sketched as follows. Given a set $S$ of suspect IDs, we first randomly connect them to the available testing servers in set $A$, where each server will receive requests from approximately the same number of clients, roughly $w_i = \lceil \frac{|S|}{|A|} \rceil$. For each test round, we identify the IDs on the negative servers as legitimate clients, and “pack” them into a number $G$ of non-testing machines. Since they need no more tests, only normal services will be provided for the following rounds. As more testing servers will speed up the tests, given at most $K$
server machines in total, $G$ is then supposed to be minimized to the least, as long as all identified legitimate clients can be handled by the non-testing $w$-capacity servers. Hence
$$G = \left\lceil \frac{n - (|S| - (|A| - d)/w)}{w} \right\rceil.$$

With the assumption $|A| > d + 1$, we have at least $|A| - d$ servers with negative outcomes in each testing round, hence at least $(|A| - d)w_i$ legitimate IDs are identified.

If any server containing only one active ID is found under attack, the only ID is surely an attacker. Then its ID is added into the black-list and all its requests are dropped.

Iterate the algorithm until all IDs are identified, malicious or legitimate. Via the “packing” strategy, legitimate clients can exempt from the influence of potential attacks as soon as they are identified.

**Example.** A 3-round detection example in Fig. 3-4A illustrates the execution of this algorithm. Matrices from up to bottom are corresponding to three detections respectively. Column 1 and 3 (in circle) are corresponding to two malicious clients, while the others are legitimate ones.
Given \( n = 10, d = 2, K = 5, \) and \( w = 4. \) Let \( |S| = 10, |A| = 5, \) and \ID 1, 3 be the attackers. In the first round, \( w_i = 2 \) yields two \ID s on each server. According to our configuration, server 1 and server 2 (containing \ID 1 and 3 respectively) will indicate being under attack, i.e., \( |I| = 2 \) and \( |A \setminus I| = 3, \) pack legitimate \ID s on the other three servers into \( G = 2 \) servers. Update \( |S| = 4 \) and \( |A| = 3. \) In the second round, \( w_i = 2 \) yields at most two \ID s on each server, assume that server 1 and server 3 are under attack, then again pack the legitimate \ID 4 in server 2 to server 4 (server 5 is already full). In the third round, server 1 and 3 only contain an attacker \ID 1 and 3, respectively. Therefore, the two attackers are identified at the end of this round.

**Performance Analysis.** With regard to the computational overhead of the matrix \( M \) in this context, it is a re-mapping of the suspect clients to the testing servers based on quite trivial strategies and previous testing outcomes, thus is negligible and up to \( O(1). \)

The time cost of each testing round is exactly \( P, \) including the re-computation time of the testing matrix.

Besides this, the overall time complexity of this algorithm in terms of the number of testing rounds are depicted in the following theorems.

**Lemma 3.4.1.** *The number of available testing server machines: \( |A| \in [K - \lceil \frac{n-d}{w} \rceil, K]. \)*

**Proof.** In each round, the total number of legitimate \ID s \( n - (|S| - (|A \setminus I|)w_i) \) is non-decreasing. Hence, the number of servers used for serving identified legitimate IDs, \( G = \lceil \frac{n - (|S| - (|A \setminus I|)w_i)}{w} \rceil \) is non-decreasing. Therefore \( |A| = K - G \) will finally converge to \( K - \lceil \frac{n-d}{w} \rceil. \)

**Theorem 3.4.1.** *The SDP algorithm can identify all the \( d \) attackers within at most \( O(\log_{\frac{K}{d}} \frac{n}{d}) \) testing rounds, where \( K' = K - \lceil \frac{n-d}{w} \rceil. \)*

**Proof.** In each round, at most \( d \) servers are under attack, i.e., at most \( \frac{d|S|}{|A|} \) IDs remain suspect. Since at the end of the program, all legitimate IDs are identified, at most \( d \)
malicious IDs remain suspect. If they happen to be handled by \( d \) different servers, i.e., each positive server contains exactly one attacker, then the detection is completed. If not, one more testing round is needed. Assume we need at most \( j \) testing rounds in total, then \( n \left( \frac{d}{|A|} \right)^{-j} = d \) where \( K - \lceil \frac{n-d}{w} \rceil \leq |A| \leq K \). Therefore, \( j \leq \log_{\left( \frac{K'}{d} \right)} \frac{n}{d} + 1 \), where \( K' = K - \lceil \frac{n-d}{w} \rceil \).

3.4.2 Sequential Detection without Packing

Algorithm 3 Sequential Detection without Packing (SDoP)

1: \( w_i \leftarrow \) number of IDs on server \( i \);  
2: \textbf{for all} server \( i \) do  
3: \( w_i \leftarrow \lceil \frac{s}{2} \rceil \) // Evenly assign clients to servers as SDP did.  
4: \textbf{end for}  
5:  
6: \textbf{while} \( |S| \neq 0 \) do  
7: Randomly reassign \( |S| \) suspect IDs to \( K \) servers, and keep legitimate IDs unmoved.  
8: \( L \leftarrow \) set of IDs on safe servers  
9: \( S \leftarrow S \setminus (S \cap L) \) // \( |S \cap L| \) IDs are identified as legitimate.  
10: \textbf{for all} server \( i \) under attack do  
11: \( Q \leftarrow \) set of IDs on \( i \)  
12: \textbf{if} \( |Q \cap S| = 1 \) then  
13: \( S \leftarrow S \setminus Q \) // The clients in \( Q \cap S \) are attacker and added into the black-list.  
14: \textbf{end if}  
15: \textbf{end for}  
16: \textbf{for all} servers \( i \) do  
17: \textbf{if} \( w_i = w \) // The volume approaches the capacity. then  
18: Reassign all \( n - |S| \) legitimate IDs to \( K \) servers, and go to 6 // Load balancing.  
19: \textbf{end if}  
20: \textbf{end for}  
21: \textbf{end while}  

Considering the potential overload problem arises from the “packing” scheme adopted in SDP, we propose another algorithm where legitimate clients do not shift to other servers after they are identified. This emerges from the observation that legitimate clients cannot effect the test outcomes since they are negative. Algorithm 3 includes
the abstract pseudocode of this SDoP scheme. Notice that in this algorithm for the DANGER mode, requests from one client are still handled by one server, as SDP did.

**Algorithm Description.** The basic idea of the SDoP algorithm can be sketched below. Given a suspect IDs set $S$ with initial size $n$, evenly connect them to the $K$ server machines, similar to SDP in the first round. Afterwards, still connect suspect IDs to the $K$ servers instead of $|A|$ available ones. For the identified legitimate IDs, never move them until their servers are to be overloaded. In this case, reassign all legitimate IDs over the $K$ machines to balance the load. For server with positive outcome, the IDs active on this server but not included by the set of identified legitimate ones, i.e., suspect IDs, will be still identified as suspect. However, if there is only one suspect IDs of this kind in a positive server, this ID is certainly an attacker.

**Example.** Fig. 3-4B contains a 2-round detection example to illustrate the execution of this algorithm. Assume $n = 10$, $d = 2$, $K = 5$ and the client 1, 3 are the attackers. In the first testing round, $w_i = 2$ yields two IDs on each server, then only server 1 and 2 will have positive outcomes. Update suspect ID set $S = \{1, 2, 3, 4\}$. In the second testing round, since no attackers have been identified and no servers are overloaded, we re-assign all suspect IDs to $K$ servers while do not move the legitimate clients, and server 1 and 4 turn out to be under attack. Hence the legitimate ID set is $L = \{2, 4, 5, 6, 7, 8, 9, 10\}$ and the two attackers are captured within two testing rounds.

**Performance Analysis.** It is trivial to see that the computation overhead of the testing matrix $M$ is similar to that of the SDP algorithm, therefore can be ignored. The following theorems exhibits the overall detection delay of the algorithm, in terms of the number of testing rounds.

**Lemma 3.4.2.** The number of IDs $w_i$ on server $i$ does not exceed server capacity $w$ till round $j$ where $j = \log_2 \frac{n}{n-w(K-d)}$.

**Proof.** At most $d$ servers get positive outcomes in each round. If we assume round $j$ is the last round that has no overloaded servers, then the maximum number of IDs on one
server at round $j$ is: $w_{\text{max}}^j = \sum_{i=1}^{j} \left( \frac{n}{k} \right) \left( \frac{d}{k} \right)^{i-1}$. Since round $j + 1$ will have at least one overloaded server, we have $w_{\text{max}}^j = w$, then $j = \log \frac{k}{d} \frac{n}{n-w(k-d)}$.

Lemma 3.4.3. All malicious IDs are identified within at most $O(\log \frac{n}{d})$ testing rounds if $w \geq \frac{n-d}{K-d}$.

Proof. In each round with no overloaded servers, at most $\frac{d}{k} |S|$ legitimate IDs remain suspect. After $i = \log \frac{k}{d} \frac{n}{K}$ rounds at most $K$ suspect IDs are left, SO we need only one more round to finish the testing. Hence, we need $i + 1 = \log \frac{k}{d} \frac{n}{d}$ rounds. Since it requires that no servers are overloaded within these rounds, we have $i + 1 < j = \log \frac{k}{d} \frac{n}{n-w(k-d)}$, which yields $w \geq \frac{n-d}{K-d}$.

Lemma 3.4.4. All malicious IDs are identified within at most $O(\log \frac{n}{d})$ testing rounds if $1 \leq w \leq \frac{n-d}{K-d}$.

Proof. If $w \leq \frac{n-d}{K-d}$, reassignments for legitimate IDs will be needed after round $j = \log \frac{k}{d} \frac{n}{n-w(k-d)}$. Assume that the algorithm needs $x$ more testing rounds in addition to $i + 1 = \log \frac{k}{d} \frac{n}{d}$ to reassign and balance legitimate IDs on all servers, since the position changes of these legitimate IDs will neither influence test outcomes nor delay the identifications of suspect set $S$ (they take place simultaneously), we hence have

\[ x \leq i + 1 - j \leq \log \frac{k}{d} \frac{n}{d} - \log \frac{k}{d} \frac{n}{n-w(k-d)} \leq \log \frac{k}{d} \frac{n-w(K-d)}{d} \leq \log \frac{n}{d} \]

Therefore, the total number of testing rounds is at most $O(\log \frac{n}{d})$.
Theorem 3.4.2. The SDoP algorithm can identify all attackers within at most $O\left(\log_{\frac{n}{d}} n\right)$ testing rounds.

Proof. Directly from Lemma 3.4.3 and 3.4.4. 

3.4.3 Partial Non-adaptive Detection

Considering that in the two sequential algorithms mentioned, we cannot identify any attackers until we isolate each of them to a virtual server with negative outcome. It will not only lead to a huge number of testing rounds, but also too luxury to let one virtual server to handle only one client. Therefore, we propose a hybrid of sequential and non-adaptive method in this section. In this scenario, the requests from the same client will be received and responded by different servers in a round-robin manner. Different from SDP and SDoP, a $d$-disjunct matrix is used as the testing matrix in this scheme and attackers can be identified without the need of isolating them into servers.

A matrix $M$ is called $d$-disjunct if no single column is contained in the boolean sum of any other $d$ columns, and all positive items can be identified within one round. Numerous construction methods for $d$-disjunct matrix have been proposed in [17, 20]. Among them Du’s method [17] has a better performance in terms of the size complexity(number of rows). Therefore we adopt this method to generate $M$ for each testing round.

Lemma 3.4.5. Let $T(n)$ be the smallest number of rows in the obtained $d$-disjunct matrix by Algorithm 3, then

$$T(n) = \min\{(2 + o(1))d^2 \log_2 n / \log_2 (d \log_2 n), n\}$$

Next we introduce our PND detection method using $d$-disjunct matrix, for now we assume that the row weight of the constructed matrix does not exceed the server capacity $w$. Algorithms with respect to the weight-constraint will be investigated in the future. Due to the dynamic value of $T(n)$ and maximum machine amount $K$, the algorithm differs in two cases.
Algorithm 4 Construct a $d$-disjunct matrix based on a finite field $GF(q)$

1: function dDisjunct($n$)
2: Consider a finite field $GF(q)$, choose $s,q,k$ satisfying:
3: \[ kd \leq s \leq q, \quad n \leq q^k \]
4: if $qs \geq n$ then
5: Return an $n \times n$ identity matrix
6: else
7: \[ \triangleright \text{Construct matrix} \ A_{s \times n} \]
8: for $x \in [0, s - 1]$ do
9: for all polynomials $p_j$ of degree $k$ do
10: $A[x, p_j] \leftarrow p_j(x)$
11: end for
12: end for
13: \[ \triangleright \text{Construct matrix} \ M_{t \times n} \]
14: for $x \in [0, s - 1]$ do
15: for $y \in [0, q - 1]$ do
16: for all polynomial $p_j$ of degree $k$ do
17: if $A[x, p_j] == y$ then
18: $M[(x, y), p_j] \leftarrow 1$
19: else
20: $M[(x, y), p_j] \leftarrow 0$
21: end if
22: end for
23: end for
24: end for
25: Return $M$
26: end if

**Case 1:** $K \geq T(n)$. All $n$ suspect IDs can be assigned to $T(n)$ servers according to a $M_{T(n) \times n}$ $d$-disjunct matrix, and thus identified with one testing round.

**Case 2:** $K < T(n)$. With inadequate servers to identify all $n$ clients in parallel, we can iteratively test a part of them, say $n' < n$, where $T(n') \leq |A|$. The detailed scheme is depicted as follows and the pseudocode is shown in Algorithm 5.

**Algorithm Description.** First evenly assign $n$ suspect IDs to all $K$ servers, as the other algorithms did. After the first round, we can pack all identified legitimate IDs into $G = \lceil \frac{n - |S|}{w} \rceil$ servers. Therefore, $|A| = K - G$ servers are available for testing afterwards. If $|A| \geq T(|S|)$, we can construct a $d$-disjunct matrix $M_{|A| \times |S|}$ by calling function $dDisjunct(|S|)$ in Algorithm 3 and identify all IDs in one more testing rounds.
Nevertheless, if $|A| < T(|S|)$, we can firstly find the maximum $n'$ ($1 \leq n' \leq |S|$) such that $T(n') \leq |A|$; then secondly partition set $S$ into two disjoint sets $S_1$ and $S_2$ with $|S_1| = n'$; and thirdly pack all $S_2$ to $G$ non-testing server machines and call subroutine $dDisjunct(|S_1|)$ to identify all IDs in $S_1$; then drop the attackers and pack all legitimate IDs into $G$ machines and continue to test $S = S_2$. Iterate this process until all IDs are identified.

**Algorithm 5** Partial Non-adaptive Detection with $K < T(n)$

1. Evenly assign $n$ IDs to $K$ servers with no two servers having the same client IDs. Pack all legitimate IDs on into $G$ servers.
2. $|A| \leftarrow K - G$ // Update the number of testing servers.
3. `if $|A| \geq T(|S|)$ then`
4. `Run $dDisjunct(|S|)$, decode from the obtained $d$-disjunct matrix and identify all IDs; $S \leftarrow 0$; // Testing finishes.`
5. `else`
6. `while $|S| \neq 0$ do`
7. `Partition $S$ into $S_1, S_2$ where $|S_1| = n'$ and $n' = \max n'' \in [1, |S|]$ satisfying $T(n'')||A|$
8. `Run $dDisjunct(|S_1|)$, decode from the obtained $d$-disjunct matrix and identify all IDs in $S_1$`
9. `Pack legitimate IDs in $S_1$ to $G$ machines, update $G$ and $A$; $S \leftarrow S_2$. // Iteration with suspect ID set $S_2$.`
10. `end while`
11. `end if`

**Performance Analysis.** The time complexity of the Algorithm 4 is $O(sq n)$, i.e. $O(Kn)$ with $sq \leq K$. As mentioned in Section 3.3, this can be decreased to $O(1)$ time cost in real-time. Since it is too complicated to use the upperbound $(2+o(1))d^2 \log_2 n / \log_2(d \log_2 n))$ for $T(n)$ in analysis, we then use $T(n) = n$ to get a rough estimate on the algorithm complexity in terms of the number of rounds. Moreover, we further investigate its performance based on an optimal $d$-disjunct, so as to demonstrate the potential of this scheme.

**Lemma 3.4.6.** The PND algorithm identifies all the attackers within at most $O(dw^2/(Kw− w − n))$ testing rounds.
Proof. Let us consider the worst case. In each round, assume no attackers are identified and filtered out previously, we have $T(n') + \lceil \frac{n-n'}{w} \rceil = K$. Since $T(n') \leq n'$ then $n' \geq \frac{Kw-w-n}{w-1}$. Therefore the maximum number of testing rounds needed is: $\frac{dn}{K} / \frac{Kw-w-n}{w-1} \leq dw^2/(Kw-w-n) = O(dw^2/(Kw-w-n))$.

This complexity for PND is not as good as that of the previous two algorithms. However we are still interested in finding out how it works if we have the $d$-disjunct matrix construction method improved. Since some lower bounds for $T(n)$ have been proposed, we then use one of them [18] to compute $T(n)$ and investigate the corresponding performance of PND next.

**Lemma 3.4.7.** (D’yachkov-Rykov lower bound) [18] Let $n > w \geq d \geq 2$ and $t > 1$ be integers. For any superimposed $(d-1, n, w)$-code ($(d, n, w)$-design) $X$ of length $t$ ($X$ is called a $(d-1)$-disjunct matrix with $t$ rows and $n$ columns), the following inequality holds:

$$t \geq \lceil \frac{dn}{w} \rceil$$

The following are the related performance analysis based on this lower bound of $T(n)$.

**Corollary 3.4.1.** In the PND algorithm, given $w \geq d \geq 1$, we have: $n' = \min\{\frac{|A|w}{d+1}, n - Kw + w + |A|w\}$

Proof. According to Lemma 3.4.7, with number of columns $n'' \in [1, |S|]$, we have

$$|A| \geq \left\lceil \frac{(d+1)n''}{w} \right\rceil \geq \frac{(d+1)n''}{w} \Rightarrow n' = \max n'' \leq \frac{|A|w}{d+1}$$

Meanwhile in the PND algorithm, for each round $i$, we have:

$$|A_i| \geq K - \left\lceil \frac{n - n'}{w} \right\rceil \Rightarrow n' \leq n - Kw + w + |A|w$$

**Lemma 3.4.8.** In any testing round $i$:
1\( a' = \frac{|A|w}{d+1} \text{ when } K \in (d, \frac{dw+n+w}{w}) \);

2\( a' = n - Kw + w + |A|w \text{ when } K \in [k_1, \frac{dn+n+w}{w}) \text{ with} \)

\[
k_1 = \frac{dw + w + n + \sqrt{(dw + w + n)^2 + 4wnd^2}}{2w}
\]

Proof. According to Corollary 3.4.1,

1we have

\[
K \leq \frac{dw + n + w}{w} \iff wK \leq dw + n + w
\]

\[
|A| \geq d + 1 \iff \frac{|A|wd}{d+1} \geq wd
\]

hence

\[
n + w - Kw + |A|w \geq \frac{|A|w}{d+1}
\]

2In order to get

\[
\frac{|A|w}{d+1} \geq n - Kw + w + |A|w
\]

we need

\[
|A| \leq \frac{((K - 1)m - n)(d + 1)}{wd}
\]

which requires

\[
K - \frac{(K - d)n}{Kw} \leq \frac{((K - 1)m - n)(d + 1)}{wd}
\]

hence we need

\[
wK^2 - (dw + w + n)K - nd^2 \geq 0
\]

Solving this inequality, we have \( K \in [k_1, +\infty) \). Note if \( K \geq \left\lceil \frac{dn+n}{w} \right\rceil \), we need not do partitions in PND algorithm and since

\[
k_1 \leq \frac{dn + n + w}{w}
\]

we have

\[
K \in [k_1, \frac{dn + n + w}{w})
\]

Moreover

\[
k_1 > \frac{dw + w + n}{w} \geq \frac{d^2 + w + n}{w}
\]

there are thus no overlaps between these two intervals.
Therefore, we split \( K \in (d, +\infty) \) into four disjoint intervals and study which interval of value \( K \) yields \( O(1) \) testing rounds in worst case besides the interval \( K \in \left[ \frac{dn+n+w}{w}, +\infty \right) \) shown above, as well as complexity for other intervals.

- **I**: \( K \in (d, \frac{dw+n+w}{w}] \) yields \( n' = \frac{|A|w}{d+1} \);
- **II**: \( K \in (\frac{dw+n+w}{w}, k_1) \) yields \( n' = \min\{\frac{|A|w}{d+1}, n - Kw + w + |A|w\} \);
- **III**: \( K \in [k_1, \frac{dn+n+w}{w}) \) yields \( n' = n - Kw + w + |A|w \);
- **IV**: \( K \in \left[ \frac{dn+n+w}{w}, +\infty \right) \) yields ONE testing round in total.

**Lemma 3.4.9.** The PND algorithm needs at most \( O(1) \) testing rounds with \( K \in [k_2, \frac{dw+n+w}{w}] \), where

\[
d \leq \frac{w + \sqrt{w^2 - 4n^2(n-w)}}{2(n-w)}
\]

and

\[
k_2 = \frac{n + w + \sqrt{n^2 + w^2 + 2nw - 4n^2w + 4d^2wn + 4dwn}}{2w}
\]

**Proof.** Since at least one server gets positive outcome at the first testing round, we have

\[
|A_0| \geq K - \left\lceil \frac{(K-1)n}{Kw} \right\rceil
\]

With simple algebraic computations, we can reach the interval \([k_2, \frac{dw+n+w}{w}]\) on the condition that

\[
d \leq \frac{w + \sqrt{w^2 - 4n^2(n-w)}}{2(n-w)}
\]

within interval I; however, for interval II and III, no such detailed subintervals of \( K \) yielding \( O(1) \) testing rounds can be obtained.

\[\square\]

**Lemma 3.4.10.** Within interval I, PND algorithm can identify all IDs with \( O(d + \frac{K}{\sqrt{n}}) \) testing rounds.

**Proof.** We derive the time complexity from the following recurrence:

**Starting round 0:** \( |S_0| \leq \frac{dn}{K}, \) and \( K - \left\lceil \frac{(K-1)n}{Kw} \right\rceil \leq |A_0| \leq K - \left\lceil \frac{(K-d)n}{Kw} \right\rceil \)
Ending round $T$: $0 < |S_T| \leq \frac{|A_T|}{d+1}$

Iteration: For $\forall i \in [0, T-1]$ we have

$$|S_{i+1}| = |S_i| - \frac{|A_i|w}{d+1}$$

and

$$K - \left\lceil \frac{n - |S_{i+1}|}{w} \right\rceil \leq |A_{i+1}| \leq K - \left\lfloor \frac{n - |S_{i+1}| - d}{w} \right\rfloor$$

hence

$$\begin{cases} 
|A_{i+1}| \geq K - \frac{n}{w} + \frac{|S_i|}{w} - \frac{|A_i|}{d+1} - 1 \\
S_0 \leq \sum_{i=0}^{T} \frac{|A_i|w}{d+1}
\end{cases}$$

In order to estimate the maximum time cost, use $|S_0| = \frac{dn}{K}$ to initiate the worst starting case. Solving this recurrence, we get the following inequality:

$$\frac{Kw}{2(d+1)} T^2 - (\frac{Kw}{2(d+1)} + \frac{dn}{K} - K + \frac{n}{w} + w) T - (K - \frac{(K-1)n}{w} - \frac{dn(d+2)}{K} + w - 1) \leq 0,$$

therefore

$$T \leq \frac{d+1}{Kw} \left( \alpha + \sqrt{\frac{\beta}{d+1} + \alpha^2} \right)$$

where

$$\alpha = \frac{Kw}{2(d+1)} + \frac{dn}{K} + \frac{n}{w} - K + w$$

and

$$\beta = 2K^2w - 2K^2n + 2Kn - 2Kw + 2Kw^2 - 2d(d+2)nw$$

Since $\frac{n}{w} \leq K$, $wK \leq dw + n + w$ and $n > w$, we have $\alpha > 0$ and $\beta > 0$. So with trivial computation we can get

$$T \leq \frac{2\alpha(d+1)}{Kw} + \sqrt{\frac{\beta}{d+1}}$$

$$< 1 + \frac{2(d+1)^2}{K} + \sqrt{\frac{(4K-2)(d+1)}{n} + \frac{2(d+1)}{d+1}}$$

$$< 1 + 2(d+1) + \sqrt{\frac{4K(d+1)}{n} + 2}$$

$$< 3 + \sqrt{2} + 2d + \frac{2K}{\sqrt{n}}$$

50
Therefore, \( PND \) will complete the identification within at most \( O(d + \frac{K}{\sqrt{n}}) \) testing rounds.

\[ \square \]

Note that since \( K \) is always much smaller than \( n \), the complexity will approach \( O(d) \) in fact.

**Lemma 3.4.11.** Within interval III, PND can identify all IDs with at most \( O(d) \) testing rounds.

**Proof.** Similarly we derive the time complexity from the following recurrence:

Starting round 0: \( |S_0| = \frac{dn}{K} \) and \( K - \lceil \frac{(K-1)n}{Kw} \rceil \leq |A_0| \leq K - \lceil \frac{(K-d)n}{Kw} \rceil \)

Ending round \( T \): \( 0 < |S_T| \leq n - Kw + w + |A_T|w \)

Iteration: \( \forall i \in [0, T-1], |S_{i+1}| = |S_i| - (n - Kw + w + |A_i|w) \) and \( K - \lceil \frac{n - |S_{i+1}|}{w} \rceil \leq |A_{i+1}| \leq K - \lceil \frac{n - |S_{i+1}| - d}{w} \rceil \). By solving this recurrence, we can finally get

\[ T \leq 2d + 1 + 2\sqrt{\frac{1 + d}{4}} \]

Therefore, \( PND \) will complete the identification within at most \( O(d) \) testing rounds.

\[ \square \]

**Corollary 3.4.2.** Within interval II, PND algorithm can identify all IDs with \( O(d + \frac{K}{\sqrt{n}}) \) testing rounds.

**Proof.** According to Lemma 3.4.10 and 3.4.11, the time complexity of \( PND \) algorithm depends on the value of \( n' \) at each round, and since \( n' \) within interval II oscillates between \( \frac{|A|w}{d+1} \) and \( n - Kw + w + |A|w \), the time complexity is at most \( O(d + \frac{K}{\sqrt{n}}) \).

\[ \square \]

**Theorem 3.4.3.** Given \( 1 \leq d \leq w \), the PND algorithm can identify all IDs within

- at most \( O(d + \frac{K}{\sqrt{n}}) \) testing rounds when \( K \in (d, k_1) \), whilst at most \( O(1) \) testing rounds when \( K \in [k_2, \frac{dw+n+w}{w}] \) on condition that \( d \leq \frac{w+\sqrt{w^2-4n^2(n-w)}}{2(n-w)} \);
- at most \( O(d) \) testing rounds when \( K \in [k_1, \frac{dn+n+w}{w}] \);
• **at most** $O(1)$ **testing rounds** when $K \in \left[\frac{dn+w}{w}, +\infty\right)$; where $k_1 = \frac{dw+w+n+\sqrt{(dw+w+n)^2+4wnd^2}}{2w}$ and $k_2 = \frac{n+w+\sqrt{n^2+w^2+2nw-4n^2w-4d^2wn+4dwn}}{2w}$.

Despite the number of needed testing rounds differs for these three algorithms above, the time complexity of calculating each testing round for these algorithms are approximate in practice. It is trivial to see that this costs for **SDP** and **SDoP** are negligible, but not for **PND** algorithm which involves polynomial computation on Galois Field. However, considering that the upperbound of both the number of clients $n$ and attackers $d$ are estimated, the detection system can pre-compute the $d$-disjunct matrices for all possible $(n, d)$ pairs offline, and fetch the results in real-time. Therefore, the overhead can be decreased to $O(1)$ and the client requests can be smoothly distributed at the turn of testing rounds without suffering from long delays of matrix update.

### 3.5 Simulation Configurations and Results

To demonstrate the theoretical complexity results shown in the previous section, we conduct a simulation study on the proposed system, in terms of four metrics: **average testing delay** $T$ which refers to the length of the time interval from attackers starting sending requests till the system recovering to NORMAL mode; **average false positive rate** $f_p$ and **false negative rate** $f_n$; as well as the **average number of testing rounds** $R_{test}$ which stands for the number of testing rounds needed for identifying all the clients by each algorithm.

#### 3.5.1 Configurations

The purpose of this simulation is not to fully implement a detection system for use, but instead to validate all the theoretical results we derived. Although we make several assumptions below to simply the implementation of the simulation environments, as we will show later, these issues are orthogonal to the performance of our scheme. Therefore, the simulation results could provide a reliable overview of the applicability and practical performance of our framework for general network types.
To this end, we implement a simulator in Java by modeling both the $n$ clients and $K$ virtual servers as independent threads. Consider that all back-end servers are independent testing domains, we only simulate one back-end server as we did for the algorithms, and thus does not test the flash crowd scenario which was mentioned and settled in Section 3.3 using multiple back-end servers.

In order to mimic the real client (including attacker) behavior and dynamic network environment, we implement the client/server system as follows:

- each legitimate client joins in and leaves the system at random times which are uniformly distributed, while the attacker threads arrive at time $t = 30s$ and keep live until being filtered out (More complicated client behaviors might be more favored, but since our scheme contains a learning phase over various client behavior and adjusts the threshold accordingly, the performance of this system will not significantly decay by applying it to a different network type, i.e., the network type is orthogonal with the scheme.)
- both legitimate and malicious clients send requests which are with a random inter-arrival rate and CPU processing time (workload) to the virtual servers, however, legitimate one have a much smaller random range than that of the attackers.
- each virtual server is equipped with an infinite request buffer and all the client requests arrive at the buffers with 0 transmission and distribution delays, as well as 1 ms access time for retrieving states from the shared memory; each server handles the incoming requests in its own buffer in FCFS manner and responds to the client on completing the corresponding request; the average response time and incoming request aggregate are recorded periodically to generate the test outcomes by comparing them to the dynamic thresholds fetched from established legitimate profiles.

The purpose of assuming both transmission and distribution delays to be 0 is to quantify the length of the whole detection phase (testing latency). With regard to the transmission delay, it can be large due to the geographical distances in large-scale distributed system and possibly can bring up the testing latency if the client sends request in a stop-and-wait manner (it does not send a request until the previous requests are all responded, therefore the request rate is quite low and the length of each testing round is required to be longer), yet since the detection will be completed
just in several rounds, such increases in the detection length is not significant. The assumption of \( \mathcal{O} \) distribution delay is also practical, since the computational overheads for the testing matrix and dynamic thresholds can be negligible by pre-computing and fetching the results from the profiles. For the 1ms state maintenance time, since all the clients are residing in one physical servers, and all the virtual servers can quickly retrieve the client states from the shared memory, this is also a practical assumption.

With regard to the details of client behavior, the legitimate request inter-arrival rate is randomized from 1 to 3 request per ms, and the legitimate workload is randomized from 1 to 3 ms CPU processing time. On the contrary, the malicious request inter-arrival rates range from 5 to 20 per ms, and malicious workload range from 5 to 20 ms CPU time. Although requests with arbitrarily large rate or workload are favored by attackers, they are in fact easier to be discovered, so we consider malicious requests with small margin from legitimate ones.

3.5.2 Results

By setting \( K = 50, n = 1000, w = 100, d = 40, P = 1 \) second, we first show the efficiency of our detection system using PND algorithm, in terms of reducing the service resource usage ratio \((SRUR)\) and the average response time \((ART)\), as shown in Fig. 3-5A and 3-5B. The values of \( SRUR \) and \( ART \) climb up sharply at \( t = 30s \) when the attack starts, and then gradually falls to normal before \( t = 100s \). Therefore, it takes only 70s for the system to filter out attackers and recover to normal status. Notice that the length of actual detection period should be shorter than this, because the threshold of \( ART \) for the system to convert from DANGER mode back to NORMAL mode is slightly higher than normal \( ART \). Therefore, the system \( SRUR \) and \( ART \) will recover to normal shortly after the detection period ends.

In the following we show the robustness of the performance toward different environment setting: an increasing number of 1) given virtual servers \( K \), 2) malicious clients \( d \), 3) all clients \( n \).
In Fig. 3-6, we simulated to identify \( d = 10 \) attackers out of \( n = 1000 \) clients with the number of virtual servers ranging in \([25, 200]\). It can be seen, one one hand, that all the false negative (Fig. 3-6(a)) and positive (Fig. 3-6(b)) rates are upper-bounded by 5% and decreasing as \( K \) goes up for all the three algorithms. This makes sense since the most possible case for an attacker to succeed hiding itself is that it is accompanied by many clients with low request rate and workloads in a testing pool. In this case, it is quite possible that the aggregate inter-arrival rates and workloads in this server is still less than that of a server with a same number of legitimate clients. Therefore, the less clients serviced by each server, the less possibly this false negative case happens. On the other hand, the testing latencies and number of testing rounds keep declining from less than 11s and 4 rounds respectively, which is because the identification will be speeded up with more available virtual servers. With respect to the three different algorithms, they obtained approximate performances except that \( SDoP \) incurs slightly higher false negative rate than the other two. This is because of those legitimate clients who are identified at earlier stages and staying in the servers till the end of the detection. Since their request rates and workloads are likely to be smaller than normal (that is why they are identified earlier), they may camouflage attackers in the following rounds.
Due to the space limitation, we briefly summarized the results for the following two experiments. In Fig. 3-7, we simulated the identifications for $n = 1000$, $K = 50$, $w = 100$, $P = 1s$ with $[1, 15]$ attackers, and found that all the values of the four measures slowly increase as $d$ goes up. Overall, the false negative/positive rates are limited to less than 5% and the testing latencies are smaller than 6 seconds with 5 testing rounds. Similar as the previous experiments, $PND$ and $SDP$ exhibits better performances than $SDoP$ in terms of false negative rate, but for the other measures, they are approximately the same.

Fig. 3-8 shows its robustness for the cases $d = 10$, $K = 50$, $w = 100$, $P = 1s$ with $[1000, 2000]$ clients. Apparently, both the false rates and number of testing rounds keep stably below 5% and 4 rounds respectively, toward increasing client amount. The
testing latencies grow up from 10 seconds to 20 seconds for all three algorithms, due to the increasing time costs for state maintenance toward a double number of clients (from 1000 to 2000). However, this small latency is still tolerable in real-time applications and can be further reduced by decreasing the state maintenance time cost within the same physical machine.

Overall, the simulation results can be concluded as follows:

- in general, the system can efficiently detect the attacks, filter out the malicious clients, and recover to NORMAL mode in a short period of time, in real-time network scenarios;
- all the three detection algorithms can complete the detection with short latency (less than 30s) and low false negative/positive rate (both less than 5%) for up to 2000 clients. Thus they are applicable to large scale time/error-sensitive services;
Figure 3-8. Robustness by different total number of clients $n$

- the $PND$ and $SDP$ algorithms achieve slightly better performance than the $SDoP$ algorithm. Furthermore, the efficiency of the $PND$ algorithm can be further enhanced by optimizing the $d$-disjunct matrix employed;

- the detection delay can be further reduced by decreasing the state maintenance time cost.

### 3.6 Conclusions and Discussions

#### 3.6.1 Conclusions

We proposed a novel technique for detecting application DoS attack by means of a new constraint-based group testing model. Motivated by classic GT methods, three detection algorithms were proposed and a system based on these algorithms was implemented. Theoretical analysis and preliminary simulation results demonstrated
the outstanding performance of this system in terms of low detection latency and false positive/negative rate.

Our focus of this chapter is to apply group testing principles to application DoS attacks, and provide an underlying framework for the detection against a general case of network assaults, where malicious requests are in-distinguishable from normal ones. We presented several promising detection algorithms to shorten the testing latency for real-time implementation, while our future work will be around several limitations, which include: (1) in SDP and SDoP, the identification of any single attacker can only be completed by isolating it in a virtual server with positive outcome. This would probably bring up testing overhead, and we would improve the sequential algorithms to overcome this limitation. (2) more efficient \(d\)-disjunct matrix (with less rows) is required to enhance the performance of the PND algorithm. What is more, the constraint in the row weight \((w)\) is necessary to be considered in the matrix construction. Hence we will propose new construction method for the matrix, but it is non-trivial and can be a major theoretical work for another paper. (3) there are still some implementation details left to be handled, e.g., the overhead of maintaining the state transfer among virtual servers, which can be further decreased by more sophisticated techniques. (4) Notice that both the false positive and negative rates of this system have been shown to be quite low in the simulation, however, we still can further decrease them via false-tolerant group testing methods, as discussed next.

### 3.6.2 Discussions over False Rate

The false positive/negative rate of this detection system can possibly originate from either inaccuracy of matrix \(M\) or the phase of outcome generation.

(1) The accuracy of testing matrices refers to whether these matrices are exactly \(d\)-disjunct. Note that in [34], the matrices were generated using randomized construction method[25], thus were \(d\)-disjunct with high probability but not deterministic. Hence, false positive/negative rates are inevitable. On the contrary, Du’s matrix construction algorithm
has no such indeterminacies, and thus guarantees the accuracy of our $d$-disjunct matrices for testing.

(2) In our system, not only the triggering of detections (DANGER mode), but also the generation of outcomes for tests, are dependent upon predefined thresholds. Since these thresholds are obtained by studying legitimate traffics, constructing legitimate profile for ART distributions and approximately fitting them by Normal Distribution, inaccuracies are thus brought in. Although sincerely reflecting the dynamic traffic in real networks, the proactive learning on legitimate traffic cannot completely avoid the influence of the traffic bursts from legitimate users, moreover, not all hostile behaviors or bandwidth abuses can be ruled out. Therefore, the ART samples obtained possibly tower above that of legitimate traffic.

With respect to the fitting for the ART distributions, previous studies have concluded that none existing distributions can perfectly fit them [13][78][8], yet different distributions can provide best fittings for different scenarios. Considering the computational time complexity for fitting with several classic distributions, like Normal Distribution and Diffusion, ex-Gaussian, Gamma, Weibull[78], we adopt Normal Distribution with the shortest time cost. However, with relatively large bias from the real distribution, Normal Distribution possibly incurs inaccuracies. We will search for better distribution to enhance our system performance in the future work.
CHAPTER 4
A TRIGGER IDENTIFICATION SERVICE FOR DEFENDING REACTIVE JAMMERS IN WIRELESS SENSOR NETWORKS

Since the last decade, the security of wireless sensor networks (WSNs) has attracted numerous attentions, due to its wide applications in various monitoring systems and invulnerability toward sophisticated wireless attacks. Among these attacks, jamming attack where a jammer node disrupts the message delivery of its neighboring sensor nodes with interference signals, has become the most critical threat to WSNs. Thanks to the efforts of researchers toward this issue, as summarized in [73], various efficient defense strategies have been proposed and developed. However, a reactive variant of this attack, where jammer nodes stay quite until an ongoing legitimate transmission (even has a single bit) is sensed over the channel, emerged recently and called for stronger defending system and more efficient detection schemes.

Existing countermeasures against Reactive Jamming attacks consist of jamming (signal) detection and jamming mitigation.

On the one hand, detection of interference signals from jammer nodes is non-trivial due to the discrimination between normal noises and adversarial signals over unstable wireless channels. Numerous attempts to this end monitored critical communication related objects, such as Receiver Signal Strength (RSS), Carrier Sensing Time (CST), Packet Delivery Ratio (PDR), compared the results with specific thresholds, which were established from basic statistical methods and multi-modal strategies [59][73]. By such schemes, jamming signals could be discovered, however, how to locate and catch the jammer nodes based on these signals is much more complicated and has not been settled.

On the other hand, various network diversities are investigated to provide mitigation solutions [60]. Spreading spectrum [73][28][57] making use of multiple frequency bands and MAC channels, Multi-path routing benefiting from multiple pre-selected routing paths [60] are two good examples of them. However, in this method, the capability of
jammers are assumed to be limited and powerless to catch the legitimate traffic from the camouflage of these diversities. However, due to the silent behavior of reactive jammers, they have more powers to destruct these mitigation methods. To this end, other solutions are in great need. A mapping service of jammed area has been presented in [70], which detects the jammed areas and suggests that routing paths evade these areas. This works for proactive jamming, since all the jammed nodes are having low PDR and thus incapable for reliable message delay. However, in the case of reactive jamming, as we will show later, this is not always the case. Only a proportion of these jammed nodes, named as trigger nodes, whose transmissions wake up the reactive jammers, are required to be blocked to avoid the jamming effects.

In this chapter, we present an application-layer real-time trigger-identification service for reactive-jamming in wireless sensor networks, which promptly provides the list of trigger-nodes using a lightweight decentralized algorithm, without introducing neither new hardware devices, nor significant message overhead at each sensor node.

This service exhibits great potentials to be developed as reactive jamming defending schemes. As an example, by excluding the set of trigger nodes from the routing paths, the reactive jammers will have to stay idle since transmissions can be sensed. Even though the jammers move around and detect new sensor signals, the list of trigger nodes will be quickly updated, so are the routing tables. As another example, without prior knowledge of the number of jammers, the radius of jamming signals and specific jamming behavior types, it is quite hard to locate the reactive jammers even the jammed areas are detected (e.g. by [70]). However, with the trigger nodes localized, the possible locations of reactive jammers are significantly narrowed down.

Although the benefits of this trigger-identification service are exciting, its hardness is also obvious, which dues to the efficiency requirements of identifying the set of trigger nodes out of a much large set of victim nodes, that are affected jamming signals from reactive jammers with possibly various sophisticated behaviors. To address these
problem, a novel randomized error-tolerant group testing scheme as well as minimum disk cover for polygons are proposed and leveraged.

The basic idea of our solution is to first identify the set of victim nodes by investigating corresponding links’ PDR and RSS, then these victim nodes are grouped into multiple testing teams. Once the group testing schedule is made at the base station and routed to all the victim nodes, they then locally conducts the test to identify each of them as a trigger or non-trigger. The identification results can be stored locally for reactive routing schemes or delivered to the base station for jamming localization process.

In the remainder of this chapter, we first present the problem definition in Section 4.2, where the network model, victim model and attacker models are included. Then we introduce three kernel techniques for our scheme, Randomized Error-Tolerant Non-adaptive Group Testing, Clique-independent Set and Minimum Disk Cover in a Simple Polygon in Section 4.3. The core of this chapter: trigger identification procedure and its error-tolerant extension toward sophisticated jammer behaviors are presented respectively in Section 4.4 and 4.5. A series of simulation results for evaluating the system performance and validating the theoretical results are included in Section 6.5. We also present some related works in Section 5.1 and summarize the whole chapter in Section 6.6.

4.1 Related Works

Existing countermeasures against jamming attacks in WSN can be categorized into two facets: signal detection and mitigation, both of which have been well studied and developed with various defense schemes. On the one hand, a majority of detection methods focus on analyzing specific object values to discover abnormal events, e.g., Xu et. al [75] studied a multi-model (PDR, RSS) to consistently monitor jamming signals. Work based on similar ideas [10][51][39] improved the detection accuracy by investigating sophisticated decision criteria and thresholds. However, reactive jamming attacks, where the jammer node are not continuously active and thus unnecessary to
cause huge deviations of these variables from normal legitimate profiles, cannot be efficiently tackled by these methods. In addition, some recent works proposed methods for detecting jammed areas [70] and directing normal communications bypass possible jammed area using wormhole [9]. These solutions can effectively mitigate jamming attacks, but their performances rely on the accuracy of detection on jammed areas, i.e. the transmission overhead would be unnecessarily brought up if the jammed area is much larger than its actual size. On the other hand, mitigation schemes which benefit from channel surfing [74], frequency hopping and spatial retreats[73], reactively help legitimate nodes escape from the jammed area or frequency. Unfortunately, being lack of pre-knowledge over possible positions of hidden reactive jammer nodes, legitimate nodes cannot efficiently evade jamming signals, especially in dense sensor network when multiple mobile nodes can easily activate reactive jammer nodes and cause the interference. For the sake of overcoming these limitations above, in [56] we studied on the problem of identification trigger nodes with a short period of time, whose results can be employed by jamming-resistant routing schemes, to avoid the transmissions of these trigger nodes and deactivate the reactive jammer nodes. In this chapter, we complete this trigger identification procedure as a lightweight service, which is prompt and reliable to various network scenarios.

4.2 Problem Models and Notations

4.2.1 Network Model

We consider a wireless sensor network consisting of \( n \) sensor nodes and one base station (larger networks with multiple base stations can be split into small ones to satisfy the model). Each sensor node is equipped with omnidirectional antennas, \( m \) radios for in total \( k \) channels throughout the network, where \( k > m \). For simplicity, the power strength in each direction is assumed to be uniform, so the transmission range of each sensor can be abstracted as a constant \( r_s \) and the whole network as a unit disk graph (UDG) \( G = (V, E) \), where any node pair \( i, j \) is connected iff the Euclidean distance between \( i, j \):
\[ \delta(i,j) \leq r_s. \] We leave asymmetric powers and polygonal transmission area for further study.

### 4.2.2 Attacker Model

We consider both a basic attacker model and several advanced attacker models in this chapter. In the next sections, we will first illustrate our framework solution toward the basic attacker model, and then validate its performance toward multiple advanced attacker models theoretically and experimentally.

#### 4.2.2.1 Basic attacker model

Conventional reactive jammers [73] are defined as malicious devices, which keep idle until they sense any ongoing legitimate transmissions and then emit jamming signals (packet or bit) to disrupt the sensed signal (called jammer wake-up period), instead of the whole channel, which means once the sensor transmission finishes, the jamming attacks will be stopped (called jammer sleep period). Three concepts are introduced to complete this model.

**Jamming range** \( R \). Similar to the sensors, the jammers are equipped with omnidirectional antennas with uniform power strength on each direction. The jammed area can be regarded as a circle centered at the jammer node, with a radius \( R \), where \( R \) is assumed greater than \( r_s \), for simulating a powerful and efficient jammer node. All the sensors within this range will be jammed during the jammer wake-up period. The value of \( R \) can be approximated based on the positions of the boundary sensors (whose neighbors are jammed but themselves not), and then further refined.

**Triggering range** \( r \). On sensing an ongoing transmission, the decision whether or not to launch a jamming signal depends on the power of the sensor signal \( P_s \), the arrived signal power at the jammer \( P_a \) with distance \( r \) from the sensor, and the power of the background noise \( P_n \).

According to the traditional signal propagation model, the jammer will regard the arrived signal as a sensor transmission as long as the Signal-Noise-Ratio is higher than
some threshold, i.e., \( SNR = \frac{P_a}{P_n} > \theta \) where \( P_a = \frac{P_s}{\xi} \cdot Y \) with \( \theta \) and \( \xi \) called jamming decision \textit{threshold} and \textit{path-loss} factor, \( Y \) as a log-normally random variable. Therefore, \( r \geq \left( \frac{\theta \cdot P_n}{P_s \cdot Y} \right)^{\frac{1}{\xi}} \) is a range within which the sensor transmission will definitely trigger the jamming attack, named as \textit{triggering range}. As will be shown later, this range \( r \) is bounded by \( R \) from above, and \( r_s \) from below, where the distances from either bounds are decided by the jamming decision threshold \( \theta \). For simplicity, we assume triggering range is the same for each sensor.

\textbf{Jammer distance.} Any two jammer nodes are assumed not to be too close to each other, i.e., the distance between jammer \( J_1 \) and \( J_2 \) is \( \delta(J_1, J_2) > R \). The motivations behind this assumptions are three-fold: 1) the deployment of jammers should maximize the jammed areas with a limited number of jammers, therefore large overlapping between jammed areas of different jammers lowers down the attack efficiency; 2) \( \delta(J_1, J_2) \) should be greater than \( R \), since the transmission signals from one jammer should not interfere the signal reception at the other jammer, otherwise, the sensed sensor signals mixed with the jamming signals from the other jammer will not invoke this jammer; 3) the communications between jammers are impractical, which will expose the jammers to anomaly detections at the network authority.

4.2.2.2 Advanced attacker model

Although the basic reactive jamming model is quite energy-efficient, the attackers may alter their behaviors to evade the detection, for which two advanced reactive jamming models: \textit{probabilistic attack} and \textit{asymmetric response time delay} are considered in this chapter. In the first one, the jammer responds each sensed transmission with a probability \( \eta \) independently. In the second one, the jammer delays each of its jamming signals with an independently randomized time interval.

We do not specify the possible changes of jamming range \( R \) as an advanced model, since the trigger set in this case will not change, though the victim set varies. Further, we do not theoretically analyze the effects of various jamming decision threshold \( \theta \) in
this paper version, but we evaluate all these above factors in the simulation section.
Jammer mobilities are out of the scope of this chapter, which assumes that the jammers are static during our trigger-identification phase. This is quite reasonable, since the time length of this phase is short, as to be shown later.

4.2.3 Sensor Model

Besides monitoring the assigned network field and generating alarms in case of special events (e.g., fire, high temperature), each sensor periodically sends a status report message to the base station, which includes a header and a main message body containing the monitored results, battery usage, and other related content. The header is designated for anti-jamming purpose, which is 4-tuple: Sensor_ID as the ID of the sensor node, Time_Stamp as the sending out time indicating the sequence number, as well as a Label referring to the node’s current jamming status and TTL as the time-to-live field which is initialized as the $2D$ where $D$ is the diameter of this network.

According to the jamming status, all the sensor nodes can be categorized into four classes: trigger nodes $TN$, victim nodes $VN$, boundary nodes $BN$ and unaffected node $UN$. Trigger nodes refer to the sensor nodes whose signals awake the jammers, i.e. within a distance less than $r$ from a jammer. Victim nodes are those within a distance $R$ from an activated jammer and disturbed by the jamming signals. Since $R > r$, $TN \subseteq VN$. Other than these disturbed sensors, $UN$ and $BN$ are the unaffected sensors while the latter ones have at least one neighbor in $VN$, hence $BN \subseteq UN$, and $VN \cap UN = \emptyset$. The Label field of each sensor indicates the smallest class it belongs to. The relationships among these classes are shown in Fig. 4-2, where nodes in grey and blue are victim nodes around jammer nodes, and blue nodes are also trigger nodes, which invoke the jammer nodes. Nodes surrounding the jammed are are boundary nodes, while the others are unaffected nodes.
We assume that the detection of jammed signals can be 100% correctly completed via comparing the SNR, PDR and RSS, as shown in [59] in this work. Although this detection problem is also quite challenging, it is orthogonal to the service framework proposed in this chapter. We will dig into this problem in our future work, where various real-time applications embedded with this service framework will be developed.

4.3 Three Kernel Techniques

In this section, we mention three kernel techniques that we resort to in the proposed protocol. Most existing anti-jamming works consider only proactive jammers, while reactive jammers can bring up larger damage due to efficient attack and hardness to detect. To this end, we embed a group testing process, i.e., the randomized error-tolerant group testing by means of our designed randomized \((d, z)\)-disjunct matrix, to the routing update scheme, which avoids unnecessarily large isolated areas as [70] does. Moreover, most existing topology-based solutions [41][42] can only handle the single-jammer case, since lacking of knowledge over the jamming range and inevitable overlapping of the jammed areas bring ups the analytical difficulties. Regarding these issues, we resort to a minimum disk cover problem in within simple polygon problem and a clique-independent set problem.

4.3.1 Minimum Disk Cover in a Simple Polygon

Given a simple polygon with a set of vertices inside, the problem of finding a minimum number of variable-radii disks that not only cover all the given vertices, but also are all within the polygon, can be efficiently solved.

The latest results due to the near-linear algorithm proposed recently by [31], which investigates the medial axis and voronoi diagram of the given polygon, and provides the optimal solution using \(O(\varpi + \kappa \log \varpi + \log^6 \kappa)\) time and \(O(\varpi + \kappa \log \log \kappa)\) space, where the number of edges of the polygon is \(\varpi\) and nodes within it as \(\kappa\). We employ this algorithm to estimate the jamming range \(R\).
4.3.2 Clique-Independent Set

Clique-Independent Set is the problem to find a set of maximum number of pairwise vertex-disjoint maximal cliques, which is referred to as a maximum clique-independent set (MCIS) [27]. Since this problem serves as the abstracted model of the grouping phase of our identification, its hardness is of great interest in this scope. To our best knowledge, it has already been proved to be NP-hard for cocomparability, planar, line and total graphs, however its hardness on UDG is still an open issue. We prove the NP-hardness of this problem on UDG via a polynomial-time reduction from the Maximum Independent Set problem on planar graph with maximum node degree 3 to it.

From [23], the Maximum Independent Set problem is NP-hard on planar graph with maximum degree 3, and from [65], any planar graph $G$ with maximum degree 4 can be embedded in the plane using $O(|V|^2)$ area units such that its vertices are at integer coordinates and its edges consist of line segments of the form $x = i$ or $y = j$, for any integers $i$ and $j$.

**Theorem 4.1.** Clique-Independent Set problem is NP-hard on Unit Disk Graph.

**Proof.** Given an instance $G' = (V', E')$ of such a MIS problem, whose optimal value is denoted as $MIS(G')$, we construct an instance $G = (V, E)$ of the CIS problem as follows:

- Embed $G'$ in the plane in the way mentioned above [65].
- For each node $v_i \in V'$, attach two new nodes $v_{i1}$ and $v_{i2}$ to it and form a triangle $N_i = \{v_{i1}, v_{i2}, v_{i3}\}$, where each edge of this triangle $N_i$ is of a unit length $r = \sqrt{3}/3$.
- Since each nodes $v_i$ is incident to at most three edges, for all edges $(v_i, u), \cdots, (v_i, v)$, move their endpoint from $v_i$ to different $v_j$s, e.g., $(v_1, u)$ changes to $(v_{11}, u)$ and $(v_1, v)$ to $(v_{12}, v)$. Afterwards, for each of such edges $e = (u, v)$, assume that it is of length $t$, we divide it into $t$ pieces and replace each piece with a concatenation of 2 triangles (not necessarily equilateral), as shown in Fig. 4-1B. Therefore, any edge $e_y = (v_i, v_{i}) \in E'$ of length $|e_y|$ becomes a concatenation of $2|e_y|$ 3-cliques, denoted as $\{c_{y1}, c_{y2}, \cdots, c_{y1}^{s_1}, c_{y2}^{s_2}\}$. Because of the triangles $N_i$s, the two triangles at each corner of Fig. 4-1B may need slight stencches, which can be done in polynomial time.
The resulting graph $G$ is then a unit disk graph with radius $r = \frac{\sqrt{3}}{3}$.

![A](image1.png)  
**A** $G' = (V', E')$

![B](image2.png)  
**B** $G = (V, E)$

Figure 4-1. Polynomial Time Reduction

The reduction is as follows:

$(\Rightarrow)$: if $G'$ has a maximum independent set $M$, for each $u_i \in M$, we choose cliques of two kinds in the corresponding instance $G$: (1) the clique $N_i$ at $u_i$; (2) for each incident edge $e_{ij} = (u_i, u_j)$, choose cliques $\{c_{ij}^{1,2}, c_{ij}^{2,2}, c_{ij}^{3,2}, \ldots, c_{ij}^{\mid e_{ij} \mid,2}\}$. Since the clique $N_j$ at $u_j$ shares a vertex with $c_{ij}^{\mid e_{ij} \mid,2}$, it cannot be selected. For any edge $e_{jk} = (u_j, u_k)$ where $u_j \notin M$ and $u_k \notin M$, choose cliques $\{c_{jk}^{1,2}, c_{jk}^{2,2}, \ldots, c_{jk}^{\mid e_{jk} \mid,2}\}$. It is easy to verify that all the cliques selected are vertex-disjoint from each other.

Assume that after embedding $G'$ into the plane, each node $v_i \in V'$ has coordinate $(x_i, y_i)$, then edge length $|e_{ij}| = \| v_i - v_j \|_1 = |x_i - x_j| + |y_i - y_j|$. Therefore if we have an independent set of size $|M| = k$ for $G'$, we then have a clique independent set of size $k' = k + \sum_{(i,j) \in E'} |e_{ij}|$.

$(\Leftarrow)$: if $G$ has a clique independent set of size $k'$, since the lengths of the embedded edges are constant, then $G'$ has exactly an independent set of size $k = k' - \sum_{(i,j) \in E'} |e_{ij}|$. The proof is complete.

There have been numerous polynomial exact algorithms for solving this problem on graphs with specific topology, e.g., Helly circular-arc graph and strongly chordal graph [27], but none of these algorithms gives the solution on UDG. In this chapter,
we employ the \textit{scanning disk approach} in [26] to find all maximal cliques on UDG, and then find all the \textit{MCIS} using a greedy algorithm. In fact, by abstracting this problem as a \textit{Set Packing} problem, we can obtain a $\sqrt{n}$-approximation algorithm, however, it exhibits worse performance than the greedy algorithm proposed in our trigger identification procedure.

4.4 Trigger Identification Procedure

We propose a decentralized trigger-identification procedure. It is lightweight in that all the calculations occur at the base station, and the transmission overhead as well as the time complexity is low and theoretically guaranteed. No extra hardware is introduced into the scheme, except for the simple status report messages sent by each sensor, and the geographic locations of all sensors maintained at the base station. Three main steps of this procedure are as follows:

1. \textit{Anomaly Detection} – the base station detects potential reactive jamming attacks, each boundary node tries to report their identities to the base station.

2. \textit{Jammer Property Estimation} – The base station calculates the estimated jammed area and jamming range $R$ based on the locations of boundary nodes.

3. \textit{Trigger Detection} –

   \begin{itemize}
   \item the base station makes a \textit{short} testing schedule message $Z$ which will be broadcasted to all the boundary nodes.
   \item boundary nodes keep broadcasting $Z$ to all the victim nodes within the estimated jammed area for a period $Q$.
   \item all the victim nodes locally execute the testing procedure based on $Z$ and a global uniform clock, identify themselves as trigger or non-trigger.
   \end{itemize}

4.4.1 Anomaly Detection

Each sensor periodically sends a status report message to the base station. However, once the jammers are activated by message transmissions, the base station will not receive these reports from some sensors. By comparing the ratio of received reports to a predefined threshold $\psi$, the base station can thus decide if a jamming attack is happening in the networks. When generating the status report message,
each sensor can locally obtain its jamming-status and decide the value of the \textbf{Label} field (Initially trigger "TN"). In detail, if a node $v$ hears jamming signals, it will not try to send out messages but keep its label as \textit{victim}. If $v$ cannot sense jamming signals, its report will be routed to the base station as usual, however, if it does not receive ACK from its neighbor on the next hop of the route within a timeout period, it tries for 2 more retransmissions. If no ACKs are received, it is quite possible that that neighbor is a \textit{victim} node, then $v$ updates \textbf{Label} tuple as boundary "BN" in its status report. Another outgoing link from $v$ with the most available capacity is taken to forward this message. If the status report is successfully delivered to the base station with \textbf{Label} = \textbf{TN}, the corresponding node is regarded as \textit{unaffected}. All the messages are queued in the buffer of the intermediate nodes and forwarded in an FCFS manner. The \textbf{TTL} value is reduced by 1 per hop for each message, and the message will be dropped once its \textbf{TTL} = 0, to avoid self-loops.
The base station waits for the status report from each node in each period of length $P$. If no reports have been received from a node $v$ with a maximum delay time, then $v$ will be regarded as victim. The maximum delay time is related with graph diameter and will be specified later. If the aggregate report amount is less than $\psi$, the base station starts to create the testing schedule for the trigger nodes, based on which the routing tables will be updated locally.

4.4.2 Jammer Property Estimation

We estimate the jamming range as $R$ and the jammed areas as simple polygons, based on the locations of the boundary and victim nodes.

In the sparse-jammer case where the distribution of jammers is relatively sparse and there is at least one jammer whose jammed area does not overlap with the others, like $J_2$ in Fig. 4-2. By denoting the set of boundary nodes for the $i^{th}$ jammed area as $BN_i$, the coordinate of this jammer can be estimated as

$$(X_J, Y_J) = \left( \frac{\sum_{k=1}^{BN_i} X_k}{|BN_i|}, \frac{\sum_{k=1}^{BN_i} Y_k}{|BN_i|} \right)$$

where $(X_k, Y_k)$ is the coordinate of a node $k$ is the jammed area $BN_i$ and then further the jamming range $R$ can be estimated as

$$R = \min_{i \in BN_i} \{ \max_{k \in BN_i} (\sqrt{(X_k - X_J)^2 + (Y_k - Y_J)^2}) \}$$

since we assume all the jammers have the same range.

Otherwise in the dense-jammer case, as shown in Fig. 4-3, we need to first estimate the jammed areas, which are simple polygons (unnecessarily convex) containing all the boundary and victim nodes. This process consists of three steps: (1) discovery of convex hulls of the boundary and victim nodes, where no unaffected nodes are included in the generate convex polygons. (2) for each boundary node $v$ not on the hull, choose two nodes on the hull and connect $v$ to them in such a way that the internal angle at this reflex vertex is the smallest, hence the polygon is modified by
replacing an edge (dotted one in Fig. 4-3) by the two new ones. The resulted polygon is the estimated jammed area. (3) execute the near-linear algorithm [31] to find the optimal variable-radii disk cover of all the victim nodes, but constrained in the polygon, and return the largest disk radius as $R$.

**4.4.3 Trigger Detection**

Since the jammer behavior is reactive, in order to find all the trigger nodes, a straightforward way is that each sensor broadcasts one by one, and monitors if the jammers are invoked by sensing the jamming signals. However, this individual detection is quite time-consuming and all the victim nodes thus have to be isolated for a long detection period, or even returns wrong detection result in the presence of mobile jammers. In this case, the network throughput would be dramatically decreased. Therefore, to promptly and accurately find out these triggers from a large pool of victim nodes, emerges as the most challenging part of the proposed protocol, for which the idea of group testing is applied.

In this section, we only consider a basic attack model where the jammers *deterministically* and *immediately* broadcasts jamming signals once it senses the sensor signal. Therefore as long as at least one of the broadcasting victim nodes is a trigger, some jamming signals will be sensed, and vice versa. The performance of this protocol toward
Table 4-1. Message Containing Trigger Detection Schedule

<table>
<thead>
<tr>
<th>Time Slot</th>
<th>Channel</th>
<th>Node List</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$f_1$</td>
<td>$v_1, v_3, \cdots, v_n$</td>
</tr>
<tr>
<td>0</td>
<td>$f_2$</td>
<td>$v_1, v_2, v_4, \cdots, v_{n-1}$</td>
</tr>
<tr>
<td>0</td>
<td>$\vdots$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>0</td>
<td>$f_m$</td>
<td>$v_2, v_5, \cdots, v_n$</td>
</tr>
<tr>
<td>1</td>
<td>$f_1$</td>
<td>$v_2, v_4, \cdots, v_{n-2}$</td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\cdots$</td>
</tr>
</tbody>
</table>

sophisticated attacker models with probabilistic attack strategies will be validated in the next section.

All the following is the testing schedule over all the victim nodes, which is designed at the base station based on the set of boundary nodes and the global topology, stored as a message (illustrated in Table 4-1) and broadcasted to all the boundary nodes. After receiving this message, each boundary node broadcasts this message one time using simple flooding method to its nearby jammed area. All the victim nodes execute the testing schedule and indicate themselves as non-triggers or triggers. Since all the sensor nodes are equipped with a global uniform clock, and no message transmissions to the base station are required during the detection, the mechanism is easy to implement and practical for applications.

As shown in Table 4-1, for each time slot, $m$ sets of victim sensors will be tested. The selection of these sets involves a two-level grouping procedure.

First-level, the whole set of victims are divided into several interference-free testing teams. Here by interference-free we mean that if the transmissions from the victim nodes in one testing team invokes a jammer node, its jamming area will not reach the victim nodes in another testing team. Therefore, by trying broadcasting from victim nodes in each testing team and monitoring the jamming signals, we can conclude if any members in this team are triggers. In addition, all the tests in different testing teams can be executed simultaneously since they will not interfere each other. Fig. 4-4 provides an example for this. 3 maximal cliques $C_1 = \{v_1, v_2, v_3, v_4\}$, $C_2 = \{v_3, v_4, v_5, v_6\}$,
\( C_3 = \{ v_5, v_7, v_8, v_9 \} \) can be found within 3 jammed areas. Assume these three cliques are respectively the three teams we test at the same time. If \( v_4 \) in the middle team keeps broadcasting all the time and \( J_2 \) is awaken frequently, no matter the trigger \( v_2 \) in the leftmost team is broadcasting or not, \( v_3 \) will always hear the jamming signals, so these two teams interfere each other. In addition, node-disjoint groups do not necessarily interference-free, as the leftmost and rightmost teams show.

Second-level, within each testing team, victims are further divided into multiple testing groups. This is completed by constructing a randomized \((d, 1)\)-disjunct matrix, mapping each sensor node to a matrix column, and make each matrix row as a testing group (sensors corresponding to the columns with 1s in this row are chosen). Apparently tests within one group will possibly interfere that of another, so each group will be assigned with a different frequency channel.

The duration of the overall testing process is \( t \) time slots, where the length of each slot is \( L \). Both \( t \) and \( L \) are predefined, yet the former depends on the total number of victims and estimated number of trigger nodes, and the latter depends on the transmission rate of the channel. Specifically, at the beginning of each time slot, all the sensors designated to test in this slot broadcast a \( \tau \)-bit test packet on the assigned channel to their 1-hop neighbors. Till the end of this slot, these sensors keeps detecting possible jamming signals. Each sensors will label itself as a trigger
unless in at least one slot of its testing, no jamming signal is sensed, in which case, the label is converted to a non-trigger.

The correctness of this trigger test procedure is theoretically straightforward. Given that all the testing teams are interference-free, then the testing with different teams can be executed simultaneously. Given that we have an upperbound \( d \) on the number of trigger nodes and each testing group follow the \((d, 1)\)-disjunct matrix, which guarantees that each non-trigger node will be included in at least one group, which does not contain any trigger node, so each non-trigger node will not hear jamming signals in at least one time slot, but the trigger nodes will since the jammers are activated once they broadcast the test packets. Therefore, two critical issues need to be addressed to ensure this correctness: how to partition the victim set into maximal interference-free testing teams and estimate the number of trigger nodes \( d \), as follows. Though these two involve geometric analysis over the global topology, since it only takes the information of boundary and victim nodes as inputs, and is calculated at the base station, no message complexity is introduced.

4.4.3.1 Discovery of interference-free testing teams

As stated above, two disjoint sets of victim nodes are interference-free testing teams iff the transmission within one set will not invoke a jammer node, whose jamming signals will interfere the communications within the other set. Although we have estimated the jamming range \( R \), it is still quite challenging to find these interference-free teams without knowing the accurate locations of the jammers. Notice that it is possible to discover the set of victim nodes within the same jammed area, i.e. with a distance \( R \) from the same jammer node. Any two nodes within the same jammed area should be at most \( 2R \) far from each other, i.e. if we induce a new graph \( G' = (V', E') \) with all these victim nodes as the vertex set \( V' \) and \( E' = \{(u, v) | \delta(u, v) \leq 2R\} \), the nodes jammed by the same jammer should form a clique. The maximum number of vertex-disjoint maximal cliques (i.e. clique-independent set (CIS) ) of this kind provides an upperbound
of possible jammers within the estimated jammed area, where each maximal clique is likely to correspond to the nodes jammed by the same jammer.

The solution consists of three steps: **CIS discovery** on the induced graph from the *remaining* victim without test schedules, boundary-based **local refinement** and interference-free **team detection**. We iterate three steps to decide the schedule for every victim node.

**CIS discovery.** We first employ Gupta’s MCE algorithm [26] to find all the maximal cliques, then use a greedy algorithm, as shown in Alg. 4.4.3.1 to get the CIS.

**Algorithm 6** Finding Clique-Independent Set (FCIS)

1. **Input**: Induced Subgraph $G' = (W, E')$.
2. **Output**: The set $C$ of maximum number of disjoint maximal cliques.
3. Find out the set $S$ of all maximal (not disjoint) cliques by using Gupta’s MCE algorithm [26].
4. **while** $S \neq \emptyset$ **do**
5. Choose clique $C \in S$ which intersects with the minimum number of other cliques in $S$;
6. $C \leftarrow C \cup \{C\}$
7. Remove all the maximal cliques intersecting with $C$;
8. $S \leftarrow S \setminus \{C\}$
9. **end while**
10. **return** $C$

**Local Refinement.** Each clique we select is expected to represent the jammed area poisoned by the same jammer, and this area should not cover the boundary nodes.
However, we did not take this into account when discovering the CIS, and need to locally update it. Specially, for each clique, we find its circumscribed circle $CC$ and the concentric circle $CC'$ with radius $R$ of $CC$. In the case that $CC'$ covers any boundary nodes, we locally select another clique by adding/removing nodes from this clique, to see if the problem can be solve. If not, we keep this clique as it is, otherwise, we update it. This is shown in Fig. 4-5, where clique $C_1 = V_1V_2V_3V_4$ is chosen by CIS, but its $CC'$ covers boundary node $V_0$, then clique $C_2 = V_4V_5V_6V_7$ replaces $C_1$ in the testing team for the first round. Clique $V_1V_2V_3$ are left for the next round.

**Team Detection.** The cliques in CIS can also interfere each other, e.g. the clique $V_1V_2V_3V_4$ and $V_5V_6V_7V_9$ in Fig. 4-4. This is because the signals from $V_4$ will wake $J_2$, who will try to block these signals with noises and affect $V_5$ by the way. But if any two cliques $C_1$ and $C_2$ are not connected by any single edge, then they are straightforwardly interference-free, since the shortest distance between any node in $C_1$ and $C_2$ is larger than $2R$. But the farthest jammer waken by and from $C_1$ is $r < R$ distance away, whose jamming range can only reach another $R$ distance further, which is thus away from $C_2$. Therefore, the cliques in the obtained CIS of this kind are selected as testing teams. While the others are left for the next time slot.

In addition, in the worst case, any single maximal clique $C$ has at most 12 interfering cliques in the CIS, as the shadowed ones in Fig. 4-6. Therefore, at most 13 testing teams are required to cover all these cliques. If the number of channels $k$ given is larger than 13, then a frequency-division is available, i.e. these interfering cliques can still become simultaneous testing teams, on the condition each team can only use $\min\{\lceil \frac{k}{13} \rceil, m\}$ of the given channels, where $m$ is the number of radios per sensor. Otherwise, we have to use time-divisions, i.e. they have to be tested in different time slots.
4.4.3.2 Estimation of trigger upperbound

Before bounding the trigger quantity from above, the triggering range \( r \) should be estimated. As mentioned in the attacker model, \( r \) depends not only on the power of both sensors and jammers, but also the jamming threshold \( \theta \) and path-loss factor \( \xi \):

\[
\begin{aligned}
    r & \geq \left( \frac{P_n \cdot \theta}{P_s \cdot Y} \right) \frac{1}{\xi} \\
    \text{since the real time } P_n \text{ and } P_s \text{ are not given, we estimate } r \text{ based on the SNR cutoff } \theta' \text{ of the network setting. In fact, the transmission range of each sensor } r_s \text{ is a maximum radius to guarantee}
\end{aligned}
\]

\[
\begin{aligned}
    SNR = \frac{P_a}{P_n} = \frac{P_s \cdot Y}{P_n \cdot r_s^\xi} \geq \theta'
\end{aligned}
\]

Therefore, we can estimate \( r \) as

\[
    r \approx r_s \left( \frac{\theta}{\theta'} \right)^\frac{1}{\xi}
\]

where \( \theta' \) and \( \xi \) are parts of the network input, while \( \theta \) is assumed as a constant, which indicates the aggressiveness of the jammer. For this estimation, \( \theta \) can be first set as 10db, which is the normally lower bound of SNR in wireless transmission, and then adaptively adjusted to polish the service quality.
With estimated $r$, since all the trigger nodes in the same team should be within a $2r$ distance from each other, by finding another induced graph $G'' = (W_i, E'')$ from the victim nodes $W_i$ in team $i$, with $E'' = \{(u, v) \in E'' \text{ if } \delta(u, v) \leq 2r\}$, the size of the maximal clique indicates the upperbound of the trigger nodes, thus can be an estimate over $d$.

As mentioned above, all the parallel testing teams selected are interference-free, therefore we roughly regard each team to be the jammed area of one jammer. As a deeper investigation, the number of jammers that can be invoked by the nodes in the same team (six 3-clique within the red circles) can be up to 6, since the minimum distance between two jammers is greater than $R$ and $r \leq R$, as shown in Fig. 4-7. Therefore on the induced graph, the largest 6 cliques form the possible trigger set. However, since the jammer distribution cannot be that dense for the sake of energy-conserving, the former estimate over $d$ is large enough.

4.4.4 Analysis of Time and Message Complexity

**Time complexity:** By time complexity we mean the identification delay counted since the attack happens till all the nodes successfully identify themselves as trigger or non-trigger. Therefore, the complexity break downs into four parts: (1) the detection of jamming signals at local links $T_d$; (2) the routing of sensor report to the base station from each sensor node, and the testing schedule to each victim node from the base station, aggregated as $T_r$; (3) the calculation of CIS and $R$ at the base station $T_c$; (4) the testing at each jammed area $T_t$.

The local jamming signal detection involves the statistical properties of PDR, RSS and SNR, which is orthogonal to our work. We regard $T_d$ as $O(1)$ since it is an entirely local operation and independent with the network scale.

The routing time overhead is quite complicated, since congestions need to be considered. For simplicity, we consider that all the 1-hop transmission takes $O(1)$ time and bound $T_r$ using the diameter $D$ of the graph. As mentioned earlier, the base
station waits at most $O(2D)$ for the reports, so that is the upperbound of the one-way routing. As to the other way, we also bound it using $O(2D)$ to match any collision and retransmission cases.

The calculation of CIS resorts to the algorithm in [26], which finds $O(I \Delta)$ maximal cliques on UDG within $O(I \Delta^2)$ time, where $I = |E|$ and $\Delta$ refers to the maximum degree. We used a greedy algorithm to find a MCIS from these $O(I \Delta)$ cliques with $O(I^3 \Delta^3 Q)$ time: $O(I \Delta)$-time for each clique to check the overlapping with other cliques, $O(I \Delta)$-time to find a clique overlapping with minimum other cliques, and $Q$ denotes the number of testing teams. Notice that in practice, sensor networks are not quite dense, so the number of edges $I$ and maximum degree $\Delta$ are actually limited to small values. On the other hand, the time complexity of estimating $R$ is up to $O(\frac{n \Delta^2}{2} + n(\log \frac{n \Delta^2}{2} + \log^6 n)$ using the minimum disk cover algorithm as mentioned.

The testing delay $T_t$ depends on the number of testing rounds and the length of each round. Since the reactive jamming signal disappears as soon as these sensed 1-hop transmission finishes, each round length is then $O(1)$. The number of testing rounds is however complicated and bounded by Theorem 4.2.

**Lemma 1.** Based on the ETG algorithm, the number of tests to identify $d$ trigger nodes from $|W|$ victim nodes is upperbounded by $t(|W|, d) = O(d^2 |\ln |W||)$ w.h.p.

**Theorem 4.2. (Main)** The total number of testing rounds is upper bounded by

$$O\left(\max_{i=1}^{Q} \frac{13 \min\{d_i^2 |\ln |W_i||, |W_i|\}}{m}\right)$$

w.h.p, with $d_i = \min\{\sum_{s=1}^{6} |c_s(G_i)|, |W_i|\}$ and $c_s(G_i)$ is the $s^{\text{th}}$ largest clique over an induced unit disk subgraph $G_i = (W_i, E_i, 2r)$ in the testing team $i$.

**Proof.** First, from Lemma 1, at most $\frac{t(|W|, d)}{m} = \frac{d^2 |\ln |W||}{m}$ testing rounds are needed to identify all nodes in testing team $i$. Second, the set of testing teams that can be tested in parallel is 13, as mentioned earlier. Combining with the worst-case upperbound of triggers in each team, the upperbound on round is derived. \qed
If the jamming range $R$ is assumed known beforehand, similar to [56], the whole time complexity is thus

$$O(\max_{i=1}^{Q} \frac{13d_i^2 \log |W_i|, |W_i|}{m})$$

and asymptotically bounded by $O(n^2 \log n)$. It is asymptotically smaller than that of [56]:

$$O(\sum_{i=1}^{\Delta(H)} \max_j ([2 + o(1)) \frac{d_j^2 \log_2 |W_j|}{\log_2 (d_j \log_2 |W_j|)/m})$$

where $\Delta(H)$ refers to the maximum degree of the induced graph $H$ (in this new solution, maximum degree is not involved). By taking the calculation overhead for $R$ into account, the overall time complexity is asymptotically $O(n^2 \log n + n \log^6 n)$, which is $O(n \log^6 n)$ for $n \geq 4$.

**Message Complexity:** On the one hand, the broadcasting of testing schedule $Z$ from the base station to all the victim nodes costs $O(n)$ messages in the worst case. On the other hand, the overhead of routing reports toward the base station depends on the routing scheme used and the network topology as well as capacity. The upperbound is straightforward obtained in a line graph with the base station at one end, whose message complexity is $O(\frac{n(n-1)}{2})$.

With regard to the message overhead of the testing process. Considering that there are approximately $\frac{|W_i|}{d+1}$ victim nodes in each testing group of team $W_i$ (mentioned in the construction of randomized $(d, z)$-disjunct matrix in Appendix), the overhead of each testing group in a testing round is $\frac{|W_i|}{d+1}$ 1-hop testing message broadcasted by all victim nodes in each group of team $W_i$. Therefore, the over message complexity is

$$O(n^2 + \sum_{i=1}^{Q} |W_i| \max_{i=1}^{Q} \frac{d_i \log |W_i|, |W_i|}{m})$$

which is $O(n^2 \log n)$.

### 4.5 Advanced Solutions Toward Sophisticated Attack Models

In this section, we consider two sophisticated attacker models: probabilistic attack and variant response time delay, where the jammers rely each sensed transmission
### Table 4-2. Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T^+$</td>
<td>The number of false positive outcomes</td>
</tr>
<tr>
<td>$T^-$</td>
<td>The number of false negative outcomes</td>
</tr>
<tr>
<td>$u(i)$</td>
<td>The number of trigger nodes in test $i$</td>
</tr>
<tr>
<td>$x(i)$</td>
<td>The reaction time of jammer toward test $i$</td>
</tr>
<tr>
<td>$g(i)$</td>
<td>The outcome of test $i$</td>
</tr>
</tbody>
</table>

with different probabilities, instead of deterministically, or delay the jamming signals with a random time interval, instead of immediately. This may mismatch with the original definition of reactive jamming, which targets at transmission signals, instead of nodes or channels. However, clever jammers can possibly change their strategies to evade possible sensed detections. Also, a common sense indicates that as long as an activity is sensed by the jammer, it is quite possible that some other activities are following this. So delaying the response time still guarantees the attack efficiency, but minimize the risk of being caught by reactive detections.

Since our scheme is robust and accurate in the steps of grouping, generating disjunct matrix and decoding the testing results, the only possible test errors arise from the generation of testing outcomes. Nevertheless, by using the error-tolerant disjunct matrix and relaxing the identification procedures to asynchronous manner, our scheme will provide small false rates in these cases. Some notations can be found in Table 4-2. In this section, the terms `test` and `group`, the terms `column` and `nodes` are interchangeable.

#### 4.5.1 Upperbound on the Expected Value of $z$

First, we investigate the properties of both jamming behaviors and obtain the expected number of error tests in both cases through the following analysis. Since in practice, it is not trivial to establish accurate jamming models, we derive an upperbound of the error probability which does not require the beforehand knowledge of the objective jamming models, which is therefore feasible for real-time identifications. Since it is a relaxed bound, it could be further strengthened via learning the jamming history.
4.5.1.1 Probabilistic jamming response

A clever jammer can choose not to respond to some sensed ongoing transmissions, in order to evade the detection. Assume that each ongoing transmission has an independent probability \( \eta \) to be responded. In our construction algorithm ETG, where each matrix entry is IID and has a probability \( p \) to be 1, therefore for any single test \( i \) with \( i \in [1, t] \):

\[
\Pr[u(i) = x] = \binom{d}{x} p^x (1 - p)^{d-x}
\]  

Hence for each test \( i \), the event that it contains no trigger nodes but returns a positive result, has a probability at most:

\[
\Pr[g(i) = 0 & u(i) \geq 1] = \sum_{x=1}^{d} (1 - \eta)^x \binom{d}{x} p^x (1 - p)^{d-x}
\]

\[
= \left[ (1 - \eta)p + 1 - p \right]^d - (1 - p)^d
\]

\[
= (1 - \eta p)^d - (1 - p)^d < (1 - \eta)p
\]

Meanwhile, the event that it contains at least one trigger but returns a negative result, has a probability:

\[
\Pr[g(i) = 1 & u(i) = 0] = 0
\]  

Since in practical \( \eta \geq \frac{1}{2} \), we therefore have the expected number of false positive and negative tests is respectively at most \( pt/2 \) and 0.

4.5.1.2 Variant reaction time

The introduction of group testing techniques aims to decrease the identification latency to the minimum, therefore, if the jammer would not respond immediately after sensing the ongoing transmissions, but instead wait for a randomized time delay, the test outcomes would be messed up. Since it is expensive to synchronize the tests among sensors, we use a predefined testing length as \( L \), thus the test outcome of test \( i \in [1, t] \)
is generated within time interval \([([\frac{i}{m}] - 1)\mathcal{L}, \lceil \frac{i}{m} \rceil \mathcal{L}]\). There are two possible error events regarding any test \(i\).

- \(Fp(i)\): test \(i\) is negative, but some jamming signals are delayed from previous tests and interfere this test, where we have a false positive event;

- \(Fn(i)\): test \(i\) is positive, but the jammer activated in this test delayed its jamming signals to some subsequent tests, meanwhile, no delayed jamming signals from previous tests exists, where we have a false negative event.

Since the jammers in this chapter are assumed to block communications only on the channels where transmissions are sensed, for the following analysis, we claim that the interferences can only happen between any two tests \(i, j\) with \(i \equiv j \pmod{m}\). Denote the delay of jamming signals as a random variable \(X = \{x(1), x(2), x(3), \cdots x(t)\}\) where \(x(i)\) is the delay for possible jamming signals arisen from test \(i\). (1) For event \(Fp(i)\), consider the test \(i - m\), in order to have its jamming signals delayed to test \(i\), we have a bound on \(x(i - m) \in (0, 2\mathcal{L})\). Similarly, in order to have the signals of any test \(j\) delayed to \(i\), we have \(x(j) \in [([\frac{i}{m}] - 1)\mathcal{L}, ([\frac{i}{m}] + 1)\mathcal{L}]\). Further assume the probability density function of \(X\) is \(P(i) = \Pr[X = x(i)]\). Consider all the tests prior to \(i\), which are \(i\%m, 1 + i\%m, \cdots, i - m\), we then have the probability for \(Fp(i)\):

\[
(1 - p)^d \sum_{j=i\%m}^{i-m} \int_{([\frac{i}{m}] - 1)\mathcal{L}}^{([\frac{i}{m}] + 1)\mathcal{L}} P(w) dw (1 - (1 - p)^d)
\]

(4–3)

To simplify this expression, we assume that \(X/\mathcal{L}\) follows a uniform distribution within the range \([0, \beta]\) with a small \(\beta\), which is reasonable and efficient for attackers in practice. Since the nature of jamming attacks lies in adapting the attack frequency due to the sensed transmissions, too large delay does not make sense to tackle the ongoing transmissions. Under a uniform distribution, the probability of \(Fp(i)\) becomes:

\[
(1 - (1 - p)^d)(1 - p)^d \sum_{j=\max(i\%m, i-m-\beta-1)}^{i-m} \frac{2}{\beta}
\]

\[
= (1 - (1 - p)^d)(1 - p)^d([\frac{i}{m}] - 1) \frac{2}{\beta}
\]
Therefore, the expected number of false positive tests is at most

\[ T^+ \leq \sum_{i=1}^{t} (1 - (1 - p)^d)(1 - p)^d(\beta) \frac{2}{\beta} \]

\[ \leq 2 \sum_{i=1}^{t} (1 - (1 - p)^d)(1 - p)^d \]

\[ \leq 2(1 - (1 - p)^d)(1 - p)^d t \]

(2) For event \( F_n(i) \), following the similar arguments above, we have an upperbound of the probability for \( F_n(i) \) (assume that any delays larger than \( l \) at test \( i \) will interfere the tests \( j \) following \( i \) where \( j \in [\max(i / m, i - m - \beta - 1), i - m] \):

\[
(1 - (1 - p)^d) \int_{l}^{+\infty} \mathcal{P}(w)d\omega \\
\cdot \left( 1 - \sum_{j} \int_{\left(\frac{i-j}{m}+1\right)\mathcal{L}}^{\left(\frac{i-j}{m}-1\right)\mathcal{L}} \mathcal{P}(w)d\omega(1 - (1 - p)^d) \right) \]

\[ \leq (1 - (1 - p)^d)(1 - 2(1 - (1 - p)^d))(\beta - l)/\beta \]

\[ \leq (1 - (1 - p)^d)(1 - 2(1 - (1 - p)^d)) \]

So the expected number of false negative tests is at most

\[ T^- \leq (1 - (1 - p)^d)(1 - 2(1 - (1 - p)^d)) t \]  \( (4-4) \)

Therefore, we could use a union bound and obtain a worst-case error rate of each test:

\[ \gamma = \frac{p}{2} + 2(1 - (1 - p)^d)(1 - p)^d \]

\[ + (1 - (1 - p)^d)(1 - 2(1 - (1 - p)^d)) \]

\[ = (10\tau - 8\tau^2 - \tau^{-d} - 1)/2 \]

where \( \tau = (d/(d + 1))^d \). Intuitively, we can have an upperbound on the number of error tests as \( z = \gamma t = (10\tau - 8\tau^2 - \tau^{-d} - 1)/2 \), and take it as an input to construct the \( (d, z) \)-disjunct matrix. However, notice that \( z \) depends on \( t \), i.e., the number of rows of
the constructed matrix, we therefore derive another bound of $t$ related to $\gamma$, as shown by Corollary 2.0.1 in the appendix.

4.5.2 Error-tolerant Asynchronous Testing within Each Testing Team

By applying the derived worst-cast number of error tests into the ETG construction, we can obtain the following algorithm where tests are conducted in an asynchronous manner to enhance the efficiency.

As shown in Algorithm 7, after all the groups are decided, conduct group testing on them in $m$ pipelines, where in each pipeline any detected jamming signals will end the current test and trigger the next tests while groups receiving no jamming signals will be required to resend triggering messages and wait till the predefined round time has passed. These changes over the original algorithm, especially the asynchronous testing are located in each testing team, thus will not introduce significant overheads, however, the resulted error rates are limited to a quite low level.

4.6 Experimental Evaluation

4.6.1 Overview

As a lightweight distribute trigger-identification service, our solution will be experimentally evaluated from four facets:

- in order to show the benefit of this service, we compare it with JAM [70] in terms of the end-to-end delay and delivery ratio of the detour routes from the base station to all the sensor nodes, as the number of sensors $n$, sensor range $r_s$, and number of jammers $J$ vary within practical intervals.

- in order to show the acceleration effect of the clique-independent set in this solution, we compare the complexity of this solution to our previous centralized one [56], with varying the above four parameters, where both jamming and triggering range $R$ and $r$ are assumed to be known beforehand.

- in order to show the accuracy of estimating the jamming range by using the polygon disk cover algorithm, we provide the estimated jamming ranges as well as the error rate to the actual values.
Algorithm 7  Asynchronous Testing

1: **Input**: \( n \) victim nodes in a testing team.
2: **Output**: all trigger nodes within these victim nodes.
3: Estimate \( d \) as mentioned.
4: Set \( \gamma = (10\tau - 8\pi^2 - \tau^{-d} - 1)/2. //upper bound of error probability for each test. 
5: Set \( t = \left\lfloor \frac{\ln n(d+1)^2}{(d+1)^2} \right\rfloor. //number of rows. 
6: Construct a \((d, z)\)-disjunct matrix using ETG algorithm with \( t \) rows, and divide all the \( n \) victim nodes into \( t \) groups accordingly \( \{g_1, g_2, \cdots, g_t\} \).
7: /* For each round, conduct group testing on \( m \) groups using \( m \) different channels (radios). The testing is asynchronous in that, the \( m \) groups tested in parallel do not wait for each other to finish the testing, instead, any finished test \( j \) will trigger the test \( j + m \), i.e., the tests are conducted in \( m \) pipelines. */
8: for \( i = 1 \) to \( \left\lceil \frac{t}{m} \right\rceil \) do
9: Conduct group testing in groups \( g_{im+1}, g_{im+2}, g_{im+m} \) in parallel;
10: If any nodes in group \( g_j \) with \( j \in [im+1, im+m] \) detects jamming noises, the testing in this group finishes and start testing on \( g_{j+m} \).
11: If no nodes in group \( g_j \) detect jamming noises, while at least one other test in parallel detects jamming noises, let all the nodes in group \( g_j \) resend 3 more messages to activate possible hidden jammers. If no jamming signals are detected till the end of the predefined round length (\( L \)), return a negative outcome for this group and start testing on \( g_{j+m} \).
12: end for

- in order to show its performance and robustness towards tricky attackers, we provide its false positive/negative rate, when taking into account those two advanced jammer models, as well as the estimation of \( R \).

The simulation is developed using C++ on a Linux Workstation with 8GB RAM. A 1000 \times 1000 square sensor field is created with uniformly distributed \( n \) sensor nodes, one base station and \( J \) randomly distributed jammer nodes. All the simulation results are derived by averaging 20 random instances.

4.6.2 Benefits for Jamming-resistent Routing

JAM[70] proposed a jamming-resistent routing scheme, where all the detected jammed areas will be evaded and packets will not pass through the jammed nodes. This method is dedicated for proactive jamming attacks, which sacrifices significant packet delivery ratio due to the unnecessarily long routes selected, though the effects of jamming signals are avoided. We compare the end-to-end delay between each
Figure 4-8. Benefits for routing

sensor node and the base station, of the selected routes by evading the jammed areas detected by JAM, with that of the ones evading only trigger nodes. Although there are many existing routing protocols for unreliable network environments, the aim of this experiment is to show the potential of this service to various applications, instead of being a dedicated routing protocol.

Three key parameters for routing could be the number of Jammers $J$, jamming range $R$, jamming threshold $\theta$. As mentioned earlier, $\theta$ indicates the aggressiveness of the attacker and the triggering range $r \approx r_s (\frac{\theta}{\theta'})^{\frac{1}{2}}$. Therefore, with $r_s$, $\theta'$ and $\xi$ as fixed network inputs, the effect of $\theta$ can be exactly indicated by studying the effect of $r$ instead.
The whole network has $n = 1500$ nodes and sensor transmission range $r_s = 50$.

The results with respect to the three parameters $J \in [1, 20]$, $R \in [100, 200]$, $r \in [50, 150]$ are included in Fig.4-8A, 4-8B and 4-8C respectively. Notice that for each experiments, the other two parameters are set as the median value of their corresponding intervals. Therefore, $R = 150$ for Fig.4-8C, which matches the extreme case $R = r$. Furthermore, for the nodes that are in jammed areas for JAM and that are triggers for our method, in another word, unable to deliver packets to or from the base station, we count the delay as $n + 1$, which is an upperbound of the route length.

As shown in Fig. 4-8A and 4-8B, when $j$ and $R$ increases, the routing delay goes up, which is quite reasonable since the jamming areas get larger and more detours have to be taken. The length of routes based on JAM quickly climbs up to the upperbound, while that of our trigger method is much lower and more stable, specifically keeps less than 900 seconds. When triggering range $r$ is small, as in Fig.4-8C, the end-to-end delay of Trigger-based routing is much smaller than the other, while as $r$ increases the two approaches each other, since more victim nodes are triggers now.

4.6.3 Improvements on Time Complexity

In our previous work [56], we proposed a preliminary idea of this trigger detection, and provided a disk-based solution. However, its high time complexity limits its usage in real-time networks. As mentioned above, the time complexity of our new clique-based detection is proved to be asymptotically lower than the previous, while the message complexities are approaching each other.

Although the computational overhead for estimating $R$ is asymptotically huge, the phase is not the key part of our scheme, and can be easily improved by machine learning techniques. Therefore, in this section, we assume that both $R$ and $r$ are known beforehand, and validate the theoretical results through simulations on network instances with various settings. Specifically, the network size $n$ ranging from 450 to 550 with step 2, transmission $r_s$ from 50 to 60 with step 0.2 and number of jammers $J$ from 3...
to 10 with step 1. Parameter values lower than these intervals would make the sensor network less connected and jamming attack less severe, while higher values would lead to impractical dense scenarios and unnecessary energy waste.

Since the length of each reactive attack is equal to the transmission delay of the object sensor signal, note that in our trigger detection, only one message is broadcast by each sensor in the testing groups. Therefore, it is reasonable to predefine the length of each testing round as a constant. We set this as 1 second, which is far more enough for any single packet to be transmitted from one node to its neighboring nodes. Henceforth, the time cost shown in Fig. 4.6.3 only indicates the number of necessary rounds to find out all the triggers, and can be further reduced. The message complexity is measured via the average message cost on each sensor node.

As shown in Fig. 4-9A and 4-9B, this clique-based scheme completes the identification with steadily less than 10 seconds, compared to the increasing time overhead with more than 15 seconds of the disk-based solution, as the network grows denser with more sensor nodes. Meanwhile, its amortized communication overheads are only slightly higher than that of the other solution, whereas both are below 10 messages per victim node. Therefore, the new scheme is even more efficient and robust to large-scale network scenarios.

With the sensor transmission radius growing up, the time complexity of the disk-based solution gradually ascends (Fig. 4-9D and 4-9C) due to the increased maximum degree $\Delta(H)$ mentioned in the above analysis. Comparatively, the time cost of clique-based solution remains below 10 seconds, while the message complexity still approximates the other one.

Since sensor nodes are uniformly distributed, the more jammer nodes placed in the networks, the more victim nodes are expected to be tested, the identification complexity will therewith raises, as the performance of disk-based scheme shows in Fig. 4-9F and 4-9E. Encouragingly, the proposed scheme can still finish the identification promptly
Figure 4-9. Time and Message complexity
<table>
<thead>
<tr>
<th>J=5</th>
<th>Actual R</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated R</td>
<td>51.9542</td>
<td>61.378</td>
<td>72.5228</td>
<td>80.7886</td>
<td>92.9285</td>
<td>104.826</td>
<td></td>
</tr>
<tr>
<td>∆R</td>
<td>3.91%</td>
<td>2.29%</td>
<td>3.60%</td>
<td>0.99%</td>
<td>3.25%</td>
<td>6.21%</td>
<td></td>
</tr>
<tr>
<td>J=10</td>
<td>Actual R</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>Estimated R</td>
<td>52.9438</td>
<td>63.496</td>
<td>73.4763</td>
<td>82.4191</td>
<td>93.9339</td>
<td>104.202</td>
<td></td>
</tr>
<tr>
<td>∆R</td>
<td>5.88%</td>
<td>5.83%</td>
<td>4.96%</td>
<td>3.02%</td>
<td>4.37%</td>
<td>4.21%</td>
<td></td>
</tr>
<tr>
<td>J=15</td>
<td>Actual R</td>
<td>50</td>
<td>60</td>
<td>70</td>
<td>80</td>
<td>90</td>
<td>100</td>
</tr>
<tr>
<td>Estimated R</td>
<td>51.6574</td>
<td>65.5034</td>
<td>73.5997</td>
<td>83.4615</td>
<td>96.6998</td>
<td>107.21</td>
<td></td>
</tr>
<tr>
<td>∆R</td>
<td>3.31%</td>
<td>9.17%</td>
<td>5.14%</td>
<td>4.33%</td>
<td>7.44%</td>
<td>7.21%</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4-10. Estimation error of $R$

with less than 10 seconds, which grows up much slower than the other. It has slightly more communication overheads (10 messages per victim nodes) but is still affordable to power-limited sensor nodes.

4.6.4 Accuracy in Estimating Jammer Properties

Though the estimate of jamming range $R$ is only to provide an upperbound for $R$, such that the testing teams obtained accordingly are interference-free, we are also interested in the accuracy of this estimation. As shown in Fig. 4-10, we investigate the error rate $\Delta R$ for $R = [50, 100]$ when there are respectively $J = 5, 10, 15$ jammers.

Two observations are straightforward from these results: (1) all the estimated values are above the actual ones, however, less than 10% difference. This meets our requirement for a tight upperbound of $R$. (2) the error rates in case of fewer jammers are relatively lower than those with more jammers. This is because jammers could have large overlaps in their jamming areas, which introduces estimate inaccuracies. Thanks to the accurate estimation of $R$, the overall false positive/negative rate is quite small, as to be shown next.
4.6.5 Robustness to Various Jammer Models

In order to show the precision of our proposed solution under different jamming environments, we vary the two parameters of the jammer behaviors above: Jammer Response Probability $\alpha$ and Testing Round Length/Maximum Jamming Delay $L/X$ and illustrate the resulted false rates in Fig. 4-11A and 4-11B. To simulate the most dangerous case, we assume a hybrid behavior for all the jammers, for example, the jammers in the simulation of Fig. 4-11A not only launch the jamming signals probabilistically, but also delay the jamming messages with a random period of time up to $2L$. On the other hand, the jammers in the simulation of Fig. 4-11B respond each sensed transmission with probability 0.5 as well. All the simulation results are derived by averaging 10 instances for each parameter team.

As shown in both figures, we consider the extreme cases where jammers respond transmission signals with a probability as small as 0.1, or delay the signals to up to 10 testing rounds later. This actually contradicts with the nature of reactive jamming attacks, which aim at disrupting the network communication as soon as any legitimate transmission starts. The motivation of such parameter setting is to show the robustness of this scheme even if the attackers sense the detection and intentionally slow down the attacks. The overall false rates are below 20% for any parameter values.

In Fig. 4-11A, when $\alpha > 1/2$ which corresponds to practical cases, we find that the false negative rates generally decrease from 10% to 5% as $\alpha$ increases. Meanwhile the false positive rate grows gently, but is still below 14%, this is because as more and more jamming signals are sent, due to their randomized time delays, more and more following tests will be influenced and become false positive. In Fig. 4-11B, considering the practical cases where $L/X > 1/2$, both rates are going down from around 10% to 1%, since the maximum jamming delay becomes shorter and shorter compared to the testing round length $L$, in which case, the number of interferences between consecutive tests is decreasing.
A Probabilistic Jammer Response

B Random Jamming Delay

Figure 4-11. Solution Robustness

4.7 Discussion and Conclusions

One leftover problem to this service framework is the jammer mobility. Although the identification latency has been shown small, it would not be efficient toward jammers that are moving at a high speed. This would become an interesting direction of this research.

Another leftover problem is the application of this service. Jamming-resistent routing and jammer localizations are both quite promising, yet the service overhead has to be further reduced to for real-time requirements.

As a summary, in order to provide an efficient trigger-identification service framework, we leverage several optimization problem models and provide corresponding algorithms to them, which includes the clique-independent problem, randomized error-tolerant group testing, and minimum disk cover for simple polygon. The efficiency of this framework is proved through both theoretically analysis toward various sophisticated attack models and simulations under different network settings. With abundant possible applications, this framework exhibits huge potentials and deserves further studies.
CHAPTER 5
AN EFFICIENT MULTI-LINK FAILURE LOCALIZATION SCHEME IN ALL-OPTICAL NETWORKS

Due to the side benefit of electronic switching in optical networks [43], the problem of link failure localization, namely efficiently finding critical errors like fiber cuts and optical amplifier breakdowns, has seized an increasing attentions[3][58][68].

Besides the conventional monitoring schemes where each link is equipped with a monitor, major recent solutions [44][79][72][29][61][1] employ specific set of light-paths each with a pair of transmitter and receiver. Wavelength signals are launched at the transmitter and monitored at the receiver for any syndromes like signal disruption, anomaly high signal-to-noise ratio (SNR) or bit-error-rate (BER). The failed links exist in the light-paths with such syndromes and are localized via combinatorial analysis.

Most solutions based on this strategy target at single link failure in a small network, and try to minimize the number of light-paths (we call it path-trial). However, to our best knowledge, few of them provide an efficient solution to multiple link failure localization. It is obvious that as the number of link failures increases, the hardness of this problem also goes up. In addition, existing solutions rely on centralized calculations and thus cannot be applied to large-scale networks, where the quantity of link failures is also relatively larger as more links are involved. Specifically, among the latest works, Harvey et al. in [29] provide a complete theoretical review over the application of non-adaptive group testing to the probe selection for the failure localization over various graph topologies, however, only preliminary centralized solutions are sketched along with theoretical complexity bounds. In [61], Tapolcai et. al propose an efficient random code assignment based solution, however, only single-link failure can be handled. Ahuja et al. in [1] prove several interesting necessary and sufficient conditions on the graph connectivity for the link failures to be unambiguously localized. But the centralized solution they propose is based on the discovery of \((k + 2)\)-edge-connected subgraph.
for \( k \) link failures. The solution is limited to small networks due to its expensive subgraph search.

Besides these limitations of the existing solutions, some practical constraints have not been investigated. The first and the most important one is that the number of failed links cannot be known beforehand, or tightly upper-bounded. We formulate the problem of estimating this failure quantity as a graph optimization problem, \( \text{avg}-\beta\)-disruptor, which is within the scope of network topology vulnerability assessment [15][76]. Based on this estimated value, our proposed schemes go through. The second concern is the limited capacity of the WDM (wavelength-division multiplexing) technique as summarized in [7]. Dense-WDM (DWDM) provides up to 128 channels in a single fiber, which requires more power, higher accurate lasers and wave filters, as well as more expensive EDFA’s for amplifiers than Coarse-WDM (CWDM) which has only 18 channels. So taking the set up expense into account, some networks may adopt CWDM and the capacity of each link is limited. Third, some links may cause transmission failures only to a proportion of wavelengths [1], and some performance metrics, like SNR may not always correctly reflect the results of the path-trials.

To our best knowledge, this is the first attempt to minimize the number of path-trials for multiple link failure localization in various network scenarios. Our contributions are:

- For small networks with central control and multiple monitoring locations, we provide a centralized tree-decomposition based method, to localized multiple link failures whose quantity is proportional to the network size, instead of a constant.
- For large-scale networks without central control and only one monitoring location, we provide a localized random-walk based algorithm.
- For various network scenarios with specific constraints, we eliminate unrealistic assumptions and provide efficient adaptations to the proposed schemes.
- Besides theoretical supports for the performance complexity, we present extensive simulation results for networks with different topologies and sizes.

The rest of this chapter is organized as follows: Section 5.2 provides a review over the concept of the path-trial solution and the formal definitions of the localization
problem, as well as the breakdown of our algorithms. Section 5.3 presents our centralized localization method and Section 5.4 shows the localized solution. In Section 6.5, simulation results are exhibited while the related works are included in Section 5.1. In Section 5.6, we provide adapted versions of the two schemes with respect to practical constraints. Section 6.6 summarizes this whole chapter.

5.1 Related Works

For the existing works using the trial of light-paths [44][79][72][71][29][61][1], there are two measures over the expense of this scheme, the number of path-trials taken and the monitoring locations required. They are different since multiple light-paths can share the same receiver therefore monitors are always less than the light-paths. Among the latest works, [29] and [61] focus on minimizing the former measure, while [1] the latter. These three papers represent three directions of solving this problem. Ahuja et al. in [1] employ monitoring cycles and paths and aim to minimize the monitoring locations. Their solution reveals the relationship between the graph connectivity and the expense of single-link failure detection, which is quite interesting. However, limited by the high computational overhead in finding the highly-edge-connected subgraph, this technique is hard to be engrafted to large-scale networks with multiple link failures. In [61], Tapolcai et al. propose a random code assignment based solution, which uses the binary representation of each link as their alarm codes, therefore each single link can be unambiguously diagnosed, combined with the detection syndromes. The algorithm is simple and the local swapping optimization can be completed at the off-line stage, however, how to handle multiple link failures has not been discussed. Harvey et al. in [29] provide the first application of group testing theory into this localization topic and present a series of theoretical bound over the preliminary solution on various topologies. Nevertheless, the biggest difference between these two fields is whether the testing pool (path-trial) is required to be connected on the graph. By resorting to a random walk based scheme, we overcome this gap and present an efficient solution. Among existing
works, many assumptions like the pre-knowledge of the failure quantity have been made, which bring down the efficiency when applied to industrial applications. Moreover, as the scale of all-optical networks keeps growing, simple but efficient solutions that can handle large networks is becoming a great demand. To this end, we include our schemes in this chapter.

5.2 Background and Problem Branches

As stated in [1][61], a path-trial is a lightpath which consists of multiple links. The two ends of the path-trial are called transmitter and receiver respectively. Each path-trial will be equipped with a monitor at the receiver end, which monitors the received wavelength signal after it is injected at the transmitter and traverses through the path, to catch any abnormal signs like high SNR or BER (which we call positive) to indicate if any of the links within the path are failed. Note that each link can be included by multiple path-trials, and a failed link will poison all the trials containing it, therefore by carefully designing the path-trials and observing the monitored result of each one, it is possible to catch the failure links via combinatorial methods.

We include an example of this solution in Fig. 5-1: Given two graphs, both of which contain 5 vertices and 8 edges. Black nodes refer to monitoring locations. The red arrows indicate the path-trials consisting of multiple edges, e.g. the arrow in the left-hand graph $s \rightarrow v \rightarrow w \rightarrow s$ is a path-trial where a wavelength signal is injected into $s$, goes through this trial, and monitored at $s$ for abnormal signs. If $s$ fails to receive the signal, then at least one of $(s, v), (v, w), (s, w)$ is failed, otherwise all are good links. The left-hand and right-hand graph refer to the two problem branches studied in this chapter: Multi-LOC and $st$-LOC, i.e., multiple monitoring locations $(s, t, v, \cdots)$ and single-pair locations $(s, t$ only). The latter problem is harder since each trial is restricted to be starting from $s$, ending at $t$.

Therefore, given that any path-trial containing at least one link failure will return a positive monitoring result, and those consisting of operative links will return negative
Figure 5-1. An example of Path-trial

![Diagram of Path-trial](image)

Adaptions to three practical constraints:
- k-par: each WDM fiber can handle at most only k path-trials.
- d-unknown: no pre-knowledge over the upper-bound of the failure quantity.
- q-fail: each failure link still works for q wavelengths.

Figure 5-2. Multiple branches of the M-LFL (Multi-Link failure localization) problem

results, the problem of localizing multi-link failures is thus converted to a trial-selection problem, which can be abstracted as the following graph optimization problem $M$-$LFL$ (Multi-Link failure localization):

**Definition 5.2.1 (M-LFL).** Given an undirected connected graph $G = (V, E)$, where $V$ refers to the vertex set and $E$ edge set. We denote the cardinality of the vertex set by $|V|$ and edge set by $|E|$, where $|V| = n$ and $|E| = m$ are two given constants. Given that $d > 1$ edges are failed, the problem asks us to minimize the number of path-trials required to localize them.
Table 5-1. Main Notations

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{V}(g)$</td>
<td>vertex set of graph $g$</td>
</tr>
<tr>
<td>$\mathcal{E}(g)$</td>
<td>edge set of graph $g$</td>
</tr>
<tr>
<td>$\mathcal{D}(g)$</td>
<td>diameter of graph $g$</td>
</tr>
<tr>
<td>$n, m$</td>
<td>the number of nodes and links in the network</td>
</tr>
<tr>
<td>$\text{deg}(v)$</td>
<td>the node-degree of $v$</td>
</tr>
<tr>
<td>$d$</td>
<td>upperbound of the number of link failures</td>
</tr>
<tr>
<td>$\Psi(v)$</td>
<td>$v$-Span graph rooted at vertex $v$</td>
</tr>
<tr>
<td>$\mathcal{R}$</td>
<td>upperbound of the number of random walks taken</td>
</tr>
<tr>
<td>$T$</td>
<td>the number of path-trials used</td>
</tr>
</tbody>
</table>

We study basically two branches of this problem, i.e., Multi-LOC and st-LOC, which respectively refer to the case with unlimited number of monitoring locations and only a node pair $(s, t)$ as the transmitter and receiver, i.e., a single monitor at $t$. These two branches in fact match different application scenarios: for networks with relative smaller size and simpler topology, it is practical to freely place monitors at arbitrary nodes. Since centralized algorithms are easy to execute on this underlying structure, we provide a centralized tree-decomposition algorithm for it. On the other hand, for large-scale networks with complicated topologies, it is expensive to equip monitors and transmit results at the intermediate nodes but only a few terminal nodes, so we present a localized random walk based algorithm returning the set of path-trials starting and ending at a single node-pair. Beyond this, we consider three practical constraints, i.e., $k\text{-par}$, $d\text{-unknown}$ and $q\text{-fail}$ as mentioned. Fig. 5-2 shows the chapter flow.

The main notations for this chapter are included in Table 5-1. The terms node and vertex, link and edge, network and graph are interchangeable.

5.3 A centralized algorithm for moderate sized networks

In this branch, we can place a monitor at any node within the graph and want to minimize the number of path-trials used. Since there exist many single-link failure localization schemes, it is straightforward to consider partitioning the graph into small pieces where each is expected to contain a single-link failure, and locating them one by one. However, lots of challenging problems are hidden behind this idea: (1) how to do
the partition such that each subgraph contains only one failure? (2) even the partition is feasible, how to minimize the cost of the localization in each subgraph? (3) given that the localization in each subgraph is efficiently handled, is the total cost also low enough?

We design a centralized algorithm TDD (Alg. 8), which uses a tree-decomposition process to partition the graph into edge-disjoint tree-subgraphs, and employs an efficient single-link failure localization algorithm with theoretical cost bound on each tree. To handle these challenging problems and optimize the solution, we embed several techniques, like cycle-break operation, $L$-degree based greedy as well as local priority strategy, into the algorithm. In addition, a redundant tree-subgraph scheme is introduced to further improve the algorithm performance. All these will be illustrated in details in the following three subsections.

### 5.3.1 Tree-decomposition

It is reasonable to assume that the link failures fall into a Bernoulli distribution [68], where each edge $e$ has a uniform probability $f_e = p$ to be failed for some constant $p \in (0, 1)$. Then Lemma 5.3.1 follows.

**Lemma 5.3.1.** Given that at most $d$ out of $m$ links in failed, and the link failure rate follows an i.i.d Bernoulli distribution, the expected size of subgraphs with at most 1 link failure is $\frac{m}{d}$.

Knowing this size upperbound for each subgraph, we aim to address the left two question. We define a $v$-Span graph and $L$-degree for each vertex $v$, and adopt a greedy strategy.

**Definition 5.3.1.** ($v$-Span graph) Given graph $G = (V, E)$ and constant $d << |E|$, for any vertex $v \in V$, a $v$-Span graph $\Psi(v)$ is constructed as follows:

- do BFS (breadth-first-search) starting from $v$;
- include any vertex reached by BFS into $\Psi(v)$;
- if any edge $e$ reached by BFS introduces a cycle, eliminate the endpoint with smaller degree from $\Psi(v)$, as well as all its incident edges.
Algorithm 8 TDD (tree-decomposition based detection)

1: **Input**: Graph $G = (V, E)$ and constant $d$.
2: **Output**: The set $F$ of failed link set.
3:
4: $F \leftarrow \emptyset$;
5: /* Decompose the graph into a set of edge-disjoint tree-subgraphs $T = \{T_1, T_2, \ldots, T_d, \ldots, T_r\}$ */
6: $T \leftarrow \emptyset$;
7: $V_s \leftarrow V$;▷ local priority .
8: while $E \neq \emptyset$ do
9: Find the $v$-Span graph of each vertex $v$ as $\Psi(v)$ (they may overlap).
10: if $V_s == \emptyset$ then
11: $V_s \leftarrow V$;▷ if no abandoned vertices left, the greedy selection scope is the whole graph, otherwise only within the set of abandon vertices.
12: end if
13:
14: For all the vertices in $V_s$, sort them by their $L$-degree in a non-decreasing order as $v_1, v_2, \ldots$;▷ greedy on $L$-degree
15: $T \leftarrow T \cup \Psi(v_1)$;▷ use the $v$-Span graph of the first vertex in this order
16: $E \leftarrow E \setminus \mathcal{E}(\Psi(v_1))$;▷ edge-disjoint
17: $V_s \leftarrow V_s \setminus \{v_1\}$;▷ update the set of abandoned vertices
18: $V_s \leftarrow V_s \cup$ new abandoned vertices in finding $v$-Span for $v_1$.
19: end while
20:
21: /* Localize single-link failure on each tree-subgraph */
22: Run a single-link failure localization algorithm $HA(T_i)$ on each tree-subgraph $T_i$ with $i = 1, \ldots, r$ in parallel.
23: if $HA(T_i)$ returns one failed link $e$ then
24: $F \leftarrow F \cup \{e\};$
25: end if
26:
27: return Failed link set $F$

- **iterate this process till the number of edges in $\Psi(v)$: $\mathcal{E}(\Psi(v)) = \frac{m}{d}$**.

**Definition 5.3.2.** For any vertex $v$, the $L$-degree of $v$ is

$$D(\Psi(v)) + \log |\mathcal{E}(\Psi(v))|$$

$$|\mathcal{E}(\Psi(v))|$$

This $L$-degree serves as the measure in our greedy strategy for decomposing the graph. Specifically, we iteratively choose a vertex with the minimum $L$-degree, and taking its corresponding $v$-Span graph as a tree-subgraph till that each edge in the original graph is covered in exact one tree-subgraph.
There are three details to be noticed in this scheme. First, the third step in forming $\nu$-Span graph (Definition 5.3.1) is a cycle-break operation, which is used to avoid isolated edge. An isolated edge refers to the one all of whose neighboring edges (those sharing endpoints with this edge) have been included in some tree-subgraph already in the decomposing process, in which case we will have to use another tree-subgraph to cover this singleton edge and a path-trial only consisting of it, which is quite luxury. Second, $V_s$ at line 7 in $TDD$ (Alg. 8) is introduced to provide local priority and avoid abandoned vertex. Abandoned vertices refer to those which have been visited by the spanning process of previously chosen trees, but not included. In other words, they are involved with cycles and ruled out by the cycle-break operation. It is easy to see that normally vertex with small node-degree has a large $L$-degree, therefore these abandoned vertices get less likely to be chosen for spanning a tree-subgraph as more and more edges are removed from the graph. They will become small leftover fragments if we only span the tree-subgraph from vertices with global smallest $L$-degree. Similar to the case of isolated edge, this decomposition can be quite inefficient. Therefore, these abandoned vertices which are closed to the previously selected subgraph are included in a set $V_s$ and given higher priorities to be selected. Third, the definition of $L$-degree reflects the cost of the single-link failure localization algorithm in each tree, as to be explained later.

Although the detection of $\nu$-Span tree-subgraphs is an iterative process. For networks with fixed topology, this is an off-line algorithm which means all the path-trials can be pre-calculated for the need of real-time failure localization.

5.3.2 Single-Link Failure Localization on Tree-subgraphs

There are plenty of existing algorithms for single-link failure localization, e.g. depth-based method in [29] and RCA method in [61]. Generally, they all can be used as subroutine for the second step of our $TDD$ algorithm framework. In this chapter, we
employ the depth-based algorithm proposed by Harvey [29] (denoted as \(HA(G)\) on
graph \(G\)) for tree graph.

The algorithm can be sketched in two steps: first randomly pick a root from the tree,
and test level by level to localize the level \(l\) where the failure happens; second use the
all the edges in level \(0, \cdots, l-1\) as hubs to locate the single failure in level \(l\). There are
two depths defined for each edge, to handle the heavily unbalanced trees: \(depth\) refers
to the number of nodes in the path from \(e\) to the root; \(light\)-\(depth\) refers to the number of
\(light\) edges on that path. By defining the \(weight\) of node as its number of children, a path
from a node to its \(heaviest\) child is a \(heavy\) edge, otherwise \(light\) edge. All the edges of
the same \(depth\) or \(light\)-\(depth\) is included in a subtree traversed by a trial. With regard
to the tree in Fig. 5-3, where \((1, 0)\) means a \(depth\) 1 and \(light\)-\(depth\) 0. Assume edge
\((b, f)\) fails, it can be localized by the above 3 trials along with the depth information. Its
complexity is straightforward.

**Lemma 5.3.2.** For any tree \(G\), when the number of failure edges \(d = 1\), we have the
minimum number of trials \(L(G)\) as

\[
L(G, 1) = O(D + \log m)
\]

where \(m = D(G)\) and \(m = |E(G)|\).

**Proof.** It takes at most \(D\) trials to locate the first level with failures, and the binary search
within this level will take at most \(O(\log m)\) trials. \(\square\)

It can be noticed that the definition of \(L\)-degree reflects the cost in each tree-subgraph,
according to Lemma 5.3.2. With different single failure localization subroutine used, this
definition can be adapted accordingly.

5.3.3 Redundant Tree-Subgraphs

As is shown, if each of the tree-subgraph contains at most one failed link, the
localization can be completed within a single iteration. Otherwise, the multiple-link
failures contained by a single subgraph become false negative ones (positive items
diagnosed as negative). It is natural to further diagnosis these subgraphs after the first iteration of localization. However, for networks with long geographic distances between node pair, and thus large transmission latency, the tradeoff in time complexity of multi-iteration localization is not worthwhile.

We resort to redundant tree-subgraphs to address this problem. The redundant subgraphs share edges with those selected edge-disjoint tree-subgraphs. In another word, we generate more \( v \)-Span graphs to provide redundant path-trials. Since the most inefficient part of the tree-decomposition scheme lies in those isolated edges or small trees with few edges, we select another set of tree-subgraphs to cover all the edges, which are spanned from vertices with small \textit{node-degree}.

5.4 A localized algorithm for large-scale networks

Since most of existing failure localization schemes are centralized or requiring excessive local message exchange, they are inefficient for industrial implementation and maintenance in large-scale networks. Without any central controls and arbitrary monitoring locations, the localization problem becomes more challenging and is beyond the capability of our centralized tree-decomposition method. Moreover, due to the
transparency requirement of the fiber data-connections, monitors may only be placed at specific locations, so every path-trial will be restricted at both ends, and even $O(|E|)$ trials cannot provide reliable results.

In order to solve this problem, we provide a localized random walk based algorithm, which requires negligible implementation expense and results in great performance. The kernel of this scheme is a concept of $d$-disjunct matrix [16], which originates from group testing theory and perfectly matches the localization problem. Furthermore, we embed a local rarest first strategy to enhance the performance of the scheme.

### 5.4.1 Random Walk based Algorithm

A $d$-disjunct matrix is a binary matrix $M$ where within any combination of $d + 1$ columns, for each single column $c_0$ and the other $d$ columns $c_1, \cdots, c_d$, there is at least one row $r$ with $M[r, c_0] = 1$ and $M[r, c_i] = 0$ for all $i = 1, \cdots, d$. Due to the simple decoding method of this matrix kind, it has been widely applied to the anomaly localization out of various large-scale instances. In the context of link failure localization, assume the number of link failures is upper bounded by $d$, we let each column of the matrix represent a link, and each row represent a set of links. If each row forms a path-trial, since any good link will exist in at least one path-trial with negative monitoring results (no anomaly detected), guaranteed by the property of $M$ mentioned above, by eliminating all the links contained by negative path-trials, we can identify all the leftover links as faulty ones.

The implementation of this idea is non-trivial, since firstly the construction of $d$-disjunct matrix itself is quite challenging [16], yet restricting all the rows as paths on the given graph makes it even more difficult. In this chapter, we provide a random walk based scheme which returns a feasible $d$-disjunct matrix on the graph, and thus a solution to localize the failures. We refer it as $R$-walk, as shown in Alg. 9.
Algorithm 9 R-walk (Random Walk)

1: Input: Network $G = (V, E)$, a transmitter $s$, receiver $t$.
2: Output: The list $F$ of failed links.
3: 
4: Each link contains a unique identifier within $[1, \ldots, |E|]$.
5: The set of failed links is maintained at $t$, initialized as $F \leftarrow E$, i.e. all the links.
6: Launch a predefined $R_0$ number of wavelength signals from $s$ at the same time $C_0$, which is known by $t$. /* The value of $R_0$ is critical. */
7: Each node forwards its received signals randomly to an incident link, and attaches the link identifier in a header of the signal.
8: $t$ waits for a predefined length of time, retrieves from each successfully received signal the identifier of each traversed link, and exempts them from $F$.

5.4.2 Correctness and Complexity

From the following lemmas, we can see that with regarding these $R_0$ path-trials as rows, they will form a $d$-disjunct matrix with a high probability. Based on the definition of $d$-disjunct matrix, each good link will be traversed by at least one path-trial, which does not go through any of the $d$ failed links. Therefore, all good links will be exempted from the failed link list. The following lemmas reveal that with a high probability, the path-trials taken by Alg. 9 form a $d$-disjunct matrix.

Lemma 5.4.1. Given that $G$ is not bipartite, the stationary distribution of the edge random walk is $\pi_{(y, z)} = \frac{1}{2|E|}$ for any $(y, z) \in E$.

Proof. Denote the transition matrix as $P$, it is straightforward that $P((x, y), (y, z)) = \frac{1}{\deg(y)}$ for the transition probability from edge $(x, y)$ to edge $(y, z)$. Therefore the distribution on any edge $(y, z)$ is

$$(P^T \cdot \pi)_{(y, z)} = \sum_{x: (x, y) \in E} \frac{1}{\deg(y)} \cdot \frac{1}{2|E|} = \frac{1}{2|E|}$$

Based on this stationary distribution, we further investigate the probability of a walk traversing a specific vertex but not any $d$ others. Due to the space limit, we only provide the conclusion and omit the proof, which is similar to that in [12].
Lemma 5.4.2. Any random walk $W$ with length $t$ passes any single edge $e \in E$ with probability $\Omega(\frac{t}{|E|^2})$ where $T(|E|)$ is a $(\frac{1}{4|E|})^2$-mixing time of $G$, i.e., the smallest length $t_0$ of random walks which ends up with a distribution $\mu'$ as $\|\mu' - \mu\|_\infty \leq (\frac{1}{4|E|})^2$.

Lemma 5.4.3. Given any an edge $e$, an edge set $S \subseteq E$ with $e \notin S$ and $|S| \leq d$, the probability that any edge random walk in R-walk passes $e$ but not any edges in $S$ is $\Omega(\frac{1}{d^2 T^4(n)}).

Theorem 5.4.1. By choosing up to

$$\mathcal{R} = O(d^3 T^4(|E|) \log(|E|/d))$$

random walks, the constructed matrix $M$ is $d$-disjunct with probability $1 - o(1)$.

Remark: Since social networks with power-law topology are claimed to be fast mixing, i.e., $O(\log n)$-mixing time, $\mathcal{R} = O(d^3 \log^4(|E|) \log(|E|/d))$ is enough to approach the $d$-disjunctness.

5.4.3 Local Rarest First

Theorem 5.4.1 provides a complexity upper-bound of this algorithm. We can furthermore improve this algorithm by a Local Rarest First strategy usually which is favored by Bit-torrent techniques. The idea is to setup a counter $\rho(e)$ for each edge $e$, which stands for the number of different walks that have traversed it. Instead of randomly choosing one incident edge for the next step, each walk chooses the one with the minimum counter, called rarest to proceed. The implementation for this is quite simple, since each node is only required to maintain the counters of all of its incident links, and forward the signals to the rarest link.

The purpose of this scheme is to span the walks over the whole networks in a minimum latency. In fact, this adaptation to the algorithm $R$-walk still keeps the upperbound in Theorem 5.4.1, though unnecessarily tight. It is easy to see that if the counter values of all the edges sharing the same endpoint come to the same, then each
of this edge has been traversed by the same amount of path-trials, which coincides the expected outcome of the random walk.

5.5 Simulation Results

As mentioned above, our work is the first attempt to solve the multi-link failure localization problem on general graphs, with minimizing the number of path-trials. Therefore, in this current version, we do not compare the performance of the two schemes with the existing solutions, i.e., \((k + 2)\)-edge-connected subgraph based MP/MC algorithm in [1] with a different objective function (minimizing the number of monitoring locations); ILP and RCA-based solutions addressing a subproblem (only single-link failure) in [71] and [61] respectively. Instead, we present the experimental evaluations over the performance of our solutions for different network scenarios in terms of network sizes, topologies and link failure rate. The simulation is implemented by C++ on the Visual C++ 2005 platform, and executed on a Windows PC with 4GB RAM. All the results are averaged values from 100 instances.

5.5.1 The Centralized Algorithm

*TDD* is designed for networks with moderate size, therefore we adopt one real network ARPANET with \(n = 20\) vertices, \(m = 32\) edges, as well as three generated networks with \(n' = 100\) vertices, \(m' = 1000\) edges, which follow Erdos-Renyi, Power-Law [47] and Waxman [67] models respectively. The results are included in Fig. 5-4 where two measures: the ratio \(\alpha = \frac{T}{m}\) i.e., number of path-trials over number of edges and the ratio of correctly localized failed links \(\beta\) are investigated, as the link failure rate \(d/m \in [1\%, 9\%]\).

**Effect of Redundant Tree-subgraphs.** The introduction of redundant tree-subgraphs dramatically enhances the localization efficiency. As is shown, without redundancy, ARPANET requires less than \(0.4m = 12\) path-trials to locate up to 3 failures with success probability greater than 90%, while for these random networks, *TDD* requires around \(0.3m' = 300\) path-trials to locate up to 90 failures with success probability greater
than 60%. On the other hand, with redundant tree-subgraphs, ARPANET costs around 10% more path-trials and the success probability is slightly better. This is because the number of link failures in ARPANET is not larger than 3, so the probability of each tree-subgraph from $T$-$div$ contains no more than one failed link is quite high. So the redundancy helps little for ARPANET. However, for the random networks, with a trade-off of around $25\% m' = 250$ more path-trials, the success rate stays above 90% for up to 90 failures. The number of path-trials used only accounts is up to $0.6 m'$, which is $O(|E|)$, but it is much more efficient than convectional schemes where each link is a monitor. Since multiple path-trials share vertices, which can be used as monitoring locations. The number of monitoring locations is far less than $|V|$. How to optimize the number of monitoring locations, for these selected path-trials will be our future work.

Effect of Link Failure Rate. An increasing number of link failures brings up the difficulty of the localization, however, we conclude that the localization complexity is almost stable as $d/m$ increases from the experimental results. As shown in Fig. 5-4C and Fig. 5-4D, the number of path-trials keeps below $0.6 m' = 600$ with the $\beta$ greater than 90%, as the failure rate increases from 0.01 to 0.09.

Effect of Topology Variation. Results for the three topology models are different due to their variation of the node-degree distribution. Compared to Erdos-Renyi and Waxman model, Power-Law model has a larger degree variation, which results in many low-degree nodes and isolated edges in the tree-decomposition process. Therefore, without redundant tree-subgraphs, the power-law network has relatively lower success probability than others, while by including redundant subgraphs spanned from low-degree nodes, the number of isolated edges is decreased, and its success probability increases to the same level with the others.

5.5.2 The Localized Algorithm

To investigate the performance of the localized algorithm, we test it on large-scale random sparse networks, which are generated following the Power-law model with
average node-degree 2 and 3. On one hand, we gradually increase the size of $R_0$ and stop when the average false positive rate is 5% or lower (According to the decoding algorithm of $d$-disjunct matrix, all intact links are identified and all leftover ones are diagnosed as failures, so false negative ratio is 0, this is different from TDD). On the other hand, for the set of random walks passing no failed links among these $R_0$ ones, we also check their average length, which helps the receiver to decide how long it should wait for a signal to arrive, or regard it as a failure. Therefore, it reflects the online time complexity of this algorithm. We investigate the robustness of the algorithm to increasing network size in terms of the number of links $m$ and link failure rate defined as $d/m$ as above. For each network instance, we fix two vertices with the largest node-degree as the transmitter and receiver.
Upperbound on the number of path-trials. We generate networks with a range of 500 to 2500 nodes, respectively 1000 to 5000 links. The failure rate $d/m$ is set to 5% which is larger than normal cases. *R-Walk* is different from *TDD* in that the number of path-trials is pre-defined, so we vary this value and check the resulted localization success rate. As shown in Fig. 5-5A and 5-6A, as the network size or the failure rate increases, the required number of path-trials to localize the link failures is increasing as well. This is inevitable, since the more link failures we have, the harder the localization is. However, from Fig. 5-5B and 5-6B, we can see that the expense is much smaller than $d^2 \log m$, which is the theoretical upperbound [16] of the number of rows of the $d$-disjunct matrix, i.e. the number of trials to locate $d$ failures without the graph constraint, let alone the theoretical bound $d^3 \log^5 m$ as in Lemma 5.4.1. Furthermore, from Fig. 5-5C and
Figure 5-6. Robustness of \textit{R-Walk} localized algorithm to Link Failure Rate

5-6C, we can obtain a rough upperbound as

\[ \# \text{path-trials for } d \text{ failures in } m \text{ links} = cd \log m \]

where the constant factor \( c \) is less than 5 for a network with up to 5000 links and 250 failed ones.

\textbf{Average length of each negative path-trial.} On one hand, as shown in Fig. 5-5D, the average negative trial length increases from around 30 to 100 as the network size increases from 1200 to 5000 links. Since we only have a pair of transmitter and receiver, their diameter can be as large as the diameter of the graph, which increases as the graph gets larger. Based on this, the transmitter can use the maximum transmission time of 100 hops as the deadline for the WDM signals. On the other hand, Fig. 5-6D
shows as more links within the same network fail, the average trial length almost keeps the same at round 35 for a network with 1000 links.

**Effect of Topology Density.** As shown in Fig. 5-5A, from average node-degree from 2 to 3, the expense of the localization increases. With a denser topology, a large amount of random walks will traverse the same set of links or cycles, which brings down the efficiency of the path-trials. However, it is still bounded by $O(d \log m)$ and in the case of small $d$, the algorithm can also handle dense networks.

### 5.6 Adaptation to Additional Constraints

In this section, we consider three scenario types, $k$-par, $q$-fail and $d$-unknown, where the first two are relatively easier yet the last one is quite challenging. The cases in real networks could be a mixture of these three types. For simplicity, we only consider them individually and leave the complicated case for further work. Due to the space limit, we only provide the simulation results of these adaptation to the random walk algorithm for large-scale networks, yet for TDD on moderate sized networks, we only sketch the adaptation idea.

#### 5.6.1 Case 1: $k$-par

$k$-par refers to the case that each WDM link (not including out-edges of $s$ and in-edges of $t$) can only be traversed by in a total of $k$ path-trials. Although the capacity of WDM link refers to the maximum number of signals it can simultaneously support, since unexpected transmission delays always happen, we can precisely pre-determine neither the time-line of each signal, nor the set of signals traversing the same link at any time-point. Since an overload at any single link may ruin the overall localization result, we strengthen the constraint on the link capacity and upperbound the total trial quantity traversing each link, instead of simultaneous ones.

It is easy to adapt our algorithms to this case. For TDD, after locating the $v$-Span graph for each vertex $v$, since the path-trials on each tree are deterministic, the load of each link (the number of trials traversing it) is also known, and we can label the links
with heavy load ($> k$). By avoiding selecting these tree-subgraphs with heavy load links, the algorithm is feasible. On the other hand, for $R$-$Walk$, since the sole transmitter and receiver have inevitable transmission load, we do not consider this constraint on the edges outgoing from the transmitter or entering the receiver, but it applies to all the other edges. Straightforwardly, threshold on the trial counter used for our local rarest first strategy addresses this problem. Fig. 5.7A shows that in a Power-Law network with average degree 2, as the link failure rate ranges from 1% to 4.5% when restricting the maximum number path-trials traversing each link as $k = 18$ (according to the standard of CWDM), the required number of path-trials keeps almost the same with the case of un-restricted $k$. When the failure rate grows bigger than 4.5%, we require more path-trials since the constraint on $k$ starts to disqualify some random walks. As the failure rate grows, the gap could be bigger, but since most real networks have a smaller failure rate than 5%, the solution is generally feasible.

5.6.2 Case 2: $q$-fail

In the case that the wavelength signals of some path-trials can still pass some failed links, false negative rate increases. Assume that each failed link can never support more than $q$ wavelengths, then the certificate of a good link is to appear in at least $q + 1$ negative path-trials. Therefore, for $TDD$, by duplicating each path-trial by $(q + 1)$ different wavelength signals, i.e. with $(q + 1) \min((\mathcal{D}(G) \log m), (\mathcal{D}(G) + \log^2 m))$ path-trials, all the failures can be unambiguously localized.

On the other hand, for $R$-$Walk$, the $error$-$tolerant$ $(d, q)$-disjunct matrix which can be used to localize $d$ positive items in the presence of $q$ error tests, comes into picture. More formally, it is required that each column $C_i$ has at least $q + 1$ different rows $r_j$ where $M[r_j, c_i] = 1$ and $M[r_j, c_k] = 0$ for any other $d$ columns $c_k$ with $k \neq i$. 

117
Figure 5-7. Performance of adaptation to various constraints

Through the same induction as in Section 5.4 and [12], the $R$-walk algorithm for handling this scenario requires

$$\frac{2q^2d^3 \log^4(m)}{1 - 2\sqrt{\frac{4q}{\log(m/d)}} + 1 + 2q}$$

i.e., $O(qd^3 \log^4 m)$, path-trials in the worst-case. As shown in Fig. 5-7B, when $q$ increases from 1 to 5, the cost increases at most by 4 times (when $d/m = 7\%$), which matches the derived bound. However, the curve for $q = 10$ almost coincides with that for $q = 5$, so when $q$ is bigger enough, its influence over the number of trials is negligible. Therefore, $R$-walk is feasible for various values of $q$. 
5.6.3 Case 3: \(d\)-unknown

When the number of faulty links is unknown, all the existing solutions fail at the first step. Therefore, how to estimate this value has to solved beforehand. Trial-and-error methods are inapplicable since the long localization latency is not affordable, while simple guess over \(d\) may result in an unnecessary large value, which draws too many path-trials.

Since the failure localization procedure will not keep running all the time, the event that triggers it could be some anomalies detected at the network, for example, the throughput or PDR (packet delivery ratio) between specific node-pair sharply decreases. We simply denote this measure for the global network \(G\) as \(\mathcal{M}(G)\), and assume that the localization procedure is triggered when \(\mathcal{M}(G)\) falls below specific threshold. Apparently, \(\mathcal{M}(G)\) is related with some function regarding with the graph topology. Therefore, it is possible to investigate the graph topology in the off-line step to provide an estimation over \(d\) corresponding to the \(\mathcal{M}(G)\) value. Since our work is orthogonal with this detection phase, but focuses on the localization step, for simplicity, we formulate a function **pairwise routing connectivity**

\[
CONN(G) = \sum_{s,t \in V} \zeta(s, t)
\]

where the routing connectivity \(\zeta(s, t)\) between any node pair \((s, t)\) is defined as follows:

**Definition 5.6.1.** Any paths without a failed link are called operational paths. Given a node pair \((s, t)\) and a set of routing paths \(Q_{(s,t)}\) predetermined in the routing table, the routing connectivity \(\zeta(s, t)\) is:

\[
\zeta(s, t) = \begin{cases} 
1, & \text{if at least one path in } Q_{(s,t)} \text{ is operational} \\
0, & \text{otherwise}
\end{cases}
\]

Therefore, \(CONN(G)\) refers to the communication ability between any node pair in the presence of the link failures.
Based on this observation, we resort to an avg-$\beta$-disruptor problem, which finds an average size of edge set whose deletion from the graph makes $\text{CONN}(G) \leq \beta$, to approach the size of the real link failures. Specifically, for a network with $\text{CONN}(G) \leq \beta$, we first guess a value $d_x$ as the number of failures, randomly choose $d_x$ links as failures and check if the corresponding $\text{CONN}(G) \leq \beta$, if not, increase $d_x$ till this is satisfied. For each network instance, we obtain an estimated $d_x$ by averaging the results of 100 executions. We investigate its performance on a Power-Law network with $m = 1000$ links and an increasing failure ratio from 1% to 9%, using the $R$-Walk algorithm. According to the simulation results in Fig. 5-6C, $5d \log m$ path-trials is enough for the networks with the same size, topology density and failure rate, so we use $t = 5d_x \log m$ path-trials to check the localization error rate (Fig. 5-7C) and the cost ratio over the theoretical bound (Fig. 5-7D). Therefore, at least 90% failed links can be correctly localized and the number of path-trials almost approaches the one in the case with known $d$.

Notice that we introduce $\text{CONN}(G)$ to illustrate the framework for estimating $d$, but it is not limited to this and can be applied to various $\mathcal{M}(G)$.

5.7 Summary

In this chapter, we present two efficient path-trial algorithms to tackle the multiple link failure localization problem in all-optical networks. The centralized tree-decomposition method catches all the link failures with high probability for moderate sized networks with un-restricted monitoring locations, and the localized random-walk based algorithm handles large-scale networks with single-pair monitoring location. For implementation purpose, we also consider various practical constraints and include adaptations to our scheme correspondingly. These adaptations, especially the one for unknown failure quantity, provide a general solution framework associated with network vulnerability assessment, which could enlighten the further researches and developments. Since the principle of the provided solutions matches the group testing theory, they are also
capable for anomaly detection problem in various network types, not limited in all-optical networks.
Network topology vulnerability attracts more and more attentions since the last decade. As summarized in [15], numerous evaluation metrics have been proposed for this purpose, most of which are related with the network connectivity, specifically, how fragmented the network it is in the presence of failures. However, to our best knowledge, none of them take the network quality of service (QoS) into consideration. In this chapter, we observe that even before the network being fragmented into pieces, its QoS may already drop to an intolerant low level, and the network can no longer provide services. To this end, we present a novel QoS-aware vulnerability assessment framework.

QoS-aware topology vulnerability is critical for the Internet. As the Internet serves as the main carrier of more and more real-time applications, it has to satisfy several QoS measures with predefined thresholds, which include jitter, delay, bandwidth, packet loss and etc. Plenty of QoS routing protocols, e.g., Q-OSPF and PNNI [5], have been developed to meet these requirements. In practical networks, malfunctions often take place at intermediate network nodes/links for routing, consequently, *even the optimal routing path from source to destination can satisfy few of the QoS constraints*. In this cases, only improvements over routing protocols cannot enhance the robustness in unreliable network environments. Therefore, we are interested in study how many node/link failures are required to break down the network to such an extent. Notice that even the *subproblem* of this study, detecting an optimal routing path satisfying a set of QoS constraints, is nontrivial.

In practical applications, the constraints that are satisfied by the QoS optimal routing path can be categorized into additive and non-additive ones. Specifically, jitter, delay and packet loss of a routing path are the sum of each metric over all the links belonging to this path. However, constraints like bandwidth are not additive from edge to edge, but min/max or multiplicative functions. In principle, multiplicative
measures can be converted into additive ones in a logarithmic manner and min/max non-additive measures can be satisfied by ruling out all the unsatisfied single links. Therefore, classic theoretical studies over the QoS routing are normally formulated as a multi-additive-constraint path (MCP) problem [36][37][53]: Consider a network $G(V, E, s, t)$ with designated source node $s$, destination node $t$, and $m$ additive constraint $(c_1, \ldots, c_m)$, where each edge $(u, v) \in E$ has $m$ additive weights $w_i(u, v) \geq 0$, $i \in [1, \ldots, m]$. Find a path $P$ from $s$ to $t$ with

$$w_i(P) \triangleq \sum_{(u,v) \in P} w_i(u, v) \leq C_i$$

for all $i \in [1, m]$, if it exists.

MCP has been shown to be NP-complete but not strongly NP-complete [37], thus it is tractable for practical network sizes. Xue et al. [77] proposed approximation algorithms towards this problem and several other variants, however since MCP is merely a subproblem of our assessment, to decide if a given set of node/link is a feasible solution to our problem, the approximation ratio to MCP cannot be grafted. As summarized by Khadivi et al. in [33], plenty of mixed-metric based heuristics have been proposed to tackle MCP, however, any inaccuracies brought by the heuristics will possible provide a solution set that is far from optimal to our problem. In [40], Li et al. modified classic dynamic programming algorithm for MCP and provided a fast exact solution by effectively compacting the search space. However, as to be further defined later, the paths studied in our problem are not limited to those satisfying all given constraints, but may satisfy only a subset of them. Therefore, it is much more challenging than MCP.

The purpose of vulnerability assessment is to discover the weakness of the object network topology, whose result can be applied to optimizing network topology design, enhancing network robustness or destroying terrorist networks. We refer to these weak nodes/links as critical nodes/links, specifically, the minimum set of nodes/links whose
failure can bring down the network QoS to a certain low level are called QoS critical node/link set. Therefore, given two networks of the same size, the one which has a smaller QoS critical node/link set is of course more vulnerable. In this chapter, we measure the network QoS by the optimal QoS source-destination routing path, which satisfy the most QoS constraints over all routing paths. By requiring how much such an optimal path satisfies the multiple QoS constraints, we put a threshold on the network QoS and discover the QoS critical node/link set correspondingly.

Our contributions in this chapter are: (1) provide the first graph-theoretic QoS-ware vulnerability assessment method; (2) abstract the assessment problem as a graph optimization problem and study its hardness; (3) present one exact solution and two efficient heuristic algorithms for general network cases.

The rest of this chapter is organized as follows. Notations and the problem model are included in Section 6.1, where an integer program and preliminary hardness discussions are also presented. We include an exact algorithm using branch-and-bound and pareto optimality in Section 6.3. A near-optimal solution for a network with small constraint quantity and an efficient betweenness-based heuristic for general networks are introduced in Section 6.4. Extensive simulation studies over the performance of the proposed assessment algorithms are presented in Section 6.5. Section 6.6 summaries the whole chapter.

6.1 Problem Model

The system definition of this problem is: given a network with several additive QoS measures and a designated source-destination pair, there exists a set of source-destination (s − t) paths, each of which satisfies the constraints of some of these measures. Each measure is assumed with different credits (for example, delay is much more important than others in real-time applications), then satisfying the constraint on the most significant measure will earn a greatest credit for the source-destination path. Therefore, the level of satisfying multiple constraints by each path is quantified as a
credit, which we call **satisfactory score** of the specific path. With a threshold on this satisfactory score, an \( s-t \) path is called **QoS operational**. A network **fails** if **QoS operational** \( s-t \) path can be found. Henceforth, by checking how many node/link failures a given network topology can tolerate before it fails.

This evaluation process can be abstracted into a graph optimization problem called QoS-Critical Vertices (QoSCV) / QoS-Critical Edges(QoSCE) as following.

Given a directed graph \( G(V, E, s, t) \) with \( m \)-dim edge weight vector \( (u, v) \in E \): \( (w_1(u, v), w_2(u, v), \cdots, w_m(u, v)) \). The weight vector for each \( s-t \) path \( P \) is defined as \( W = (w_1(P), w_2(P), \cdots, w_m(P)) \) where \( w_i(P) = \sum_{(u,v) \in P} w_i(u, v) \) for all \( i \in [1, \cdots, m] \).

Given a constraint threshold vector \( C = (c_1, c_2, \cdots, c_m) \) with corresponding credit vector \( \Lambda = (\lambda_1, \lambda_2, \cdots, \lambda_m) \), we define a SAT score function \( F \) for path \( P \) as

\[
F(P, C, \Lambda) = g(W, C) \cdot \Lambda
\]

where

\[
g_i(W, C) = \begin{cases} 
1 & \text{if } w_i(P) \geq c_i \\
0 & \text{otherwise} 
\end{cases}
\]

The problem QoSCE asks for the a minimum subset \( S \) of \( E \) (each edge has a uniform cost as 1), where \( F(P, C, \Lambda) \leq \rho \) for any \( s-t \) path \( P \) in \( G(V, E \setminus S) \). The definition of QoSCV is similar.

The solution edges/vertices are referred as **QoS critical** edges/vertices respectively.

Notice that QoSCV can be readily converted into a non-uniform cost version of QoSCE through the following construction: for each vertex \( u \in V \), replace \( u \) with two new vertices \( u_1 \) and \( u_2 \), then add an edges \((v, u_1)\) for every edge \((v, u)\) \( \in E \) and an edge \((u_2, w)\) for every edge \((u, w)\) \( \in E \). Set \( \text{cost}(v, u_1) = +\infty \), \( \text{cost}(u_2, w) = +\infty \), \( \text{cost}(u_1, u_2) = 1 \), as shown in Fig. 6-1. Therefore, we only consider QoSCE problem in the remaining of this paper.
We can formulate the problem as an integer program. By introducing several variables: \( X_e = 1 \) if edge \( e \) is NOT removed in the optimal solution, 0 otherwise; \( Y_{P_i} = 1 \) if a \( s \rightarrow t \) path \( P \) has \( w_i(P) \leq c_i \), 0 otherwise; a large constant \( \epsilon = \max_i \left\{ \sum_{e \in E} w_i(e) \right\} \), we can describe QoSCE by the following integer program in Fig. 6-2.

The first constraint demands that the remaining graph satisfies the SAT score threshold. The second constraint demands that w.r.t any constraint, for any path \( P \), if no edge of \( P \) is removed (\( \sum_{e \in P}(1 - X_e) = 0 \)) and \( P \) satisfies the \( i^{th} \) constraint (\( Y_{P_i} = 1 \)), then we require \( w_i(P) \leq c_i \). Otherwise, \( w_i(P) \) is unrestricted. The third constraint demands that if no edge of \( P \) is removed, and \( P \) does not satisfy constraint \( i \), then \( w_i(P) > c_i \). It can be seen that the number of constraints of this program can be exponential, and thus hard to be solved by standard LP methods. Furthermore, the problem is NP-hard and cannot be approximated with a factor 2 even when \( m = 1 \), as shown next.
6.2 Hardness

QoSCE problem is quite challenging and its complexity class is still an open issue. We find that it does not belong to NP class and it cannot be approximated with a factor of 2.

**Lemma 6.2.1.** QoSCE is not in NP.

*Proof.* Given an edge subset \( S \) as a certificate to QoSCE problem on graph \( G \), the process of verifying this certificate equals to another decision problem QoS-SP on the remaining graph \( G \setminus S \): does there exist a path \( P \in G \setminus S \) satisfying \( \phi(P) \leq \rho \)? This problem is NP-Complete by reduction from MCP\[37\] problem with letting \( \rho = \sum_{i=1}^{m} \lambda_i \). Then if there exists a solution for the QoS-SP problem, then the path is a feasible path to MCP, otherwise, MCP has no solution.

Therefore, the certificate of QoSCE is not verifiable in polynomial time, thus not in the class of NP.

**Lemma 6.2.2.** QoSCE is NP-hard and it has no 2-approximation algorithm unless \( P=NP \).

*Proof.* Consider a special case of QoSCE where \( m = 1 \) and the problem becomes an edge blocker problem: find a minimum set of edges to remove, such that s-t shortest path is longer than a threshold. This problem has been proved as NP-hard in \([4]\) and not able to be approximated by 2 by reducing from vertex cover \([6]\).

6.3 Exact Algorithm

Despite the negative results mentioned above, for sparse networks of a small size we can still approach the exact solution using a brute force search, accelerated by a combined method of branch-and-bound and pareto optimality.

First of all, a concept of Pareto Optimal Path (PO), which is tightly related the problems of multi constraint shortest path \([36][37][33][53]\), is sketched as follows.

**Definition 6.3.1.** A path \( p \) is Pareto Optimal iff \( p \) is not dominated by any other paths, i.e. there does not exist a path \( q \) with \( w_i(q) < w_i(p) \) for all \( i \in [1, \cdots, m] \).
Algorithm 10 ParetoSearch

1: **Input:** direct graph $G = (V, E)$, constant $\rho$.
2: **Output:** a minimum set of edges $S \subset E$.
3:
4: $\beta \rightarrow$ size of min s-t cut of $G$; $\triangleright$ upperbound of $S$
5:
6: for every edge $e \in E$ do
7: \hspace{1cm} $h(e) \leftarrow 0$; $\triangleright$ # of PO s–t paths containing $e$, updated by the FindFeasiblePath function.
8: end for
9: Sort all edges as $e_1, e_2, \cdots e_{|E|}$ with $h(e_i) \geq h(e_{i+1})$;
10:
11: $r \leftarrow 1$; $\triangleright$ lowerbound for $|S|$ in the binary search
12: while $r \leq \beta$ do
13: \hspace{1cm} $\gamma \leftarrow r + (\beta - r)/2$;
14: \hspace{1cm} $\text{find}[^{\gamma}] \leftarrow \text{false}; \text{set}[^{\gamma}] \leftarrow \emptyset$;
15: \hspace{1cm} $L \leftarrow$ all $\gamma$-combination of edges; $\triangleright$ \binom{|E|}{\gamma}$
16: \hspace{1cm} Sort $L$ as $l_1, \cdots, l_{|L|}$ with $\sum_{e \in l_i} h(e) \geq \sum_{e \in l_{i+1}} h(e)$
17: \hspace{1cm} for $j \leftarrow 1$ to $|L|$ do
18: \hspace{2cm} $T \leftarrow L_j$;
19: \hspace{2cm} if FindFeasiblePath $(G, \rho, T) = \text{true}$ then
20: \hspace{3cm} continue; $\triangleright$ try next combination
21: \hspace{2cm} else
22: \hspace{3cm} $\text{find}[^{\gamma}] \leftarrow \text{true}; \text{set}[^{\gamma}] \leftarrow T$; break;
23: \hspace{2cm} end if
24: end for
25: if $\text{find}[^{\gamma}] = \text{true}$ then
26: \hspace{1cm} $\beta \leftarrow \gamma - 1$;
27: else
28: \hspace{1cm} if $\text{find}[^{\gamma + 1}] = \text{true}$ then
29: \hspace{2cm} return $\text{set}[^{\gamma + 1}]$;
30: \hspace{2cm} else
31: \hspace{3cm} $r \leftarrow \gamma + 1$;
32: \hspace{2cm} end if
33: end if
34: end if
35: end while
Algorithm 11 FindFeasiblePath

1: Input: directed graph $G = (V, E)$, constant $\rho$, a $t$-subset of edges $T$.
2: Output: true if satisfiable paths exist, false otherwise. Update $h(e)$ for each edge $e \in E$.
3: $G \leftarrow G(V, E \setminus T)$;
4: $\triangleright$ find the weight lower-bound for any path from $s$ AND from each vertex $v$ to the destination $t$.
5: Construct reverse graph $G' = (V, E')$ where edge $(u, v) \in E'$ iff edge $(v, u) \in E$.
6: for all vertices $v \in V$ do
7: use Dijkstra's algorithm on $G$ and $G'$ to find $[LB_1(s, v), LB_2(s, v), \ldots, LB_m(s, v)]$ and $[LB_1(v, t), LB_2(v, t), \ldots, LB_m(v, t)]$ where $LB_i$ is the weight of the shortest path on $i^{th}$ metric.
8: $V \leftarrow V \setminus \{v\}$ if the set of $m$ concatenated shortest paths through $v$ still has lower SAT s-core than $\rho$; $\triangleright$ ruling out impossible vertices for PO paths.
9: end for
10: $\triangleright L_v \doteq [(W_1(P_1), W_2(P_1), \ldots, W_m(P_1), prev(P_1)), \ldots, (W_1(P_k), W_2(P_k), \ldots, W_m(P_k), prev(P_k))]$
11: where $P_1, \ldots, P_k$ refer to all $k$ PO paths from $s$ to $v$, and $prev(P_k)$ refers to the predecessor of $v$ on path $P_k$.
12: $L_v \leftarrow \emptyset$ for each $v \in V \setminus \{s\}$;
13: $S \leftarrow \{s\}; L_s \leftarrow [(0, \ldots, 0, s)]$;
14: while $S \neq \emptyset$ do
15: extract any $u$ from $S$;
16: $\triangleright$ check using weight lower bound from $u$ to $t$
17: for each PO path $P_i$ from $s$ to $u$ do
18: upper bound of SAT score $\phi'(P_i) \leftarrow \sum_{j:W_j(P_i)+LB_j(u,t) \leq \lambda} e_j$;
19: if $\phi'(P_i) < \rho$ then
20: Delete tuple $(W_1(P_i), W_2(P_i), \ldots, W_m(P_i), prev(P_i))$ from $L_u$
21: end if
22: end for
23: end for
24: if $L_u$ is $\emptyset$ then
25: continue;
26: end if
27: for any edge $(u, v)$ do
28: $L'_v \leftarrow L_u \oplus (u, v)$; $\triangleright$ concatenate $s - u$ path and $(u, v)$
29: if Path of $L'_v$ dominates that of $L_v$ then
30: $L_v \leftarrow L'_v$; $S \leftarrow S \cup \{v\}$;
31: end if
32: end for
33: end while
34: if $L_t$ is $\emptyset$ then
35: return false;
36: else
37: for each $e \in E$ do
38: $h(e) \leftarrow \#$ of PO paths containing $e$;
39: end for
40: return true;
41: end if
Straightforwardly, after removing a set of edges, the max SAT score is less then $\rho$ if and only if $F(P, C, \Lambda) < \rho$ for any PO path $P$.

However, in the worst case, the number of PO paths is still beyond any polynomials of $|V|$. Even in [49], this is limited to $O(|V|^{k-1})$ through experiments, it is still too large for it to be solved as an IP.

**Weight Lower Bound Vector:**
$$LB(u, v) = (LB_1(u, v), LB_2(u, v), \ldots, LB_m(u, v))$$
where $LB_i(u, v) = \{ W_i(q) \mid q = \arg\min_{p=\text{path}(u,v)} W_i(p) \}$.

Our exact solution is included in Algorithm 10. The basic idea consists of two parts:

- The size of possible solution is upper bounded by the size $r$ of min $s - t$ cut. Do binary search on $[1, r]$ to check if each $\gamma$-set of edges is a feasible solution. For each value $\gamma$, edges are picked in an decreasing order of greedy function $h(e)$, which is number of Pareto Optimal $s - t$ paths containing $e$.

- For each $\gamma$-set of edges, span pareto optimal paths from $s$ to $t$. Branch and cut on each intermediate vertex $v$, check with the weight lower bound $LB$ to see if a potential satisfiable path can be through $v$, as shown in Fig. 6-3.

Although several acceleration techniques are embedded in this algorithm, it is still not scalable due to the binary search phase and update progress of pareto optimal paths. In the following section, we first provide an efficient network flow based heuristic for networks with a limited number of constraints, which generates near-optimal results; then we further present a fast and scalable greedy heuristic for general networks with arbitrarily large size and constraint quantity. Although the results returned by the greedy
heuristic have some gaps from the optimal solution, it is still accurate in the sense of vulnerability assessment, as shown in the simulation results later.

6.4 Heuristic Solutions

In this section, we provide two heuristic solutions MFMCSP and SDOP where the former one is near-optimal to networks with a small set of constraints and the latter one is efficient for general networks with arbitrarily large constraint sets.

**Algorithm 12** MFMCSP

1: **Input:** directed graph $G = (V, E)$, constraint set $M = \{c_1, \cdots, c_m\}$, credit vector $(\lambda_1, \lambda_2, \cdots, \lambda_m)$, satisfactory score threshold $\rho$;
2: **Output:** solution set of edge of QoSCE.
3: $S \leftarrow$ all the **minimal** combinations $ss$ of $M$ with $\sum_{c_i \in ss} \lambda_i > \rho$;
4: **for** each edge $(i, j) \in E$ **do**
5: $f(i, j) \leftarrow 0$; $f(j, i) \leftarrow 0$;
6: $c_f(i, j) \leftarrow 1$ and $c_f(j, i) \leftarrow 0$.
7: **end for**
8: **while** $S \neq \emptyset$ **do**
9: $ss \leftarrow$ extracted from $S$;
10: **while** $\exists q \leftarrow$ the shortest path satisfying all the constraints in $ss$ **do**
11: **for** each edge $(u, v) \in q$ **do**
12: $c_f(q) \leftarrow \min\{c_f(u, v) : (u, v) \in q\}$;
13: $f(u, v) \leftarrow f(u, v) + c_f(q)$; $f(v, u) \leftarrow -f(u, v)$;
14: $c_f(u, v) \leftarrow c(u, v) - f(u, v)$; $c_f(v, u) \leftarrow c(v, u) - f(v, u)$;
15: **end for**
16: **end while**
17: **end while**
18: all the vertices reachable from $s$ on the residual network induces a cut $T$.
19: **return** $T$.

6.4.1 A Heuristic Solution for Small $m$

In real application scenarios, the set of constraints that are taken into account is very limited, which may only consist jitter, packet loss, delay and some others. Therefore, it is quite practical to consider $m$ as a not quite large value, which gives rise to an exact solution called MFMCSP.

The basic idea is to first enumerate all the possible satisfiable combinations of constraints, for example $(c_1, c_3, c_5)$ if $\lambda_1 + \lambda_3 + \lambda_5 \geq \rho$. Therefore $s - t$ paths that satisfy
all the constraints within such a combination is a *satisfiable* path. We only consider
the set of minimal constraint combinations, since the set of paths satisfying a set of
constraints is surely the superset of those satisfying a superset of these constraints. (set
of paths satisfying \((c_1, c_3)\) of course contains those satisfying \((c_1, c_3, c_5)\).) With this set
of minimal combinations, we revise the classic Edmonds – Karp algorithm [19] to find the
minimum size of edge cut to cut all the augmenting \(s – t\) paths which at the same time
satisfy any of these combinations.

The pseudo-code of this algorithm is included in Algorithm 12.

6.4.2 A Heuristic Solution for Large \(m\) and Large Networks

To handle general network scenarios with arbitrary number of QoS constraints, we
provide an efficient heuristic algorithm in this section.

Alg. 11 checks if the remaining graph has a satisfiable path. However, when it
comes to large-scale networks, the local search process for PO paths may take too long.
Therefore, we define a relaxed metric \(\varphi(e)\) for each edge \(e\), and propose an algorithm
called SAT_TEST (Algorithm 13) to approximately decide if there exist a satisfactory path
in the remaining graph.

---

**Algorithm 13 SAT_TEST**

1: **Input:** directed graph \(G = (V, E)\), constant \(\rho\);
2: **Output:** true if a satisfactory path probably exists, false otherwise.
3: **for** every edge \(e \in E\) **do**
4: \(\varphi(e) \leftarrow \sum_{i=1}^{m} \frac{w_e}{c_i} \lambda_i;\)
5: **end for**
6: \(p \leftarrow\) shortest \(s-t\) path on metric \(\varphi;\)
7: **if** \(\varphi(p) > \rho\) **then**
8: \(\text{return} false;\)
9: **else**
10: \(\text{return} true;\)
11: **end if**

---

Based on this test, considering that the shortest path w.r.t each measure (single
metric shortest path) is more likely to be a satisfiable path, we count a betweenness
metric, i.e., number of appearances of each edge in such path kind, and remove the
edge with the greatest number of appearances one by one until the relaxed test returns false. The greedy heuristic is called SDOP and included in Algorithm 14.

Algorithm 14 SDOP

1: **Input**: directed graph $G = (V, E)$, constant $\rho$;
2: **Output**: a set $D$ of edges to be removed.
3: Set $T \leftarrow \emptyset$;
4: **while** SAT_TEST(G) = false **do**
5: find all $m$ single metric shortest paths $\{p_1, \cdots, p_m\}$
6: **for** all edges $e \in E$ **do**
7: find the one appears in the maximum number of such path.;
8: **end for**
9: $T \leftarrow T \cup \{e\}$; $E \leftarrow E \setminus \{e\}$;
10: **end while**
11: return $T$.

6.5 Performance Evaluations

In principle, the simulations are of three-fold: (1) to show the near-optimality of MFMCSUP, we compare its result for QoSCE with the optimal solution returned by the exact solution on a small network; (2) to show the efficiency and accuracy of the heuristic SDOP, we compare its assessment results and time complexity with MFMCSUP over random generated networks following various topologies; (3) to show the scalability of SDOP, we test it on a series of power-law networks with increasing size and constraint quantity.

6.5.1 Dataset and Setup

With regard to the topologies adopted, for (1) we use a classic backbone network: NSFNET T1 1991 network with 14 nodes and 21 bidirectional edges [50]; for (2) and (3) we use a well-known Internet topology generator BRITE [45] to generate topologies at three different internet levels: Flat Router-Level only, Flat AS-Level only, Hierarchical Top-down that follow two models: Power-law and Waxman.

Regarding the edge weights, similar to [2][77], all edge weights are generated following Uniform Distribution within the range [1, 10] and the each dimension of the constraint priority vector $(\lambda_1, \cdots, \lambda_m)$ obeys the same distribution. We call $\sum_i^m \lambda_i$ as the
full score of the given topology, and in order to guarantee that the QoS-optimal path of the topology can achieve such a full score before any edges are removed, we use the weight vector of a shortest path (w.r.t hop) count in the given topology as the constraint vector. The satisfactory rate is defined as the ratio of the threshold over full score, i.e. $\frac{\rho}{\sum_{i}^{m} \lambda_i}$. All the tests are implemented in C++ and performed on a 2.33GHz Linux Workstation with 8GB RAM.

6.5.2 Efficiency of Heuristic Solutions

We run the exact algorithm ParetoSearch and heuristic MFMCSP on the NSFNET backbone with $m = 5$ and 30 teams of random assigned edge weights, threshold and credit vectors.

First we investigate how fast MFMCSP is compared to ParetoSearch. As shown in Table 6-1, the time cost of MFMCSP is much lower than that of ParetoSearch, where the latter one can be up to 65 times of the former one.
Then our interest is to see how likely MFMCSP will return the exact solution, which in our simulation is quantified as the ratio of the instances where MFMCSP catches the optimal (we call it success probability).

As shown in Fig. 6-5, when $\rho \leq 0.6$, with at least probability 0.7, MFMCSP can catch the exact solution. Since from the attacker point of view, targeting at a $\rho$ larger than 0.6 makes little sense, the slightly lower success probability for $\rho \in [0.7, 0.9]$ does not ruin the algorithm. Generally, this success probability goes down as the $\rho$ increases. This is natural since the small $\rho$ is, the nearer the solution is to the min s-t cut. When $\rho$ is large, it is harder for the max flow algorithm to catch the minimum solution.

Furthermore, we would like to see how far is the result of MFMCSP from the exact solution. Also shown in Fig. 6-5, at least 83% MFMCSP results are less than 1.5 times of the exact solution.
As stated above, MFMCSP can serve as a near optimal solution for network small number of constraints. However, its scalability is still limited by its constraint enumeration and flow augmentation phase. Therefore, we would like to see if SDOP can accelerate this calculation process, as well as retain the correctness of the assessment.

Fig. 6-6A reports the solution sizes of QoSCE detected by MFMCSP on six 500-node 1500-edge networks with moderate $m = 5$, as shown. For each model, the solution size of QoSCE w.r.t satisfactory rates ranging with $[0.05, 0.95]$ with a step of $0.05$ are presented. Since larger QoS solution sizes indicates a higher vulnerability, we can derive the following rough sequence with the QoS vulnerability, which starts with the most robust topology: Router-PowerLaw > AS-Waxman > Hierarchical- Waxman >
Router- Waxman > Hierarchical- Powerlaw > AS-Powerlaw. Based on this evaluation, for Router-only level QoS-sensitive networks, PowerLaw model is more robust, while for AS-only level networks, Waxman model is a better choice. Another observation is that the QoSCE solution sizes of most network topologies are quite stable despite the changes of satisfactory rate. This due to the uniformly distributed edge weights assumed, where a large amount of $s - t$ paths exist and have similar satisfactory scores, therefore, all these paths are required to be cut and the minimum $s - t$ cut of the graph becomes the optimal solution of QoSCE, which remains unchanged as the satisfactory rate decreases. The time cost of MFMCSP on these topologies are bounded by around 15 seconds, as shown in Fig.6-6B. As the satisfactory rate decreases, the time complexity increases because more satisfactory paths are discovered and flow-augmented. Though this network has only 1500 edges, since the flow-augmenting process in MFMCSP only visits each edge once, this solution can also be applied to a large-scale network. However, the proposed heuristic SDOP is shown to achieve the similar performance vulnerability assessment with much less time expense, so we do not test MFMCSP on large network.

Fig. 6-6C presents the QoSCE solution sizes derived by SDOP for the same six topologies as above. Although the solution sizes are larger than the optimal values returned by MFMCSP, it provides the same robustness sequence. Therefore, the SAT_TEST does provide a good estimation over the network. Furthermore, the time expense of the assessment is reduced from 15 seconds to 2 seconds by use of this algorithm. Therefore for real-time networks with dynamic links and QoS measure requirements, this algorithm can be applied by promptly recalculate the QoSCE problem for each time slots with different inputs.

To show the scalability of the SDOP algorithm and this assessment framework, we test the time expense on random networks generated from the Power-Law model with size increasing from 1000-node to 10000-node, as well as the density increasing to 6
Table 6-2. The time cost of SDOP (Single Dimension Optimal Path) for large networks

<table>
<thead>
<tr>
<th>n</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
<th>5000</th>
<th>7000</th>
<th>9000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Cost(sec)</td>
<td>6.624</td>
<td>7.533</td>
<td>8.522</td>
<td>17.996</td>
<td>28.205</td>
<td>37.184</td>
<td>41.553</td>
</tr>
</tbody>
</table>

Table 6-3. The time cost of SDOP for large constraint amount

<table>
<thead>
<tr>
<th>m</th>
<th>5</th>
<th>15</th>
<th>20</th>
<th>30</th>
<th>50</th>
<th>75</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Cost(sec)</td>
<td>6.343</td>
<td>18.769</td>
<td>23.809</td>
<td>38.321</td>
<td>64.811</td>
<td>99.320</td>
<td>134.576</td>
</tr>
</tbody>
</table>

edges per node. The results are included in Table 6-2. As the network size increases, the time cost goes up from around 6 seconds to 41 seconds, which is surely affordable by real-time applications. A majority of the time expense arises from the calculation of shortest paths w.r.t each constraint, therefore, SDOP can be further accelerated through sampling techniques, instead of enumerate all the shortest paths. Furthermore, we test its efficiency for a Power-Law network with 5000-node up to 100 constraints, by taking into account possible extremely many constraints in the next-generation Internet. As shown in Table 6-3, SDOP can terminate within 135 seconds in this extreme case.

6.6 Summary

We propose the first QoS-aware assessment framework for network topology vulnerabilities, which is formulated as a graph optimization problem whose theoretical intractability is shown. By providing one near-optimal solution and one efficient heuristic, the scalability of this framework is validated through extensive simulation studies over several popular network models.
In this thesis, we investigate the application of discrete optimization techniques, in particular Group Testing and Graph Optimizations, in four network security and reliability problems. The results we derived include

- a 2-mode real-time malice detection framework against application DoS attacks;
- a trigger detection service for freezing and locating wireless reactive jamming attacks;
- a graph-constrained group testing based multi-link failure detection algorithm for all-optical networks;

Most of these works provide a brand new perspective over the problem in the corresponding field, by overcoming the difficulty of applying the classic mathematical models to practical problem scenarios. Thanks to the long research history of discrete optimizations, more and more network problems could find sufficient theoretical supports in there, which will be the focus of my future work, besides improving the solutions of the listed problems.
REFERENCES


BIOGRAPHICAL SKETCH

Ying Xuan received his Doctor of Philosophy in computer information science and engineering from the University of Florida in the fall of 2011. He obtained his Bachelor of Engineering in computer science and engineering from the University of Science and Technology of China in the summer of 2006. His research topics include group testing theory, network security, network reliability and social networks.